UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – TELECOMUNICACIONES

MECANISMOS PARA LA ASIGNACIÓN DINÁMICA DE RECURSOS EN UNA ARQUITECTURA DE RED.

TESIS
QUE PARA OPTAR POR EL GRADO DE:
DOCTOR EN INGENIERÍA

PRESENTA:
ALFREDO PIERO MATEOS PAPIS

TUTOR PRINCIPAL
DR. FRANCISCO JAVIER GARCÍA UGALDE. FACULTAD DE INGENIERIA. UNAM

COMITÉ TUTOR
DR. JOSÉ ALBERTO DOMINGO INCERA DIÉGUEZ. PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
DR. DEMETRIO FABIÁN GARCÍA NOCETTI. INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS Y EN SISTEMAS. UNAM

MÉXICO, D. F., NOVIEMBRE 2013

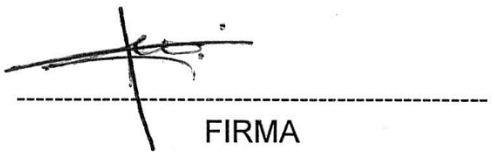
JURADO ASIGNADO

Presidente:	DR. HÉCTOR BENÍTEZ PÉREZ
Secretario:	DR. VÍCTOR RANGEL LICEA
Vocal:	DR. FRANCISCO JAVIER GARCÍA UGALDE
Primer Suplente:	DR. MICHAEL PASCOE CHALKE
Segundo Suplente:	DR. FEDERICO JOSÉ KUHLMANN RODRÍGUEZ

Lugar donde se realizó la tesis: México, Distrito Federal

TUTOR DE TESIS:

DR. FRANCISCO JAVIER GARCÍA UGALDE



FIRMA

Dedicatoria

Para mi esposa Rosa María, por su apoyo y afecto incondicionales. Sin ella este proyecto no hubiese llegado a término.

Agradecimientos

El autor de esta Tesis agradece:

De manera muy especial al Dr. Francisco Javier García Ugalde, presidente del Comité Tutor, por su atención, asesoría y apoyo.

Al Dr. José Alberto Domingo Incera Diéguez y al Dr. Demetrio Fabián García Nocetti, miembros del Comité Tutor, por su orientación y atención.

Al Dr. Fernando Arámbula Cossío, el Dr. Héctor Benítez Pérez y el Dr. Javier Gómez Castellanos, presidentes en turno del Subcomité Académico por Campo de Conocimiento, de Ingeniería Eléctrica de la UNAM, y al Dr. Luis A. Álvarez Icaza Longoria, Coordinador del Programa de Posgrado en Ingeniería de la UNAM, por su atención, apoyo y respuesta que hicieron posible que el autor de esta Tesis pudiese cumplir con los requerimientos académicos y administrativos del Programa de Maestría y Doctorado de la UNAM.

A la UNAM por permitir realizar los estudios de Doctorado en un tiempo prolongado.

A los miembros del jurado, Dr. Héctor Benítez Pérez, Dr. Víctor Rangel Licea, Dr. Francisco Javier García Ugalde, Dr. Michael Pascoe Chalke, Dr. Federico José Kuhlmann Rodríguez, por su revisión de las versiones previas de esta Tesis.

A la Universidad Autónoma Metropolitana, Unidad Cuajimalpa, por dar las facilidades para poder realizar el trabajo de esta Tesis, como profesor asociado de esta institución.

A la M. I. Abigail María Elena Ramirez Mendoza, Oficina del SACC de Ingeniería Eléctrica de la UNAM, por las atenciones brindadas en los diversos trámites para cumplir los requisitos administrativos del Programa de Maestría y Doctorado de la UNAM.

Índice General

Índice de Figuras

Índice de Tablas

Resumen

En las redes[1] fijas que ofrecen Calidad de Servicio, en donde los flujos[2] están restringidos a seguir rutas específicas, donde existe un proceso de admisión de flujos, el cual es distribuido por ruta, y en donde el tráfico es sensible a retrasos (como el tráfico MPEG4[3] –Moving Picture Experts Group) surge el problema de la interferencia entre rutas en la red, el cual tiene poca investigación hasta el momento, en la literatura especializada.

Esta Tesis propone un método llamado Método STP (nombre proveniente de *Short Term Protection*- Protección de Corto Plazo), de atención de tráfico, para aplicarse en este tipo de red, observando una sola clase de flujos. En cada enlace congestionado (en una red así) en donde se interfieren dos o más rutas, en el corto plazo el Método STP protege la porción de ancho de banda de cada ruta contra el incremento de tráfico en rutas de interferencia. En el largo plazo este método asegura a cada ruta el tener un ancho de banda proporcional a su demanda relativa promedio, medida, de ancho de banda. Los nodos actúan de manera autónoma, sin administración central. Se pretende que el Método STP: 1) Ayude a los administradores de la red[4] a tener confianza, dentro de una ventana de tiempo, acerca de la cantidad de ancho de banda con la que cuentan para cada ruta; 2) Permita una administración simple de la red, con prospecto de ser escalable hacia al menos decenas de nodos.

El resultado de esta Tesis es que el Método STP se comporta según lo esperado en cuanto a su protección de corto plazo a las rutas, y que en este aspecto el mismo es favorable en comparación con los otros métodos, existentes, evaluados en esta Tesis.

En conclusión, el Método STP protege efectivamente, en el corto plazo, a las rutas más largas (más vulnerables) contra el aumento de tráfico de las rutas más cortas, en términos del retraso observado en los flujos, estableciendo una forma de reservación de corto plazo del ancho de banda para las rutas. En el largo plazo, este método permite la competencia libre de

[1] Para simplicidad de denominación, bajo el nombre “redes de datos”, o simplemente “redes”, se hace referencia a las redes de paquetes IP, donde IP significa “Internet Protocol”. Con el nombre de “paquetes” se hace referencia a los paquetes IP.
Ver subcapítulo 2.1.1 que da el significado de red. Básicamente una red está compuesta de nodos y enlaces (aristas).

[2] En esta Tesis, un flujo es una secuencia de paquetes relacionados que entran en una red a través de un mismo nodo de entrada, y dejan la red a través del mismo nodo de salida.

[3] En la página 187 de esta Tesis se incluye un Índice de Acrónimos

[4] En esta Tesis, la palabra “administrador” se refiere a una entidad que administra, la cual puede integrarse con herramientas de automatización, con el uso de computadoras digitales, pero que finalmente opera bajo la vigilancia y dirección de una o más personas. Una administración centralizada conoce lo que sucede en cada punto de la red.

las rutas por el ancho de banda. Este método puede escalar para operar en redes de al menos decenas de nodos.

Por otro lado, el método convencional de atención de tráfico (que mezcla el tráfico proveniente de las rutas interferentes en una sola cola para pasar por el enlace), puede ser atractivo en su aplicación para redes que ofrecen Calidad de Servicio, debido a su simplicidad. Para esto, es necesario el poder limitar el uso de cada enlace de la red (en un rango tan bajo como del 50% al 75%), aun cuando esto implique su subutilización.

Esta Tesis presenta: 1- La operación general y detallada del Método STP; 2- La evaluación del Método STP, por medio de experimentos realizados mediante simulación[5] por computadora[6], en el escenario de trabajo de esta Tesis; 3- La evaluación de métodos alternativos, existentes, con motivos de comparación; 4- Un diálogo sobre un posible escenario de aplicación; 5- Conclusiones y posibles direcciones para investigación futura.

Palabras Clave: Calidad de Servicio (QoS –Quality of Service); Compartición de Ancho de Banda; Control Distribuido de Tráfico.

Abstract

In fixed networks which offer Quality of Service, where flows are constrained to follow specific routes, where there exist an admission-process for flows, which is distributed per-route, and where the traffic is sensitive to delays (like MPEG4 traffic), the problem of *interference between routes* arises. This problem has received little research attention, so far, in the literature.

This thesis proposes a method called the Short Term Protection (STP) Method, for traffic attention, applied in this type of network, observing a single class of flows. In each congested link (in such a network) where two or more routes interfere, in the short term the STP Method preserves the bandwidth-share of each route against the increment of traffic on interfering routes. In the long term, this method ensures that every route gets a bandwidth proportional to its relative, average, measured demand of bandwidth. The nodes act autonomously, without central administration. The STP Method is expected: 1) To help network administrators to have confidence, within a time-window, about the amount of bandwidth available for

[5] En esta tesis se sobreentiende que una simulación es una simulación por computadora.
[6] En esta Tesis la realización de un experimento se refiere a la realización de varias simulaciones por computadora, con las mismas condiciones iniciales, en el escenario de trabajo de la Tesis, pero donde los tráficos generados, al tener características aleatorias, producen diferentes resultados en cada simulación. Al promediar los resultados de las simulaciones se obtiene el resultado del experimento.

every route; 2) To allow having a simple bandwidth management in the network, with prospect to be scalable to at least tenths of nodes.

The result of this thesis is that the STP method behaves as expected in terms of short-term protection of the routes, and in this aspect, it is favorable compared to other existing methods evaluated in this thesis.

In conclusion, the STP Method effectively protects, in the short term, the longer routes (which are more vulnerable) against the increase of traffic on the shorter routes , in terms of the delay of the flows, establishing a form of short-term bandwidth- reservation for the routes . In the long-term, this method allows free competition for bandwidth between routes. This method can scale to operate in networks of at least tens of nodes.

On the other hand, the conventional method for traffic attention (which mixes traffic from interfering routes in a single queue to go through the link) may be attractive in its application to networks offering Quality of Service, due to its simplicity. For this, it is necessary to be able to observe and limit the use of every link of the network (in a range as low as 50% to 75 %), even if it means its underutilization.

This thesis presents: 1) The general and detailed operation of the STP Method; 2) The evaluation of the STP Method, through computer simulations, using the work scenario or this thesis; 3) The evaluation of alternative, existing, methods for comparison purposes; 4) A discussion of a possible scenario of application; 5) Conclusions and possible directions for further research.

Keywords: Quality of Service (QoS); Bandwidth Sharing; Distributed Traffic Control.

Capítulo 1. Introducción

Introducción

Los administradores de las redes buscan optimizar el uso del ancho de banda de los enlaces en las redes para aprovechar al máximo los recursos disponibles y evitar congestionamientos. Esta optimización se llama "Ingeniería de Tráfico" [1]. El enfoque más extendido[7] para el estudio de este tipo de optimización ha tratado sobre las redes en donde se restringe a los flujos a seguir rutas determinadas. En este tipo de redes, una forma de crear rutas es con uso de una tecnología llamada MPLS (Multiprotocol Label Switching Architecture) [2].

Tradicionalmente en las redes no se ofrecen garantías a los flujos, y esto incluye a las redes en donde se realiza Ingeniería de Tráfico. En estas redes el servicio que se ofrece a los flujos se llama Best Effort [3] (mejor esfuerzo). La operación de Best Effort implica que cualquier flujo no requiere solicitar permisos para poder enviar (cursar) sus flujos por dicha red.

Las redes han permitido a los programas informáticos (también llamados aplicaciones) el intercambio de datos, con lo que se pueden tener diversos servicios, como el intercambio de archivos de datos, o de imágenes.

A un servicio se le llama *inelástico* cuando el mismo requiere operar con al menos un mínimo de recursos y/o un mínimo nivel de desempeño, para poder ser satisfactorio (o útil); por ejemplo, un servicio que requiere operar con un ancho de banda de al menos un mínimo dado, y con un retraso no mayor a un máximo dado. Se dice que las aplicaciones que ofrecen servicios inelásticos son sensibles, por ejemplo, a pérdidas, o a retrasos.

Por el contrario, un servicio se considera *elástico* cuando éste puede tolerar el operar con márgenes más amplios en recursos o desempeño (sin necesariamente estar estos márgenes bien determinados). Un ejemplo de una aplicación elástica es aquella con la cual se envían archivos de datos, en donde es satisfactorio que los archivos lleguen completos, a su destino, ya sea en 5 segundos o en 5 minutos. Las primeras aplicaciones que aprovecharon a las redes para el intercambio de datos lo hicieron con servicios, más bien elásticos, como el intercambio de archivos de datos.

Desde hace aproximadamente 10 años existen aplicaciones que son sensibles a retrasos, a pérdidas, o a Jitter (variaciones en los retrasos), que se están utilizando más y más en las redes. El poder otorgar servicios de transporte a los flujos de estas aplicaciones, de forma que se cumplan las expectativas mínimas relativas a las características de sensibilidad citadas, es lo que se conoce como "otorgar servicios con calidad". Cuando una red ofrece este tipo de servicios se le llama red con "Calidad de Servicio" (que por facilidad se designa como QoS – del inglés Quality of Service [4] [5]). En toda red que ofrece QoS se requiere que se otorguen

[7] Otro enfoque es la Ingeniería de Tráfico basada en IP [35] [36] [37] [38].

"Permisos de admisión" a los flujos que cursan por la misma. Para otorgar admisión a un flujo se predice si la QoS que se brindaría en la red seguiría siendo satisfactoria después de la admisión, desde luego, las características del flujo a admitir[8].

Dado que la característica de QoS lleva a complicar la operación de las redes, esta característica se usa en las redes privadas. Esta característica no se utiliza en una red tan grande como la Internet, que ha tenido éxito gracias a la sencillez del concepto Best Effort, el cual no provee garantía alguna.

Cuando una red dispone de recursos sobrados con relación a la cantidad de tráfico que maneja, la misma puede ofrecer servicios aceptables para los flujos. Esto sucede actualmente cuando se cursan flujos sensibles a retrasos, por la Internet. Pero a veces este servicio falla pues la Internet no siempre tiene las condiciones de tráfico para ofrecer los servicios que los flujos requieren. Las redes tienden a crecer en capacidad, y por las mismas se tiende a demandar que más flujos cursen por ellas. Esta es una carrera de oferta contra demanda en la que no es claro si con tan sólo el disponer de exceso de recursos sea suficiente, y por cuánto tiempo, para otorgar un servicio de calidad a los flujos. Por lo anterior, la QoS se continúa estudiando, proponiendo alternativas de solución [6] [7] [8] [9] [10] [11] [12] [13]. Con relación a las metas de diseño, y los objetivos de las redes, el Sector de Telecomunicaciones y Estandarización de la ITU (ITU-T) [14] indica que las redes del futuro deberán poder proveer servicios de diversas características, como ancho de banda y QoS, y que un reto de las redes futuras es mejorar la QoS a un costo más bajo. Con relación a la QoS en redes de datos, existen actualmente otras tendencias para redes metropolitanas, en donde la QoS se basa en la forma de configurar los nodos para formar redes virtuales[9].

El administrador de una red tiene interés en maximizar las utilidades "económicas" que obtiene por la prestación de servicios de transporte de flujos. Un ingrediente para maximizar estas utilidades puede ser el utilizar lo más posible los recursos de la red. Otro ingrediente

[8] Ver explicación en subcapítulo 2.1.5.

[9] En [32] se presenta un algoritmo para ayudar a los operadores (personas) en las redes Metro Ethernet a mejorar la eficacia de su trabajo en la configuración de los conmutadores (Switches) Ethernet, para la mejor utilización del ancho de banda de las redes y el buen cumplimiento de los contratos de servicio (Service Leve Agreeements –SLA) con los clientes de las redes. En estas redes un Switch suele operar usando redes virtuales (VLAN). El protocolo tradicional de Ethernet con el que se definen los caminos de intercomunicación entre los Switches es el protocolo de Árbol de expansión (Spanning Tree Protocol –STP), el cual tiene problemas de falta de rapidez de convergencia y de ineficiencia en el aprovechamiento del ancho de banda de los enlaces. Para mejorar el mencionado problema de la ineficiencia, en lugar de utilizar una sola instancia de STP se utilizan múltiples instancias de ese algoritmo, es decir STP Múltiple –MSTP, cada instancia asociada a un grupo diferente de VLANs. Dado que a la fecha MSTP no tiene métodos genéricos que indiquen cómo hacer dicha asociación, los operadores (personas) ejecutan diversos ejercicios de prueba y error para la asociación manual. En [32] se presenta un algoritmo para automatizar esta asociación, buscando buenos resultados en utilización de ancho de banda y el buen cumplimiento de los contratos de servicio (Service Leve Agreeements –SLA) con los clientes de las redes.

puede ser cobrar caro (económicamente) por los servicios otorgados de la red. También se puede hacer una combinación de ambos ingredientes. Para cobrar por los servicios de la red se debe cumplir con los compromisos de otorgamiento de QoS contraídos con los dueños de los flujos. Los ingredientes anteriores pueden ser contrarios, uno del otro. Esto sucede porque cuando hay muchos flujos en la red se emplean más recursos de la misma, y se tiende a deteriorar la QoS otorgada a los flujos. Así que, ¿cómo se cobraría caro por un mal servicio?

En las redes en donde hay rutas predefinidas para el curso de los flujos, existe la llamada interferencia entre rutas[10], cuando dos o más rutas comparten enlaces y por lo mismo deben compartir el ancho de banda de los mismos. Si no hay una administración centralizada en la red, en la que se sabe (o se intenta saber) lo que pasa en toda la red, no se puede saber en qué cantidad una ruta, al admitir tráfico, afectaría a las rutas que interfiere.

Esta tesis precisamente propone un método de atención de tráfico, el Método STP, que mitiga el problema de la interferencia entre rutas de redes que no tienen una administración centralizada. Este método no permite que una ruta que admite tráfico cause efectos adversos en las rutas que interfiere, sino paulatinamente. La operación de este método se da en cada nodo de la red.

La interferencia entre rutas no es un problema si los enlaces de la red mantienen poca utilización (50% o menos[11]), aun después de la admisión, pero en utilizaciones mayores, como 75% o más, la interferencia causa efectos importantes. Esta es la razón por la cual en este tesis como en otros trabajos que tratan sobre aspectos de QoS en redes, se usan escenarios de trabajo con redes con alta utilización (redes en estrés).

Motivación

El autor de esta Tesis no ha encontrado estudios, en la literatura, que aborden la interferencia entre rutas, sino en dos casos. En un caso, esta situación se menciona, y en otro se aborda de manera diferente a como se hace en esta Tesis (como se explica en los siguientes párrafos en esta sección). Se puede especular que la atención a esta interferencia la pueden haber atendido en soluciones o productos existentes privados, cuyos detalles no se exponen al público, o que la comunidad de investigación considere que con el uso apropiado de los métodos de atención de tráfico existentes, se puedan limitar los efectos de la interferencia; sin embargo, el autor de esta Tesis no ha encontrado afirmaciones en torno a esto, en la literatura.

En los trabajos de investigación que tratan de redes con administración no centralizada, no se indica, hasta donde el autor de esta Tesis haya conocido, para qué tamaño de las redes se

[10] Ver subcapítulo 2.1.9.
[11] Ver subcapítulo 6.1.2.

puede, o no, lograr tener una administración centralizada que opere satisfactoriamente; es decir, dichos trabajos parten de que sus redes de estudio operan bajo una condición descentralizada. En algunos de estos trabajos, [15] [16], se estudian redes que tienen rutas predefinidas, con admisión "distribuida por ruta", es decir, cada ruta tiene un administrador que permite o no la admisión, donde este administrador solamente observa el retraso que hay en el curso de los paquetes de los flujos al cursar por *su* ruta[12] (en esta Tesis a este retraso por simplicidad se le refiere como "retraso en la ruta"). Estos trabajos no estudian el problema que se causa en las rutas por la interferencia de las mismas (aunque en [16] se menciona el problema) sino sólo estudian los algoritmos de admisión. La interferencia entre rutas se da porque dos o más rutas comparten nodos intermedios en la red[13]. El reto de estos trabajos es llevar la admisión al extremo tal que se utilice lo más posible la red, cumpliendo la QoS en la misma, para aprovechar al máximo los recursos de la red.

En otros trabajos, como en [17], se estudian redes donde se confina a los flujos a cursar por rutas y donde se observan cruces de rutas en sus diagramas, pero donde no se estudia el problema de dicho cruce, sino el objeto de estudio es buscar una optimización en la utilización de la red.

Sí hay trabajos que han abordado el tema de la interferencia entre rutas en redes. Por ejemplo, la propuesta del algoritmo MIRA (Minimum Interference Routing) en [18] [19] [20], para limitar el deterioro debido a dicha interferencia, con base en la observación del deterioro del ancho de banda de los enlaces al admitir nuevo tráfico, asociando a cada enlace una forma de precedencia para su uso (un "peso") en la selección de rutas para cursar "nuevo" tráfico (tráfico recientemente admitido) por la red, y con una operación basada en la centralización de los datos observados. En estos trabajos se propone que se puede establecer (sin decir cómo) una relación entre el ancho de banda utilizado en los enlaces de la red, y el retraso de los flujos al cursar por la red. En 2012 se reporta [21] que dicho algoritmo es el "más popular".

Según el conocimiento del autor de esta Tesis, no hay trabajos de investigación en donde se estudie el problema de la interferencia entre rutas en redes con admisión descentralizada por ruta, ni su impacto con relación al retraso "extremo a extremo" en el curso de los flujos por su ruta.

---

[12] En esta tesis, el retraso en una ruta se refiere al tiempo de los paquetes en cursar de extremo a extremo en una ruta, desde que el paquete ha llegado completamente a la ruta, hasta que el paquete ha salido completamente de la misma. Este retraso depende de las velocidades de transmisión en los enlaces de la ruta, y del tiempo de espera en las colas formadas en las interfaces de salida de los nodos a lo largo de la ruta. Solamente se considera encolamiento en un nodo en la interfaz de salida [33].

[13] Comparten aristas.

La ITU-T[14], en [14] indica que debido a las limitaciones en la operación administrativa de las redes actuales, se desarrolla un enfoque de administración descentralizado que utiliza auto-organización, autonomía y autonomicidad[15], como sus conceptos fundamentales. El enfoque de esta Tesis está en acuerdo con el interés de la descentralización.

En esta Tesis se evalúa el impacto en las redes debido al problema de interferencia indicado, con el uso del método convencional (o tradicional) de atención de tráfico en los nodos de la red, en el cual se utiliza una sola cola por cada interfaz de salida de los nodos (método de atención referido en esta Tesis como Método q1[16]). El parámetro de QoS que se observa es el retraso en las rutas[17].

Asimismo, en esta Tesis se evalúa el impacto del problema de interferencia indicado, cuando en las redes se utilizan otros dos métodos alternativos de atención de tráfico: en el primero, cada ruta tiene una reservación fija de ancho de banda en los nodos en donde se tiene la interferencia (método de atención referido en esta Tesis como Método q3f[18]), y se puede considerar que es un método tradicional para reservación de ancho de banda; en el segundo se utiliza un método propuesto en [22], llamado Método del Servicio de Despacho de Multi-salto Coordinado (*Coordinated Multi-hop Scheduling Service*), aplicado a su operación en una sola clase para proteger a rutas.

Finalmente, se propone un método de solución novedoso, denominado Método STP[19], para enfrentar el problema planteado, y se evalúan y comparan sus resultados contra aquellos de los otros métodos evaluados.

En esta Tesis también se evalúa si el Método q1 ofrece protección contra interferencia entre rutas en una red.

Cada enlace de una red cuenta con una capacidad para el envío de tráfico, a la que se le refiere como “ancho de banda”. Cuando un enlace se comparte por varias rutas, cada ruta utiliza una parte del ancho de banda del enlace. La asignación de ancho de banda, ya sea a usuarios o a rutas, se denomina “aprovisionar” ancho de banda. El Método STP garantiza una porción del ancho de banda de cada uno de los enlaces compartidos en una red, a cada ruta

[14] Sector de Estandarización en Telecomunicaciones de la Unión Internacional de Telecomunicaciones.
[15] Capacidad de hacer las cosas sin ocupar la mente con los detalles de bajo nivel necesarios, lo que le permite convertirse en un patrón de respuesta automática. Por lo general es el resultado del aprendizaje con la repetición.
[16] Las siglas q1 se refieren a 1 cola.
[17] Ver pie de página 12.
[18] Las siglas q3f se refieren a 3 colas con ancho de banda fijo (dado que en esta Tesis se aplica a la interferencia de 3 rutas).
[19] STP se acuñó del término, en inglés, Short-Term Protection, que está en inglés porque el método se presentó en dos artículos del autor de esta Tesis, escritos en inglés: [96] [80].

que lo comparte. Esta garantía es temporal, es decir, la cantidad garantizada de ancho de banda va cambiando con el tiempo, en favor de las rutas que mayormente demanden ancho de banda. La razón de cambio con el tiempo obedece a un algoritmo del Método STP. Con esto, el administrador de cada ruta tiene un conocimiento del ancho de banda con el que cuenta, al menos en una ventana de tiempo.

La hipótesis general de esta Tesis es que los administradores de las rutas tienen interés en poder predecir el ancho de banda con el que cuentan, para poder tomar decisiones con relación al tráfico que vaya a cursar por las rutas.

El Método STP apoya el interés de esta hipótesis pues limita la razón de cambio de ancho de banda de las rutas de la red en donde se aplica. Con el interés de preservar la QoS otorgada en las rutas, los administradores de las mismas limitan la magnitud de los cambios en el tráfico en las rutas de la red.

Trabajos como [23] presentan situaciones en donde las redes aprovisionan ancho de banda en la cantidad justa, para atender al tráfico que tienen. Se presenta el problema de poder predecir el tráfico, para aprovisionar con tiempo el ancho de banda, de tal forma que el ancho de banda con el que se cuente sea el adecuado.

La característica del Método STP de aprovisionar, de forma paulatina, ancho de banda a la ruta que más demanda, en un cruce de interferencia entre rutas, a conocimiento del autor, anteriormente no se había explorado.

Objetivo General de esta Tesis

Proponer y evaluar un método que mitigue el problema de la interferencia entre rutas, en una red fija que ofrece QoS, con admisión distribuida por ruta, donde la QoS toma en cuenta los retrasos en los flujos para cursar las rutas. La evaluación del Método STP sería en comparación con otros métodos alternativos, incluyendo el Método q1.

Se busca que el método sea sencillo de aplicar y administrar. Se pretende que el método permita la competencia entre rutas, por el ancho de banda en los enlaces de la red, pero que opere de tal forma que a corto plazo se proteja a las rutas más vulnerables, como aquellas que son más largas (en términos de nodos cruzados), contra las rutas cruzadas menos vulnerables, más cortas, cuando estas últimas aumentan su tráfico. Se pretende que a largo plazo las rutas que más demandan recursos los vayan adquiriendo poco a poco, de tal forma que las rutas más vulnerables puedan tener tiempo para tomar medidas, como liberar tráfico, cuando detecten que su QoS se va perdiendo. A largo plazo, se pretende que la red otorgue mayores recursos a las rutas que mayores recursos demanden. Esta operación de largo plazo es la operación que se asume tener presente en una red tradicional, donde hay una cola por

cada interfaz de salida. En ésta, la ruta que demanda mayores recursos, de manera natural tiene mayores recursos de las colas únicas, en las interfaces de salida.

Se concibe al método para operar de forma autónoma en cada nodo, sin administración centralizada, y se pretende que el mismo ayude a los administradores de cada ruta en la red a tener confianza, dentro de una ventana de tiempo, acerca de la cantidad de ancho de banda con la que cuentan, y permita una administración simple de la red, con prospecto de ser escalable hacia al menos decenas de nodos.

Objetivos Particulares de esta Tesis

Proponer un escenario de trabajo de la Tesis, para evaluaciones mediante experimentación por simulación por computadora[20].

Encontrar los niveles de tráfico en donde se pueda encontrar que se manifiesta “de manera importante” el problema planteado. Con esto, proponer las características de tráfico con que se han de hacer las evaluaciones, así como la metodología de prueba.

Preparar y adecuar el simulador a usar (el simulador ns-2[21]) para que pueda operar con el nuevo método planteado, y con el Método q1 y otros métodos alternativos, a evaluar.

Evaluar el desempeño del Método q1, en el escenario de trabajo de esta Tesis. Evaluar otros dos métodos alternativos de atención de tráfico, todos en términos del retraso de los flujos.

Proponer el método con las características del objetivo general (el Método STP). Se incluye la propuesta del algoritmo de cómo deben cambiar las proporciones de atención a las colas con el tiempo, según las condiciones de tráfico (las proporciones se denotan con valores reales positivos llamados pesos).

Evaluar el cumplimiento del comportamiento del Método STP, y el desempeño de la aplicación del método en el escenario de trabajo de esta Tesis, en términos del retraso en las rutas, como estrategia eficaz para resolver el problema de la interferencia entre rutas, con las características operativas con las cuales se concibió, observando el nivel de operación de tráfico en donde resulta benéfico, así como sus limitaciones.

Comparar los resultados de los métodos evaluados, con lo cual se puedan observar cuantitativamente las ventajas entre uno y otro, así como sus limitaciones.

---

[20] Como se indica en el pie de página 5, la palabra evaluación significa evaluación por computadora.
[21] ns: Network Simulator.

Establecer las principales conclusiones y contribuciones del trabajo desarrollado.

Objetivos del Método STP

El objetivo del Método STP (por sus siglas en inglés Short-Term Protection) se refiere a la situación en que dos o más rutas de tráfico en una red convergen al llegar a un nodo y las mismas se juntan a la salida del nodo (en alguna de las interfaces de salida del nodo), pudiendo volver a separarse en nodos posteriores, hacia su destino. Esta situación se conoce como interferencia, intersección o cruce de rutas en el nodo indicado, donde los flujos de las rutas compiten por el ancho de banda de la interfaz de salida del nodo (o bien por el ancho de banda del enlace de salida asociado).

Un objetivo del Método STP es que cuando el flujo de una ruta aumente su tráfico promedio, y que por lo mismo el flujo resienta un aumento de retraso promedio, este impacto "se reduzca" en el "corto plazo", en los flujos de otras rutas que se interfieren en el nodo. Se espera que esta reducción de impacto permita a los flujos afectados tener un tiempo mayor para resentir el impacto completo del retraso que se tiene con el uso del Método q1. El largo plazo para este método, en esta Tesis, es del orden de 16 minutos, el cual es un tiempo que podría ser diferente, dependiendo de los valores de los parámetros del método. El autor de esta Tesis seleccionó este valor por tres razones: 1- Por considerarlo suficientemente grande para poder observar la reacción del método; 2- Porque es similar al de otros trabajos relacionados.[22].

Otro objetivo del Método STP es que en el "largo plazo" los tiempos de espera promedio del tráfico de dos o más rutas que se interfieren en un nodo tiendan a igualarse, al medirse la espera adentro del nodo. Se persigue este objetivo por la comparación con el Método q1, donde el tiempo de espera promedio para los flujos de una y otra ruta es el mismo, adentro del nodo.

El Método STP opera como si hubiese una reservación de corto plazo para las rutas afectadas, con el fin de darles tiempo para tomar medidas cuando alguna ruta interferente aumente su tráfico en promedio. Las medidas serían disminuir el tráfico. Se prevé que en una red con un tráfico equilibrado entre todas las rutas, aquellas que podrían aumentar más fácilmente su tráfico serían las más cortas (en términos de nodos cruzados), y las más afectadas serían las rutas largas.

[22] Por ejemplo en [13] se simulan 10 minutos de operación; en [65] 8.33 minutos; [64] de 16 minutos. En [17] donde se requieren observaciones de "larga duración" los experimentos duran 5.33 horas, y en [60], que se compara con [17], los experimentos duran 50 minutos.

Metodología de Trabajo en esta Tesis

La metodología utilizada en esta Tesis consiste en plantear los antecedentes relativos al tema de investigación, incluyendo los trabajos de investigación relacionados, para presentar el problema de investigación y justificar su importancia.

Con base en lo anterior, evaluar el impacto del problema, con el uso del Método q1 y con el uso de otros métodos alternativos de atención de tráfico en los nodos. Posteriormente, proponer una forma nueva de solución (que resulta en el Método STP), y evaluar la validez de sus hipótesis de operación, y su desempeño, en el escenario de trabajo de esta Tesis. Se utiliza la simulación como herramienta de evaluación.

Comparar los resultados, obtener conclusiones, evaluar y plantear los logros obtenidos, y plantear las posibles rutas de investigación futura.

El retraso manejado en esta Tesis es el percentil del 98 del retraso medido, es decir, éste es el retraso de los paquetes que se encuentran en el límite del 2% de paquetes más retrasados al cursar por la ruta. Por la naturaleza de la obtención de este retraso, durante cada simulación no se van obteniendo resultados de retraso, sino que se espera a que concluya la misma para obtener un archivo de traza y de ahí obtener los resultados. Por lo anterior, lo que procede en cada simulación es iniciar la misma con una cantidad de tráfico prestablecida en cada una de las rutas de la red, y a partir de un punto casi intermedio en el tiempo de la simulación, se aumenta el tráfico en una o más rutas de la red, para observar los efectos de este aumento, en términos de los retrasos medidos.

Las condiciones de cada simulación se fijan desde el principio de las mismas. Estas condiciones son parte del escenario de trabajo. Por ejemplo, condiciones prefijadas de tráfico son la cantidad de tráfico que hay al principio de la simulación y en qué momento del tiempo de simulación cambia el tráfico y en cuánto.

Para las evaluaciones también se plantea un indicador de ganancia que suma 1 *punto* por cada paquete que cursa su ruta por debajo de un tiempo establecido, o penaliza con 10 *puntos* si no lo hace. Este tiempo se asigna igual para todas las rutas en el escenario de trabajo.

Acotamientos de esta Tesis

Se estudia tanto el Método STP como otros métodos alternativos de atención de tráfico, incluyendo el método q1 (de 1 cola por cada interfaz de salida), para observar la problemática de la interferencia entre rutas en los métodos, y hacer comparaciones, en términos del retraso medido. Las evaluaciones se realizan mediante experimentación por simulación.

Se utiliza un escenario de trabajo que tiene una red donde los flujos están restringidos a cursar por rutas, iniciando en nodos fuente y llegando a nodos destino, afuera de la red. Dicha red tiene una ruta larga y seis rutas cortas que interfieren a la primera. Las rutas cortas se interfieren, a su vez, por pares. El tipo de tráfico utilizado en los experimentos es generado por fuentes con características de tráfico MPEG, sensible al retraso. Los enlaces entre los nodos interiores de la red tienen 30 *Mb*/*s* (estos son los enlaces centrales), y los enlaces desde los nodos fuente de tráfico hacia la red, o desde la red hacia los nodos destino tienen 100 *Mb/s*.

En las situaciones de operación evaluadas, en la red del escenario de trabajo se inician las evaluaciones con una utilización del 74.5% del ancho de banda de los enlaces centrales de la red, con las rutas teniendo igual cantidad de tráfico[23]. Con esta utilización se observan retrasos entre 10 y 15 *ms* en las diversas rutas de la red (el retraso se observa al percentil del 98% del retraso más alto). Con esto, se experimentan situaciones donde se mide el tráfico y el retraso en la red, durante 26 minutos (1560 *s*). En la red, hay un nodo central en donde se enfocan las mediciones de retrasos por nodo. Para algunas evaluaciones se aumenta el tráfico hasta una utilización del 95.22% del ancho de banda en el enlace de salida de dicho nodo.

En cada experimento se parte de la utilización indicada de enlaces, y a los 600 *s* del tiempo experimental, una de las rutas "cortas", aumenta su tráfico. Con esto se observa el impacto en el retraso en las rutas de la red, especialmente en la ruta larga. El aumento de tráfico que se maneja en la mayor parte de los experimentos en el escenario de trabajo representa hasta el 11.1% del tráfico total inicial, lo que causa un aumento del retraso observado de hasta 25 *ms* en una de las rutas, y durante un tiempo experimental de 16 minutos posteriores al aumento de tráfico se observa cómo ese aumento va impactando en los retrasos de las rutas. Ya con este aumento se tiene una utilización del 82.8%[24] del enlace de salida del nodo referido. Se pretende que con este aumento de tráfico los retrasos de las rutas estén cerca del límite de retraso admitido, que es el punto de trabajo en donde se pueden observar las ventajas y desventajas de los métodos de atención de tráfico para protección por interferencia entre rutas.

La razón de la "alta" utilización indicada es que, cuando se tienen utilizaciones menores, los retrasos son "pequeños" y no hay poco impacto en los retrasos de las rutas que se interfieren, cuando una de ellas aumenta su tráfico. Por ejemplo, cuando la utilización de los enlaces sube de 37.26% a 57.96% debido al aumento de tráfico en una o más de ellas, el retraso en la ruta

[23] En [61] también se realizan evaluaciónes por simulación con un escenario de trabajo que tiene una red con rutas, y usando igual cantidad de tráfico en las mismas, observándose que cuando se utiliza cerca del 80% del ancho de banda del enlace compartido por dichas rutas se empieza a notar retraso significativo en las mismas.
[24] Considerando los datos que se tienen en el Capítulo Capítulo 4. Se obtiene de realizar (16+16+16+4)*0.621/30 = 0.828.

larga sube de 3 a 5 *ms*[25], aproximadamente, aun cuando hay reservación de ancho de banda en las rutas.

También se evalúa el impacto en el retraso de la ruta larga de la red, cuando dos o hasta tres rutas cortas que le interfieren aumentan su tráfico, y este aumento es simultáneo.

Para cuantificar el desempeño de los diversos métodos evaluados, se añade a las evaluaciones un límite único máximo de retraso para todas las rutas, que es de 19 *ms*, con el que se calcula una ganancia.

El valor de 19 *ms* se utiliza porque en la validación experimental del método q1, en el escenario de trabajo de esta Tesis, al aumentar el tráfico inicial en un 11.1% se llega a 20 *ms* de retraso en la ruta más larga. Los pasos de aumento de tráfico que se usan en las rutas del escenario de trabajo son "pequeñas", esto es, de 5.55% y hasta 11.1%[26]. Se podría usar un valor límite más alto, como 25 *ms*, en esta Tesis, pero eso implicaría manejar valores mayores de tráfico.

Este límite de retraso puede compararse con el límite de retraso del tráfico de video codificado con MPEG4[27]. Con MPEG4, el orden de transmisión es diferente al orden de despliegue de los cuadros de video comprimidos. Con las características del escenario de trabajo de esta Tesis, tan solo por la disparidad de orden indicada, se tendría un retraso, por espera en la decodificación, de al menos 100 *ms* entre un cuadro de video transmitido y recibido. A este retraso habría que agregar el tiempo de retraso del curso del tráfico en las rutas de la red del escenario de trabajo de esta Tesis, más el tiempo para que cada cuadro completo concluya su llegada al lado receptor (la media del tamaño de los cuadros es de 2,578 bytes – ver. Con estos tiempos el retraso de 100 *ms* se ve incrementado.

Según [24], [25], el tener 100 *ms* de retraso, de extremo a extremo para una aplicación interactiva, no causa incomodidad en los usuarios. Si la aplicación de video fuese de distribución en un sentido se podría manejar mayor retraso. El análisis cuidadoso de los tiempos, según las aplicaciones de video, queda fuera del alcance de esta Tesis.

---

[25] Ver subcapítulo 6.1.2.

[26] Esto sucede en el enlace de interferencia (donde hay interferencia entre rutas) de la red del escenario de trabajo (esta situación se puede dar en más de un enlace en la red), cuando una de las rutas cortas interferentes aumenta su tráfico en 4 fuentes MPEG4, llegando a una utilización promedio del ancho de banda de ese enlace de 82.8% (Se podrá ver en el capítulo que habla de las condiciones del escenario de trabajo (Capítulo 4), que esto se obtiene de calcular (12*3+4)*0.621/30= 82.8%.

[27] Para mayor información con relación a los datos aquí indicados sobre el tráfico MPEG4, el Apéndice L explica al respecto. El Apéndice K da detalles sobre la la incoportación del código para generar tráfico MPEG4 en el simulador ns-2.

Por último, no se evalúa, en esta Tesis, la diferencia o impacto que pueda haber con el uso del Método STP con relación al Método q1, para aprovechar mejor el ancho de banda de la red, partiendo de niveles "bajos" de utilización de la red.

Se considera que la admisión en las rutas funciona bien. No se implementa algoritmo de admisión alguna en las evaluaciones que se hacen en esta Tesis.

No se incorpora algoritmo de admisión alguno, ni en el escenario de trabajo ni en la operación del simulador. En el análisis de resultados de cada experimento se hacen consideraciones sobre qué hubiese decidido un algoritmo de admisión, en vista del efecto causado por el aumento de tráfico.

Sobre la QoS en IPv4 o IPv6[28]. En su concepción, el paquete IPv4 incluyó un campo llamado Tipo de Servicio (con siglas TOS por su significado en inglés: "Type of Service"), previsto para contener un código para su marcado, para otorgarle una precedencia y un correspondiente tratamiento especial en cada nodo de la red. En el desarrollo de los estudios de QoS se encontró que para ofrecer QoS en una red se requería, no solamente del tratamiento especial a los paquetes en cada nodo de la red, sino también de un conjunto de funciones coordinadas en la red, en los diversos elementos de la misma, y desde luego de reglas de operación (aspectos que reunidos componían el concepto llamado "Arquitectura"). Por lo anterior, la concepción original de uso de dicho campo quedó obsoleta para redefinirse, posteriormente, como "el campo DS", concebido para su uso en la Arquitectura de Servicios Diferenciados.

En un paquete IPv6 existe también un campo previsto para usarse en el otorgamiento de QoS, llamado "Traffic Class", que puede ser equivalente al campo TOS del paquete IPv4, y con el cual se puede también indicar una precedencia para su tratamiento, sin embargo, de igual manera que el caso del campo TOS de IPv4, para otorgar QoS se requiere de una arquitectura.

Por lo anterior, los trabajos de investigación sobre QoS en redes de datos proponen aspectos que pueden mejorar o completar las arquitecturas de QoS sin depender del uso de una u otra versión de IP. Esta tesis tampoco tiene dicha dependencia.

La implementación de las funciones para QoS en la práctica, con el uso de los paquetes IPv6, es similar a aquellas usadas para los paquetes IPv4 [26].

En esta tesis no se considera el tráfico de control de enrutamiento en las redes, porque en las redes de datos de enrutadores, típicamente es suficiente con considerar el uno por ciento para el tráfico de enrutamiento [27], y la carga de enrutamiento se considera insignificante

[28] IP. Del inglés Internet Protocol. El protocolo requerido en Internet para llevar los datos en forma de paquetes. Los postfijos v4 o v6 se refieren a la versión de IP: la versión 4, actualmente universalmente difundida en su uso en Internet, y la versión 6 que se abre camino actualmente.

para protocolos como RIP o EIGRP [28][29]. Asimismo, la red del escenario de trabajo no es una red de miles de enrutadores, como sucede en las redes exteriores con el uso del protocolo de enrutamiento BGP y tablas de enrutamiento, en los enrutadores, hasta con 400,000 entradas.

En autor de esta tesis hizo la cuenta del tráfico de control que habría, usando los protocolos RIP y OSPF para redes de 20 enrutadores, obteniendo que el porcentaje de tráfico se puede considerar no-significativo[30].

Se considera que la red del escenario de trabajo tiene enrutadores; y no Switches, por lo que no se tiene el uso del protocolo Spanning Tree [29] que tiene un tráfico más intenso por los paquetes "Keep Alive" de ese protocolo, que se usan para detectar lazos, y más aún cuando el protocolo se aplica a VLANs con una instancia operando en cada una de ellas [30] [31] [32].

Logros

Se planteó inicialmente proponer un método que protegiera a las rutas vulnerables en una red, contra las rutas menos vulnerables, con relación la situación de la interferencia entre rutas cuando existe admisión independiente, por ruta.

Se propone un método (el Método STP), el cual tiene cambios, a lo largo del avance del trabajo de investigación de esta Tesis, y con el cual, al final, se obtienen resultados positivos que concuerdan con los propósitos iniciales de esta Tesis.

Se aprecia que el Método STP tiene ventajas cuando se tiene una utilización grande (del 74.5% o más) de la red, y su ventaja no es apreciable en utilizaciones menores, contra el Método tradicional (Método q1). Se encuentra que el Método STP tiene también otras características de comportamiento que apoyan su ventaja de protección de la ruta larga.

En términos de un indicador llamado *ganancia*, manejado en esta tesis, se aprecia que la protección que ofrece el Método STP es a costa de algún deterioro, en menor medida, en

---

[29] En la práctica, cuando se clasifica el tráfico en enrutadores, se recomienda dejar el 25% de ancho de banda sin clasificar para el curso de tráfico Best Effort mezclado con tráfico de enrutamiento [101].

[30] Con el protocolo de enrutamiento RIP se causan mensajes de control no solicitados, enviados cada 30 *s* por cada nodo, para enviar su tabla "vector-distancia". En una red de *n* nodos, donde el número de *Bytes* por cada paquete donde se envíe dicha tabla es de (4 + 4x5x(*n*-1)) *Bytes* [102], si *n* = 20 se tendría un tráfico no mayor a 2.0 *Kb*/*s*.
Para el protocolo de enrutamiento OSPF para los paquetes Link State Update Packet, que son el alma de operación del protocolo, usados por cada enrutador para anunciar el estado de cada uno de sus enlaces, el paquete tendría 24 + 4 + 20 +4 + (*t* x 4) x *j Bytes*, donde *t* es el tipo de enlace a anunciar (teniendo los valores, en *Bytes* de 4 para el tipo 1, 1 para el tipo 2, 1 para el tipo 3, 1 para el tipo 4, y 3 para el tipo 5), y *j* es el número de enlaces a anunciar. Estos paquetes se colocan directamente en el paquete IP. Considerando que se enviara en promedio un paquete cada 30 *s* por enrutador, para una red de 20 enrutadores, donde cada enrutador tuviese 4 enlaces tipo 1 a anunciar, la cantidad de tráfico en la red por el control de enrutamiento no rebasaría 0.620 *Kb*/*s*.

otras rutas interferidas de menor longitud, con relación a las ganancias observadas con el uso del Método q1. Haciendo un balance de la protección a todas las rutas en general (un promedio) se observa que el desempeño de los métodos q1 y STP es similar, y sin embargo, el Método STP sigue teniendo el valor de poder proteger a la ruta larga, como no puede hacerlo el Método q1. Con este útlimo método se llega a observar una afectación a la ruta larga de hasta un 100%, en términos de ganancia, en los resultados de esta Tesis).

El desempeño del Método q1 resulta ser lo esperado, con excepción de que se encuentra que con poca utilización de la red (entre aproximadamente 37 a 58) el problema de la interferencia entre rutas es poco perceptible[31] con el uso de ese método.

Contribuciones

En esta Tesis se presenta un método de atención de tráfico (el método STP), en el escenario de trabajo de esta Tesis, que da apoyo para prevenir los efectos de la interferencia entre rutas, en una red, que se vuelven significativos cuando existen utilizaciones grandes (del 74.5% o más) de la capacidad de los enlaces de la red.

El método protege a las rutas más vulnerables de la red (como la ruta más larga) contra la interferencia causada por las rutas más cortas. En este sentido, el método STP supera al método tradicional de atención de tráfico (el Método q1), el cual pierde habilidad para proteger a la ruta larga, y más aún cuando más de una ruta interferente aumenta su tráfico. No hay, a conocimiento del autor de esta tesis, un método que tenga esta orientación.

Existe un método llamado Método CMS (Coordinated Multihop Scheduling[32]), que puede orientarse para una protección con el mismo objetivo que el Método STP. En esta tesis se realiza una evaluación de este último método, y se concluye que el mismo no es mejor que el Método STP para los objetivos planteados.

En esta tesis se evalúa también el Método q1 en el escenario de trabajo, y los resultados permiten argumentar que el Método q1 podría beneficiarse con la ayuda de una administración centralizada que tan solo observara la utilización de cada enlace de la red (esto queda fuera del alcance de esta Tesis).

[31] En el subcapítulo 6.1.2 se concluye que con una utilización inicial del 37.26% (6x3x0.621/30), y de ahí un aumento de tráfico en un 55.55% (10/18) para llegar al 57.96% de utilización ((6*3+10)*0.621/30), apenas da un aumento de retraso menor a 3 *ms* en las rutas.
[32] Que se presenta en [22].

Organización de esta Tesis

El Capítulo 1 contiene la introducción, motivación, objetivo general y los objetivos particulares de este trabajo, la metodología y los logros en este trabajo.

El Capítulo 2 contiene los antecedentes y definiciones del trabajo, el planteamiento del problema a resolver del trabajo,

El Capítulo 3 contiene el sumario de los trabajos relacionados.

El Capítulo 4 presenta las características y condiciones del escenario de trabajo para el estudio, mediante experimentación por simulación, incluyendo la presentación de la herramienta de simulación a utilizar.

El Capítulo 5 presenta el impacto del problema a resolver, mediante la presentación de los resultados experimentales al utilizar el Método q1 en el escenario de trabajo, en términos de los retrasos observados en los flujos.

El Capítulo 6 presenta resultados similares a los del capítulo anterior, pero cuando se aplican otros dos métodos, alternativos, para la atención del tráfico en el escenario de trabajo. Se exponen los motivos por los cuales estos dos métodos no se continúan evaluando en este trabajo.

El Capítulo 7 presenta el planteamiento del Método STP. Se presentan inicialmente algunos conceptos que permiten plantear las hipótesis de operación del método. Se presentan las características esperadas en la operación y resultados del método, y sus ventajas esperadas.

En el 0 se evalúa el Método STP, aplicado al escenario de trabajo de esta Tesis, mediante experimentación por simulación, obteniendo resultados en términos de los retrasos observados de los flujos, al cursar por las rutas de la red. Se comparan estos resultados con aquellos obtenidos en el Capítulo 5, por la aplicación del Método q1 en el mismo escenario de trabajo. Se presenta el indicador de ganancia para obtener resultados que representan los retrasos en las rutas, de forma cuantitativa. Se presentan las comparaciones en términos cuantitativos y se emiten conclusiones.

El Capítulo 9 presenta posibles escenarios de aplicación del Método STP.

El Capítulo 10 presenta las conclusiones del trabajo, las contribuciones y las posibles rutas de trabajo para investigación futura.

A continuación se incluyen las referencias, e índices de acrónimos y de términos.

El apéndice A contiene una lista de los trabajos emanados de esta investigación, y la copia de dos artículos publicados en revistas.

Los siguientes apéndices, del B al M, contienen información adicional de conceptos teóricos relacionados con el trabajo; del desarrollo de trabajo previo antes de llegar a la forma final del Método STP; de detalles del desarrollo de algoritmos y programación del simulador utilizado para las evaluaciones realizadas en este trabajo.

Capítulo 2. Antecedentes y Conceptos Fundamentales

Introducción al Capítulo

En este capítulo se hace una revisión de los antecedentes teóricos y definiciones utilizadas en las redes. Se revisan los diversos métodos para optimizar el uso del ancho de banda de los enlaces de las redes. Se presenta el concepto de QoS en las redes y su relación con la disponibilidad sobrada de recursos en las mismas. Se presentan dos de los métodos más sobresalientes para la implementación de QoS en redes. Se observa la relación de los métodos de QoS con los métodos de optimización en redes.

Ya con los antecedentes indicados se presentan las razones por las que existen redes en donde los flujos están restringidos a tomar rutas específicas, y en este ámbito se identifica el problema que se toma como el problema a resolver en este trabajo.

2.1 Antecedentes y Definiciones

2.1.1 Redes, Flujos y Recursos

Las redes están compuestas de nodos y enlaces. Las redes proveen el servicio de transporte de flujos.

Un flujo se suele definir como el tráfico que cursa una red, desde una fuente determinada hacia un destino determinado. El tráfico de la fuente entra normalmente por un mismo nodo de ingreso, y sale de la red por un mismo nodo de egreso.

En una red cada flujo es propiedad de algún usuario, o cliente, de la red. Se puede decir que dicho servicio se otorga a los flujos, o a los dueños de los flujos. También se dice que un flujo pertenece a alguna aplicación de un usuario. Cuando el dueño de un flujo intenta pasar su flujo por una red se dice, por facilidad, que el flujo intenta pasar por la red, yendo de un nodo de entrada a un nodo de salida, específicos, de la red.

Los recursos de una red están conformados por un conjunto de medios físicos, de servicios y de instalaciones. Los medios físicos se conforman por nodos[33] y enlaces. Cada nodo se conecta a un enlace mediante una de sus interfaces. De hecho, se puede considerar que la interfaz de conexión es parte del enlace. Uno de los recursos físicos que se asocia a cada enlace unidireccional es la cantidad de *bit*/*s* que se envía por el enlace (concepto que en el mundo de las redes se conoce como “ancho de banda del enlace”). Otro recurso físico del enlace es el espacio de almacenamiento de paquetes en la interfaz de salida del nodo transmisor [33].

[33] En el ámbito de las redes de paquetes IP, un enrutador y un nodo son equivalentes.

Todo flujo, al cursar por una red, utiliza recursos de la red. Los recursos de una red son limitados, por lo que mientras más flujos cursan por la misma se producen más complicaciones para poder otorgarles el servicio de transporte en la red.

2.1.2 Ingeniería de Tráfico

Los administradores de las redes, con el objetivo de maximizar su utilización, buscan evitar las sobre congestiones en los enlaces de las mismas, es decir, el administrador de una red trata de balancear el tráfico para evitar cuellos de botella, y poner el tráfico donde haya más disponibilidad de ancho de banda. Para esto se deben considerar las demandas de tráfico actuales, o inclusive las previstas, en la red. Este objetivo se puede reflejar en la minimización de una función de la utilización de los enlaces, sujeta a diversas condiciones de frontera, como no sobrepasar la capacidad de cada enlace. Por ejemplo, la función de utilización del ancho de banda de los enlaces puede ser la minimización del valor del uso relativo (tráfico / capacidad) máximo que pudiese encontrarse en algún enlace en la red. La herramienta para administrar el uso de los recursos de una red se llama "Ingeniería de Tráfico"[34].

La ingeniería de tráfico sirve para hacer transitar tráfico por donde el ancho de banda requerido esté disponible (buscando evitar tener enlaces que representen "cuellos de botella" por congestionamiento en los mismos)[35] [1]. Así, la naturaleza de la Ingeniería de Tráfico es efectivamente la optimización de enrutamiento para mejorar la capacidad de servicio de la red sin causar congestión en la misma. En [34] se dice (traducido): "La ingeniería de tráfico es una solución eficaz para el control de congestión en una red, y para optimizar el rendimiento de la misma. La ingeniería de tráfico abarca la aplicación de principios científicos y tecnológicos en la medición, el modelado, la caracterización, y el control del tráfico en la red. El propósito de la ingeniería de tráfico es el de facilitar la operación de la red, de una forma eficiente y confiable, y a la vez optimizando la utilización de los recursos de la red y su desempeño en el manejo de tráfico."

Fue en las redes en donde se establecen rutas para el curso de los flujos en donde se introdujo el concepto de Ingeniería de Tráfico (Traffic Engineering)[36] [1]. Posteriormente la Ingeniería de Tráfico se aplicaría en redes sin rutas establecidas. A esta vertiente se le llama Ingeniería de Tráfico basada en IP [35] [36] [37] [38].

---

[34] Definiciones más detalladas de Ingeniería de Tráfico se pueden encontrar en [98].

[35] Usando funciones de penalización de utilización que penalizan más a la mayor utilización (funciones crecientes y convexas hacia abajo).

[36] A dichas redes les denomina "dominios MPLS", donde las rutas se crean con el uso de la arquitectura MPLS (Multiprotocol Label Switching Architecture) [2], y a las rutas se les denomina LSP (Label Switched Paths).

La Ingeniería de Tráfico se puede clasificar [1] según si se aplica dentro de un sistema administrativo o entre sistemas administrativos. Un sistema administrativo se refiere a un conjunto contiguo de nodos donde hay un administrador que decide diversas tecnologías a utilizar relacionadas con la forma de envío de paquetes (concepto también conocido como enrutamiento de paquetes), además de políticas (reglas de operación). En esta Tesis se usa el término "Sistema Autónomo" [37] (o AS[38] por sus siglas en inglés), que es un término general para referirse a un sistema administrativo como el referido. Entonces, a la ingeniería de tráfico aplicada dentro de un sistema autónomo se le refiere como "Intra-AS", y a aquella entre sistemas autónomos se le refiere como "Inter-AS".

Asimismo se clasifica a la Ingeniería de Tráfico según sus estrategias de operación: "Fuera de Línea" o "En Línea". En el tipo "Fuera de Línea" se calculan las rutas a partir de una matriz de tráfico ofrecido obtenida con base en especificaciones de tráfico[39], o a partir de mediciones, y se aplican las nuevas rutas todas a un tiempo, a una red sin tráfico, buscando minimizar el uso de los recursos de los enlaces (se indica [1] que en la práctica el ciclo entre dos cálculos consecutivos de este tipo es semanal o mensual). Este tipo de estrategia no se adapta a cambios más dinámicos del tráfico. No se resuelve la atención al arribo de nuevo tráfico, conforme va llegando. Cuando se realiza un recálculo en el acomodo del tráfico, aun cuando se busca la optimización en el uso de los recursos de ancho de banda de los enlaces, no se observa la continuidad del cumplimiento en la QoS que se pudiese haber estado entregando. En [18] se indica que un algoritmo "fuera de línea" crea re enrutamientos frecuentes, lo que no va de acuerdo a la forma de operación real de las redes.

En el tipo "En Línea" a la llegada de un nuevo flujo se le acomoda "óptimamente" en una de las rutas ya establecidas o en una ruta nueva, con la intención de que se converja, sin intervención humana, hacia la maximización del acomodo del nuevo tráfico, y sin causar congestión. En [1] se indica que una situación importante en la Ingeniería de Tráfico Intra-AS y En-Línea es lo que denominan "interferencia" entre rutas (ver capítulo 2.1.9).

2.1.3 Servicio Best Effort

El tipo de servicio para el envío de paquetes, tradicionalmente otorgado en las redes, incluyendo la Internet, es el servicio llamado *Best Effort* [3] (mejor esfuerzo). La operación de *Best Effort* implica que cualquier usuario de una red que utilice *Best Effort* no requiere solicitar

[37] En el ámbito MPLS el sistema administrativo se llama "Dominio MPLS".
[38] AS:"Autonomous System".
[39] Estas especificaciones se plantean en un documento técnico llamado "Especificación a Nivel de Servicio" (o SLS que quiere decir Service Level Specification).

permisos para poder enviar sus flujos por dicha red. Estos permisos se llaman permisos "de admisión. Una red sin "control de admisión" es aquella que no impone permisos de admisión.

Hay que aclarar que muchas redes privadas filtran los flujos que dejan cursar, por razones administrativas o de seguridad, mediante dispositivos específicos, como los "*Firewalls*", pero esto no se debe confundir con que se requieran permisos de admisión.

Una de las razones fundamentales del éxito de la Internet se debe a la simplicidad de su tipo de servicio tradicional y general, *Best Effort*, que no considera el concepto de admisión. La implantación del control de admisión en las redes conlleva una sofisticación que crece conforme dichas redes crecen y se enlazan entre sí.

Cuando no se tiene control de admisión no es posible establecer una cota máxima de la cantidad de flujos que intenten cursar por la red. Dado que cada flujo requiere de un ancho de banda específico, y que la red tiene un límite de ancho de banda, conforme se admiten más y más flujos, el servicio *Best Effort* otorgado se deteriora. Esto se puede explicar con la analogía de una sala de cine, en donde muy generalmente hay control de admisión a la misma. Es evidente que el control de admisión está relacionado con la QoS que pueda otorgarse.

Es importante aclarar que el control de admisión no solamente se maneja globalmente para que un flujo entre a una red; este concepto también puede enfocarse a lugares o ambientes específicos dentro de una red. Un caso es cuando cada nodo en una red ejerce, por sí mismo, un control de admisión (como se explica en el subcapítulo 2.1.8).

2.1.4 Despachador de Paquetes

Un despachador de paquetes (o del inglés: "Packet Scheduler") es un seleccionador ubicado en cada una de las interfaces de salida de un nodo, que selecciona el siguiente paquete que habrá de salir por la interfaz, de entre dos o más colas de espera de paquetes, alojadas en dicha interfaz.

Al envío de un paquete se le refiere, también, como el despacho, la atención o el servicio del paquete. Cuando un paquete está saliendo por la interfaz, se dice que el paquete "está siendo atendido" (o se está atendiendo al paquete). También se indica que el despachador atiende "tráfico".

El despachador divide el ancho de banda de la interfaz de salida entre dichas colas. En términos generales se puede decir que un paquete se aloja en una cola específica porque pertenece a algún flujo que, a su vez, pertenece a alguna categoría de flujos a la que se asigna u otorga una fracción del ancho de banda de la interfaz de salida.

A la forma en que un despachador reparte su trabajo para atender a las colas se le llama "estrategia (o disciplina) de atención de colas". Un despachador puede atender a un paquete a la vez. De entre los paquetes de una cola, se les atiende en una forma FIFO (el primero que entra a la cola es el primero en ser servido).

Una estrategia de atención de colas común es la "atención por turnos" o "Round Robin" (por su denominación en inglés). Al despachador que utiliza esta estrategia se le suele llamar "despachador Round Robin". En esta modalidad cada cola se atiende por turnos, a manera rotatoria. La cantidad de tráfico a enviar en el turno de atención en una cola puede depender de un peso asignado a la misma, en cuyo caso la versión del despachador se llama "despachador con atención por turnos con ponderación de pesos" o "Weighted Round Robin" (WRR).

Cuando una cola tiene paquetes por enviar se dice que la misma tiene "trabajo por hacer" o está "ocupada". Un despachador puede operar con reservación, de tal forma que cuando en el turno de atención de una cola, ésta no tiene trabajo por hacer, el despachador no enviaría tráfico alguno, ni de esa cola ni de cualquier otra cola que sí tuviese trabajo por hacer. Se dice que este despachador desperdicia energía.

Un despachador es "conservador de trabajo" (o de energía) [39] si envía a tasa máxima cuando hay trabajo por hacer en al menos una de las colas de espera. En este despachador las colas ocupadas pueden usar el ancho de banda, de la interfaz de salida, asignado a las colas desocupadas. Aun así, el peso de cada cola le reserva una cantidad mínima de ancho de banda de la interfaz de salida cuando la cola tiene trabajo por hacer.

Otro despachador muy referido es el llamado de "Espera Equitativa Ponderada" (en inglés Weighted Fair Queuing –WFQ), también llamado "Procesador Generalizado de Repartición Paquete por Paquete" [40] (en inglés Packet-by-Packet Generalized Processor Sharing – PGPS)[40]. Este despachador, conservador de energía, reparte el tráfico (de paquetes) entre colas siguiendo cercanamente el comportamiento indicado por otro despachador, ideal, llamado "Procesador Generalizado de Repartición" (en inglés Generalized Processor Sharing – GPS) el cual puede servir a múltiples sesiones simultáneamente y donde el tráfico es divisible infinitamente, de tal forma que puede garantizar exactamente a cada cola una atención (un mínimo ancho de banda) g_i, donde $g_i = r\,\phi_i \Big/ \left(\sum_j \phi_j\right)$, y donde ϕ_1, ϕ_2, … ϕ_N, son números reales positivos que caracterizan a cada cola, que se pueden llamar "pesos", y donde *r* es el

[40] En este trabajo WFQ y PGPS se utilizan de forma indistinta.

ancho de banda disponible del despachador (también referido como ancho de banda del servidor o enlace). Se puede llamar al factor $\phi_i \Big/ \left(\sum_j \phi_j \right)$ el peso relativo de la cola i. Un caso particular de los valores de los pesos es cuando los mismos suman 1.

En los despachadores conservadores de energía que operan con tráfico de paquetes, incluyendo PGPS, cuando se atiende a un paquete de una cola, el paquete debe atenderse de inicio a fin, aunque el paquete sea muy grande y se sobrepase la cota designada para atención a la cola del paquete, por lo que con estos despachadores, cuando los paquetes son de tamaños diferentes, se dice que los despachadores intentan servir a cada cola con una tasa proporcional a su peso relativo.

En la operación del despachador PGPS, se requiere que el despachador GPS opere en paralelo. GPS no sirve tráfico en realidad, pero opera como si recibiese el mismo tráfico del despachador PGPS, solamente considerando que el tráfico puede ser divisible infinitamente, y llevando sus propias cuentas del tráfico que va sirviendo. GPS sirve de pauta para la operación de PGPS, indicándole cuál es el siguiente paquete que debe servir. Por la manera de operar de PGPS, el mismo se considera una excelente aproximación [40] de GPS, con una diferencia, en el servicio de paquetes, menor al tiempo que toma servir al paquete de mayor tamaño. Considerando esta diferencia, WFQ resulta favorable en comparación con otros tipos de despachadores, como WRR.

Por la cantidad de trabajo de cálculo requerida por WFQ, el mismo no ha sido preferido para su utilización en nodos que requieren atender a muchos miles de paquetes por segundo. El uso de WFQ se observa más en simuladores, con motivos de estudio. En el Apéndice D se explica con más detalle la operación del despachador GPS/PGPS (WFQ).

2.1.5 Calidad de Servicio (QoS)

Concepto

Las aplicaciones que son sensibles a retrasos, a pérdidas, o a *Jitter* (variaciones en los retrasos), se están utilizando más y más en las redes. El poder otorgar servicios de transporte a los flujos de estas aplicaciones, de forma que se cumplan las expectativas mínimas relativas a las características de sensibilidad citadas, es lo que se conoce como “otorgar servicios con calidad”, y a dichas características se les puede llamar “características de calidad” o “parámetros de calidad”. Un servicio con calidad es un servicio diferente al convencional *Best Effort*. En el ámbito de las redes se dice que cuando una red ofrece servicios con calidad, ésta ofrece, o tiene, “Calidad de Servicio” concepto que suele abreviarse con las siglas, provenientes del inglés, QoS (*Quality of Service*). La QoS [4] [5], similarmente a lo que sucede con la admisión, puede enfocarse a una parte de la red, o a una clase en la red. Una clase en una red es una

categoría a la que se asocian los paquetes que fluyen por la red. Según la clase a la que pertenezca un paquete sería el tipo de tratamiento al mismo, en su curso por la red[41].

La Admisión para QoS

Conforme más flujos cursan por una red los recursos de la misma deben compartirse más, y por consiguiente la QoS que se otorga en la red, a los flujos, tiende a deteriorarse. Así, para que un flujo sea admitido (se le permita el ingreso) en una red (a este flujo se le llama *nuevo flujo*) se puede hacer una "prueba de admisión[42]" con la que se evalúa si la QoS que se brindaría en la red seguiría siendo satisfactoria[43], tanto para los flujos anteriormente admitidos como para aquél que se vaya a admitir, si es que se hiciera la admisión. Entonces, la admisión tiene una característica predictiva.

Formas de Admisión para QoS

Las pruebas de admisión consisten en modelos matemáticos predictivos, así que inicialmente se llaman "Pruebas de Admisión por Modelos". Si dichos modelos contienen parámetros provenientes de mediciones, entonces las pruebas de admisión se llaman "Pruebas de Admisión por Medición". Una de las primeras introducciones a los modelos de admisión basados en medición se dan en [41].

También, las pruebas de admisión se pueden clasificar en pruebas deterministas o estadísticas.

Forma Determinista.

Con esta forma se reserva (se garantiza), para el nuevo flujo, una cantidad de ancho de banda del ambiente. La QoS que se otorga al flujo no implica el cumplimiento de otras características de QoS como retraso, *Jitter* o pérdidas. Una prueba de admisión bajo esta forma se llama prueba determinista.

Forma Estadística.

En esta forma [42] los recursos del ambiente no se reservan. En este caso se asocia una probabilidad (pequeña) de incumplimiento sobre la característica de QoS prevista para los flujos.

---

[41] En la definición de los protocolos IPv4 (IP versión 4) e IPv6, en el encabezado de los paquetes, se define un octeto de bits relacionado con la asociación a clases. En el caso de IPv4 el octeto se llamaba "Tipo de Servicio", y en el caso de IPv6 el octeto se llamaba Clase de Tráfico. Tecnologías posteriores, como DiffServ (ver subcapítulo 2.1.8), usarían dicho octeto, de forma modificada, para la asociación de paquetes a clases (a las clases en el ámbito DiffServ se les llama "Agregados de Comportamiento" –ver pie de página 50).

[42] Esta prueba se suele llamar, en inglés, "Schedulability Test" [91].

[43] Los términos en que se sustenta la satisfacción se deben aportar en un contrato, usualmente llamado Acuerdo de Nivel de Servicios (Service Level Agreement –SLA [4] [5]).

Se considera que en un ambiente en donde no se reservan recursos hay mayor eficiencia en la utilización de los mismos. Una prueba de admisión bajo esta forma se llama prueba estadística.

2.1.6 Administradores de Redes o de Ambientes con QoS

De la existencia del concepto de QoS surge la necesidad de la existencia de un administrador para cualquier red, o ambiente, donde se otorgue QoS. Este administrador, y el dueño de cada flujo que transite por la red, o ambiente, contraen compromisos, por un lado sobre la QoS que la red o ambiente brinda al flujo, y por otro sobre el comportamiento que el flujo debe guardar durante su tránsito.

Los compromisos de parte de este administrador se basan en el trato que se dé a los paquetes del flujo en su tránsito, desde su ingreso hasta su entrega en su punto de egreso de la red, o ambiente.

Los compromisos que adquiere el dueño del flujo, sobre el comportamiento del mismo, pueden incluir el valor medio de la tasa de bits, el valor de las “ráfagas” de bits y la duración de las mismas.

El administrador de una red tiene interés en maximizar las utilidades por la prestación de servicios de transporte de flujos. Un factor para maximizar estas utilidades puede ser el utilizar lo más posible los recursos de la red. Otro ingrediente puede ser cobrar caro por los servicios otorgados de la red. Se puede hacer una combinación de ambos ingredientes. Para cobrar por los servicios de la red se debe cumplir con los compromisos de otorgamiento de QoS contraídos con los dueños de los flujos. Los ingredientes anteriores pueden ser contrarios, uno del otro. Esto sucede porque cuando hay muchos flujos en la red se emplean más recursos de la misma, y se tiende a deteriorar la QoS otorgada a los flujos. Así que, ¿cómo se cobraría caro por un mal servicio?

2.1.7 Recursos Sobrantes para la QoS

El administrador, o dueño, de una red, en donde se tiene un conocimiento estadístico de la demanda de los usuarios, con el fin de resolver el problema de lograr otorgar QoS, puede enfocar su solución en disponer de recursos sobrados en la red, sin requerir admisión.

Sobre esto, surgen dos dudas: 1- ¿Funcionaría este enfoque para otorgar QoS?, y 2- ¿Se tendría la rentabilidad esperada? (ya que los recursos tienen un costo).

Se pueden enlistar las siguientes condiciones que pudiesen impedir que la QoS se mantuviese, en una red, con tan solo disponer de recursos sobrados:

1- Los detonantes de la carrera de oferta contra demanda, en la sociedad, emanan de las mejoras tecnológicas, administrativas y legales que se van dando, de los cambios en los estilos de vida y en los valores de la gente, y de la creciente injerencia de la sociedad en la tecnología misma. La gente observa un aumento de capacidades y de alternativas, y una disminución de costos en los servicios a los que se puede acceder, lo que promueve una mayor demanda de servicios. Las redes están contundentemente inmersas en este tipo de carrera por lo que el ofrecer buena QoS con base en disponer de recursos sobrados implicaría tener siempre el poder de compra de dichos recursos. Algo así como tener una "fuerza bruta inagotable".

2- Los grandes cambios de tasa en el tráfico que tienen algunas nuevas aplicaciones propician que crezca la diferencia entre el uso promedio y el uso máximo del ancho de banda de las redes. Esto dificulta la evaluación de cuáles son los recursos mínimos requeridos en las redes, para poder otorgar una "buena" QoS.

3- La existencia de una concatenación de redes con características diversas, donde algunas de ellas tienen un cuantioso ancho de banda[44], mientras que otras tienen un ancho de banda reducido. En la Internet, las redes portadoras (*Carrier Networks*), que son las más centrales, usualmente están sobre dimensionadas para propósitos de simplicidad operacional y por confiabilidad, lo que conlleva a una utilización observada muy por debajo de niveles críticos[45] . No obstante, en la zona de redes portadoras puede haber situaciones que lleven a limitación de recursos. Por ejemplo, esto se ha suscitado con la falta de acuerdos entre los administradores de distintas redes portadoras, lo que lleva a tener recursos limitados en los puntos de interconexión (*Peering Points*) [43] entre dichas redes. Por último, en las redes menos centrales de la Internet es usual que NO sobren recursos.

En contraposición al uso de recursos sobrados, como enfoque para otorgar QoS, existen las llamadas "arquitecturas" de QoS. Se puede decir que una Arquitectura de QoS es el conjunto de elementos funcionales y físicos, en las redes, para que éstas puedan ofrecer QoS (se les puede llamar "mecanismos" de QoS).

---

[44] En 2005 se indicaba que los enlaces céntricos en las comunicaciones de paquetes estaban sobre-aprovisionados [97].

[45] Se cita (traducido) a [43]: "Hoy en día, la práctica común es sobredimensionar las redes portadoras para tener tanto simplicidad operacional como confiablidad, lo que hace que la utilización observada esté muy por debajo de los niveles "críticos". Es bien sabido que la mayoría de los proveedores mejoran sus enlaces internos, con más capacidad, cuando la utilización media rebasa un umbral arbitrario…".

Es importante indicar que se ha argumentado que los mecanismos de QoS no serán implantados en gran escala en la Internet por la dificultad de poner de acuerdo a los Proveedores de Servicio de Internet (Internet Service Providers –ISP), y que estos mecanismos se restringirían a usarse en menor escala, en sistemas administrativos en donde pudiese haber consenso sobre la aplicación de tales mecanismos. En gran escala se ha utilizado el enfoque de aprovechar los recursos de la Internet sin requerir aplicar mecanismos de QoS en la misma; más bien se intenta aprovechar mejor los recursos de la Internet y minimizar el empleo de comunicaciones de larga distancia en la misma, como lo hacen las redes de entrega [44] (ver subcapítulo 9.1).

2.1.8 Arquitecturas de QoS

IntServ

Entre 1994 y 1996 se definió una “arquitectura” para la protección de aplicaciones que requieren QoS en las redes, llamada IntServ (Integración de Servicios) [45]. En una red con IntServ, se determina una ruta específica para el paso de cada flujo, y los nodos por donde pase la ruta de un flujo estarán “al tanto” de la existencia del mismo y de su tratamiento particular (se dice que los nodos llevan “estado por flujo”). Cada nodo por donde pasa la ruta de un flujo deberá haber aceptado el paso del flujo y lo hace con relación al ancho de banda que tiene disponible para su paso.

IntServ ofrece dos tipos de servicio. En el primero, el Servicio Garantizado [46], se realiza un aseguramiento de ancho de banda en los enlaces de la ruta, y se usa una admisión con enfoque determinista. Este servicio se aproxima cercanamente a aquél que sería provisto por un alambre dedicado, con una capacidad (ancho de banda) de R *bit*/s entre la fuente y el receptor. En el segundo, el Servicio de Carga Controlada [47] se realiza una evaluación de los recursos disponibles, y se usa una admisión con enfoque estadístico. El comportamiento de extremo a extremo provisto por una serie de elementos de la red que proporcionan servicio de Carga Controlada estrechamente se aproxima a la conducta visible del servicio que se recibe por el servicio Best Effort en condiciones “sin carga” por la misma serie de elementos de la red.

Es interesante notar que desde IntServ se reconoce un problema que en DiffServ se conoce como "distorsión de tráfico". Se traduce de [46]: “Dentro de la red, la vigilancia (Policing) no produce los resultados deseados porque los efectos de encolamiento ocasionalmente causan

que un flujo que ha entrado en una red cumpliendo con las características de tráfico comprometidas, ya no cumpla en algún elemento de red río-abajo[46].

En una arquitectura anterior a IntServ, llamada ATM (*Asynchronous Transfer Mode*), se maneja, al igual que en IntServ, el establecimiento de rutas para los flujos y el control de admisión a nivel de nodos. En ATM la admisión de un flujo se indica como la admisión de una llamada y al control de admisión se le llama "*Call Admission Control*" (CAC). Sobre ATM se traduce, de [48], lo siguiente, que aplica también para IntServ: "El problema general de admisión de llamadas en la red se compone de dos partes. En primer lugar la red tiene que encontrar una ruta candidato a una llamada - esto se llama *el problema de enrutamiento*. Las llamadas son rechazadas si no se puede encontrar tal ruta. En segundo lugar, después de que un candidato se ha encontrado, cada nodo, a lo largo de la ruta, debe decidir individualmente si puede aceptar la llamada".

IntServ no se considera "escalable", es decir, si una red que maneja IntServ crece, en términos de flujos manejados, entonces un nodo de esta red, por lo general, requeriría aumentar el número de flujos que maneja. Conforme se da este crecimiento, el nodo puede llegar a verse imposibilitado de manejar un estado por flujo porque esto le demanda memoria y capacidad de procesamiento no disponibles. El concepto de escalabilidad tiene indeterminación pues con la mejora en la tecnología (lo que deriva en un incremento en las capacidades de los nodos), los límites relativos al concepto van cambiando. IntServ no resuelve, tampoco, el problema de crecimiento por la interconexión de redes.

DiffServ.

Entre 1997 y 2001 se definió una "arquitectura" competidora de IntServ, llamada DiffServ (Diferenciación de Servicios o DS) [4] [5] [49] [50] [51] [52] [53] [54] para ofrecer QoS en redes[47]. A una red en donde se ofrece DiffServ se le denomina "Dominio DiffServ", al que, por simplicidad, se le refiere simplemente como "dominio". El dominio está conformado por dos tipos de nodos: 1- nodos de frontera (aquellos que se comunican tanto con nodos que están afuera del dominio como con nodos que están adentro del dominio), y 2- nodos internos o centrales (aquellos que se comunican solamente con nodos adentro del dominio). El interior de un dominio está conformado por sus nodos interiores, y los enlaces entre éstos[48]. Un nodo de frontera puede ser un nodo de ingreso, un nodo de egreso, o ambos a la vez. Un flujo ingresa a un dominio por un nodo de ingreso, y sale del dominio por un nodo de egreso. Una

[46] Río Abajo: Más adelante, en el curso del flujo.
[47] Una explicación sobre DiffServ se puede encontrar "en línea" en un reporte [86] del autor de esta Tesis.
[48] En este interior también se podrían incluir las interfaces de los nodos de frontera que se conectan hacia los nodos interiores, incluyendo los enlaces de conexión. Este detalle no se observa en las descripciones de DiffServ de la literatura, quizá por considerarse poco significativo en el contexto general.

vez que entra un flujo al dominio, se mantiene conocimiento de éste en los nodos de ingreso y de egreso. Dado que en los nodos de frontera se lleva cuenta de los flujos "individuales", se dice que en la frontera, o borde, del dominio se lleva un "estado por flujo".

En DiffServ cada flujo debe pertenecer a una clase, que en DiffServ se llama "Agregado de Comportamiento" (Behavior Aggregates)[49]. Por definición, en un domino existen pocas clases, pero no se indica cuánto es "pocas". Para que un flujo pertenezca a una clase, cada paquete del flujo se "marca" con un código que lo asocia con esa clase, de esta forma, el flujo se incorpora a la clase, en el dominio.

A un paquete asociado a un Agregado de Comportamiento se le trata, en cada nodo, de forma igual y específica. Ese trato se llama: "Comportamiento por Salto" (Per-Hop Behavior" – PHB[50]). En el interior de un dominio no se lleva cuenta de los flujos individuales (no se lleva "estado por flujo"), sino únicamente "estado por clase".

Se considera que los nodos interiores del dominio conducen más flujos que los nodos de frontera. Dado que los nodos interiores no llevan estado por flujo, el hecho de que condujesen una gran cantidad de flujo no les impone una limitación en su capacidad de manejo de estado. Por esta razón, a DiffServ se le clasifica como un enfoque para ofrecer QoS que puede ser escalable, aunque este último término es indeterminado.

En un nodo de ingreso existe la función de vigilancia (o Policing en Inglés) de los flujos que entran al dominio, por ese nodo, para verificar que los mismos cumplan con las características acordadas de tráfico. Para un flujo dado que no cumple con estas características, el nodo de ingreso puede realizar sobre el flujo una función llamada "acondicionamiento" con el que se procesa el flujo para buscar que el mismo cumpla con las características acordadas[51].

[49] Cada paquete se añade a una clase escribiendo el código de la clase en su campo llamado Codepoint

[50] DiffServ no determina cómo deben ser los PHB, pero sí existen propuestas como lo son: AF [88] y EF [62] (ver Apéndice B.2).

[51] Este acondicionamiento consta de cuatro elementos básicos [95]: medidor (Meter), marcador (Marker), Conformador (Shaper) y Descartador o Desechador (Dropper). Con el conformado los paquetes del flujo son redispuestos en el tiempo, sin desordenarlos, buscando que el flujo cumpla sus características acordadas de tráfico. Con el marcado los paquetes pueden ser cambiados de clase (a una de manor jerarquía), si existe. Con el desechado los paquetes se descartan.

El conjunto de nodos operando, cada uno, con un mismo PHB, junto con acciones de vigilancia y acondicionamiento determinados pueden dar lugar a un tratamiento general al flujo en el dominio, que se le llama "servicio" (ver Apéndice B.2).

Debido al tratamiento "por clase" en lugar de "por flujo", con DiffServ no se pueden ofrecer garantías a los flujos. En concordancia con esto, diversos autores, como en [55], indican que en el interior de los dominios existe la llamada "distorsión de tráfico", que, de manera similar a lo indicado sobre IntServ, es el fenómeno donde un flujo que ha entrado en el dominio cumpliendo con las características de tráfico comprometidas, ya no las cumpla en algún elemento del interior del dominio. En [56] se indica que, aun existiendo control de admisión en las "redes" DiffServ, cuando hay "sobrecarga" (Overload) de tráfico en las mismas los flujos sufren de degradación en la QoS.

Nota. A veces se usa el término "admisión" de un flujo a un dominio, en donde la acepción del término solamente implica que el flujo cumple con sus características comprometidas de tráfico; así que este término no tiene el mismo significado que aquél usado en esta Tesis, que implica la aplicación de un modelo predictivo.

Indefiniciones de IntServ y DiffServ.

En ambas arquitecturas hay indefiniciones, las cuales no son un defecto, sino que están dadas a propósito para permitir que las mismas puedan atenderse con mayor detalle, posterior a la definición de las arquitecturas; sin embargo, la dificultad para encontrar solución a estas indefiniciones deja a ambas arquitecturas con un horizonte incierto.

Las indefiniciones incluyen lo siguiente (para ambas arquitecturas).

1- ¿Cómo generar servicios y sus correspondientes características de QoS con base en los elementos que componen las arquitecturas?

2- ¿Cómo administrar los servicios?, lo que incluye el manejo de las solicitudes de servicio, buscando maximizar el tráfico que pueda admitirse, respetando las garantías acordadas sobre QoS (lo que implica la interacción con el administrador de la red). Se incluyen las funciones de control de admisión.

3- ¿Cómo realizar la Ingeniería de Tráfico?

4- ¿Cómo resolver la concatenación de redes o dominios que otorguen individualmente QoS, y qué se requiere para que la QoS se extienda en la concatenación [15]?

5- ¿Cuál es el tamaño o capacidad de las redes para que la operación de estas arquitecturas se considere viable?

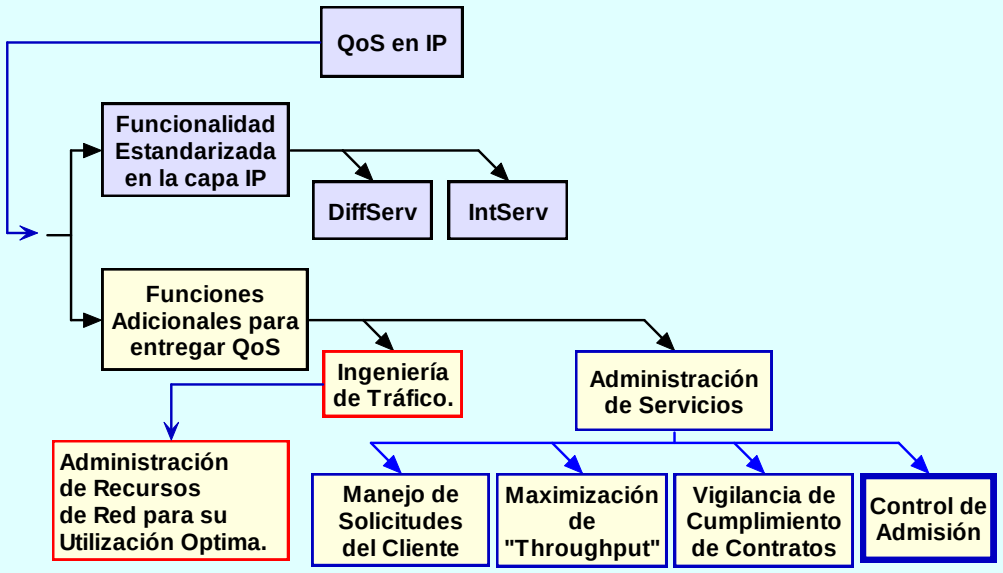


Figura 1. Componentes no definidos en DiffServ e IntServ, necesarios para proporcionar QoS.

En la Figura 1 se representan los componentes de Ingeniería de Tráfico y de Administración de Servicios, de las arquitecturas IntServ y DiffServ, necesarios para entregar QoS de extremo a extremo (diagrama obtenido de los conceptos de [57]).Estos componentes se encuentran en el "plano de control" de las arquitecturas, a diferencia de las funciones de vigilancia, conformado, clasificación y enrutamiento, que se ubican en el llamado "plano de datos".

La operación (en el plano de datos) de IntServ y DiffServ se ha incorporado, en forma parcial, en equipos para su implantación en forma comercial [58]. DiffServ, además, se puso en operación en algunas redes, de prueba, como por ejemplo en el proyecto TEQUILA [57] [59], que operó entre 2000 y 2002. Este proyecto utilizó funciones[52] que llenaban, en mayor o menor medida, el hueco existente por las indefiniciones indicadas.

2.1.9 Redes donde los Flujos están Restringidos a Cursar por Rutas

Una variante de redes con QoS, en donde se pueden hacer admisiones, son aquellas redes en donde cada flujo está restringido a cursar por una ruta dada (dígase que la red está sujeta a tráfico por rutas). En este caso cada admisión se haría a la entrada de la ruta por donde habría de pasar el flujo (no en cada nodo de la ruta).

[52] Que incluía un conjunto de herramientas para definición de servicios, y para ingeniería de tráfico, para obtener garantías cuantitativas punta-a-punta de QoS.

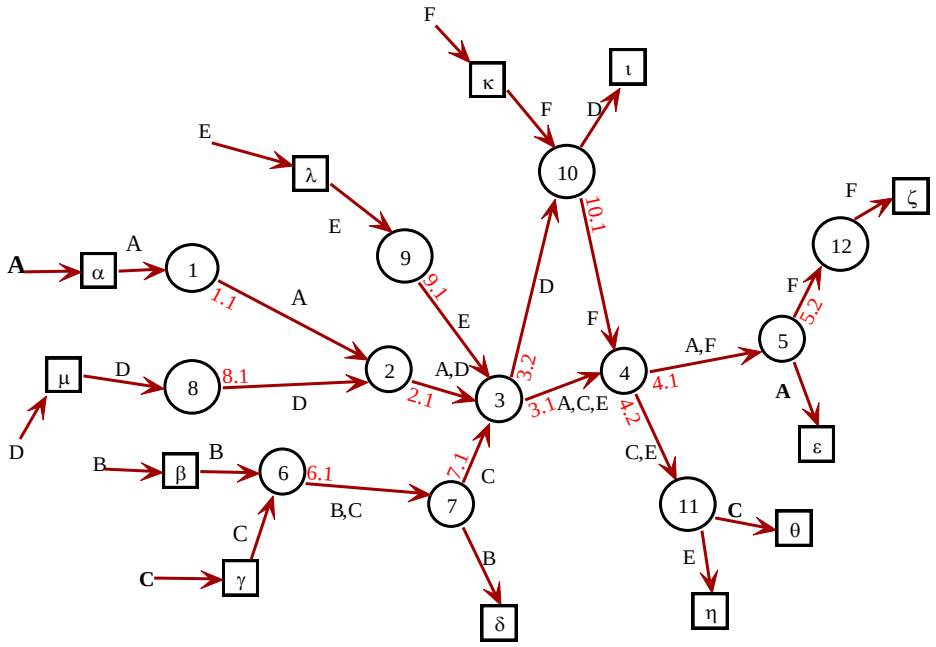


Figura 2. Ejemplo de una red donde los flujos están restringidos a cursar en rutas predefinidas. La ruta *A* se interfiere, o cruza, con las rutas *D*, *C*, *E*, y *F*.

Una red sujeta a tráfico por rutas no necesariamente usa DiffServ (podría hacerlo), donde en algunas o en todas las rutas, dos o más clases compartirían los recursos.

Para ejemplificar el uso de rutas dentro de una red, en el ejemplo de la Figura 2 se tiene una red (con enlaces unidireccionales) sujeta a tráfico por ruta, la cual tiene nodos de frontera y nodos interiores. Las rutas están indicadas con letras mayúsculas. Los nodos interiores se representan con círculos, que tienen números, y los nodos de frontera con cuadrados, que tienen letras griegas. Como ejemplo, los enlaces que forman parte de la ruta “A” están señalados con dicha letra. Esta ruta inicia en el nodo de frontera (de ingreso) “α” y concluye en el nodo de frontera (de egreso) “ε”. Se puede observar que la ruta *A* comparte al menos un enlace con las rutas *D*, *C*, *E*, y *F*. Esto se conoce como intersección, interferencia, o cruce entre rutas. Se puede decir que la ruta *A* se interfiere con las rutas *D*, *C*, *E* y *F*.

Una forma de proteger a rutas contra rutas cruzadas es reservar ancho de banda de una ruta con relación a las rutas cruzadas, lo que suele llevar a la utilización ineficiente del ancho de banda de las rutas, y de la red en general.

2.2 *Problema a Resolver Identificado*

Dentro de los tipos de redes que implementan QoS sin reservación de ancho de banda, aquellos que ponen en práctica la admisión distribuida, por rutas, representan un subtipo importante, donde se atenúa la dificultad del proceso de admisión.

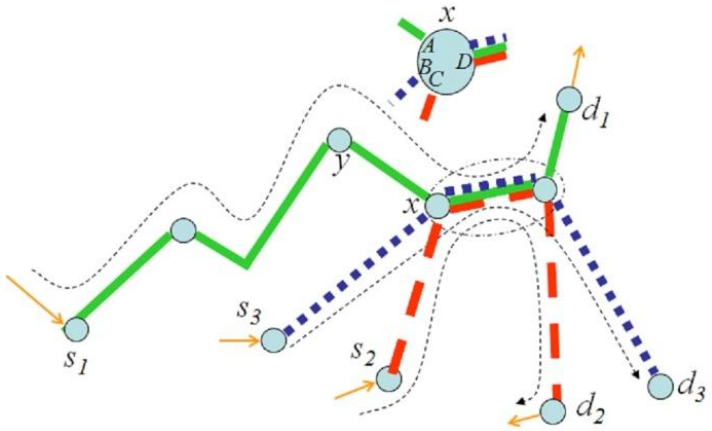


Figura 3. Sección de una red donde los flujos están restringidos a cursar por rutas.
Las líneas punteadas indican las rutas. Las rutas $s_1 - d_1$, $s_2 - d_2$ y $s_3 - d_3$ se interfieren en la interfaz de salida *x* del nodo. Este nodo tiene tres interfaces de entrada: *A*, *B* y *C*, y una interfaz de salida *D*, donde se produce la interferencia, en la cual el tráfico de las tres rutas indicadas convergen o confluyen.

Por ejemplo, la Figura 3 representa parte de una red que tiene tres rutas, $s_1 - d_1$, $s_2 - d_2$ y $s_3 - d_3$, que convergen en el nodo *x*, llegando, cada una al nodo por cada una de sus interfaces: *A*, *B* y *C*, respectivamente, y compartiendo el ancho de banda de la interfaz de salida, *D*, del nodo. En este trabajo, a este nodo y a las rutas referidas se les llama "nodo de interferencia *x*" y "*rutas de interferencia*", respectivamente. Estas rutas convergen, específicamente, en la interfaz de salida, *D*, del nodo *x*. Se puede decir que las rutas convergen en el enlace que sale del nodo *x*, a través de la interfaz de salida *D*.

Cuando en una ruta se aumenta el tráfico, se puede causar una disminución en el ancho de banda disponible en las rutas que la interfieren (sus rutas de interferencia), con un posible deterioro de la QoS ofrecido en todas esas rutas. La ruta que aumenta el tráfico no "sabe" sobre dicho deterioro. Este es el problema de estudio de esta Tesis.

En esta Tesis se evalúa el impacto en las redes debido al problema indicado, con el uso del Método q1, en los nodos de la red.

Asimismo, se evalúa el impacto de este problema cuando en las redes se utilizan otros métodos, alternativos, de atención de tráfico.

Finalmente se propone un método de solución (el Método STP), y se evalúa y comparan sus resultados contra aquellos de los otros métodos evaluados.

Resumen del Capítulo

En este capítulo se han visto diversos conceptos fundamentales relativos a la redes y al tráfico en las mismas. Ya con los conceptos presentados se identifica el problema a resolver en las redes donde los flujos están restringidos a cursar por rutas específicas.

Se destacan los siguientes aspectos, presentados en el capítulo:

1- Las redes que tienen recursos limitados, mientras ofrecen una QoS satisfactoria a sus flujos de admitidos, pueden sufrir un deterioro de la QoS ofrecido después de aumentar su tráfico (tal vez como consecuencia de la admisión de nuevos flujos). Este deterioro puede ser pequeño, y por consiguiente, admisible, mientras que todavía hay recursos sobrados en la red, pero con el aumento de más y más tráfico en la red este deterioro puede crecer hasta el punto en que se vuelva un factor importante en la QoS.

2- El proceso de admisión, para que un flujo entre en una red, implica la operación de un algoritmo predictivo sobre la QoS que habría en la red después de admitir, para evitar que la concesión de admisión haga que la QoS ofrecida en la red se torne inaceptable.

3- Una red en la que cada flujo se admite con la restricción de transitar por la red a través de una ruta específica, donde toda decisión de admisión la realiza un administrador independiente por cada ruta, es una red con admisión por ruta. Cuando no hay reservación de ancho de banda en ruta alguna se permite una mayor flexibilidad en el uso de su ancho de banda, con el inconveniente de que las rutas puedan interferirse una con la otra.

4- Un proceso de admisión por-ruta, en una red, puede ser centralizado o distribuido. El proceso es centralizado si en el mismo, se mantiene conocimiento de las condiciones de tráfico en todas las rutas de la red, de modo que antes de otorgar una admisión en una ruta se pueden considerar los posibles efectos de la admisión en todas las rutas de la red. El proceso es distribuido si hay un proceso de admisión, por separado, para cada ruta en la red, sin considerar los posibles efectos de la admisión en las otras rutas de la red. Al evaluar si una admisión a una ruta se concede o no, es más fácil considerar las condiciones del tráfico solamente en esa ruta. Esta simplicidad hace que los procesos de admisión distribuidos puedan ser llevados a ambientes de mayor tamaño (según los términos de redes se suele decir “más escalables”), en comparación con su contraparte,

la admisión centralizada, sin embargo, esta simplicidad puede traer consigo un deterioro de la QoS por la interferencia entre rutas.

5- En esta Tesis se evalúa el impacto en las redes debido al problema de interferencia entre rutas, y se propone un método de solución (el Método STP), y se evalúa y comparan sus resultados contra los de otros métodos alternativos, entre ellos el Método q1 (el método convencional en donde se tiene una cola por cada interfaz de salida en cada nodo).

Capítulo 3. Trabajos Relacionados

Introducción al Capítulo

En este capítulo se presentan los trabajos publicados, relacionados con el problema identificado a resolver en este trabajo. Para algunos de estos trabajos se presentan topologías ilustrativas sobre los problemas que resuelven, y se relacionan estos problemas con el problema a resolver en este trabajo.

Al final del capítulo se hace un cuadro para resumir las principales características de los trabajos relacionados, aquí presentados.

3.1 Sumario de Trabajos Relacionados

El método de solución para el problema de estudio de este trabajo, está motivado por algunas de las características del tipo de los nodos utilizados por el modelo DiffServ [4] [5] [49] [50] [51] [52] [53] y por las condiciones de red propuestas en [15] [16]. En estos dos últimos trabajos se presenta un método de admisión de flujos en redes con rutas predefinidas, que atiende a diversas clases, y donde la admisión es distribuida, por cada ruta, la que se realiza en el nodo de egreso de la ruta. En la red, cada interfaz de salida de cada nodo central tiene una sola cola de espera para los paquetes de los flujos, la cual debe ser compartida por los flujos de las rutas que confluyan en el nodo y que se continúen por esa interfaz de salida. Por lo anterior, existe interferencia entre las rutas en la red.

El método tiene un algoritmo que mide el tráfico de salida de la ruta y a partir de las mediciones evalúa los recursos disponibles en la ruta (en términos de un “envolvente de servicio disponible”), y predice el retraso en la ruta si se aumentase una cantidad de tráfico de un flujo solicitante, sin requerir conocer la “disciplina” de servicio ni el tráfico de en los nodos intermedios en la red, para buscar otorgar la admisión si el retraso en la ruta no se afecta inadmisiblemente. Aun cuando la interferencia entre rutas impone inconvenientes las variaciones medidas, en los recursos disponibles, incorporan dicha interferencia.

Se pretende aprovechar al máximo los recursos de la red que no estarían disponibles en caso de que se manejase reservación de recursos, por eso la admisión que se hace la clasifican los autores como admisión estadística, y el servicio que da la red se llama servicio estadístico.

La abstracción del sistema de estudio se observa en la Figura 4.

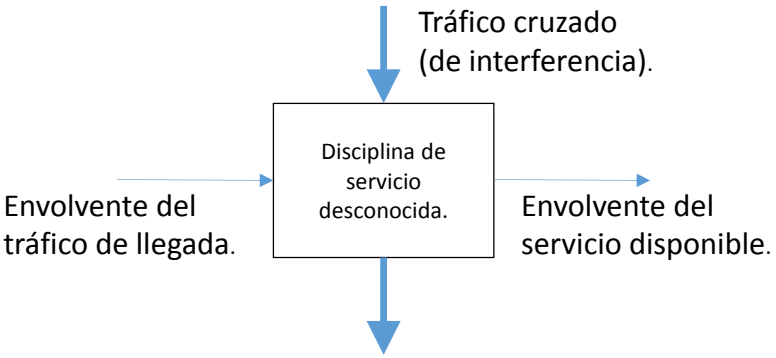


Figura 4. Abstracción del sistema de estudio en **[15]** y **[16]**.

Para medir los retrasos de los paquetes, en cada paquete se requiere el estampado del tiempo en el nodo de ingreso, y posteriormente, ya sea el estampado del tiempo en el nodo de egreso (para lo que se requeriría sincronización de tiempos en los nodos de ingreso y egreso), o el estampado del tiempo de espera en cada nodo subsecuente en la red.

Se realizan experimentos por simulación, con base en un escenario con una topología que se muestra en la Figura 5, donde el tráfico cursa del nodo *A* al *A*' y del *B* al *B*', y donde el nodo central *D* tiene una cola para alojar todo el tráfico que se dirige al nodo *C*. Los resultados se dan en términos del retraso del flujo de una clase que recibe servicio estadístico, que va del nodo *A* al nodo *A*', contra la utilización que la clase hace del enlace central *D*-*C* (a mayor utilización mayor retraso). Se comparan los resultados experimentales de retrasos, con aquellos obtenidos al evaluar la expresión matemática del algoritmo de admisión.

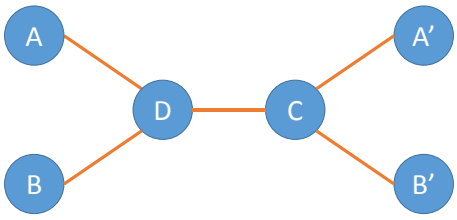


Figura 5. Topología de experimentación en **[15]** y **[16]**.

Esta utilización promedio del enlace es el ancho de banda promedio de la clase entre el ancho de banda promedio disponible del enlace (que es el ancho de banda del enlace menos el ancho de banda usado en promedio por las demás clases).

En la Figura 5 se observa que en el nodo *D* hay interferencia de las rutas *A*-*A'* y *B*-*B'*.

Finalmente se realizan otros experimentos con base en una la topología de la red UUNet (ver Figura 6) con tráfico entrando por Houston hacia Toronto vía Atlanta y Chicago, con una idea de experimentación similar a la de la topología planteada en la Figura 5, en donde también se observa interferencia entre rutas.

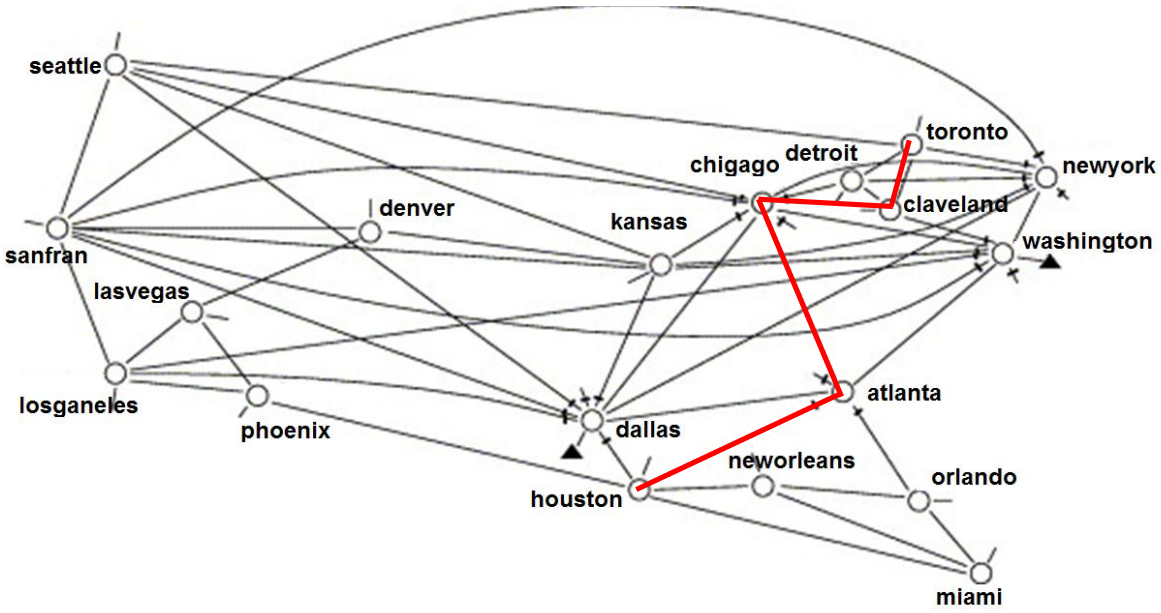


Figura 6. Topología alternativa de experimentación en **[15]** y **[16]**.

Estos trabajos hacen mención del problema de la interferencia entre rutas, y se refieren a un método en que una ruta puede obtener información sobre el tráfico de otras rutas, para poder tomar decisiones de admisión y no impactar inaceptablemente a esas otras rutas, con el costo de afectar la simplicidad de su método de admisión propuesto.

Un enfoque para la repartición de un ancho de banda común entre las clases en una red se presenta en [17]. En este caso el tráfico de cada clase se aloja en una cola diferente, en la interfaz de salida de los nodos centrales de una red, y la repartición de ancho de banda se da en términos de cuánta atención da el despachador de la interfaz a cada cola.

Se explica que las clases en una red pueden tener diferentes tratamientos, y ser valoradas de manera diferente, por sus características. Por ejemplo, los flujos de las aplicaciones de “misión crítica”, como aquellas para el control en tiempo real, se asignarían a una clase en la red; y los flujos de otras aplicaciones que fuesen tolerantes a fallas ocasionales en la QoS otorgada en la red, se asignarían a otra clase. Estas últimas aplicaciones podrían ser las transacciones en línea” y las videoconferencias.

El trabajo de [17] propone un método llamado RLAP (Reinforcement Learning Adaptive Provisioning o –Aprovisionamiento Adaptivo por Aprendizaje por Reforzamiento) que obtiene una política para maximizar la utilización de la red y minimizar los costos de utilización de la misma. Estos factores de utilización y costos se conjuntan en un indicador llamado "ganancia total de la red", el cual se busca maximizar.

Se indica que se busca hacer que esta maximización sea de "largo plazo", por lo que al método se considera proactivo; al contrario de los métodos que buscan una maximización a corto plazo, que se consideran reactivos. Se indica que el método propuesto se puede adaptar a condiciones de tráfico, planes de precio y especificaciones de QoS que cambien con el tiempo.

El método opera con un agente en cada nodo de la red, y con un administrador central de la red, el cual está en contacto con los agentes. A intervalos regulares, el administrador central colecta medidas de tráfico de los agentes, como los bits transmitidos, medida que implica un ingreso económico; y como las pérdidas por congestión, los retrasos y el incumplimiento de *rendimiento* (*Throughput*[53]), medidas que implican una penalización o pérdida económica. Con estas medidas y con los valores de las tarifas de cobro o de penalización, el administrador central calcula la ganancia total en cada intervalo, y una ganancia total acumulada sopesada.

El administrador realimenta la ganancia acumulada al agente de cada nodo, el cual, de manera independiente, busca aprovisionar el ancho de banda, a las diversas clases en el nodo, de tal forma que con cada nuevo aprovisionamiento se produzca una mayor ganancia de la red.

El aprovisionamiento en cada nodo se hace mediante el cambio de pesos en el despachador o despachadores WFQ del nodo. Cada *T s* el agente de cada nodo genera un cambio aleatorio en los pesos mediante una "unidad gaussiana" que incluye una media y varianza para cada peso, que sirven para la nueva generación aleatoria. Un ejemplo de cómo se van haciendo los ajustes es: si después de que se aumentó la media de un peso, la siguiente ganancia obtenida aumentó, entonces la tendencia será a aumentar nuevamente el valor de la media.

Se estudia el método en un escenario con una red con rutas predefinidas para cada clase. Cada clase tiene una cola en cada interfaz de salida de los nodos intermedios (ver Figura 7).

[53] El *Throughput* es la tasa promedio de la entrega de tráfico exitoso, de un nodo fuente a un nodo destino.

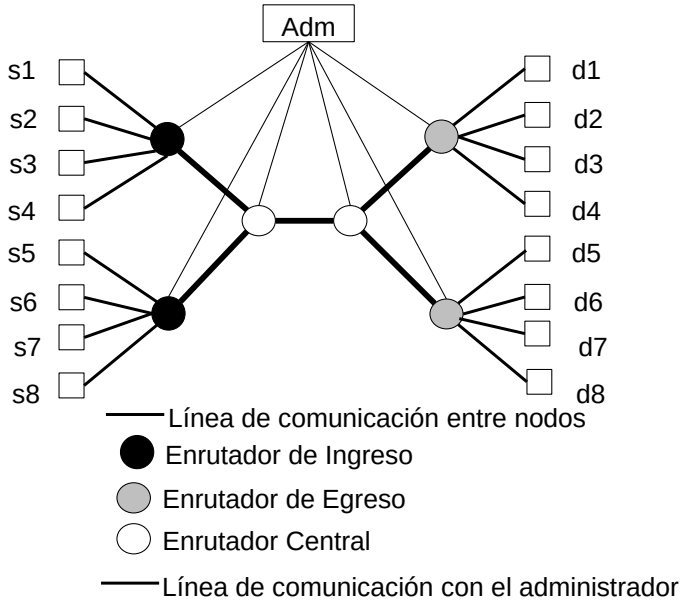


Figura 7. Topología de la red en el escenario de estudio de [17]

Se hacen experimentos por simulación. Los experimentos duran 20,000 *s* (5.55 horas), y en ellos *T* = 500s (8.33 min). En los experimentos se observa que se llega a una estabilización como a los 10,000 *s*, obteniendo un resultado favorable contra una solución con pesos constantes.

Las fuentes s_i y los destinos d_i son elementos que se añaden a los nodos de ingreso, y de egreso, respectivamente. El tráfico se genera para emplear la capacidad "casi" completa de la red. Cada fuente de una clase fluye de un nodo s_i a un nodo d_i). Se usa tráfico fijo en todo el tiempo experimental, o tráfico que cambia a los 10,000 y 15,000 *s*. Los requerimientos de retraso son de 15 y 12 *ms* para el tráfico de la clase más importante (que va del nodo s_0 al d_0), y de 30 a 35 *ms* para el tráfico de las otras clases. Existe un plan de precios y penalizaciones diferente para cada clase.

En la red representada en la Figura 7 se observa que hay interferencia entre rutas en el tráfico que pertenezca a una sola clase. En [17] no se estudia la problemática de la interferencia entre rutas, que se da en los dos nodos centrales, para los flujos de una sola clase.

En comparación, el Método STP, a presentar en este trabajo, considera que la repartición del ancho de banda no se da entre clases sino entre rutas que se interfieren, dentro de una clase. El Método STP asigna ancho de banda a través de ajustes dinámicos de los pesos de las colas, en cada interfaz de salida de cada nodo en donde dos o más rutas se cruzan. En el Método STP no hay autoridad central alguna que asigne ancho de banda o que evalúe una ganancia total: los nodos operan de forma autónoma cuando asignan ancho de banda.

En [60] se presenta un método llamado LAAP, que es muy similar al método RLAP presentado en [17], de hecho lo comparan compitiendo directamente con RLAP. Con LAAP se hace un aprovisionamiento de ancho de banda en los nodos de una red Diffserv, para las diversas clases en la red, para obtener mejor QoS en términos de pérdidas, retraso y Throughput. El aprovisionamiento se hace mediante la variación de pesos en los despachadores WFQ de los nodos, donde cada peso es proporcional al ancho de banda mínimo asegurado a cada clase. Con LAAP se tiene una función de ganancia muy similar a aquella de RLAP, y se tienen agentes en los nodos que operan, al menos en parte, de forma autónoma.

Se especifica que hay dos tipos de agentes. El primer tipo es el *agente que aprende*, el cual mide, en el nodo en donde se encuentre, los bits enviados y los paquetes perdidos en colas. El segundo tipo es el *agente contador*, colocado en alguno de los nodos de destino. Este agente lleva la cuenta de los paquetes que no cumplen con su retraso objetivo al llegar al nodo, y los intervalos de tiempo en donde los flujos no cumplen con el Throughput objetivo. Los *agentes contadores* envían reportes a los *agentes que aprenden*, los cuales calculan la ganancia y ajustan sus colas. Con esta última afirmación, no "parece" que en [60] se utilice una administración central que indique los valores de los pesos de los nodos (en este sentido LAAP sería diferente a RLAP). La topología de red que se utiliza para la simulación es básicamente igual a la que se usa en [17].

Un método referido como "de despachadores coordinados" CMS (Coordinated Multihop Scheduling) se presenta en [22], el cual ofrece un servicio garantizado, en términos de retraso de extremo a extremo en una red. Este método trata de proveer un ancho de banda equitativo para todos los flujos, de tal forma que cuando hay flujos que aumentan su tráfico fuera de su comportamiento normal (se "comportan mal") lo otros flujos quedan protegidos.

En este método, cualquier interfaz de salida de un nodo central tiene una cola para manejar todas las clases de la red. En el método, los nodos de la red modifican el índice de prioridad de cada paquete que reciben (el índice de un paquete está almacenado en su encabezado), que representa el tiempo elegible para que el paquete sea servido (atendido por el despachador). Cualquier paquete que fue servido tarde, en un nodo "*río arriba*", como consecuencia del exceso de tráfico en otras clases, o flujos, contendientes de ancho de banda en ese nodo, llega al nodo "*río abajo*" con un favorable índice de prioridad, en comparación con los paquetes de otros flujos contendiendo por el servicio, y que podrían haber sido servidos temprano, en sus respectivos nodos "*río arriba*".

Este método proporciona una coordinación natural entre los nodos, para lograr un comportamiento global, y en el mismo los nodos operan de forma autónoma. El método requiere un estado por flujo solamente en los bordes de la red, y tiene la complejidad de que requiere marcar cada paquete.

En [22] se generan evaluaciones con experimentos por simulación, en un escenario que incluye una red con una ruta larga (cruza los nodos 1 al nodo 6), y que es interferida por diversas rutas cortas. Se indica que los flujos cruzados son de otras clases. Los nodos intermedios usan una cola en su interfaz de salida. En los diversos experimentos por simulación, la ruta objetivo tiene desde 25 hasta 60 flujos, y las otras rutas generan otros 250 flujos de salida por cada nodo, para un total desde 275 hasta 310 flujos de salida por nodo. Se comparan los resultados del uso del Método CMS contra métodos como EDF (Erliest Deadline First) y WFQ.

Por su operación, el método está enfocado a proteger los flujos de rutas largas, así que en los experimentos, en lugar de aumentar el tráfico de las rutas de interferencia se aumenta el tráfico en la ruta larga y se observa el efecto de protección que hay sobre la misma.

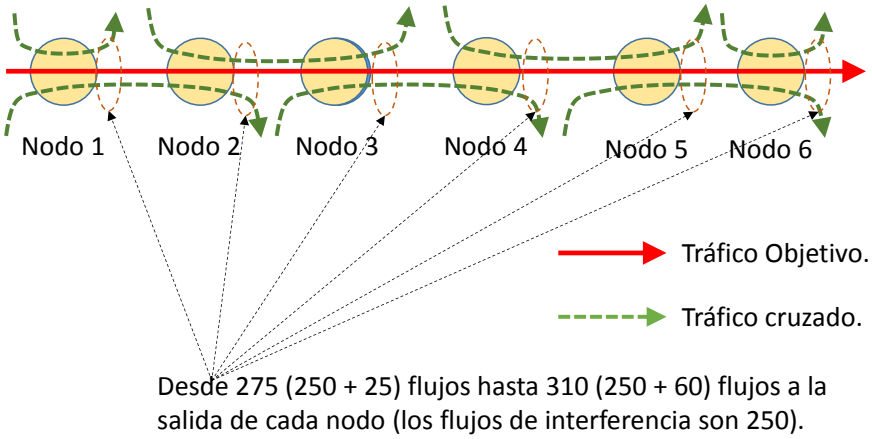


Figura 8. Red del escenario experimental para la evaluación del Método CMS en [22].

Asimismo, en las evaluaciones de este método se observa la idea de proteger a una ruta larga contra rutas interferentes cortas, y los resultados se observan en términos del retraso, en un percentil del 99%, en la ruta de interés (de inicio a fin) (ver Figura 9).

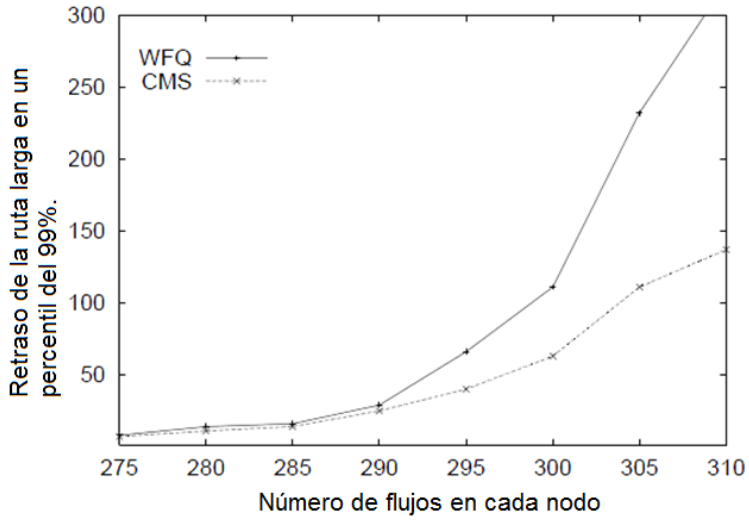


Figura 9. Forma de presentación de resultados en el escenario experimental de [22], para la evaluación del Método CMS y su comparación con otro método, WFQ.

Este método protege al flujo de la ruta larga al darle mayor prioridad en los nodos subsecuentes, pero por ello, los flujos de las rutas cruzadas, en los nodos subsecuentes, se verían afectados por dicho aumento de prioridad (ver Figura 10).

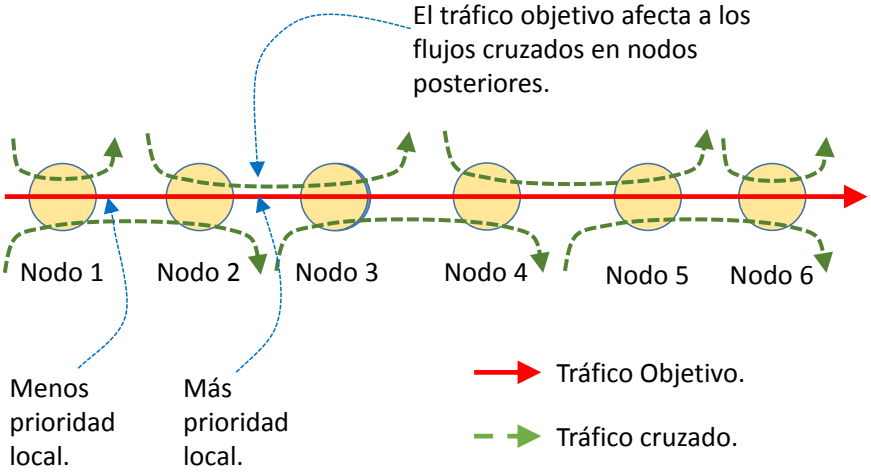


Figura 10. Red del escenario experimental para la evaluación del Método CMS en [22].

Este método es de especial interés en esta Tesis ya que puede ser utilizado para efectos de comparación, aplicado a proteger una ruta contra rutas de interferencia, dentro de una sola clase.

En [61] se propone un método de atención de tráfico llamado WFQ adaptivo con el que se pretende proveer un ancho de banda equitativo entre clases en una red que ofrece QoS, para mejorar la utilización del ancho de banda en la misma. En cada despachador, en las interfaces (de salida) de los nodos, este método ajusta el peso de la cola de cada clase, con base en la longitud media de la cola. Una clase obtiene más ancho de banda si el tamaño de su cola es más grande que el tamaño de las otras colas. El método es autónomo por nodo y dinámico, que se presta para esquemas en que se van admitiendo flujos. No observa restricciones de QoS globales (por operar autónomamente). No parece haber un control sobre la rapidez con la cual se hace este ajuste.

En [55] se propone un método que ajusta, de forma adaptable, los pesos de un despachador tipo "*Round Robin*" con ponderación de pesos, y que opera en cada nodo central de una red, para proteger al servicio "*Premium*" [62] absorbiendo los transitorios del Agregado de Comportamiento *Expedited-Forwarding*, en la arquitectura de Servicios Diferenciados. El peso de cada cola se ajusta de acuerdo al tamaño medio de la cola. Los nodos operan autónomamente y cooperan para obtener indicadores de QoS, para alcanzar un desempeño de baja tasa de pérdidas, bajo retardo, y bajo *Jitter*, para el servicio *Premium*.

En [63] se presenta un método cuyo propósito es garantizar límites de retraso en las diferentes clases y mantener equidad entre flujos dentro de cada clase. El método opera de acuerdo al estatus de retraso de la cola asociada a cada clase en la red. Este algoritmo se presta para esquemas en que se van admitiendo flujos y tiene dos partes. La primera parte incluye un algoritmo que opera autónomamente en cada nodo central en la red. El algoritmo cambia los pesos de un despachador WFQ cada vez que llega un paquete. Cada vez que llega un paquete a una cola, se calcula el tiempo de espera previsto para atender al paquete, y si ese tiempo es mayor que aquél calculado para el paquete anterior que llegó a la cola, entonces el algoritmo incrementa el valor del peso de la cola, para, con esto, mejorar los tiempos de atención en la cola. La segunda parte del método es un algoritmo de conformado de flujos, operando en la cola de cada clase, dentro de los nodos, para mantener la equidad entre los diferentes flujos.

Este algoritmo tiene una excesiva carga de trabajo, calculando los pesos al momento de la llegada de cada paquete y no observa restricciones de QoS globales (por operar autónomamente).

En [64] se presenta un sofisticado método de aprovisionamiento de ancho de banda, basado en medición, que tiene dos partes. La primera parte es el llamado algoritmo Auto Adaptivo, que opera autónomamente en cada nodo, midiendo las condiciones locales en el mismo y ajustando, en consecuencia, 1- los pesos de servicio dados a las clases por el despachador del nodo, y 2- los umbrales del tamaño de las colas, para la operación del proceso de desechado

de paquetes, para dar preferencia al cumplimiento de los límites de retardo, sobre el cumplimiento de los límites de pérdida de paquetes. Cuando este algoritmo observa que persistentemente se violan los umbrales locales de retraso y pérdidas, genera una alarma al algoritmo que es la segunda parte del método, el cual se llama Algoritmo de Aprovisionamiento de Núcleo, que es uno solo centralizado, que a su vez monitorea el tráfico en la red. Una de las reacciones de este algoritmo es reducir las tasas de llegada a la red, de las diversas clases, mediante el ajuste de los acondicionadores en los nodos de ingreso. Asimismo reasigna ancho de banda a las diversas clases en la red.

El algoritmo Auto Adaptivo trabaja con una cola por clase en cada interfaz de salida de los nodos, sin marcar paquetes, y puede operar autónomamente sin la operación del segundo algoritmo. Se indica que por estar distribuido en los enrutadores centrales, este algoritmo es escalable en configuraciones de redes grandes.

El método trata de cumplir umbrales de retraso y pérdidas, pero no trata de optimizar un objetivo de ganancia. Este método podría proteger a una ruta contra interferencias de otras dentro de clases.

En [65] se presenta un método llamado WFQ Dinámico, que opera en una red que atiende tráfico de diferentes clases, para tratar de mantener la relación de retrasos “contratada” (se refiere a contrato de servicio) entre las clases. El método tiene un algoritmo que opera autónomamente en cada nodo de la red, cambiando, en cada interfaz de salida, los pesos del despachador WFQ (que opera con una cola por clase). El algoritmo opera de tal forma que el peso de la cola de cada clase depende de la tasa de llegada (con una relación Afín) a la clase, con una evaluación que se realiza en periodos que oscilan entre 0.25 y 1 *s*.

Para evaluar se utiliza una red con 4 rutas prestablecidas, que se interfieren en un solo enlace de paso central. Como interesa evaluar los retrasos de paquetes, las capacidades en las colas son infinitas (no hay pérdidas por desborde). Los experimentos duran 500 *s* y los tráficos se mantienen durante cada experimento. En los resultados se indica que el método WFQ Dinámico mantiene las proporciones de retraso contratadas, de mejor forma que el método WFQ que utiliza pesos fijos. No se intenta mantener una QoS global.

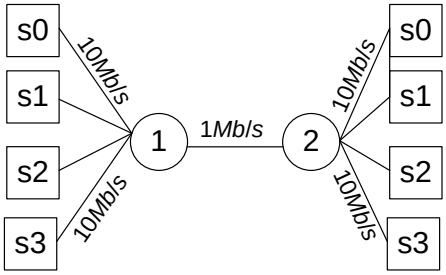


Figura 11. Red usada en el escenario de trabajo de [65] , para evaluación.

En [13] se presenta un método con el interés de mejorar la posibilidad de ofrecimiento de QoS. En este método, para el ingreso de flujos de alta prioridad (como flujos de voz), reserva un ancho de banda a cada flujo, en el nodo de frontera de forma similar a lo que hace IntServ (con el uso de RSVP). La diferencia en este caso es que los nodos centrales no requieran el conocimiento de cada reservación, como sucede en IntServ, sino que los nodos centrales funcionan exclusivamente en el ámbito DiffServ.

Para hacer esto, en el nodo de frontera, una vez establecida la reservación para los flujos de alta prioridad, se asocian estos flujos a una sola clase en DiffServ[54], en la cual se respeta la reservación de los flujos (se limita la cantidad de reservación para estos flujos para que no se aniquile el ancho de banda para los flujos de baja prioridad). Los flujos de baja prioridad pasan directamente a asociarse a otra u otras clases DiffServ.

Para las evaluaciones se utiliza un escenario con una red central que es un Dominio DiffServ, que tiene 4 nodos (dos centrales y dos de frontera). El nodo de entrada de frontera está conectado a tres conmutadores (Switches) en el lado transmisor. Cada Switch envía un tipo diferente de tráfico: el primero envía señales de voz que son de alta prioridad[55], el segundo y tercer Switches envían flujos de baja prioridad que reciben servicios tipo AF, y BE[56], respectivamente. Los flujos de estos Switches son de páginas con http[57], archivos con FTP[58]; enviadas, cada una, en espacios de tiempo iguales[59]. Los flujos siguen rutas predefinidas, y para cada clase hay interferencia en el enlace central del Dominio Diffserv.

[54] Se les escribe un mismo DSCP.
[55] Uno de los flujos es de telefonía usando la codificación G.729 A (con supresión de silencio) al que se otorga un ancho de banda por flujo de 36 *Kb*/*s* ó 48 *Kb*/*s*.
[56] Ver subcapítulos B.2.1 y B.2.2.
[57] Hypertext Transfer Protocol.
[58] FTP. File Transfer Protocol.
[59] No se indica si la transmisión es CBR ni detallan si usan TCP ni el tamaño de paquetes manejados. El tamaño máximo de las colas para los flujos de baja prioridad es de 100.
El retraso medio de punta a punta en el envío de flujos de archivos de baja prioridad tiene un promedio de retraso de aproximadamente 40 *ms*. Si se enviasen paquetes de 1000 *Bytes* para estos flujos de baja prioridad se tendría un retraso de transmisión en la ruta del cuello de botella de 1000 x 8 / 1544000 = 5.18 *ms*.

Se indica que esta propuesta no resuelve el problema de la escalabilidad que tiene IntServ.

El método propuesto se reporta con resultados favorables con relación al uso de DiffServ puro, poniendo al tráfico de voz en la clase *EF* con despachadores Priority Queueing[60].

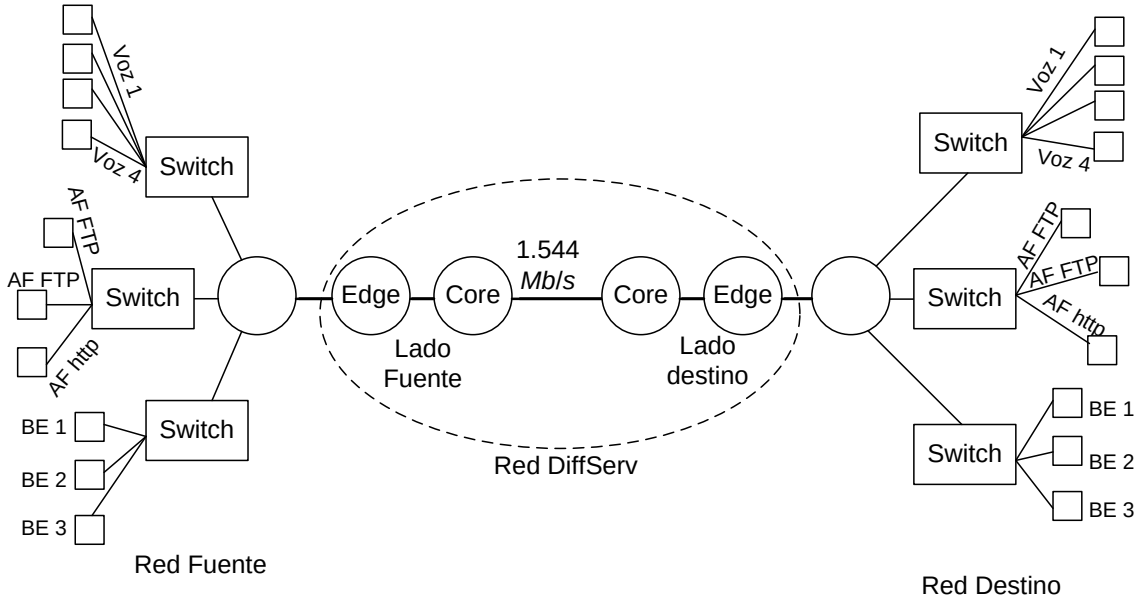


Figura 12. Red empleada en el escenario de trabajo de [13].
La red tiene una parte central que es un Dominio DiffServ, y los flujos van del lado fuente al lado destino. Se tiene un solo enlace central que es un cuello de botella. Los demás enlaces tienen 100 *Mb/s*.

En [9] se propone un algoritmo autónomo para diferenciar servicios en redes, en donde existen clases dadas por las características ya marcadas en los paquetes de los flujos en la red. Se indica que se trata de obtener equidad en el trato a los flujos (y mencionan que no hay un acuerdo para definir lo que es equidad en las redes).

Las características son: 1- Sensible o no sensible a retraso, 2- Prioridad dada. En cada nodo se proveen tres colas (en sus interfaces de salida). Para que un paquete que ha llegado a un nodo sea situado en una de esas colas, el paquete se clasifica (y éste no se remarca).

Algo que diferencia a este algoritmo de otros es que se clasifican los paquetes en forma probabilística (no de manera determinista como se haría normalmente con DiffServ) por el marcado del paquete. Además, también se involucra en esta clasificación el tamaño del paquete.

[60] Priority Queueing. Despachador en donde la clase de mayor prioridad siempre es atendida sobre las de menor prioridad, cuando tiene paquetes por enviar.

Un paquete sensible a retraso puede ponerse, probabilísticamente, en las colas 1 y 2 pero no en la 3 (la de menor prioridad). Un paquete no sensible a retraso puede ponerse, probabilísticamente, en las colas 2 y 3, pero no en la 1. Un paquete por su longitud puede ponerse probabilísticamente en las colas 2 o 3, con mayor probabilidad de ir a la cola 2 si es corto. Un paquete, según su prioridad dada, pueden ponerse, probabilísticamente, a las colas 2 o 3.

Se evalúa por simulación en una topología donde hay *N* rutas (de fuente *i* a destino *i*), que se interfieren en un enlace central, con un flujo por ruta. El enlace central es de 10 Mb/s (aquí el enlace central es más rápido que los exteriores que son de 1 Mb/s).

Se reporta mejora en desempeño en términos de Goodput[61] y retraso, para flujos sensibles a retraso, sin afectar notablemente a los flujos no sensibles a retraso, y en un índice de satisfacción de aplicación.

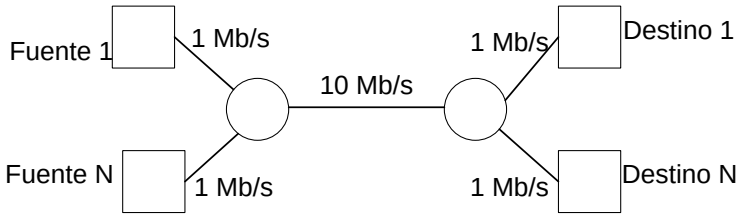


Figura 13. Topología de red usada en los experimentos por simulación realizados en **[9]**.
El número *N* significa el número máximo de flujos que se manejan, que puede ser hasta de 120. En la red cada flujo sigue una ruta y hay interferencia entre rutas en el enlace central.

En [6] se propone un modelo para otorgar QoS en una red DiffServ. La solución se propone solo en los nodos fuente, considerando que la salida de cada nodo fuente (hacia la red) le represente un cuello de botella hacia la red. Se pretende reducir las pérdidas y el retraso de paquetes, para el tráfico de tiempo real.

El nodo fuente tiene 3 clases: 1- *EF* (para tráfico de voz que se considera de tasa constante), 2- *AF* (para tráfico de video que se considera de tasa variable) y 3- *BE* (otro tráfico).

Se reparten recursos del enlace de salida mediante el peso de un despachador WRR. Los pesos calculados para *AF* son (ecuaciones similares se proponen para *EF* y *BE*):

$$q_{wAF}(t) = \frac{ql_{AF}\, xP_{AF}}{ql_{AF}(t) + ql_{EF}(t) + ql_{BE}(t)}$$

Donde P_{AF} es la prioridad inherente de *AF* (se usa 2 para *AF*, 3 para *EF* y 1 para BE), y $q_{\underline{wAF}}(t)$ es el peso calculado para la cola *AF* (además existe también un peso asociado a *AF* que se

[61] En las redes, el *GoodPut* es el *Throughput* a nivel de aplicación, es decir, el número de bits de información útiles que pasan de nodo fuente a nodo destino, sin considerar bits de protocolos.

designa con q_{wAF} tal que $K = q_{wAF} / \underline{q_{wAF}}(t)$ que indica la relación entre el peso realmente asignado y el calculado). Se prueban varios valores de *K* (mientras *K* es mayor la clase *AF* tiene menores pérdidas y retrasos)[62].

Asimismo, sí es interesante el significado de $ql_{AF}(t)$ y ql_{AF}. El primero es la longitud medida (sin especificar si es promedio ni en qué periodo de tiempo se evalúa) de la cola *AF*, y el segundo es la longitud que se necesita por *AF*, que se calcula con:

$$ql_{AF} = \frac{idr_{AF}\, xdly_{AF}\, xN_{AF}}{pktsz_{AF}}$$

Donde idr_{AF} es la tasa de llegada media por sesión de *AF*, dly_{AF} es el retraso permitido máximo, y N_{AF} es el número de sesiones que hay de *AF*. El numerador resulta ser el número promedio de paquetes en la cola, y la ecuación, en total, ofrecería el tamaño (en *Bytes* o *bits*) requerido en la cola (en promedio).

Se realizan pruebas emulando un solo nodo fuente, que tiene un enlace de salida de 2.1 *Mb*/*s*. Asimismo, se prueba, mediante simulación (con ns-2), usando 4 nodos fuente que comparten un enlace (de entrada a la red) de 8.4 *Mb*/*s* (2.1 *Mb*/*s* en promedio para cada nodo). Sobre el tráfico generado, se genera voz con codificación G723, y video con codificación H261.

Las pruebas ofrecen resultados de retraso y pérdidas de paquetes en el nodo, comparando el desempeño del tráfico *EF* contra el del tráfico *AF*. Asimismo se compara el desempeño del retraso en el tráfico *AF* para diversos valores del parámetro *K*.

En [18], [19] y [20] se presenta un método de ingeniería de tráfico "en línea" para "ir acomodando" en la red, y con garantías de ancho de banda, a cada nuevo flujo que solicita cruzar entre un nodo de ingreso y otro de egreso, por la red[63]. Dicha solicitud implica el encontrar una ruta indivisible (llamada "túnel de tráfico") [64] por la red, sin re enrutar los túneles de tráfico ya existentes, y buscando causar la mínima interferencia con los túneles de tráfico ya existentes.

La idea central del método es que para el acomodo de un nuevo túnel de tráfico se seleccione una ruta que interfiera en la menor medida con los otros pares de túneles ya acomodados en la red, buscando, así, una maximización en el acomodo de nuevos túneles.

[62] El tiempo de actualización de los pesos se menciona pero no se especifica.
[63] Se indica que los algoritmos "fuera de línea" crean reenrutamientos frecuentes, lo que no va de acuerdo a la forma de operación real de las redes.

[64] O como se le llama en dicha referencia: Label Switched Path (LSP) aludiendo a la terminología MPLS.

La interferencia se mide por la disminución de ancho de banda disponible en las rutas de la red. Se menciona que el manejo de métricas de QoS, como pérdidas y retraso, es computacionalmente complejo, por lo que se indica que lo más práctico es describir estas métricas en términos de un requerimiento de ancho de banda.

Para la admisión de un nuevo flujo se considera el ancho de banda "anunciado" por el mismo, en comparación con la red residual, que está compuesta por los enlaces cuyo ancho de banda disponible pueda aun alojar al nuevo flujo.

Se requiere emplear información centralizada sobre el estado (de ocupación) de los enlaces, donde esta ocupación proviene de los valores de ancho de banda anunciados por cada flujo que ha ingresado a la red. Lo anterior implica una forma de reservación de ancho de banda para cada flujo.

En la operación de este método se utiliza el algoritmo llamado Minimum Interference Routing Algorithm (MIRA[65]), que es una aproximación que simplifica los cálculos que se harían usando enrutamiento óptimo.

El algoritmo se corre en un lugar centralizado, pero se menciona que es deseable que se pueda correr de forma distribuida, en el nodo de ingreso de cada ruta, para no requerir centralización. Cada solicitud de ingreso llega a un servidor de rutas (directamente o a través del enrutador de ingreso) el cual determina, para cada solicitud, si hay recursos en la red y la ruta específica para el ingreso, y lleva cuenta de las capacidades disponibles registradas.

Una solicitud de ingreso puede ser rechazada si no se encuentra que en la red se tenga capacidad residual para dicho tamaño de tráfico.

Hay que notar que no es trivial hacer una relación entre ancho de banda y retraso, en una red, considerando, sobre todo, un retraso extremo a extremo entre fuente y destino. También, el acomodo de un nuevo túnel no asegura el mantener una métrica de QoS en la red, como lo serían pérdidas o retraso.

En el algoritmo MIRA para cada enlace de la red se calcula la tasa de disminución de capacidad de cada ruta existente en la red con la disminución de capacidad en ese enlace. Con la suma de estas razones se conforma un peso asignado al enlace. Este peso representa la importancia del enlace para mantener la capacidad de las rutas de la red. Entonces, mientras más peso tenga un enlace, más impacta la disminución de su capacidad en la capacidad de la red, por

[65] Sobre MIRA, en [21] dice que la minimización de retraso obtenida con ese método es lenta pero es más popular por el número de rechazo de solicitud de rutas que tiene.

lo que, para crear una nueva ruta, se pretende utilizar lo menos posible los enlaces de más peso. Ya con estos pesos, se utiliza el algoritmo de Dijkstra [66] para generar cada nueva ruta.

En la Figura 14 se representa la red del escenario de trabajo usada en [18], [19] y [20]. En los experimentos se busca el admitir flujos entre pares de nodos $s_i - d_i$, y se comparan los flujos que se pudieron admitir, en comparación con otros métodos. En la red se observa la existencia de las interferencias entre rutas.

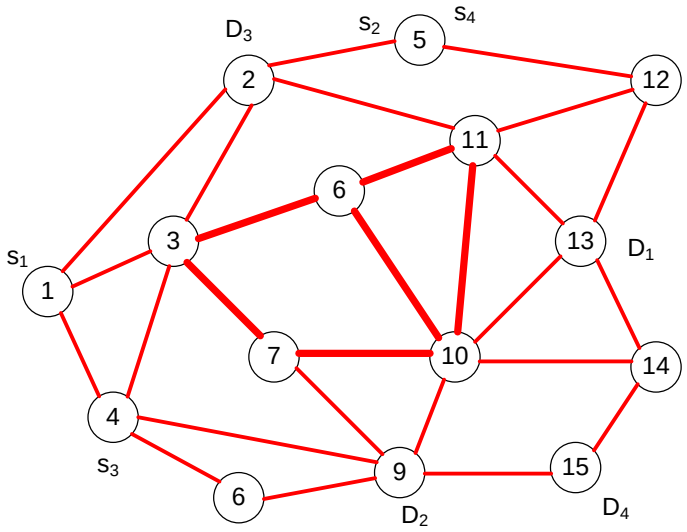


Figura 14. Topología de la red utilizada en el escenario de trabajo de **[18]**, **[19]** y **[20]**.

Un enfoque de QoS del que se tiene investigación activa actual, y que es interesante de revisar para comparar sus características con el problema a resolver en esta Tesis, es el llamado "Enrutamiento de QoS" (QoS Routing) [67] [68] [69], en el que se busca una ruta para cada flujo que solicita entrada a una red de topología fija, de un nodo de ingreso "*s*" a un nodo de egreso "*t*", tal que la misma satisfaga, simultáneamente, dos o más condiciones de QoS (como retraso, o costo, o confiabilidad). Este problema es un caso de ingeniería de tráfico en línea.

En [67] se consideran *K* condiciones de QoS, con $K \geq 2$, indicando que la resolución de este problema, con $K \geq 2$ es no polinomial (es "NP-Hard"). El problema se modela con grafos en donde cada condición de QoS se asocia a un peso en cada enlace, teniendo entonces *K* pesos ("ya dados") para cada enlace. La "distancia" de cada ruta se forma con la suma de los pesos de sus enlaces. Se propone el cumplimiento de "versiones de optimización" con las que se busca cumplir, con un margen de tolerancia, las condiciones de QoS originales. Se proponen diversos algoritmos con los que se busca cumplir las diversas versiones de optimización propuestas, que obtienen rutas que se garantiza que satisfacen la condición con la aproximación dada. Uno de los algoritmos obtiene, como peso de los enlaces, el peso máximo de cada enlace, dividido entre un parámetro fijo *W*. Ya con un peso por enlace se obtiene la ruta más

corta usando algoritmos tradicionales de búsqueda, como el de Dijkstra, o el de Bellman-Ford [66]. En las evaluaciones de ese trabajo el lograr el acomodo del mayor número de flujos en las rutas cumpliendo la QoS esperada para ellos, y tratando de causar el menor impacto en la red, en términos de QoS (al buscar ingresar un flujo en la ruta de menor distancia). Asimismo, es de interés la complejidad y el tiempo de corrida de los algoritmos. En las verificaciones no se genera tráfico, sino se utilizan algoritmos de enrutamiento.

No se indica cómo se haría la asociación inicial entre los pesos y las características de QoS en la red. Al buscarse una ruta para cada "solicitud de ingreso", este enfoque no incluye el concepto de admisión, en el sentido de que no observa el impacto del ingreso en la red, incluyendo que las rutas creadas pueden tener interferencias.

Resumen del Capítulo

Este capítulo presenta trabajos relacionados que presentan métodos para la mejora del desempeño o la solución de problemas, en redes. Todos los enfoques son de ingeniería de tráfico en línea. En algunos de estos métodos se observa la situación de interferencia de tráfico, en las topologías que presentan en sus escenarios de experimentación. Uno de estos trabajos aborda esta situación como tema central, otros dos de manera indirecta.

La influencia de esos trabajos, en esta Tesis, es la siguiente:

- El otorgar a los pesos de las colas, en las interfaces de salida de los nodos centrales, un valor proporcional al tamaño promedio de las colas (cuando se maneja una cola por clase), en [55] y [61].
- La manera en que se miden los retrasos en la red, en [22].
- La topología de red con admisión por rutas, en [15] y [16].
- La existencia de interferencia entre rutas, en los escenarios de experimentación presentados, en [15] [16], [17] y [55].
- El problema de la interferencia entre rutas, como objetivo del trabajo, en: [18] [19] [20], o como componente fundamental de la problemática abordada, en [22] y [67].
- La protección de una ruta larga contra la interferencia de rutas cruzadas cortas, en [22].
- El cambio de pesos dinámico en los pesos de despachadores, para mejorar las características de eficiencia o resolver problemas detectados en las redes, en [17], [55], [61], [63] y [64].

Además, se denota la importancia del tema sobre la interferencia entre rutas, la presentación de métodos para obtención de rutas en redes, para dar entrada a flujos minimizando el impacto en la interferencia a rutas en la red, o en la QoS otorgada en la red, como [18], [19] [20] y [67].

En la Tabla 1 se presentan características relevantes con relación a los métodos de atención del tráfico, presentados en los trabajos relacionados. Se incluye, al final de la tabla, el Método STP, a presentar como solución en esta Tesis.

Tabla 1. Comparación de características relevantes de los métodos de atención de tráfico en los trabajos relacionados, y el Método STP.

	No. colas / interfaz salida	¿Busca optimizar algún uso de la red?	Rutas en red	Considera impacto a rutas cruzadas	Qué busca el método	Tipo de algoritmo
[15] [16]	Una. No hay diferenciación del tráfico.	Si, por ruta.	Prestablecidas.	No. Solo mencionan el problema, y refieren a una posible solución.	Predecir la QoS al admitir a flujos, por ruta, para preservar la QoS en la misma. Busca asegurar la QoS en términos de retraso de flujos en las rutas.	En el nodo de egreso de cada ruta. La operación de cada ruta es independiente de las demás. Solamente se requiere medir características de tráfico en ese nodo. Marca paquetes.
[17]	Una por clase.	Sí, de la red.	No necesariamente, pero se utilizan en las evaluaciones por simulación.	No.	Maximizar la ganancia de la red (mayor utilización y menores penalizaciones). No observa el aseguramiento de QoS en la red.	En un administrador central y en agentes en cada nodo, operando autónomamente en cuando a sus mediciones, pero el administrador central les indica los pesos de las colas.
[60]	Una por clase.	Si, de la red.	No necesariamente, pero se utilizan en las evaluaciones por simulación.	No.	Maximizar la ganancia de la red (mayor utilización y menores penalizaciones). No observa el aseguramiento de QoS en la red.	Agentes operando en cada nodo, para mediciones. Para establecer los pesos de las colas hay interacción entre agentes sin intervención de un administrador central.

	No. colas / interfaz salida	¿Busca optimizar algún uso de la red?	Rutas en red	Considera impacto a rutas cruzadas	Qué busca el método	Tipo de algoritmo
[18] [19] [20]	1	Si. De la red.	Sí. Obtiene ruta para cada ingreso de flujo.	Sí.	Maximizar el uso de la red buscando el minimizar la interferencia entre rutas al admitir más tráfico. Da apoyo a la preservación de QoS optimizando el uso del ancho de banda en la red (aunque lleva cuentas de ancho de banda reservado y no el actual).	En un lugar centralizado. Lleva la cuenta del ancho de banda reservado para cada enlace de la red.
[55]	Una por clase.	No. Complemento para apoyar preservación de QoS.	No necesariamente, pero se utilizan en las evaluaciones.	No.	Proteger al servicio Premium (que trabaja en una red DiffServ con el agregado de comportamiento “Expedited Forwarding”). No asegura la QoS en la red.	Autónoma en cada nodo.
[22]	Una	No.	Prestablecidas.	Más bien de flujos que se cruzan.	Proteger a flujos preexistentes en la red contra flujos que aumenten su tráfico por encima de su comportamiento habitual (el problema se da en los cruces de rutas). Tiende a proteger mejor a flujos en rutas largas. No observa el aseguramiento de QoS en la red, pero es un apoyo para eso.	Autónoma en cada nodo. Existe una coordinación natural entre los nodos, pero no requiere haber comunicación entre los mismos. Marca paquetes.
[61]	Una por clase.	No. Complemento para mejorar eficiencia en uso de red y apoyar preservación de QoS.	No necesariamente, pero se utilizan en las evaluaciones.	No.	Proveer un ancho de banda que mencionan como “equitativo” entre clases en los nodos para apoyar a preservar la QoS en la red. No asegura la QoS en la red.	En cada nodo.
[63]	Una por clase.	No. Complemento para apoyar preservación de QoS.	No necesariamente, pero se utilizan en las evaluaciones.	No.	Mantener el retraso de cada clase dentro de un umbral. Mejorar le equidad entre flujos dentro de cada clase.	Autónoma en cada nodo.

	No. colas / interfaz salida	¿Busca optimizar algún uso de la red?	Rutas en red	Considera impacto a rutas cruzadas	Qué busca el método	Tipo de algoritmo
[64]	Una por clase.	No. Complemento para apoyar preservación de QoS.	No necesariamente, pero se utilizan en las evaluaciones.	No, pero podría aliviar el problema.	Operar manteniendo los umbrales, primero de retraso, y luego de pérdidas en la red.	Doble. Una parte autónoma en cada nodo y la otra centralizada.
[65]	Una por clase.	No.	No necesariamente, pero se utilizan en las evaluaciones.	No.	Mantener la relación de retrasos “contratada”, entre clases.	Autónomo en cada nodo.
[13]	Una por clase (una clase tiene internamente reservaciones).	No.	No necesariamente, pero se utilizan en las evaluaciones.	No.	Proteger el ancho de banda de flujos de alta prioridad, integrando reservación para ellos en una clase de DiffServ, y aprovechar la simplicidad y escalabilidad de DiffServ para llevar los flujos en la red mayor.	Opera en los nodos de frontera del Dominio DiffServ.
[9]	3. La asignación de paquetes en las diversas colas es probabilística.	No.	No necesariamente, pero se utilizan en las evaluaciones.	No.	Busca equidad y mejorar tratamiento a flujos sensibles a retraso. No asegura la preservación de la QoS en la red	Autónomo por nodo. Introduce la clasificación probabilística, y se considera, para clasificar paquetes, tanto la marca que tienen como su tamaño.
[6]	3	No.	No. Ni siquiera se plantea una red.	No pues no se plantea una red.	Reducir las pérdidas y retraso de tráfico de alta prioridad en el nodo de entrada a una red. El algoritmo plantea la adecuada repartición de ancho de banda de salida de los nodos fuente, entre diversas clases.	Autónomo en los nodos de entrada de la red. No considera lo que pueda pasar en la red.
[67]	Una total.	Sí.	Obtiene ruta para cada ingreso de un flujo.	De manera indirecta.	Ingresar la mayor cantidad de flujos en la red cumpliendo con la QoS mínima esperada para ellos. No asegura la preservación de la QoS en la red al ingresar nuevos flujos.	Centralizado.

	No. colas / interfaz salida	¿Busca optimizar algún uso de la red?	Rutas en red	Considera impacto a rutas cruzadas	Qué busca el método	Tipo de algoritmo
STP	Una por interfaz de llegada.	No. Complemento para mejorar eficiencia en uso de red y apoyar preservación de QoS.	Prestablecidas.	Sí.	A corto plazo protege a las rutas de una red contra el aumento de tráfico por rutas interferentes. A largo plazo permite la competencia entre rutas que se interfieren. No observa el aseguramiento de QoS en la red, pero es un apoyo para eso.	Autónomo en cada nodo.

Tabla 1. Comparación de características relevantes de los métodos de atención de tráfico en los trabajos relacionados.

Capítulo 4. Características y Condiciones del Escenario de Trabajo de Esta Tesis

Introducción al Capítulo

En este capítulo se presenta el escenario de trabajo de esta Tesis, que se utiliza para los experimentos, por medio de simulación, con los que se evalúan los diversos métodos de atención de tráfico, incluyendo al que se plantea en esta Tesis, como a otros métodos alternativos en los que se incluye el Método q1. Las características de este escenario incluyen la topología de la red y las condiciones de tráfico en la misma.

Este capítulo también presenta el simulador empleado, y sus características.

4.1 Topología de la Red del Escenario de Trabajo

La topología de la red del escenario de trabajo, para los experimentos de esta Tesis se muestra en la Figura 15. Esta topología es una red de tamaño limitado, similar a la encontrada en [22] (Figura 4). La red tiene tres nodos centrales, c_0, c_1 y c_2, y ocho nodos de frontera, e_1, ..., e_8. En esta red los nodos centrales son los únicos nodos en donde las rutas se interfieren, así que los nodos centrales son los nodos de interferencia. Más específicamente, las interfaces de salida donde está la interferencia, de estos nodos centrales, están en los enlaces: $c_0 - c_1$, $c_1 - c_2$ y $c_2 - e_8$. Todas las otras interfaces de los nodos centrales y las interfaces de los nodos de frontera usan solamente una cola en sus interfaces de salida.

Fuera de los límites de la red hay siete nodos fuente, s_1, ..., s_7, y siete nodos destino, d_1, ..., d_7. Todas las interfaces de salida de estos nodos tienen una cola.

En esta topología de red, hay siete rutas, $s_1 - d_1$, $s_2 - d_2$, ... $s_7 - d_7$. La ruta $s_1 - d_1$, la cual es la más larga de todas, atraviesa 7 nodos, s_1, e_1, c_0, c_1, c_2, e_8 y d_1. Las otras rutas son más pequeñas, en términos de nodos atravesados. Por ejemplo, la ruta $s_2 - d_2$ atraviesa 6 nodos, s_2, e_2, c_0, c_1, e_5 y d_2, y la ruta $s_3 - d_3$ atraviesa 6 nodos, s_3, e_3, c_0, c_1, e_4 y d_3. Estas tres rutas se interfieren en la interfaz de salida del nodo central c_0 (que se conecta con el nodo central c_1).

Las rutas $s_1 - d_1$, $s_4 - d_4$ y $s_5 - d_5$ se interfieren en la interfaz de salida del nodo central c_1 (que se conecta con el nodo c_2). Finalmente, las rutas $s_1 - d_1$, $s_6 - d_6$ y $s_7 - d_7$ se interfieren en la interfaz de salida del nodo central c_2 (que se conecta con el nodo e_8). No hay otras interferencias en esta red.

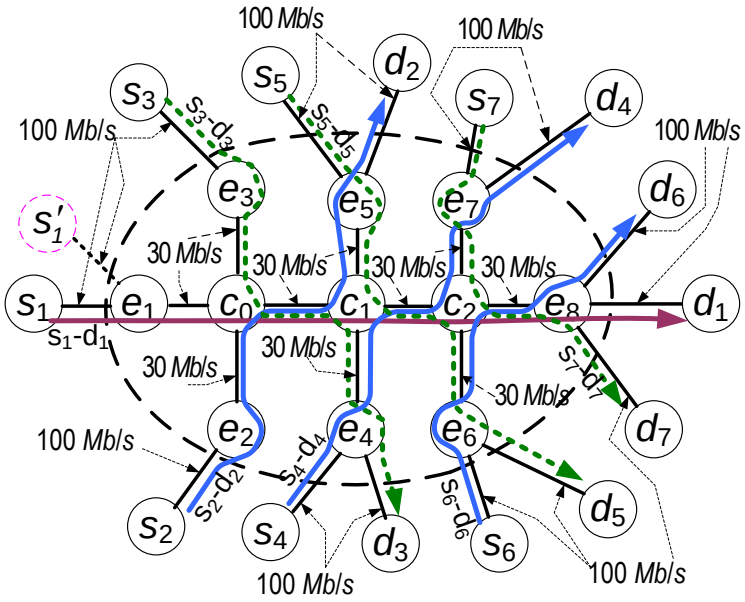


Figura 15. Topología de la red del escenario de prueba de esta Tesis.
La red cuenta con tres nodos centrales, c_i, y ocho nodos de frontera, e_i. Afuera de la red hay nodos fuente, s_i, y nodos destino d_i. La red es atravesada por siete rutas, las que se interfieren en grupos de tres.

Solamente la ruta $s_1 - d_1$ se interfiere con dos rutas, tres veces, esto es, a la interfaz de salida de cada uno de los tres nodos centrales. Todas las demás rutas se interfieren con otras dos rutas, en la interfaz de salida de solamente un nodo central.

Con el propósito de poder llevar a cabo los experimentos por simulación, los enlaces centrales de la red (aquellos que unen nodos dentro de la red) tienen un ancho de banda de 30 *Mb/s*[66]. Los enlaces que conectan los nodos de frontera de la red con los nodos exteriores a la red tienen un ancho de banda de 100 *Mb/s*. Cada enlace tiene 0.05 *ms*[67] de retardo de propagación.

[66] El uso de tasas mayores en con el simulador ns-2, cuando se guardan las trazas de resultados, para su análisis, produce archivos de resultados muy grandes que dificultan su manipulación. En el caso de esta Tesis, ya filtrando los resultados, para que tuviesen solamente la información requerida para poder calcular los retrasos y ganancias deseadas, se obtuvieron archivos de 1 *GByte* por simulación, así, un solo experimento que promedia 10 simulaciones usa 10 *GBytes* de disco.

[67] Por ejemplo, la velocidad de las ondas electromagnéticas, propagándose en una fibra óptica puede ser de aproximadamente 1.927 x 10^8 *m/s* (el índice de refracción n_1 del núcleo de la fibra óptica tiene un valor de aproximadamente 1.557 y la velocidad en el núcleo es igual a 3x10^8 *m/s* / n_1. Considerando enlaces de 8 *Km*,

La Figura 15 también muestra la ruta $s_1' - d_1$. Si se utilizara esta ruta, la misma no podría ser distinguida de la ruta $s_1 - d_1$, en los nodos c_0, c_1 y c_2, dado que ambas rutas entran en la red por el nodo e_1. En la interfaz de salida del nodo e_1 estas dos rutas podrían distinguirse, una de la otra.

4.2 Condiciones del Tráfico Experimental

En cada experimento de esta Tesis, a lo largo de todo el tiempo del experimento cada flujo sigue la misma ruta. Sólo hay un flujo considerado para cada ruta, así que a los paquetes de un flujo se les refiere como los paquetes de la ruta correspondiente.

El resultado de cada experimento es el retraso calculado de los paquetes que pasan a través de sus respectivas rutas (éste es un retraso de extremo a extremo). El resultado de retraso obtenido para cada ruta es el percentil 98 del retraso medido, es decir, éste es el retraso de los paquetes que se encuentran en el límite del 2% de paquetes más retrasados al atravesar la ruta. Este retraso se calcula para cada intervalo de 120 *s*, dentro de la duración del experimento. Los cálculos de retraso utilizan los archivos de trazas, obtenidos a partir de las pruebas realizadas (ver Apéndice C.8).

El resultado obtenido de cada experimento es el promedio de los resultados de 10 simulaciones, lo que se refiere como el resultado del experimento[68].

En los experimentos, el flujo se crea a partir del tráfico generado por las "Fuentes de tráfico" ("fuentes"). Una fuente es un lugar donde el tráfico se genera, en ns-2. Una fuente está conectada a un "nodo fuente" (como cualquiera de los nodos fuente, s_i, de la Figura 15), y entrega su tráfico a la red mediante el nodo fuente.

En el escenario de trabajo se utilizó tráfico MPEG4, con uso del servicio de transporte UDP. Para este fin, el algoritmo para generar este tipo de tráfico [70] se añadió a ns-2 (ver Apéndice K, Pág. 359). La tasa promedio, con los ajustes seleccionados para este algoritmo, resultaron

---

para una fibra monomodal, el retraso de propagación que se tiene en los mismos es de aproximadamente 41.5 $\mu s \approx 0.05\ ms$.

[68] El resultado de cada experimento es el valor medio de los resultados de 10 simulaciones. Esta es una media estimada. Con el fin de obtener el intervalo de confianza para indicar la fiabilidad de la estimación, se utilizó el método del percentil Bootstrap de Intervalos de Confianza, con 1000 diferentes re-muestreos de los 10 resultados de simulación. Este método ha tenido éxito con una amplia gama de distribuciones de probabilidad [93]. El método indicó que, con una probabilidad del 90%, el valor medio real se encuentra dentro de una ventana situada desde 6,67% por debajo hasta 6,99% por encima del valor medio estimado (ver Apéndice I).

de 0.621 *Mb/s* para cada fuente (ver Apéndice L). Las longitudes de los paquetes generados variaron, pero generalmente la longitud fue de 1000 *Bytes*.

En comparación con los nuevos métodos de generación de tráfico, que utilizan los bancos de pruebas [71], el método de generación de tráfico en esta Tesis podría ser considerado como restrictivo, en cuanto a su falta de precisión en sus tiempos de generación entre paquetes, y también en términos de su falta de capacidad de respuesta con respecto a los cambios en las condiciones de la red, incluyendo el tráfico de la red, sin embargo, los siguientes puntos pueden tenerse en cuenta:

- Dado que las memorias intermedias (*Buffers*), de los nodos utilizados en los experimentos de esta Tesis, son grandes (400 paquetes), para evitar la pérdida de paquetes[69] como resultado de la saturación de cola[70], los resultados en estos experimentos no deberían ser sensibles a ligeras desviaciones en los patrones de generación de paquetes. Las escalas de tiempo utilizadas en los experimentos de esta Tesis son, más bien, escalas para planificación de capacidad.
- La limitación, en la falta de respuesta indicada, se maneja, en esta Tesis, haciendo experimentos diversos, todos ellos iniciando con las mismas condiciones del tráfico. Cada experimento tiene 1560(*s*) de tiempo de duración de experimento, y en el tiempo 600 *s*, hay un cambio en las condiciones del tráfico, para probar la reacción de los métodos probados. Este cambio en las condiciones de tráfico se produce cuando la ruta $s_2 - d_2$ incrementa su tráfico en el tiempo experimental de 600 *s*.

En los experimentos, el tráfico inicial en cada ruta es lo suficientemente grande como para que el incremento de retraso en las rutas, causado por el incremento de tráfico en la ruta $s_2 - d_2$, se pueda observar en los resultados de los experimentos. La selección de los valores de tráfico para los experimentos se hace considerando lo siguiente:

- Se utiliza un tráfico de 36 fuentes para pasar por cada uno de los enlaces de salida de los nodos de interferencia. Para el caso del nodo c_0, se tendrían 12 fuentes por cada ruta s_1-d_1, s_2-d_2 y s_3-d_3. Con esto se ocuparía un 74.5% de los 30 *Mb/s* del enlace de salida del nodo[71]. En estas condiciones, con un aumento de 2 fuentes (el 5.5% del tráfico de las tres rutas) en el tráfico de la ruta s_2-d_2 (la ruta interferente), ya se observa, experimentalmente, un aumento en cerca de 3 *ms* en el retraso en la cola asociada a la ruta s_1-d_1 (la ruta interferida).

[69] Hay menos del 2% de paquetes perdidos en cualquier caso.
[70] En [65], por ejemplo, consideran tamaños de Buffers infinitos con el mismo propósito.
[71] Cada fuente emite en promedio una tasa de 0.621 *Mb/s*, así que el tráfico total fluyendo por la interfaz de salida del nodo c_0 tiene en promedio (12 + 12 + 12) x 0.621 *Mb/s* = 22.356 *Mb/s*.

- Desde 0 *s* a 600 *s* del tiempo de cada experimento, el flujo de cada ruta tiene 12 fuentes. En el tiempo 600 *s* hay un cambio en las condiciones de tráfico porque el flujo de la ruta $s_2 - d_2$ aumenta su número de fuentes. El propósito de este incremento es de observar su impacto en el retraso en las otras rutas de la red. Con este incremento solamente la interfaz de salida del nodo c_0 (hacia c_1) vería un aumento de tráfico en promedio, desde los 600 *s* del tiempo experimental.

Con el propósito de que la validación de los experimentos sea correcta, las condiciones experimentales, es decir, las situaciones de tráfico y de topología de los experimentos, son las mismas para todos los métodos probados para atención de tráfico. En la mayoría de los experimentos de esta Tesis se seleccionan cuatro rutas para mostrar su retraso. Estas rutas son: $s_1 - d_1$, $s_2 - d_2$, $s_3 - d_3$ y $s_4 - d_4$.

Hay una excepción en las condiciones experimentales para un caso en uno de los métodos probados, como se explica más adelante.

Es importante indicar que en cada interfaz de entrada, en los nodos de interferencia de la red, se lleva el tráfico de solo una ruta que se interfiere en ese nodo. Es por eso que a cada cola de cada una de las interfaces de salida, donde está la interferencia, se le refiere como "la cola de la ruta" específica. Debe quedar claro que esta simplificación de nombres puede no ser válida en otros casos generales.

4.3 Herramientas para la Simulación

Todas las evaluaciones en esta Tesis se hicieron mediante experimentación por simulación, usando el simulador de redes ns-2 [72] [73] [74], sobre el cual se presenta una introducción en el Apéndice C. En el simulador se emplearon herramientas y configuraciones similares a las que se usan en las propuestas posteriores (subcapítulos 3.2, 3.3 y 4).

Este simulador se empleó con sus herramientas "DiffServ" [75], empleadas para implantar la característica operativa del Método STP en donde en cada interfaz de salida de un nodo central existe una cola de espera por cada interfaz de entrada del nodo.

A las herramientas DiffServ mencionadas se les añadió la posibilidad de usar el despachador WFQ, según se hace en [76], incluyendo mejoras de implementación en el código C++ (hechas por el autor de esta Tesis), como se indica en el subcapítulo 7.3.2.

La razón de utilizar el despachador WFQ es que es aquél considerado una "excelente" aproximación [40] a un despachador ideal de flujos ideal que puede asegurar a cada cola un ancho de banda proporcional al peso de la cola, como se indica en el subcapítulo 2.1.4.

A la codificación del despachador WFQ se le añaden, en esta Tesis, posibilidades de realizar cambios dinámicos de pesos del despachador. Este algoritmo, aunque es producto de esta Tesis, no se considera como una parte fundamental aquí, porque ya existen otros trabajos reportados usando despachadores WFQ que operan con pesos variables.

La modificación del simulador ns-2 para considerar la operación del despachador WFQ con pesos que cambian según las longitudes de las colas requirió una cuidadosa revisión del código en C++ de dicho simulador, entre la que se observó que había tres archivos que se requerían modificar: dsred.h, dsred.cc y wfq-list.h (este último no estaba incluido en el simulador ns-2 sino que se trajo de la solución de [76]). El Apéndice E presenta el algoritmo de operación del despachador WFQ para operar con pesos variables.

Resumen del Capítulo

Con propósitos de una correcta validación de los resultados de los experimentos por simulación, las condiciones de tráfico y de topología de los experimentos son las mismas para todos los métodos probados para atención de tráfico. En la red hay una ruta larga y seis rutas cortas. Las longitudes se dan en términos de los nodos cruzados. La ruta larga es la única que atraviesa los tres nodos centrales de la red. En cada uno de estos tres nodos centrales se interfiere la ruta larga con dos rutas cortas.

En los experimentos a realizar, una o más de las rutas cortas pueden aumentar su tráfico en un tiempo experimental dado, y con esto se observa la reacción, en términos del retraso, en el tráfico de cada una de las rutas de la red.

Las evaluaciones en esta Tesis se hacen mediante experimentación por simulación. El simulador empleado en este trabajo es el ns-2, con adecuaciones para integrar funcionalidades de interés para esta Tesis, como el uso del despachador WFQ.

Asimismo, se hacen adecuaciones al simulador, para que los pesos de las colas, en el despachador WFQ, cambien conforme al algoritmo del Método STP.

Capítulo 5. Impacto del Problema de Estudio

Introducción al Capítulo

En este capítulo se plantea con detalle el impacto del problema de la interferencia entre rutas en una red, bajo el Método q1, en el escenario de trabajo de esta Tesis. Con el Método q1, haciendo referencia al concepto "flujo en el nodo", explicado en el Capítulo 2, ningún flujo en el nodo tiene asegurada porción alguna del ancho de banda de salida del nodo.

Para los experimentos presentados en este capítulo se utiliza el escenario de trabajo de esta Tesis, bajo las condiciones indicadas en el Capítulo 4, para observar ahí la afectación en diversas rutas de la red, debido al aumento de tráfico de una ruta interferente (o cruzada), en términos del retraso en las rutas de la red.

En este capítulo se incluye un caso de estudio donde se evalúa el retraso observado con el uso del Método q1, en términos de una de un indicador de ganancia, para una situación establecida de máximo retraso aceptable en la red.

Este capítulo se limita a mostrar los resultados para cada caso, revisando las ventajas y desventajas en cada método de atención al tráfico. Se deja para el 0, el hacer una comparación de los resultados de este capítulo, con los resultados de la aplicación del Método STP.

Nota. Se considera que la admisión en las rutas funciona bien. No se implementa algoritmo de admisión alguna en las evaluaciones que se hacen en esta Tesis[72].

5.1 Notas sobre las Etiquetas de las Figuras

A menos que se indique otra cosa, las 7 rutas (s_1-d_1 a s_7-d_7) inician los experimentos teniendo 12 fuentes cada una. Para simplificar las figuras, esta condición de inicio no se indica en las figuras.

Con relación a las etiquetas de las figuras, en todas las gráficas de esta tesis se utilizan nomenclaturas comunes. Ejemplos de estas nomenclaturas son:

- La etiqueta $\begin{bmatrix} 2s_2 \\ 0s_4 \\ 0s_6 \end{bmatrix}$ significa que en el tiempo experimental de 600 *s* se aumentan 2 fuentes en la ruta s_2-d_2, 0 fuentes en la ruta s_4-d_4, y 0 fuentes en la ruta s_6-d_6.

[72] En los diversos trabajos referidos en esta Tesis tampoco se implementa control de admisión. De hecho en [6] se indica explícitamente que se asume que hay un mecanismo de control de admisión presente en operación antes de la entrada de flujos a la red que ahí se estudia.

- La etiqueta STP P=0.10 significa que los nodos centrales, c_0, c_1 y c_2 están trabajando con el Método STP con el valor del parámetro *P* igual a 0.10.
- La etiqueta q1. Significa que los nodos centrales, c_0, c_1 y c_2 están trabajando con el método q1.
- La etiqueta s_1 se refiere a la ruta s_1-d_1.
- La etiqueta s_1-c_0 se refiere a la cola asociada a la ruta s_1-d_1. el nodo c_0 (en la (en la interfaz de salida hacia el nodo c_1).

Las demás etiquetas tienen similar significado a los ejemplos anteriores.

Al decir "retraso en el nodo c_0" se sobreentiende que es el retraso en su interfaz de salida hacia el nodo c_1.

5.2 Retrasos en Rutas cuando una Ruta Interferente Aumenta su Tráfico

En las figuras que se dan a continuación se observa la afectación en diversas rutas de la red, en términos del aumento de retraso, debido al aumento de tráfico, a los 600 *s* de tiempo experimental, en la ruta s_2-d_2, o simultáneamente en las rutas s_2-d_2, s_4-d_4 y s_6-d_6.

Recordamos que cada fuente envía un promedio de 0.621 *Mb*/*s* (ver Apéndice L), y que al inicio del experimento en enlace de salida del nodo c_0 (hacia el nodo c_1) tiene 12 x 3 = 36 fuentes (equivalente a 12 x 3 x 0.621 / 30 = 0.7452 o el 74.52% de utilización del enlace).

La Figura 16 muestra los retrasos de rutas debido al aumento de tráfico, a los 600 *s* de tiempo experimental en la ruta s_2-d_2. La diferencia de uno a otro experimento es el número de fuentes incrementadas en dicha ruta, en el tiempo 600 *s*. Los incrementos son: 2, 4 y 6 fuentes, para cada experimento, en el tiempo 600 *s*.

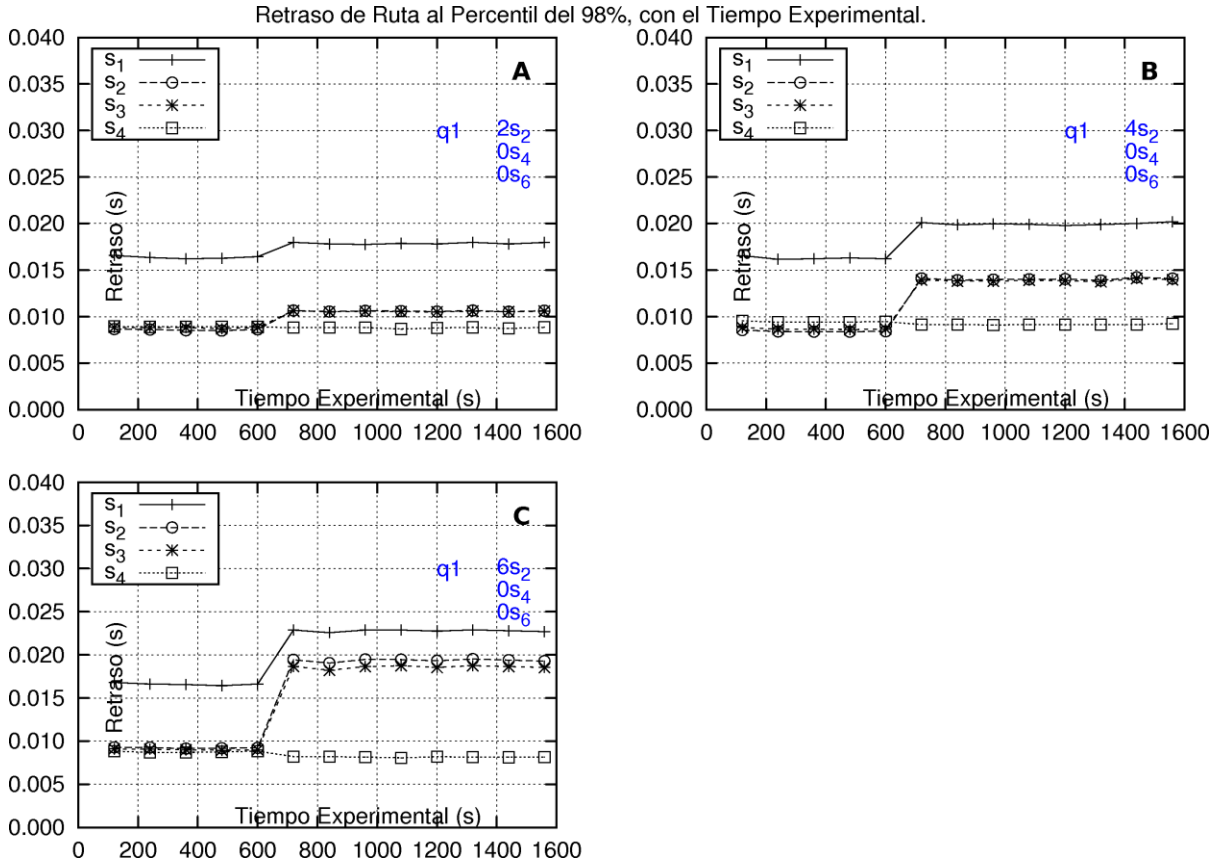


Figura 16. Retrasos en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_4-d_4 con la aplicación del Método q1 para la atención del tráfico, cuando una ruta interferente aumenta su tráfico.
En los cuadros "A", "B" y "C" la ruta s_2-d_2 aumenta de tráfico en 2, 4 y 6 fuentes, respectivamente, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

En la figura se observa lo siguiente:

- En el tiempo 720 *s* ya hay un incremento de retraso en cada una de las rutas $s_1 - d_1$, $s_2 - d_2$ y $s_3 - d_3$, que depende del aumento de tráfico en la ruta $s_2 - d_2$. El retraso en la ruta $s_1 - d_1$ es más grande que el retraso en cada una de las otras rutas, dado que la ruta $s_1 - d_1$ es la más larga de todas (en términos de nodos cruzados). Cada retraso de ruta proviene, principalmente, del retraso en la interfaz de salida en el nodo central c_0 (yendo al nodo central c_1). Todas estas rutas sufren del mismo retraso en esta interfaz de salida porque los paquetes de estas rutas son colocados en la misma cola única de esta interfaz. Se puede notar que las líneas correspondientes a los retrasos de las rutas $s_2 - d_2$ y $s_3 - d_3$ prácticamente se traslapan.
- El retraso de la ruta $s_4 - d_4$ no se afecta por el incremento de tráfico de la ruta $s_2 - d_2$ porque estas dos rutas no se interfieren.

- El aumento de tráfico en la ruta s_2-d_2 afecta a la misma ruta, y a las rutas s_1-d_1 y s_3-d_3 que le cruzan en el nodo c_0. Siempre el retraso en la ruta s_1-d_1 es mayor. Se observa también que el retraso en la ruta s_4-d_4 no se afecta pues la misma no pasa por el nodo c_0 y no se cruza con la ruta s_2-d_2.
- Conforme aumenta el tráfico en la ruta s_2-d_2 el retraso se acerca entre dicha ruta y la ruta s_1-d_1. Esto se debe a que las colas tienen un comportamiento de crecimiento y decremento en lapsos. Después de los 600 *s*, al aumentar el tráfico de la ruta s_2-d_2, la cola en c_0 tiende a crecer más que antes, en sus lapsos de crecimiento, y eso retrasa más que antes, en dichos lapsos, a los paquetes de la ruta s_1-d_1. Por lo anterior, también en dichos lapsos, se causa que decrezcan las colas en los nodos c_1 y c_2, a donde se dirigen los paquetes de la ruta s_1-d_1, así que cuando esos paquetes llegan a las colas de los nodos c_1 y c_2, pasan más rápido.

Retrasos en el nodo c_0

En la Figura 17 se presentan los retrasos dentro del nodo c_0, para las tres rutas que se cruzan en dicho nodo, s_1-d_1, s_2-d_2 y s_3-d_3. Los cuadros A y B muestran casos seleccionados donde la ruta s_2-d_2 aumenta su tráfico, a los 600 *s* del tiempo experimental, en 4 y en 10 fuentes, respectivamente.

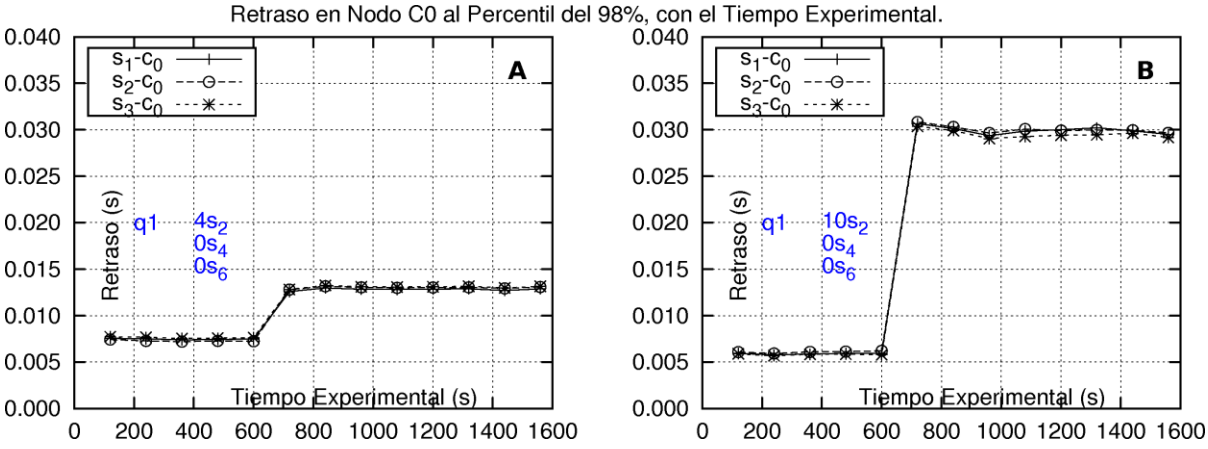


Figura 17. Retrasos en las colas de las rutas s_1-d_1, s_2-d_2, s_3-d_3, en el nodo c_0, con la aplicación del Método q1 para la atención del tráfico.
En los cuadros "A" y "B" la ruta s_2-d_2 aumenta de tráfico en 4 y 10 fuentes, respectivamente, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Se observa que los retrasos en las colas de las tres rutas s_1-d_1, s_2-d_2 y s_3-d_3 son iguales, aun cuando después de 600 *s* el tráfico de s_2-d_2 aumenta y es mayor que el de las otras dos rutas.

5.3 Retrasos en Rutas cuando más de una Ruta Interferente Aumenta su Tráfico

La Figura 18 muestra el retraso en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_4-d_4. El cuadro "A" de la figura muestra el caso seleccionado donde, a los 600 *s* del tiempo experimental, la ruta s_2-d_2 aumenta su tráfico en 4 fuentes. El cuadro "B" de la figura muestra el caso extremo en donde las rutas s_2-d_2, s_4-d_4 y s_6-d_6 aumentan su tráfico en 4 fuentes, a los 600 *s*, simultáneamente.

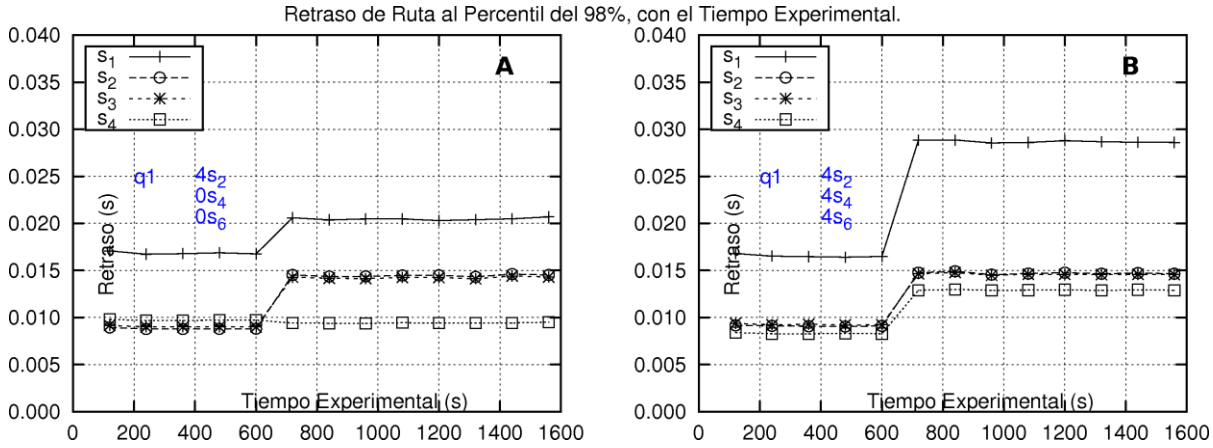


Figura 18. Retraso en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_4-d_4 con la aplicación del Método q1 para la atención del tráfico, cuando solo una, y cuando más de una ruta interferente aumentan su tráfico.
En el cuadro "A" la ruta s_2-d_2 aumenta de tráfico en 4 fuentes, y en el cuadro "B" las rutas s_2-d_2, s_4-d_4 y s_6-d_6 aumentan su tráfico simultáneamente, en 4 fuentes cada una, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

En el cuadro "A" de la Figura 18 se observa que el aumento de tráfico en la ruta s_2-d_2 afecta a la misma ruta, y a las rutas s_1-d_1 y s_3-d_3 que le cruzan en el nodo c_0, mientras que la ruta s_4-d_4 no sufre aumento de retraso pues esta ruta no pasa por el nodo c_0.

En el cuadro "B" de la Figura 18 se observa que el aumento simultáneo de tráfico en las rutas s_2-d_2, s_4-d_4 y s_6-d_6 afecta a la ruta s_1-d_1 de manera más intensa (en lugar de llegar a 20 *ms* ahora el retraso llega a 29 *ms*). Se observa que las rutas s_2-d_2 y s_3-d_3 resultan igualmente afectadas que cuando solamente la ruta s_2-d_2 aumentó su tráfico. Ahora se observa que la ruta s_4-d_4 también sufre de aumento de retraso debido a su propio aumento de tráfico.

5.3.1 Problema Observado

El problema de la aplicación del Método q1 es que la ruta interferente, en este caso la ruta s_2-d_2, al hacer una admisión independiente de tráfico, dentro de la forma distribuida, por ruta, de admisión en la red, "no sabe" sobre la afectación que causa al hacer una admisión, sobre las rutas cruzadas s_1-d_1 y s_3-d_3. En especial la ruta s_1-d_1 puede verse más afectada pues es la

ruta más larga, en términos de nodos cruzados, en la red, y por lo tanto es propensa a tener el mayor retraso en la red. A esta ruta hay que observarla para evaluar el impacto del aumento de las rutas que le cruzan (que le interfieren).

5.4 Caso de Estudio. Análisis de Ganancias con la Aplicación del Método q1

Para analizar qué pasa en un escenario en donde hay un límite máximo, y único, de retraso permitido en cualquier ruta en la red, se selecciona un límite de 19 *ms* y cuando solamente una de las rutas cruzadas aumenta su tráfico, en este caso la ruta s_2-d_2. Se observan las condiciones de retraso con relación a dicho límite, de las tres rutas que se cruzan en el nodo c_0.

El cuadro "A" de la Figura 18 se repite en la Figura 19, con una línea que indica el límite de retraso de 19 *ms*.

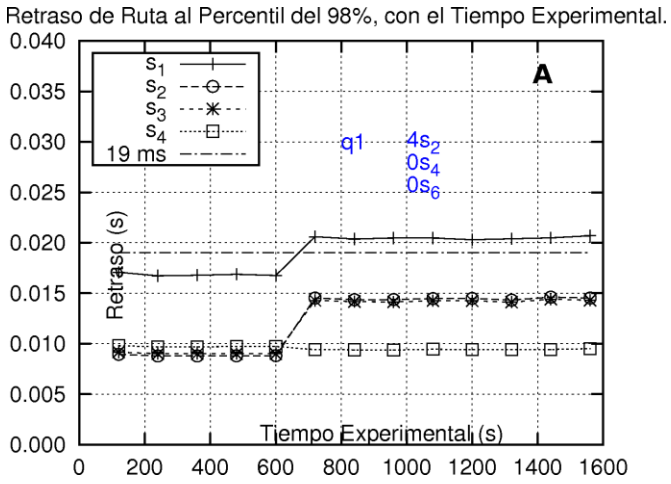


Figura 19. Retraso en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_4-d_4 con la aplicación del Método q1 para la atención del tráfico. Este es el mismo cuadro que el cuadro "A" de la Figura 18, pero incluye una línea que indica el límite de retraso de 19 *ms*.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

En la ruta s_2-d_2, antes del aumento de tráfico en la misma, de 4 fuentes, se tiene un retraso de 9 *ms*. El aumento indicado se da a los 600 *s*, y a los 720 *s* se observa que el retraso en la ruta es de 14.5 *ms*. Por este valor, que es menor a 19 *ms*, se considera que dicha ruta permitiría el aumento de este número de fuentes. Es importante mencionar que es posible que algunos paquetes de la ruta s_2-d_2 sí sobrepasen los 19 *ms* pero serían menos del 2%.

Con este aumento de fuentes, el retraso en la ruta s_1-d_1 pasa de 17 a 20.5 *ms*, lo que sobrepasa el límite indicado, sin que la ruta s_2-d_2 "se percate". La ruta s_3-d_3 sufre un aumento de retraso similar a la ruta s_2-d_2.

5.4.1 Ganancia

Para cada ruta se utiliza un indicador de ganancia al que se suma 1 *punto* por cada paquete que cursa la ruta en un tiempo menor al límite máximo establecido, y restan 10 *puntos*, como una penalización, por cada paquete que cruza en un tiempo mayor a dicho límite. Esta forma se toma de la propuesta de [17]. Estos puntos pueden transformarse en dinero, es decir, esta ganancia se puede transformar en ganancia económica. Así, la finalidad de utilizar esta ganancia es establecer un indicador de la ventaja económica de aumentar el tráfico en la red.

En la Figura 20 se observan las ganancias por ruta, para las rutas s_1-d_1, s_2-d_2, s_3-d_3, y la ganancia total (considerando el promedio de las tres rutas indicadas). Los cuadros “A” y “B” muestran las ganancias por ruta, y total, respectivamente, cuando la ruta s_2-d_2 aumenta 2 fuentes a los 600 *s*. Los cuadros “C” y “D” son similares a los anteriores, pero cuando la ruta s_2-d_2 aumenta 4 fuentes.

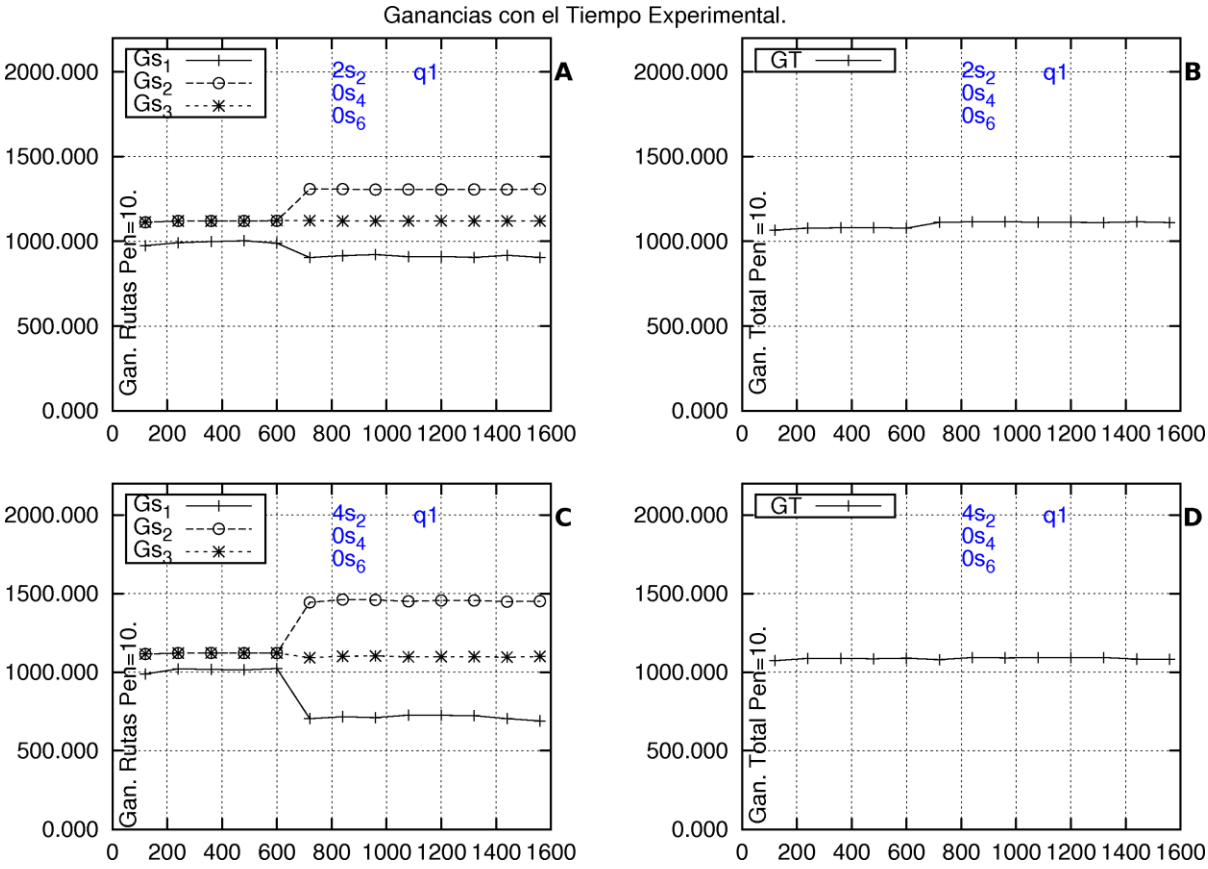


Figura 20. Ganancias en las rutas s_1-d_1, s_2-d_2 y s_3-d_3, y ganancias totales (ganancias promedio), cuando una fuente aumenta su tráfico.
En los cuadros "A" y "B" la ruta s_2-d_2 aumenta de tráfico en 2 fuentes, a los 600 *s* de tiempo experimental. Estos cuadros muestran las ganancias, por ruta y totales, respectivamente.
Los cuadros "C" y "D" son similares a los cuadros "A" y "B", respectivamente, pero para un aumento de 4 fuentes en la ruta s_2-d_2.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Para el caso de la penalización 5, a partir de los 720 *s* se observa una pérdida para la ruta s_1-d_1, y se observa una ganancia donde no se aprecia cambio para la ruta s_3-d_3. La ganancia total aumenta debido a que el aumento de la ganancia de la ruta s_2-d_2 domina.

A partir de los 720 *s* se observa que se tiene una pérdida de ganancia para la ruta s_1-d_1. La ganancia total se mantiene aproximadamente.

Dado que la ruta s_1-d_1 tiene otros puntos de interferencia, si hubiese un aumento simultáneo de tráfico en otras rutas cruzadas, además del de la ruta s_2-d_2, la ruta s_1-d_1 tendría una pérdida más pronunciada con relación a lo observado en la Figura 20 (como se muestra en el 0).

Resumen y Conclusiones del Capítulo

Sobre los resultados de la aplicación del Método q1, se resume lo siguiente:

En una red con tráfico por rutas, con admisión independiente por ruta, con el uso del Método q1 en los nodos, y donde existe un solo límite de retraso máximo para cursar por las rutas, cuando una ruta aumenta su tráfico, ésta no puede conocer la afectación que causa a las rutas que le cruzan. En este caso, una ruta larga (aquella que cruza más nodos) es vulnerable a la interferencia de las rutas cortas que la cruzan.

Con el Método q1, el hecho de que exista admisión por ruta no permite a una ruta aumentar su tráfico más allá de una cantidad, lo que promueve la protección a las rutas cruzadas. Mientras más parecida sea la *longitud* de las rutas en una red, y el tráfico sea equilibrado en ellas, el algoritmo de admisión de cada ruta será más efectivo para proteger a las rutas cruzadas.

Con el Método q1 operando en una red, se tiene una operación sencilla al utilizar solo una cola por cada interfaz de salida de los nodos de la red.

En la Figura 17 se observa que el retraso en una cola, al tener una utilización de menos del 75% de su ancho de banda, es menor a 8 *ms*[73]

Se verifica que los retrasos de todos los paquetes que llegan a una única cola de la interfaz de salida de un nodo, no importa de qué ruta provengan, y cuánto tráfico haya por ruta, son iguales.

El autor de esta tesis tiene la hipótesis de que el Método q1 se podría mezclar con un método que centralizara la información sobre el porcentaje de utilización de los enlaces en la red, para sujetar la utilización de los enlaces a menos del 75.5%, y con esto lograr contrarrestar los efectos de las interferencias entre rutas, sin requerir conocer cuales rutas interfieren a cuáles otras. El análisis de esta hipótesis va más allá de los alcances de esta tesis.

[73] En los resultados experimentales de otros trabajos que presentan métodos de atención de tráfico para observar QoS, se puede apreciar, en sus gráficas de resultados, que los indicadores de QoS que manejan mantienen valores bajos para utilizaciones bajas del ancho de banda. Por ejemplo, en [61] y [64] se tienen pérdidas 100 veces menores al máximo que reportan o retrasos de la mitad o menos del máximo que reportan, con utilizaciones de ancho de banda menores al 65%.

Capítulo 6. Aplicación de 2 Métodos de Atención de Tráfico Alternativos

Introducción al Capítulo

Este capítulo muestra los resultados de dos métodos de atención de tráfico, diferentes al Método q1, que son aplicados en el escenario de trabajo, para tener un punto de comparación con el desempeño de estos métodos.

Los resultados obtenidos ya no se comparan con aquellos del Método STP, debido a las desventajas que presentan, y que se explican en este capítulo.

En el primer método presentado en este capítulo, en cada interfaz de salida de los nodos centrales de la red, en donde se puede tener interferencia entre rutas, se tiene una cola por cada interfaz de entrada al nodo, y cada cola tiene un peso fijo. Con el uso de este método, cada flujo en el nodo tiene una porción mínima asegurada que es fija, del ancho de banda de la interfaz de salida. A este método se le llama q3f. Este método tiene similitudes con el Método STP, a presentarse más adelante, con la diferencia fundamental de que los pesos de las colas, en aquel método no son fijas, sino que se mueven de acuerdo a un algoritmo.

Se aprovecha la aplicación del Método q3f para observar los retrasos que hay en las rutas cuando una ruta interferente aumenta su tráfico, en el caso en que cada ruta inicia el experimento con un tráfico del 50% con relación al que se maneja en los demás experimentos de esta Tesis.

El segundo método presentado en este capítulo es la aplicación del método llamado Método del Servicio de Despacho de Multisalto Coordinado[74] (*Coordinated Multi-hop Scheduling Service* -CMS).

Este método puede aplicarse a la protección de tráfico en rutas cruzadas. En esta aplicación, si el tráfico de una ruta es retrasado, como consecuencia del incremento de tráfico de una ruta cruzada (interferente), el flujo de la ruta afectada adquiere mayor prioridad en el siguiente nodo de interferencia por donde pasa esa ruta, permitiendo al flujo “recuperarse”, en términos de retraso. Con esto se pretende proteger el retraso de las rutas, contra rutas interferentes que aumentan de tráfico.

Este último método se presenta bajo dos escenarios de condiciones de tráfico en la red. El primero tiene las mismas condiciones que el escenario de trabajo. Dado que se observó que con la aplicación de este método, y con estas condiciones, un flujo que incrementa su tráfico no daña a los flujos en rutas interferidas, se generó, además, un segundo escenario con con-

[74] Ver el Apéndice J para la explicación de las adecuaciones a ns-2 para insertar la operación de este método con la aplicación especial que aquí se presenta.

diciones de tráfico diferentes, utilizadas solamente para este segundo escenario, para observar cuándo un flujo protegido por el método, puede causar problemas en otros flujos en la red.

6.1 Resultados con la Aplicación del Método q3f

En este método de atención de tráfico, en cada interfaz de salida de los nodos centrales de la red, en donde se puede tener interferencia entre rutas, se tiene una cola por cada interfaz de entrada al nodo, y cada cola tiene un peso fijo. Con el uso de este método, cada flujo en el nodo tiene una porción mínima asegurada que es fija, del ancho de banda de la interfaz de salida.

6.1.1 Retrasos en Rutas cuando una Ruta Interferente Aumenta su Tráfico

En la Figura 21 se observa la afectación en diversas rutas de la red, en términos del aumento de retraso, debido al aumento de tráfico, a los 600 *s* de tiempo experimental, en la ruta s_2-d_2, o en las rutas s_2-d_2, s_4-d_4 y s_6-d_6.

La Figura 21 muestra los retrasos de rutas debido al aumento de tráfico, a los 600 *s* de tiempo experimental en la ruta s_2-d_2. La diferencia de uno a otro experimento es el número de fuentes incrementadas en la ruta $s_2 - d_2$, en el tiempo 600 *s*. Los incrementos son: 2, 4 y 6 fuentes, para cada experimento, en el tiempo 600 *s*.

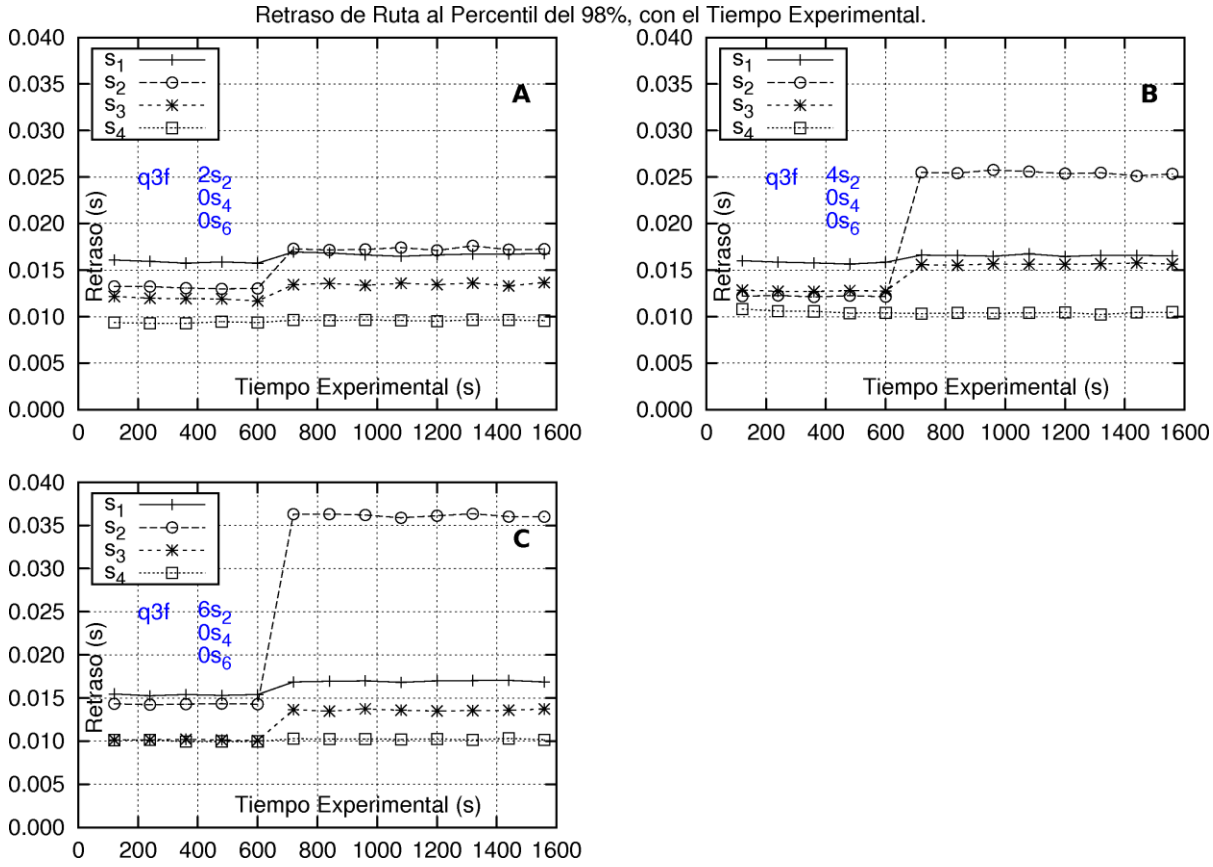


Figura 21. Retrasos en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_4-d_4, con la aplicación del Método q3f para la atención de tráfico, cuando se tiene una ruta interferente que incrementa su tráfico.
En los cuadros "A", "B" y "C" la ruta s_2-d_2 aumenta de tráfico en 2, 4 y 6 fuentes, respectivamente, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Con la aplicación de este método en la interfaz de salida de cada nodo central, c_0, c_1 y c_2, hay tres colas, una para cada una de las tres interfaces de entrada del nodo. Las tres colas tienen el mismo peso, que es fijo, y por normalización los pesos suman 1.

Se observa que, como resultado del incremento de tráfico en la ruta $s_2 - d_2$, en el tiempo 720 *s* hay un incremento de retraso en esta ruta; el retraso en la ruta $s_1 - d_1$ y el retraso en la ruta $s_3 - d_3$ son "pequeños" (de hasta 17 *ms*) con relación al retraso que se observa de la ruta $s_2 - d_2$ (con un aumento de 4 fuentes el retraso es mayor a 25 *ms*), y el retraso en la ruta $s_4 - d_4$ no varía. La ruta s_2-d_2 tiene menos oportunidades de aprovechar el ancho de banda del sistema de colas, con relación a lo que puede hacer con el uso del Método q1.

En este caso, los pesos fijos de las colas de las rutas $s_1 - d_1$ y $s_3 - d_3$, en la interfaz de salida del Nodo central c_0, han protegido a estas rutas del incremento de tráfico en la ruta $s_2 - d_2$.

La Figura 21 muestra el retraso sufrido por la ruta $s_2 - d_2$ al aumentar de tráfico, que es mayor en comparación con el retraso mostrado cuando se usa el Método q1 (referirse a la Figura 16 del Capítulo 5).

Cuando el tráfico de las rutas es menor, como se explica a continuación con los resultados de la Figura 22, la ruta s_2-d_2 sí puede aumentar su tráfico, teniendo un aumento de menos de 3 *ms* en su retraso.

Con el empleo del Método q3f, en la interfaz de salida de un nodo, no se puede proteger al tráfico de una ruta cuando éste supera el límite (fijo) de la proporción de ancho de banda asegurada para la ruta, en dicha interfaz.

6.1.2 Retrasos en Rutas cuando una ruta Interferente Aumenta su Tráfico, y se opera con un Tráfico total Menor en un 50%

Como una justificación del porqué en la red del escenario de trabajo de esta Tesis se utiliza un tráfico inicial del 75.5%[75] de la capacidad de los enlaces centrales de la red, se hacen las siguientes observaciones sobre los retrasos obtenidos cuando se utiliza la red con un tráfico significativamente menor.

- La Figura 22 muestra los retrasos en las colas en el nodo c_0, asociadas a las rutas s_1-d_1, s_2-d_2, s_3-d_3, debido al aumento de tráfico, a los 600 *s* de tiempo experimental en la ruta s_2-d_2. El aumento llega a 16 fuentes. En el cuadro "A" todas las rutas empiezan teniendo un tráfico de 6 fuentes cada una (el 50% de lo manejado en todos los demás experimentos en esta Tesis). En el cuadro "B" las rutas empiezan teniendo un tráfico de 12 fuentes cada una (como se maneja en los experimentos de esta Tesis).
- La cola de la ruta s_2-d_2 , después de su aumento llega a 16 flujos tanto para el cuadro "A", como el cuadro "B" de la figura, pero en el cuadro "A" para el tráfico de la ruta s_2-d_2 se puede aprovechar el ancho de banda disponible que pueda existir de las otras rutas, en el enlace. Por lo anterior, a los 720 *s* se observa un retraso de apenas 5.2 *ms*, mientras que en el cuadro "B" el retraso es de 22.8 *ms*. Cabe hacer notar que si en el cuadro "A" se estuviese operando con el Método q1, la posibilidad de aprovechamiento del ancho de banda del enlace es mayor, para una ruta que aumenta su tráfico, por lo que se esperaría un aumento menor de retraso (menor a 5.2 *ms*).

Por lo anterior, el autor de esta tesis seleccionó una utilización de 74.5% del ancho de banda del enlace del nodo c_0 (yendo hacia c_1) para percibir un aumento de retraso "significativo", al

[75] Según lo indicado en el Capítulo 4, dicho valor se obtiene de 12 x 3 x 0.621 / 30 = 0.755.

aumentar el tráfico en la ruta s_2-d_2, de 4 fuentes. Después del aumento dicha utilización llega a 82.8% del enlace.

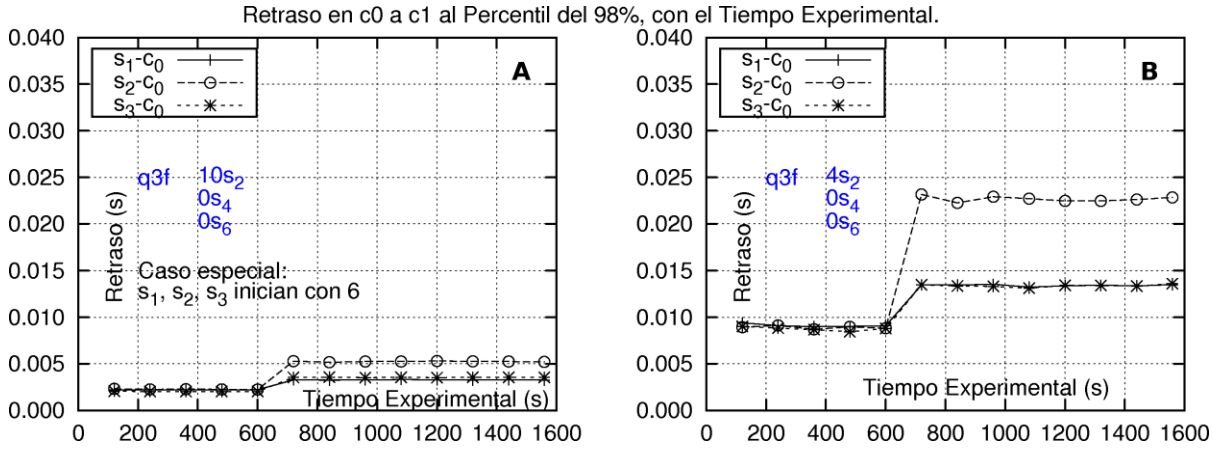


Figura 22. Retraso en las colas de las rutas s_1-d_1, s_2-d_2 y s_3-d_3, en el nodo c_0, con la aplicación del Método q3f, para la atención de tráfico, cuando se tiene una ruta interferente que incrementa su tráfico. Se hace una comparación cuando el tráfico es del 50% inicial a lo manejado en los experimentos de esta Tesis.
El tráfico inicial experimental en el cuadro "A" (de 6 fuentes en cada una de las 3 rutas) es el 50% del aquél en el cuadro "B" (de 12 fuentes en cada una de las 3 rutas). A los 600 *s* la ruta s_2-d_2 aumenta su tráfico en 10 y 4 fuentes, respectivamente en cada cuadro, para llegar, en ambos casos, a 16 fuentes.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

6.1.3 Observación de los Retrasos en las Colas del Nodo c_0 cuando los Pesos de las colas son Diferentes, y cada Peso es Igual al Tráfico Relativo en las Colas

Este subcapítulo 6.1.3 se incluye para mostrar un aspecto de interés para el método de solución presentado en esta Tesis, para el cual es de conveniencia la utilización del Método q3f. Este aspecto se refiere a la evaluación si en las tres colas del nodo c_0 (en la interfaz de salida yendo hacia c_1) se tiene el mismo retraso cuando el despachador las atiende con un peso igual a la proporción del tráfico relativo que llega a las mismas.

La Figura 23 muestra los retrasos en las colas de las rutas s_1-d_1, s_2-d_2 y s_3-d_3, en el nodo c_0, con la aplicación del Método q3f, cuando los pesos de las colas son diferentes en el experimento, pero cada peso es proporcional al tráfico relativo de las rutas. Dichos pesos y tráfico se dan en la Tabla 2. No hay aumentos del tráfico promedio, en ruta alguna, durante el experimento.

Flujos	Pesos	Trá-fico	Tráfico Relativo
s1-d1	12/40	12	12/40
s2-d2	16/40	16	16/40
s3-d3	12/40	12	12/40
Suma	1	40	1

Tabla 2. Valores de los pesos de las colas en el nodo c_0, y del tráfico, en las rutas s_1-d_1, s_2-d_2 y s_3-d_3, fijos a lo largo del experimento cuyos resultados se reportan en la Figura 23.

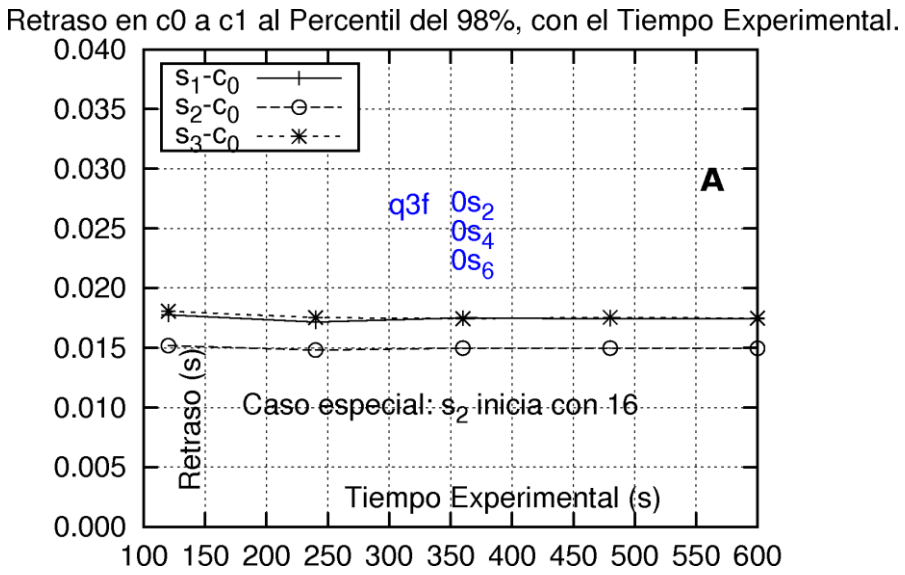


Figura 23. Retrasos en las colas de las rutas s_1-d_1, s_2-d_2 y s_3-d_3, en el nodo c_0, con la aplicación del Método q3f, para la atención del tráfico, cuando los pesos de las colas son proporcionales al tráfico relativo de las rutas. No hay aumentos de tráfico en el experimento.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Se observa que el retraso en el flujo de la ruta s_2-d_2 es de 15 *ms* y los de los flujos de las rutas s_1-d_1 y s_3-d_3 son de 17.5 *ms*.

Una de las razones de esto es que el flujo mayor (de la ruta s_2-d_2) puede aprovechar mejor el ancho de banda de las colas de los otros flujos, que tienen menos tráfico.

6.2 Aplicación del Método CMS

El Método CMS fue presentado en [22] y se presenta como un método alternativo en esta Tesis, para compararlo.

En la explicación de este método, el término “retraso de flujo” es más adecuado que el término “retraso en ruta” (aun cuando haya un flujo por ruta), debido al hecho de que este método se orienta muy específicamente a los flujos en la red.

El Método CMS es una variante, conservadora de energía, del servicio CJVC (*Core-Stateless Jitter Virtual Clock*). Este método fue presentado en [22] para proteger a los flujos de una clase contra el incremento de tráfico de otros flujos, en una red.

Con este método, en el caso en donde un flujo sea retrasado, como consecuencia del incremento de tráfico de otro flujo que comparte la misma interfaz de salida de un nodo central, entonces, el método ayuda al flujo afectado dándole una mayor prioridad en el siguiente nodo central por donde pase el flujo afectado, permitiéndole “recuperarse” (como lo indica [22]), en términos de retraso. Con lo anterior, este método trata de proteger al flujo con respecto a un aumento en su retraso total, a lo largo de su ruta completa.

En esta Tesis, este método se aplica en el escenario de trabajo, para proteger flujos en la misma clase, contra el incremento de tráfico de flujos en rutas de interferencia (el método puede ser aplicado en otras formas como para proteger a tráficos de diferentes clases que compiten por ancho de banda).

Una explicación general de la forma de operación implantada se describe como sigue. En el nodo de ingreso, el método almacena, en el encabezado de cada paquete, un índice de prioridad el cual se usa por el despachador de salida del nodo, para servir al paquete (para despacharlo) (mientras menor sea el valor del índice de prioridad, mayor será la prioridad). *El valor de este índice de prioridad depende del tiempo en que el paquete llega al nodo de ingreso* (mientras menor sea el tiempo, menor será el valor del índice). Entonces, cuando el paquete llega a un nodo central, ubicado “río abajo”, ese nodo utiliza el índice de prioridad del paquete para recalcular un nuevo índice de prioridad para ese paquete (básicamente añadiendo el tiempo máximo que podría requerirse para retransmitir el paquete), el cual se almacena, nuevamente, en el paquete. Entonces, el paquete espera a ser servido en una cola específica, correspondiente al flujo del paquete, ubicada en la interfaz de salida del nodo. El despachador de la interfaz de salida sirve a aquel paquete que tenga el menor valor de índice de prioridad (mayor prioridad), de entre aquellos paquetes que esperan en la primera posición de cada cola de la interfaz.

Específicamente, la ecuación (1) muestra el índice de prioridad, $d_{i,j}^{k}$, del paquete número *k*, del flujo *i*, en el nodo *j*, donde t_i^k es el tiempo en que el paquete número *k* del flujo *i* llega al primer nodo de la red (donde *j* = 1); l_i^k es un tamaño del paquete número *k* del flujo *i*, r_i es

el ancho de banda reservado para el flujo *i,* (de tal forma que l_i^k / r_i es el máximo tiempo para retransmitir el paquete –nótese que no se refiere al tiempo de espera en cola sino al tiempo de transmisión por la interfaz), y ξ_i^k es una variable de holgura[76] asignada al paquete número *k* del flujo *i*, en los nodos siguientes nodos por donde pasa, después del primero. La unidad del índice de prioridad se da en segundos.

$$d_{i,j}^k = \begin{cases} max\{t_i^k, d_{i,1}^{k-1}\} + \dfrac{l_i^k}{r_i} & j = 1 \\ d_{i,j-1}^k + \dfrac{l_i^k}{r_i} + \xi_i^k & j > 1 \end{cases} \quad (1)$$

Note que para este método, en las interfaces de salida de los nodos, sirve usar la forma de operación en que hay una cola por cada interfaz de entrada, y así se usa. El despachador en este método utiliza el índice de prioridad de cada paquete para seleccionar el próximo paquete a salir, de los que encabezan cada una de las colas.

6.2.1 Experimentos donde el flujo que incrementa el tráfico no daña a los flujos de rutas de interferencia

En los experimentos en esta parte del capítulo, el tráfico usado, en la red del escenario de trabajo de esta Tesis, es igual al que se usa en los demás experimentos de esta Tesis. El valor de r_i es igual a 2.4 *Mb/s*, significando que el flujo de cada una de las tres rutas de interferencia, en cada nodo central, tiene un ancho de banda reservado del 8% de la capacidad del enlace de salida del nodo central (de 30 *Mb/s*); ξ_i^k es igual a 0.0003 *s*, para el flujo que sigue la ruta larga, $s_1 - d_1$; ξ_i^k es igual a 0.000375 *s* para el flujo que sigue cada una de las rutas de longitud media, $s_2 - d_2$, $s_3 - d_3$, $s_4 - d_4$, $s_5 - d_5$; y ξ_i^k es igual a 0.0005 *s* para el flujo que sigue cada una de las rutas de longitud corta, $s_6 - d_6$ y $s_7 - d_7$.

La Figura 24 muestra, en sus cuatro sub-gráficas, los resultados de retraso en las rutas, de cuatro diferentes experimentos donde el flujo de la ruta $s_2 - d_2$ incrementa su número de fuentes en 2, 4, 6 y 10, respectivamente, en el tiempo 600 *s*.

[76] La variable de holgura es menor para las rutas largas.

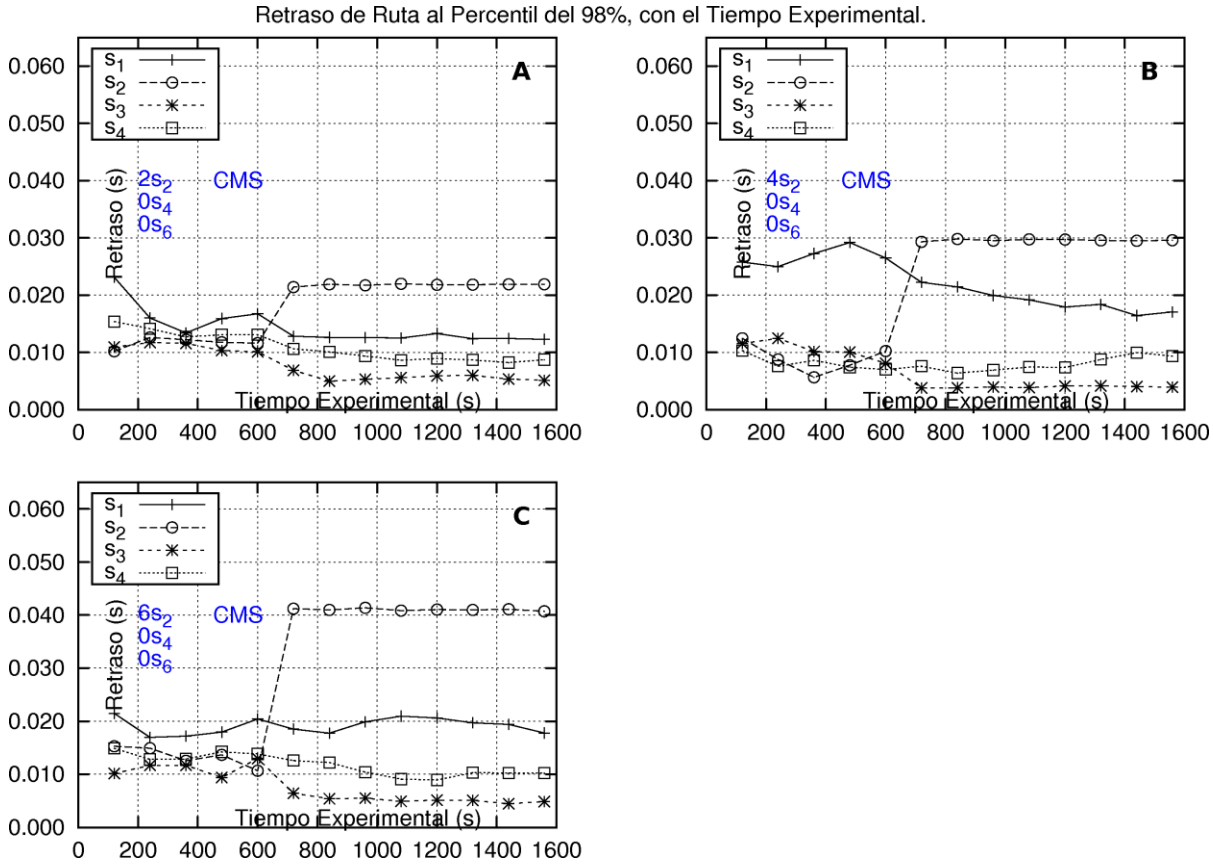


Figura 24. Retrasos en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_4-d_4, en los experimentos con la aplicación del Método CMS, para la atención de tráfico. Este método se aplica en el escenario de trabajo de esta Tesis, trabajando con una sola clase.
En los cuadros "A", "B" y "C" la ruta s_2-d_2 aumenta de tráfico en 2, 4 y 6 fuentes, respectivamente, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Se observa que, en términos de retraso, el flujo de la ruta $s_2 - d_2$ tiene un deterioro desde los 720 *s*, mientras que los flujos de las rutas $s_1 - d_1$ y $s_3 - d_3$ apenas son afectados; y el retraso del flujo de la ruta $s_4 - d_4$ no se afecta.

La razón para este impacto en el retraso, en el flujo de la propia ruta $s_2 - d_2$ es que la tasa de llegadas de paquetes, de este flujo, es mayor que la tasa de llegadas de paquetes, de los otros flujos. Dado que cada paquete, de un flujo, tiene un valor mayor de índice de prioridad (menor prioridad) que el que tienen los paquetes previos del mismo flujo, entonces, los valores, de índice de prioridad, de los paquetes del flujo de la ruta $s_2 - d_2$, encolados en la interfaz de salida del nodo central c_0, son mayores (menor prioridad) que los correspondientes valores, de índice de prioridad, de los paquetes de los flujos de las rutas $s_1 - d_1$ y $s_3 - d_3$, encolados

en la misma interfaz. Entonces, los paquetes de estos dos últimos flujos tienen ventaja, al ser evaluados para servicio, con relación a los paquetes del flujo de la ruta $s_2 - d_2$.

En este caso, los paquetes del flujo de la ruta $s_1 - d_1$ arriban al nodo central c_1 teniendo un valor de índice de prioridad no necesariamente menor, o mayor, que el que tienen los paquetes del flujos de las rutas $s_4 - d_4$ y $s_5 - d_5$, así que el método no favorece a ninguno de estos flujos, en la interfaz de salida del nodo c_1.

6.2.2 Experimentos Donde el Flujo que Incrementa Tráfico sí daña a los Flujos de Rutas Interferidas

Este experimento tiene condiciones de tráfico diferentes a las de los otros experimentas en esta Tesis. La razón de esto es poder proporcionar al flujo de la ruta $s_2 - d_2$ la mayor ventaja. En este experimento, se observan los resultados de una sola simulación.

El número inicial de fuentes es de 13, 13, 8, 14, 14, 0, 0, para los flujos de las rutas, de la $s_1 - d_1$ a $s_7 - d_7$, respectivamente (en lugar de los valores 12, 12, 12, 12, 12, 12, 12 que se tienen en el escenario experimental de esta Tesis). El flujo de la ruta $s_2 - d_2$ incrementa su tráfico, en el tiempo 600 *s*, en 6 fuentes. El ancho de banda reservado para todos los flujos es de 6 *Mb/s*, excepto para el flujo de la ruta $s_2 - d_2$, que tiene 9 *Mb/s*.

En este experimento, los valores para ξ_i^k son más pequeños, para obtener menores valores de índice de prioridad. El valor de este índice es: 0.00012 *s* para el flujo de la ruta larga $s_1 - d_1$; 0.00015 *s* para el flujo de cada una de las rutas de longitud media, $s_2 - d_2$, $s_3 - d_3$, $s_4 - d_4$ y $s_5 - d_5$; y 0.0002 *s* para el flujo de cada una delas rutas de longitud corta, $s_6 - d_6$ y $s_7 - d_7$.

El Método CMS es muy sensible a la cantidad de ancho de banda reservado para los flujos. En este experimento, el flujo de la ruta $s_2 - d_2$ tiene más ancho de banda reservado que lo que tienen cada uno del otros flujos (los flujos de las rutas $s_1 - d_1$ y $s_3 - d_3$), en la interfaz de salida del nodo c_0. Revisando la ecuación (1), los valores de índice de prioridad, que se asignan a los paquetes del flujo de la ruta $s_2 - d_2$, tienen una tasa de crecimiento menor, entre uno y el siguiente paquete, significando que la tendencia es dar un valor menor de índice de prioridad (mayor prioridad) a estos paquetes, y contribuyendo así a causar un impacto importante a los otros flujos (aquellos de las rutas $s_1 - d_1$ y $s_3 - d_3$), cuando el flujo de la ruta $s_2 - d_2$ incrementa su tráfico.

La Figura 25 muestra los resultados de este experimento.

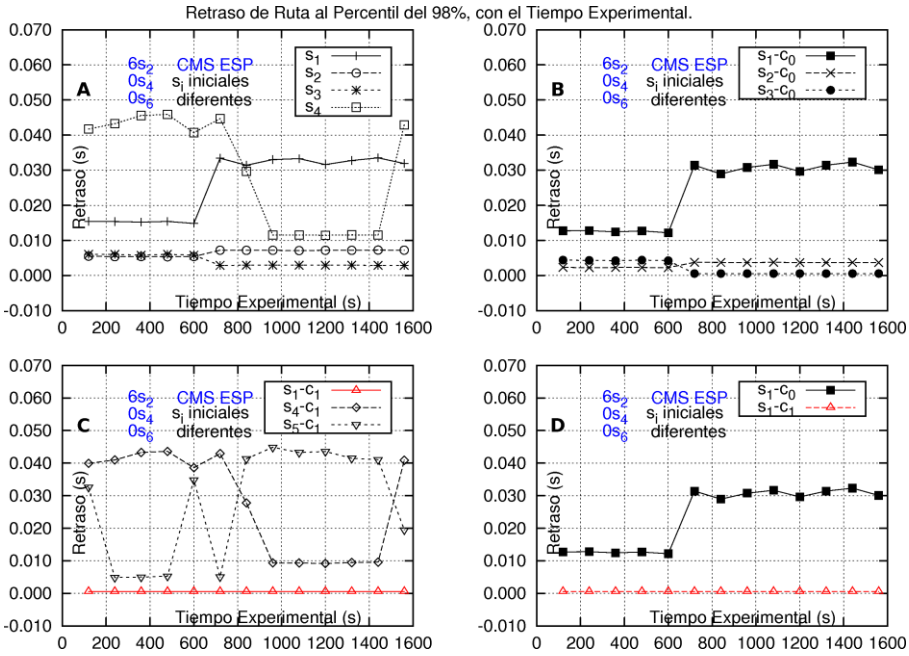


Figura 25. Resultados de retraso de los experimentos con el Método CMS, aplicado al escenario de trabajo de esta Tesis, operando con una sola clase. Aquí, el flujo de la ruta s_2-d_2 tiene ventaja sobre los otros flujos.
En todos cuadros de esta figura la ruta s_2-d_2 aumenta de tráfico en 6 fuentes, a los 600 *s* de tiempo experimental. El cuadro "A" muestra los retrasos en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_4-d_4. El cuadro "B" muestra los retrasos en las colas de las rutas s_1-d_1, s_2-d_2, s_3-d_3, en el nodo c_0. El cuadro "C" muestra los retrasos en las colas de las rutas s_1-d_1, s_4-d_4, s_5-d_5, en el nodo c_1. El cuadro "D" muestra los retrasos en las colas de la ruta s_1-d_1, en el nodo c_0, y en el nodo c_1.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

En este caso, se puede observar cómo el Método CMS protege al flujo de la ruta $s_1 - d_1$ contra el incremento de tráfico del flujo de la ruta $s_2 - d_2$, dando al flujo de la ruta $s_1 - d_1$ una mayor prioridad en la interfaz de salida del nodo c_1, pero afectando, en consecuencia, a otro flujo, de una manera inmediata (el flujo de la ruta $s_5 - d_5$ es el afectado).

El cuadro "A" muestra el retraso del flujo de cada una del rutas $s_1 - d_1$, $s_2 - d_2$, $s_3 - d_3$ y $s_4 - d_4$. Se puede observar que el flujo de la ruta $s_1 - d_1$, en el tiempo 600 *s*, sufre de un incremento de retraso, y el flujo de la ruta $s_2 - d_2$ apenas incrementa su retraso, como consecuencia de este incremento de tráfico. El flujo de la ruta $s_3 - d_3$ inclusive mejora un poco, en términos de su retraso, porque éste tiene menos tráfico, y por lo tanto, el valor de índice de prioridades de sus paquetes es menor (más prioridad). El flujo de la ruta $s_4 - d_4$ decrece su retraso, y la razón se da en la explicación del cuadro "C".

El cuadro "B" muestra los resultados de retraso del flujo de cada una de las rutas: $s_1 - d_1$, $s_2 - d_2$ y $s_3 - d_3$, en la interfaz de salida del nodo central c_0. El cuadro "C" muestra los resultados de retraso del flujo de cada ruta: $s_1 - d_1$, $s_4 - d_4$ y $s_5 - d_5$, en la interfaz de salida del nodo central c_1.

Se puede observar que el flujo de la ruta $s_1 - d_1$ prácticamente no tiene retraso a lo largo del tiempo experimental en el nodo c_1. La razón es que cada paquete de ese llega retrasado al nodo c_1, teniendo, por eso, una prioridad favorable, con relación a los paquetes de los flujos $s_4 - d_4$ y $s_5 - d_5$. A los 720 *s* se observa una afectación en el retraso de estas dos rutas, en favor del flujo de la ruta $s_1 - d_1$. El retraso del flujo de la ruta $s_4 - d_4$ es cercano a 43 *ms*, antes del tiempo 600 *s*, y éste decrece a cerca de 9.3 *ms*, después de 600 *s*. El retraso del flujo de la ruta $c_5 - d_5$ es el afectado, dado que se incrementa pasando de aproximadamente 5 *ms* a cerca de 43 *ms*, en el tiempo experimental de 720 *s*. Esta afectación es causada por:

1.- El incremento de tráfico del flujo de la ruta $s_2 - d_2$.

2.- La protección del flujo de la ruta $s_1 - d_1$ [77].

3.- Porque aun cuando los flujos de las rutas $s_4 - d_4$ y $s_5 - d_5$ tienen el mismo número de fuentes, en este experimento el flujo de la ruta $s_4 - d_4$ tiene en promedio menor tráfico que el flujo de la ruta $s_5 - d_5$.

El cuadro "D" muestra los resultados de retraso del flujo de la ruta $s_1 - d_1$, en la interfaz de salida del nodo central c_0, y en la interfaz de salida del nodo central c_1. El flujo de la ruta $s_1 - d_1$ tiene aproximadamente 12.5 *ms* de retraso en la interfaz de salida del nodo central c_0 (que va al nodo central c_1) antes de 600 *s*, y después el retraso en dicha interfaz de salida se incrementa a aproximadamente 30 *ms*. En la interfaz de salida del nodo central c_1 (que va al nodo central c_2) el retraso del flujo de la ruta $s_1 - d_1$ es, en todo el tiempo experimental,

[77] Sobre los cuadros "A" y "D" de la Figura 25 es importante notar que la suma de añadir el retraso observado en la interfaz de salida del nodo central c_0, y el retraso observado en la interfaz de salida del nodo central c_1, para los paquetes del flujo de la ruta $s_1 - d_1$, no es necesariamente menor que el retraso observado para los paquetes de este flujo en toda esta ruta, de principio a fin. La razón es que el 2% de paquetes que son más retrasados en atravesar la interfaz de salida del nodo central c_0, no son necesariamente los mismos que los paquetes dentro del 2% de paquetes más retrasados en pasar por la interfaz de salida del nodo central c_1.

muy cercano a 0.57 *ms*. Así, el retraso del flujo de la ruta $s_1 - d_1$ es compensado por este método.

Resumen del Capítulo

En este capítulo se presentaron los resultados de dos métodos alternativos, aplicados al escenario de trabajo de esta Tesis. El primer método es el q3f donde hay una cola en cada interfaz de salida de un nodo, por cada interfaz de entrada, cuando las colas tienen un peso fijo.

Con el empleo del Método q3f, en una interfaz de salida no se puede proteger al tráfico de una ruta cuando éste supera la proporción, fija, de ancho de banda asegurada para esa ruta, en la interfaz.

El segundo método es el llamado CMS, con una aplicación para proteger a flujos en rutas que se interfieren. Cuando un flujo de una ruta es afectado en un nodo, el flujo adquiere una prioridad mayor en el nodo siguiente por donde pasa la ruta, por lo que dicho flujo es favorecido con relación a los flujos de otras rutas de interferencia en dicho siguiente nodo. El problema de este método es precisamente que la protección de un flujo en una ruta, puede causar deterioro a los flujos de otras rutas que interfieren en nodos posteriores por donde pasa la ruta, formándose una cadena de afectación a las siguientes rutas interferentes cuya magnitud no es clara.

Por las anteriores observaciones, los métodos presentados en este capítulo no se consideran adecuados para la protección de rutas interferidas, en una red.

Capítulo 7. Planteamiento del Método de Solución. Método STP

Introducción al Capítulo

En este capítulo se desarrolla la propuesta de solución del problema a resolver en esta Tesis. Esta propuesta de solución es un método de atención de tráfico que aquí se llama Método STP. Se presentan las especificaciones y parámetros del modelo.

El Método STP debe proteger el tráfico de una ruta contra sus rutas cruzadas, en redes en donde el tráfico está restringido a cursar por rutas específicas. El enfoque del método es generar una protección en cada nodo por donde pase una ruta a proteger.

En la primera parte de este capítulo se presentan conceptos antecedentes que permiten exponer las razones del planteamiento de la operación del Método STP. Posteriormente, en el capítulo, se propone la operación general del método, las hipótesis de su operación, y lo que se espera del método en cuanto a sus ventajas. Al final del capítulo se presenta el algoritmo de operación del método.

7.1 Presentación de Conceptos Iniciales

7.1.1 Clases en una Red

Es de uso general que para compartir el ancho de banda de los enlaces de una red, entre dos o más clases, se formen, en cada interfaz de salida de los nodos de la red, una cola para el tráfico proveniente de cada clase, que ha de salir por el enlace de la interfaz. Para seleccionar el tráfico que va saliendo por el enlace se usa un despachador. El despachador es ahorrador de energía cuando se quiere aprovechar al máximo el ancho de banda del enlace. La selección del tráfico se hace conforme a un valor de peso que se da a cada clase, donde, por norma, la suma de los pesos es igual a 1. La manera de seleccionar los pesos depende de algún método planteado, que suele estar enfocado a maximizar alguna ganancia, o proveer alguna forma de reparto equitativo del ancho de banda de los enlaces, para las clases.

A conocimiento del autor de esta Tesis, para el manejo del tráfico de una clase en una red, se asocia al tráfico de la misma una cola por cada interfaz de salida de los nodos de la red.

7.1.2 Una Cola para una Interfaz de Salida en un Nodo

Si dos o más flujos entrantes a un nodo compiten por el ancho de banda de una misma interfaz de salida, cuando hay una “sola” cola en dicha interfaz, el tiempo promedio de espera para atención (para salir del nodo) de los paquetes de esos flujos es el mismo. Este tiempo de espera es el retraso en el nodo. Cuando uno de esos flujos eleva su tráfico, el retraso en el nodo se eleva para todos los flujos competidores.

7.1.3 Los Flujos desde la Visión del Nodo

Como se ha indicado, un flujo se suele definir como el tráfico que cursa una red, desde una fuente determinada hacia un destino determinado. El tráfico de la fuente entra normalmente por un mismo nodo de ingreso, y sale de la red por un mismo nodo de egreso.

En esta tesis interesan redes en donde los flujos están restringidos a cursar por una ruta específica. En este sentido dos flujos que cursan la red por la misma ruta se combinan para llamarse, todos ellos, el flujo de la ruta, y al correspondiente tráfico combinado se le llama el tráfico de la ruta.

En la red del escenario de trabajo de esta Tesis, similarmente a lo que sucede en las redes DiffServ, la diferenciación de los flujos solamente la pueden realizar los nodos de frontera, que pueden ser nodos de ingreso y / o de egreso. Un nodo interior o central (un nodo que no es un nodo de frontera) no puede hacer esta diferenciación.

El Método STP se concibe para que un nodo central no tenga que poder identificar si los flujos que le llegan corresponden a una ruta o a otra, pero sí debe poder identificar si los flujos provienen de una interfaz de entrada o de otra. A estos flujos diferenciados dentro de un nodo central se les puede llamar, “flujos en el nodo”.

7.1.4 Relación entre Tamaños de Colas y Tráfico de Entrada en un Sistema de Atención de Tráfico

El Teorema de John Little [77] indica que para casi cualquier sistema de colas, con diversas estructuras, que incluye la, o las colas, y el, o los servidores, se cumple que $N = \lambda T_s$, donde N es el número medio de clientes en el sistema, T_s es el tiempo medio de atención al cliente (desde que llega hasta que concluye de ser atendido –medido en *s*), y λ es la tasa media de llegada de clientes al sistema (medida en *clientes*/*s*). Una aplicación de dicho teorema es:

$$q = \lambda W \tag{2}$$

Donde q es el número medio de clientes presentes encolados, W es el tiempo medio de espera en cola.

Es importante indicar que para que se tenga estabilidad en el sistema es necesario que $\lambda < \mu$, donde $1/\mu$ es el tiempo promedio de la duración de atención de un cliente, por parte del sistema.

Cuando el servidor es un despachador ahorrador de energía, que puede atender a un cliente a la vez; siendo los clientes, paquetes, y el envío se hace por un enlace de salida con un ancho de banda B (medido en *bits*/*s*), y donde el número promedio de bits de un paquete es L_p, se cumple que $\mu = B/L_p$ (medido en *paquetes*/*s*), y por tanto que $\lambda < \mu = B/L_p$.

7.1.5 Aplicación a un Sistema de Atención que Tiene dos Flujos y una Cola por Flujo

Considerando un sistema de atención con un solo despachador, este subcapítulo analiza el significado de otorgar a cada cola i del sistema (con i = 1, .., N, con N > 1), un peso ϕ_i que es proporcional a su longitud promedio q_i , para su atención por parte del despachador. Este enfoque se utiliza en [61] y [55], y también es el enfoque adoptado en esta Tesis, para el Método STP.

Se indica en [61] que con este enfoque de asignación de pesos, el ancho de banda del despachador se distribuye de una manera *equitativa*.

En términos de ecuaciones, el enfoque se puede escribir como:

$$\phi_i = \frac{q_i}{\sum_{j=1}^{N} q_j}, \; i = 1,..N. \tag{3}$$

Para simplificar el análisis, sin pérdida de generalidad, se puede considerar que N = 2, y que todos los paquetes son del mismo tamaño. En la Figura 26 se representa un sistema estable de atención de tráfico, con dos colas y un promedio de paquetes "en cola" representados con q_1 y q_2, respectivamente[78], siendo λ_1 y λ_2 el promedio de tasa de llegadas de paquetes, a cada cola. El despachador es *conservador de energía*, el cual envía los paquetes atendidos por un enlace de salida que tiene una capacidad de r (medida en *paquetes*/*s*), y que asegura a cada cola una tasa mínima de atención de paquetes de $r\phi_1$ y $r\phi_2$, donde, por norma, $\phi_1 + \phi_2 = 1$. La estabilidad requiere que $r > \lambda_1 + \lambda_2$.

Lo anterior quiere decir que si en algún intervalo de tiempo limitado, la tasa de llegada de tráfico a la cola 1 es mayor a su tasa promedio, para dicho tráfico, en ese intervalo, se tiene disponible, al menos, una tasa de atención de $r\phi_1$. Algo similar sucede para la otra cola.

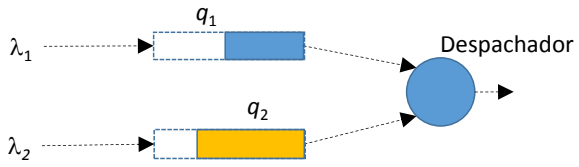


Figura 26. Estructura de un sistema de atención con 2 colas.

[78] No confundir en este subcapítulo el argumento q1 que representa el tamaño medio de la cola #1 con el Método q1 evaluado en esta Tesis.

Aplicando el teorema de Little a cada cola, se cumple que $q_1 = \lambda_1 W_1$, y que $q_2 = \lambda_2 W_2$, donde q_1 y q_2 son los tamaños promedio de las colas, y W_1 y W_2 son los tiempos de espera promedio en cada cola.

Consideración Única. Sea la porción de atención asegurada a cada cola igual a la tasa de llegada relativa promedio a la cola, esto es:

$$r\phi_i = r\frac{\lambda_i}{\lambda_1 + \lambda_2},\ i = 1, 2. \qquad (4)$$

La ecuación (4) equivale a que $(\lambda_1+\lambda_2)\phi_1 = \lambda_1$ y $(\lambda_1+\lambda_2)\phi_2 = \lambda_2$, y dado que $r > \lambda_1 + \lambda_2$, se cumple que $r\phi_1 > \lambda_1$ y $r\phi_2 > \lambda_2$. Lo anterior implica que a cada cola se le está asegurando una porción de ancho de banda mayor a su tasa media de llegada. Puede verse esto como una *repartición equitativa* de ancho de banda (o que hay una equidad en la repartición de ancho de banda[79]).

Hay que notar que podría haber una repartición de pesos diferente, donde, aun cuando se cumpliese que $r > \lambda_1 + \lambda_2$, la relación entre porciones aseguradas y tasas de llegada fuese, por ejemplo, $r\phi_1 > \lambda_1$ y $r\phi_2 < \lambda_2$, en cuyo caso, la cola 1 tendría ventajas sobre la cola 2, perdiéndose la equidad.

Combinando la ecuación (4) y la expresión $q_i = \lambda_i W_i$, i = 1, 2 (ver ecuación (2)) se obtiene que: $r\phi_i = r\dfrac{q_i / W_i}{q_1/W_1 + q_2/W_2}$, i = 1, 2. Esta expresión tiene parecido a la ecuación (3) (cuando N = 2), pero ciertamente es diferente.

Con lo anterior, se puede decir que la ecuación (3) es una aproximación para proveer una repartición equitativa, (en términos de la ecuación (4)).

En los experimentos que se realizan más adelante (en el 0) se evalúa el Método STP, en el escenario de trabajo de esta Tesis, donde los pesos de las colas se rigen según la ecuación (3). De los resultados de esos experimentos se observa que los pesos de las colas tienen un comportamiento que con el tiempo resulta cercano al tráfico relativo promedio de llegadas a las colas, es decir, también hay una *cercanía* de comportamiento con la ecuación (4).

Para dar valores a esto, en dicho capítulo, en la Figura 31D se muestra el peso de tres colas que están en la interfaz de salida de un nodo, donde se utiliza el Método STP para atender el tráfico, y donde el enlace de salida tiene un ancho de banda de 30 *Mb*/*s*. En dos de estas colas (en cada una) hay una tasa de llegada promedio de 7.45 *Mb*/*s*, y en la otra de 13.67 *Mb*/*s*

[79] En [9] se indica que no hay un acuerdo en la definición de equidad en las redes.

(para un total de tráfico que equivale al 95.22% de ocupación del enlace) (ver la Tabla 9 en la página 136). La diferencia entre el peso obtenido y la relación del tráfico relativo de llegada a la interfaz de salida, para cada cola, llega a ser tan cercana como del 1.91% (en el tiempo 1560 *s* experimental).

Es importante mencionar que no se puede afirmar aun, por los resultados de esta Tesis, que para otros posibles escenarios habría cercanía entre los pesos asignados a las colas y el tráfico relativo de llegada que tengan las mismas.

Es importante mencionar que el autor de esta Tesis ha observado el comportamiento del tamaño de las colas utilizando tráfico Pareto[80], con una distribución de probabilidad "pesada" en las colas, y la tendencia es que la proporción del tamaño de las colas crece en mayor medida al crecimiento de la proporción del tráfico relativo promedio de llegada, por lo que en ese caso la asignación de peso a una cola dada, según su tamaño relativo, cuando la cola tuviese mayor tráfico relativo que las demás, podría acentuar la preferencia de atención a esa cola, con deterioro de las demás.

7.1.6 Planteamiento de Protección Usando Varias Colas por Interfaz de Salida, en un Nodo

Dado que en un nodo central no se pueden identificar flujos de ruta, pero sí se pueden identificar los flujos provenientes de cada interfaz de entrada, y justamente dos rutas que se interfieren en un nodo llegan al mismo por interfaces de entrada diferentes, entonces, se propone que en cada interfaz de salida de los nodos centrales se genere una cola para el tráfico que llega por una interfaz de entrada diferente. Un nodo central así se puede ilustrar como el nodo "*x*" de la Figura 3, página 40.

Con esto, dos o más rutas que se interfieren en un nodo, es decir, cuyo tráfico sale del nodo por la misma interfaz de salida, tienen, cada una, una cola formada para la salida del tráfico por dicha interfaz de salida.

Para la atención del tráfico de las colas se tiene un despachador en la interfaz de salida, el cual va seleccionando el siguiente paquete a salir por la interfaz, de entre los paquetes que aguardan salida (atención) en las colas.

Para dar atención a las colas, cada una tiene un peso, de tal forma que la suma de los pesos es 1, por norma. A los despachadores que atienden a las colas según los pesos de las mismas se les refiere como despachadores de atención "pesada" o "Weighted" (del inglés). El peso en una cola es una garantía de que la cola tiene asegurado el uso de una porción del ancho de banda de la interfaz de salida, igual a su peso. Si la cola no utiliza toda dicha porción, otra

[80] Ver Apéndice G.

cola con más necesidad de ancho de banda lo puede aprovechar, pero esa cola estaría utilizando una porción mayor de ancho de banda del enlace de salida, a lo que tiene asegurado.

El Método STP se ha propuesto para operar de tal forma que cada cola, en un sistema de *N* colas, tenga un mínimo ancho de banda asegurado igual a ϕ_i, y que ϕ_i tienda a ser igual a $q_i \Big/ \sum_{j=1}^{N} q_j$, es decir, conforme el tamaño relativo de la cola i cambia, su peso asociado se mueve, pero más lentamente. La razón de esto es que al aumentar, en una cola, el tamaño medio del tráfico de entrada, se aumenta su tamaño medio. Mientras los pesos de las otras colas se mantengan en su valor, dichas colas estarán protegidas, es decir, se mantendría su ancho de banda mínimo. La pregunta obvia sería, ¿por qué no dejar a esas colas protegidas? La razón es que la protección de esas colas podría estar siendo excesiva, dejando a la cola que aumenta su tráfico sin posibilidad de ampliar su protección para el tráfico que acaba de admitir.

Por otro lado, el método permite la competencia entre rutas, de tal forma que cuando una ruta aumenta su tráfico, sí puede afectar a otras rutas interferidas, pero el método trata de que esta afectación crezca lentamente de tal forma que las rutas interferidas tengan tiempo para tomar medidas, como liberar tráfico.

Se pretende, con el método, que al dar protección de una cola en un nodo, dicha protección se extienda en la ruta asociada a dicha cola. En el largo plazo, en cada enlace de la red, la proporción de ancho de banda entre las rutas que se interfieren en ese enlace es proporcional a las demandas relativas promedio medidas de esas rutas.

Por lo anterior, el método que se propone para atención del tráfico en una red, con la protección de rutas interferentes, se llama Método de “Protección de Corto Plazo”, o Método STP (Short-Term Protection por sus siglas en inglés).

En contraste con otros trabajos que, con motivos de optimización, proponen esquemas para compartir ancho de banda entre las diversas clases de una red que implementa QoS, esta Tesis propone un método (el Método STP) que gestiona el ancho de banda disponible dentro de una sola clase, en una red que implemente QoS, para ayudar a simplificar el proceso de admisión, cuando la misma es distribuida, por ruta.

Con el fin de facilitar las explicaciones en este documento, el concepto de clase no se utiliza, el concepto de ancho de banda disponible se expresa, simplemente, como el ancho de banda disponible de una red.

El Método STP se implementa en cada interfaz de salida de los nodos centrales, en una red. Cada nodo central que trabaja con el método actúa en forma autónoma, sin necesidad de "tener conocimiento" de la existencia de rutas, o de la existencia de los flujos en la red, y sin marcar paquete alguno.

Como el parámetro de interés en la QoS, en esta Tesis es el retraso de extremo a extremo en las rutas, el término "retraso en ruta" se refiere al retardo de extremo a extremo en la ruta. Así, la protección del método se evalúa con respecto al valor de este retraso.

7.1.7 Hipótesis y Expectativas sobre el Método STP

Las hipótesis generales del Método STP son:

1- Es posible realizar un comportamiento de protección de corto plazo, en una red de decenas de nodos, a las rutas de la red sin requerir reservación en las mismas, mediante una operación autónoma en cada nodo central de la red (es decir sin que haya un administrador central que los haga operar), donde se logre una coordinación natural entre dichos nodos, y sin marcar paquetes.
2- Al haber en cada nodo por donde pasa una ruta, específicamente en la interfaz de salida por donde se da el paso, una cola específica asociada a la ruta, es posible proteger a la misma mediante la protección de su cola asociada.

La hipótesis específica sobre el comportamiento de los pesos (de las colas) es:

3- Que el comportamiento o conducta de disminución máxima de los pesos (de las colas) sigue la expresión de la ecuación (26).

Se espera que el Método STP:

1. Permita una administración simple de la red, con prospecto de ser escalable hacia al menos decenas de nodos.
2. Sea útil para los administradores de las rutas de la red, para tener confianza, por lo menos dentro de una ventana de tiempo, sobre la cantidad de ancho de banda con que se cuenta en cada ruta. Sea sencillo de manejar y se ajuste de acuerdo a las necesidades de ancho de banda de las rutas.
3. Proteja especialmente a las rutas largas, con relación a rutas cortas, de interferencia debido a que estas últimas tolerarían un mayor aumento de tráfico por tener un menor número de nodos donde pudiese haber retraso por congestión.
4. Como expectativa específica, que los valores de los pesos de las colas se aproximen, con el tiempo, a los valores del tráfico relativo de entrada a las mismas.

7.1.8 Operación del Método Planteado

En una red hay una copia del método operando en cada interfaz de salida de cada nodo central de la red.

Cada interfaz de entrada corresponde a una ruta que llega al nodo, o a una o más rutas que se han combinado río-arriba. Una representación de esta característica, en el nodo c_0 de la red, en el escenario de trabajo de esta Tesis (ver Figura 15 en la página 70) se da en la Figura 27.

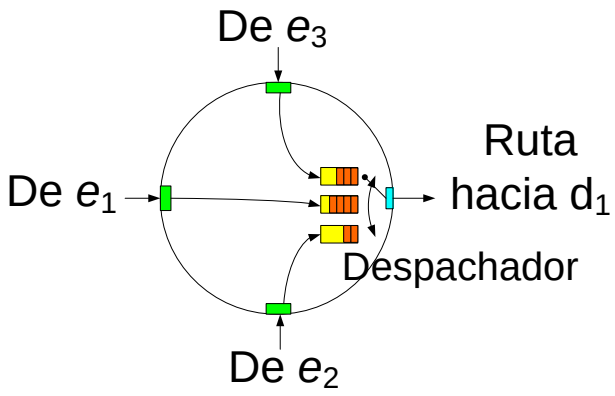


Figura 27. Representación del nodo c_0 operando con el Método STP, de la topología representada en la Figura 15.

Se considera que no hay retrasos por encolamiento en las interfaces de entrada de los nodos [33].

Al final, la protección ofrecida por este método se refleja en la protección en el retraso de las rutas.

Según el mejor conocimiento del autor de esta Tesis, sobre esta línea de investigación, no existe en la literatura un método que esté orientado a trabajar dentro de una clase, que atienda el problema de interferencia entre rutas, con admisión independiente una de otra. Tampoco existe, a conocimiento del autor de esta Tesis, un método que trabaje en la salida de las interfaces de cada nodo central, una cola diferente para cada interfaz de entrada del nodo, sin requerir marcar paquetes, como sí lo hace el Método STP.

El Método STP no protege a una ruta dada por el incremento de tráfico en la ruta misma.

La forma de cambio de los pesos, con el tiempo, con relación al tráfico, es tema de los siguientes subcapítulos.

7.1.9 Forma Propuesta para Cambio de Pesos con el Tiempo con el Método STP

El Método STP se propone para operar de tal forma que se dé a cada cola, en la interfaz de salida de un nodo, un peso igual a su longitud media relativa, como se explica en el subcapítulo 7.1.5.

7.2 Propuestas Preliminares para la Forma en que los Pesos se Adapten a la Longitud Relativa de la Cola

En el curso del trabajo de esta Tesis se hicieron dos propuestas anteriores a la definitiva, para hacer que los pesos de las colas variasen, a través del tiempo, siguiendo a las longitudes medias relativas. Estas propuestas, preliminares, se evaluaron mediante experimentación y se presentan en el Apéndice F.

A continuación se repasan, brevemente, las expresiones matemáticas de estas dos propuestas, que llevaron al planteamiento de la expresión final.

7.2.1 Propuestas

La primera propuesta para la obtención de los "nuevos" pesos, se presenta en la ecuación (5), donde $\phi_i\ t$ representa el peso propuesto para la cola i, $\overline{Q_i\ t}$ representa la longitud media medida de la cola i, ambos en función del tiempo *t*, y ϕ_i' representa el peso inicial de la cola i.

$$\phi_i\ t\ = \frac{\overline{Q_i\ t}}{\sum_{k=1}^{N} \overline{Q_k\ t}\ \Big/ N}\phi_i', \quad i = 1, ..., N \tag{5}$$

Se tomaron previsiones para evitar una indeterminación por división entre cero[81]. El factor $\overline{Q_i\ t}\ \Big/ \left[\sum_{k=1}^{N} \overline{Q_k\ t}\ \Big/ N\right]$ es la relación de la longitud de la cola i entre el promedio de la longitud de las colas, cuyo valor es positivo, y puede ser menor, igual o mayor que 1. Una vez obtenidos los pesos nuevos de las colas, éstos podrían normalizarse para sumar 1.

Un inconveniente de esta propuesta era que en cada llegada, o salida, de paquete se evaluaba un criterio para establecer si era necesario, o no, evaluar, a su vez, los nuevos pesos, con el uso de la ecuación (5). La rapidez de la variación de los pesos, con relación a la variación de las longitudes medias de las colas, no era clara.

Otro inconveniente de esta propuesta era que cada nuevo peso variaba alrededor de un peso inicial, que forzaba a que dicho peso inicial fuese una forma de ancla alrededor del cual variaba el peso de operación.

---

[81] Además, los pesos propuestos reemplazarían a los pesos presentes cuando fuese apropiado, según la operación del despachador usado. Un despachador puede requerir del paso de alguna cantidad de eventos de salida de paquetes, a partir de que se haya indicado, con algún criterio, que se debe hacer un cambio de pesos.

En una segunda propuesta los nuevos pesos se calculan como en la ecuación (6) (propuesta por el autor de esta Tesis), donde el significado de los elementos es similar a los de la ecuación (5). El valor de α es positivo cercano a 0 (del orden de 0.001) que limita la variabilidad del nuevo peso, conforme el cambio de longitudes promedio de las colas (aspecto que se quería corregir con relación a la primera propuesta).

La evaluación de los nuevos pesos se hace al final de cada periodo de duración τ, donde τ es del orden de 1 *s* (otro aspecto que se quería corregir con relación a la primera propuesta).

$$\phi_i(t+\tau) = (1-\alpha)\phi_i(t) + \alpha \frac{\overline{Q_i(t+\tau)}\,\phi_i'}{\sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'}, \quad i \in \{1,...,N\}, 0 < \alpha < 1 \tag{6}$$

Se puede observar que haciendo la sumatoria $\sum_{i=1}^{N} \phi_i(t+\tau)$ el valor de la misma resulta igual a 1. La ecuación (6) se puede despejar para observar que el incremento del peso calculado, $\phi_i(t+\tau) - \phi_i(t)$ depende del valor de α y de la diferencia de los valores $\phi_i(t)$ y $\overline{Q_i(t+\tau)}\,\phi_i' \Big/ \sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'$.

$$\phi_i(t+\tau) - \phi_i(t) = \alpha \left[\frac{\overline{Q_i(t+\tau)}\,\phi_i'}{\sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'} - \phi_i(t) \right] \tag{7}$$

En una evaluación experimental de esta propuesta (como se observa en el Apéndice F), se observó que el peso de dos colas en un nodo en una red, donde el tráfico en las colas se mantenía constante, variaba, de manera oscilante.

De la ecuación (6) se puede observar que la rapidez de acercamiento de $\phi_i(t+\tau)$ hacia $\overline{Q_i(t+\tau)}\,\phi_i' \Big/ \sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'$, además de depender de α, también dependía del valor de τ y de la diferencia entre los valores del peso y de la longitud relativa de la cola.

La dependencia de esta diferencia incrementaba la tasa de aumento de peso para una ruta que aumentaba más de tráfico. Aunque esta característica podía ser de interés, había que afinarla porque estaba causando las oscilaciones observadas.

Para solucionar lo anterior, en el desarrollo de esta Tesis, se consideró llevar el diseño de la propuesta a una en donde se pudiese establecer, mediante parámetros, la tasa de cambio de los pesos.

7.3 Modelo Conceptual del Método STP

Con el Método STP se utiliza la ecuación (7), ya no como una forma de obtener un incremento o decremento de peso de una cola, sino como un indicador para determinar si la función de peso $\phi_i(t)$ debería aumentar o disminuir, entre el tiempo *t* y el tiempo *t* + τ. Este indicador, mostrado en la ecuación (8), opera para cada cola, indicando si el tamaño promedio de una cola es menor al tamaño promedio de todas las colas. Si es así, la cola disminuye una porción fija de su peso.

El objetivo del empleo de este indicador es que la rapidez en la disminución del peso, de una cola cuyo tamaño promedio continuamente en las evaluaciones resulta menor que el tamaño promedio de todas las colas, no dependa de la diferencia observada de los tamaños. Asimismo, que la rapidez en la disminución del peso no dependa de la cantidad de actualizaciones que se hagan por segundo. El interés de tener dichas dos características se debe a que con las mismas se encontraron, como se ha indicado, comportamientos oscilatorios en el cambio de los pesos.

El uso es de la siguiente forma (el uso de este indicador se puede ver en la expresión (30)):

$$I\Delta\left(\phi_i(t)\right) = \frac{\overline{Q_i(t+\tau)}\,\phi_i'}{\sum_{k=1}^{N}\overline{Q_k(t+\tau)}\,\phi_k'} - \phi_i(t) \tag{8}$$

Si el indicador $I\Delta\left(\phi_i(t)\right)$ resultase negativo, entonces el incremento $\Delta_i(t) = \phi_i(t+\tau) - \phi_i(t)$ debería ser negativo, es decir en realidad sería un decremento. Se propuso que este decremento fuese muy pequeño, y no dependiente de la diferencia entre los valores de $\phi_i(t)$ y $\overline{Q_i(t+\tau)}\,\phi_i' \Big/ \sum_{k=1}^{N}\overline{Q_k(t+\tau)}\,\phi_k'$. Así, se propuso que el decremento fuese el resultado de multiplicar el valor del peso $\phi_i(t)$ por un factor negativo muy pequeño (del orden de–1/1000) y constante (lo que en el Método STP se puede ver en la ecuación (11)). A continuación se explica el Método STP.

7.3.1 Contexto del Método

Refiriéndose a la Figura 3 (página 40), considérese que el nodo *x* trabaja con el Método STP. Un nodo de este tipo tiene, en cada una de sus interfaces de salida, una cola para cada una de sus interfaces de entrada (teniendo en cuenta que las colas de un nodo se forman solamente en las interfaces de salida [33]). Específicamente, el nodo *x* tiene tres colas en su interfaz de salida *D*, una para cada una de sus tres interfaces de entrada. Cada cola tiene un peso que puede cambiar, y se pretende que lo haga lentamente, de acuerdo con un algoritmo de actualización que periódicamente compara el tráfico procedente de cada interfaz de entrada, a través de la evaluación de la longitud media de cada cola. Con este algoritmo, las colas en la interfaz de salida deben competir por el ancho de banda.

7.3.2 Nomenclatura y Expresiones Generales

Este subcapítulo explica el funcionamiento del Método STP, implementado en una interfaz de salida de un nodo, donde se supone que hay *N* colas, donde $N \geq 2$. Al comienzo de la operación del método las colas están vacías y todos los pesos tienen el mismo valor.

El método calcula la longitud media de cada cola, cada vez que llega, o sale, un paquete de dicha cola. Esta longitud media se calcula con el uso de la ecuación (9) (similar a como se hace en [78]), donde w_q es el parámetro para promediar, $q(t_e)$ es la longitud instantánea de la cola en el tiempo del evento, $t = t_e$,

$$\overline{Q_i(t_e)} = (1 - w_q)\,\overline{Q_i(t_{e-1})} + w_q\, q(t_e) \qquad (9)$$

Esta ecuación actúa como un filtro paso bajas. Mientras más pequeño sea el valor de w_q, más suave es la salida. Empíricamente el valor para w_q se fija en 0.002 [78], para hacer frente al comportamiento de cambios abruptos de la longitud instantánea de la cola.

Además de los cálculos de la longitud media de las colas, el método realiza la mayor parte de sus cálculos (llamado el "grueso" de los cálculos) para obtener un nuevo conjunto de pesos para las colas. Estos cálculos se realizan en la parte final de cada uno de los intervalos de tiempo consecutivos en que se divide el tiempo. Cada uno de estos intervalos tiene un tamaño igual al de los demás, y se denota con τ. Por simplicidad, a cada intervalo se le llama "un intervalo de tiempo τ". El primer intervalo comienza en el tiempo t_0. El intervalo de tiempo τ debe ser lo suficientemente grande como para poder observar muchas llegadas y salidas de paquetes. En este documento se considera $\tau = 1$ *s*. Así, cada intervalo de tiempo corre entre $(t_0 + r\tau)$ y $(t_0 + (r + 1)\tau)$, donde *r* es un número entero tal que $r \geq 0$.

El intervalo de tiempo necesario para completar el "grueso" de cálculos es muy pequeño, en comparación con τ, por lo que, en el algoritmo del método, se considera que estos cálculos

no toman tiempo, y comienzan y terminan en el tiempo $(t_0 + (r + 1)\tau)^-$, que significa "justo antes del tiempo $(t_0 + (r + 1)\tau)$)".

Justo en el tiempo $(t_0 + (r + 1)\tau)$ el nuevo conjunto de pesos para las colas, recientemente obtenido, sustituye al conjunto de pesos actuales (en uso) de las colas (los pesos que estuvieron en uso en el intervalo de tiempo τ que justo terminó)[82]. Por lo tanto, los pesos utilizados por el despachador están fijos del tiempo $(t_0 + r\tau)$ al tiempo $(t_0 + (r + 1)\tau)^-$, y los nuevos pesos entran en vigor en el tiempo $(t_0 + (r + 1)\tau)$. El peso de la cola *i*, hasta el tiempo *t*, se escribe como $\phi_i(t)$.

En esta Tesis el Método STP utiliza un despachador WFQ, pero este método no está destinado restringirse a este tipo de despachador.

En el Método STP, al comienzo de cada grueso de los cálculos, verifica si todas las colas están vacías. Si es así, se restablece el tiempo t_0, dejando los pesos de las colas como estaban. Si la suma de la longitud de las colas no es cero, entonces se lleva a cabo el cálculo de un nuevo conjunto de valores para los pesos de las colas.

En los cálculos para cada *i*, el método calcula un indicador, representado con I_i (ver la ecuación (30), Pág. 128), para comparar el peso actual de la cola con su longitud media, relativa actual. El término "relativo" significa que la longitud media de la cola se divide entre la suma de las longitudes promedio de todas las colas, en la interfaz de salida. El peso de cada cola se reduce si su longitud relativa media resulta menor que su peso actual; de lo contrario, el peso se incrementa. En este método, cada cambio de peso es siempre muy pequeño, y siempre todos los pesos de las colas suman 1. El incremento del peso de la cola *i*, del tiempo $(t_0 + r\tau)$ al tiempo $(t_0 + (r + 1)\tau)$ se escribe como $\Delta_i(t_0 + r\tau)$, tal que:

$$\Delta_i(t_0 + r\tau) = \phi_i(t_0 + (r+1)\tau) - \phi_i(t_0 + r\tau) \tag{10}$$

La longitud media de la cola *i*, en el tiempo *t*, se escribe como $\overline{Q_i(t)}$.

Siempre que, después de un cálculo para actualizar los pesos, el indicador I_i indique que la cola *i* tiene que decrecer su peso, el valor del decremento se calcula como:

$$\Delta_i(t_0 + r\tau) = f\frac{\tau}{T}\phi_i(t_0 + r\tau) \tag{11}$$

En la ecuación (11), *T* es un intervalo de tiempo tal que T/τ es un entero, $T/\tau >> 1$, y *f* es un factor negativo el cual causa que el valor $\Delta_i(t_0 + r\tau)$ también sea negativo. Si a partir del

[82] Puede haber un pequeño tiempo de espera requerido antes de poder realizar esta actualización en el despachador, -como lo requiera el funcionamiento normal del mismo, como se muestra en el procedimiento de operación del despachador WFQ en el Apéndice E.

tiempo ($t_0 + r\tau$), la cola *i* obtuviese T / τ resultados consecutivos indicando que la misma tiene que decrecer su peso, entonces se cumpliría que:

$$\begin{aligned}&\phi_i\left(t_0+\left(r+\frac{T}{\tau}\right)\tau\right)-\phi_i(t_0+r\tau)=\phi_i\left(t_0+r\,\tau+T\right)-\phi_i(t_0+r\,\tau)\\&\approx\phi_i(t_0+r\,\tau)\left(e^{f}-1\right)\end{aligned} \tag{12}$$

La prueba de la ecuación (12) se da en el subcapítulo 7.3.3, donde se hace uso de las ecuaciones (10) y (11).

En la ecuación (12) es útil hacer que *f* sea igual a $f(P) = ln(1 - P)$, donde *P* es una contante positiva pequeña tal que $P > 0$ y $P << 1$. Entonces, dado que $1 - P < 1$, resulta que $f(P) = ln(1 - P) < 0$, por lo que el valor de $\Delta_i(t_0 + r\tau)$ es negativo. Entonces, la ecuación (12) se transforma en la ecuación (13):

$$\phi_i\left(t_0+\left(r+\frac{T}{\tau}\right)\tau\right)-\phi_i(t_0+r\tau)\approx\phi_i(t_0+r\tau)\left(e^{ln(1-P)}-1\right)=-P\phi_i(t_0+r\tau) \tag{13}$$

La ecuación (13) se puede escribir como:

$$\frac{\phi_i(t_0+r\tau+T)}{\phi_i(t_0+r\tau)}\approx(1-P) \tag{14}$$

La relación, en el primer miembro de la ecuación (14), indica que el peso de la cola *i* ha perdido (*P* x 100)% de su valor, en el gran intervalo de tamaño *T*. Por ejemplo, si *P* fuese igual a 0.25, esta relación sería igual a 0.75 y el peso habría perdido el 25% de su valor. Por lo anterior, al argumento *P* se le puede llamar “el factor de pérdida”.

Con los resultados anteriores se observa que el Método STP obliga a que la pérdida de peso de una cola se limite con relación al factor *P*, en el gran intervalo de tiempo *T*. La ecuación (11), entonces, se transforma en la ecuación (15).

$$\Delta_i(t_0+r\tau)=ln(1-P)\frac{\tau}{T}\phi_i(t_0+r\tau) \tag{15}$$

También, es importante notar que, como $\tau / T << 1$, entonces $|\Delta_i(t_0 + r\tau)| << 1$, y $\sum_{\forall i}|\Delta_i(t_0+r\tau)| << 1$.

Combinando las ecuaciones (10) y (15) se obtiene:

$$\phi_i(t_0+(r+1)\tau)=\phi_i(t_0+r\tau)\left(1+\frac{\tau}{T}ln(1-P)\right) \tag{16}$$

Una comparación de la ecuación (16) (simplificada haciendo $t_0 = t$ y $r = 0$, y despejándola), con la ecuación (7) (la Segunda Propuesta de Cambio de Pesos), se da en la Figura 28, en donde se observa que en la ecuación (16) el cambio de peso, para una cola *i*, solo depende del peso anterior de la cola. En el caso de la ecuación (7) ese cambio depende del peso y de la longitud relativa, anteriores de la cola.

$$\phi_i(t+\tau)-\phi_i(t)=\left(\frac{\tau}{T}ln(1-P)\right)\phi_i(t)$$

$$\phi_i(t+\tau)-\phi_i(t)=-\alpha\,\phi_i(t)+\alpha\frac{\overline{Q_i(t+\tau)}\,\phi_i'}{\sum_{k=1}^{N}\overline{Q_k(t+\tau)}\,\phi_\ell'} \quad i\in\{1,\ldots,N\},\ 0<\alpha<1$$

Figura 28. Comparación entre ecuaciones de las propuestas segunda y tercera, para cambiar los pesos de las colas.

7.3.3 Prueba de la Ecuación (12)

Para la prueba de la ecuación (12) se acude a un desarrollo comúnmente utilizado en el cálculo del interés compuesto [79]. Esta prueba se incluye porque permite al lector tener una mejor referencia, y porque una de sus expresiones se utiliza para plantear el comportamiento o conducta de disminución máxima del valor de un peso.

Considere que la cola *i* obtiene T/τ resultados consecutivos que indican que tiene que perder peso. Como se ha indicado, τ y *T* son parámetros que representan intervalos de tiempo, siendo el primero, pequeño y el segundo, de tal forma que T/τ es un entero de tres órdenes de magnitud (por ejemplo 1000).

Las ecuaciones (10) y (11) se repiten aquí, por conveniencia, como las ecuaciones (17) y (18).

$$\Delta_i(t_0+r\tau)=\phi_i(t_0+(r+1)\tau)-\phi_i(t_0+r\tau) \tag{17}$$

$$\Delta_i(t_0+r\tau)=f\frac{\tau}{T}\phi_i(t_0+r\tau) \tag{18}$$

Hay que notar que de la ecuación (17) se puede apreciar que mientras menor sea el tamaño medio de una cola más lentamente la misma perdería peso.

Por facilidad, al intervalo (de tiempo) que trascurre desde $t_0+r\tau$ hasta $\left(t_0+(r+1)\tau\right)^-$ se le denomina como el primer intervalo a partir de t_0. Combinando las ecuaciones (17) y (18) se obtiene la ecuación (19), que expresa el peso de la cola *i* en el siguiente intervalo, después del primero, como función del peso de dicha cola en dicho primer intervalo.

$$\phi_i(t_0+(r+1)\tau)=\phi_i(t_0+r\tau)\left(1+\frac{f}{T/\tau}\right) \tag{19}$$

De manera recursiva, una relación similar, para el subsiguiente intervalo sería:

$$\phi_i(t_0+(r+2)\tau)=\phi_i(t_0+(r+1)\tau)\left(1+\frac{f}{T/\tau}\right) \tag{20}$$

Al combinar las ecuaciones (19) y (20) se obtiene la ecuación (21), que también queda en función del peso de dicha cola en el primer intervalo.

$$\phi_i(t_0+(r+2)\tau)=\phi_i(t_0+r\tau)\left(1+\frac{f}{T/\tau}\right)^2 \tag{21}$$

De igual forma, considerando los siguientes T/τ intervalos consecutivos, se obtiene la ecuación (22):

$$\phi_i(t_0+(r+T/\tau)\tau)=\phi_i(t_0+r\tau)\left(1+\frac{f}{T/\tau}\right)^{T/\tau} \tag{22}$$

Siendo el valor T/τ grande, entonces el factor con el exponente puede aproximarse como:

$$\left(1+\frac{f}{T/\tau}\right)^{T/\tau}\approx e^f \tag{23}$$

Con las ecuaciones (22) y (23) se obtiene la ecuación (24):

$$\phi_i\left(t_0+\left(r+\frac{T}{\tau}\right)\tau\right)\approx\phi_i(t_0+r\tau)\,e^f \tag{24}$$

Y, restando $\phi_i(t_0+r\tau)$ de cada miembro de la ecuación (24) se obtiene:

$$\begin{aligned}&\phi_i\left(t_0+\left(r+\frac{T}{\tau}\right)\tau\right)-\phi_i(t_0+r\tau)\approx\phi_i(t_0+r\tau)\left(e^f-1\right)\\&\phi_i\left(t_0+r\tau+T\right)-\phi_i(t_0+r\tau)\approx\phi_i(t_0+r\tau)\left(e^f-1\right)\end{aligned} \tag{25}$$

Con lo que se demuestra la ecuación (12).

7.3.4 Conducta de Cambio de los Pesos que Pierden Valor Constantemente

La ecuación (21) se puede generalizar como: $\phi_i\left(t_0+\left(r+n\right)\tau\right)=\phi_i\left(t_0+r\tau\right)\left(1+\frac{f}{T/\tau}\right)^n$

Haciendo $t_0 = 0$ *s* y $r = 0$, y $n\tau = t$, la anterior ecuación se puede aproximar como:

$$\phi_i(t) = \phi_i(0)\left(1+\frac{f}{T/\tau}\right)^{t/\tau} = \phi_i(0)\left(1+\frac{\ln(1-P)}{T/\tau}\right)^{t/\tau} \quad (26)$$

Esta ecuación representa el valor de un peso, donde el valor disminuye en cada intervalo, desde un tiempo inicial en 0 *s* hasta un tiempo final *T s* después. Se observa que si *t* = *T* se cumple que $\phi_i(T) = \phi_i(0)\left(1+\frac{f}{T/\tau}\right)^{T/\tau} \approx \phi_i(0)e^{f} = \phi_i(0)e^{\ln(1-P)} = \phi_i(0)(1-P)$, con lo que se vuelve a obtener que $\phi_i(T)/\phi_i(0) = (1-P)$, similar a la ecuación (14).

Por otro lado, la tasa de cambio del peso es $\frac{d\phi_i(t)}{dt} = \phi_i(0)\frac{d\left(1+\frac{\ln(1-P)}{T/\tau}\right)^{t/\tau}}{dt} =$

$\phi_i(0)\left(1+\frac{\ln(1-P)}{T/\tau}\right)^{t/\tau}\ln\left(1+\frac{\ln(1-P)}{T/\tau}\right)\frac{1}{\tau}$.

Por ejemplo, para *t* = 0, $\phi_i(0)$ = 1/3, *P* = 0.25, τ = 1, se tiene que $\frac{d\phi_i(t)}{dt}$ =

$\frac{-9.99046\cdot10^{-5}\cdot0.999700331^{t/1(s)}}{1(s)}$. Para *t* = 0 *s* se tiene que $\left.\frac{d\phi_i(t)}{dt}\right|_{t=0} = \frac{-9.99046\cdot10^{-5}}{1(s)}$.

Si se continuara con esta pendiente, para obtener una pérdida del 25% del peso inicial, es decir para que la delta fuese de –0.25 x 1/3 = –1/12, en lugar de requerir un tiempo *T* = 960 *s*, se requeriría un tiempo Δt tal que:

$-9.99046\text{x}10^{-5}$ x Δt = –1/12, lo que da Δt = 834 *s*.

En la Figura 29 se grafica la ecuación (26), cuando el peso inicial $\phi_i(0)$ = 1/3, para diversos valores de *P*, *T* = 960 *s* y τ = 1 *s*.

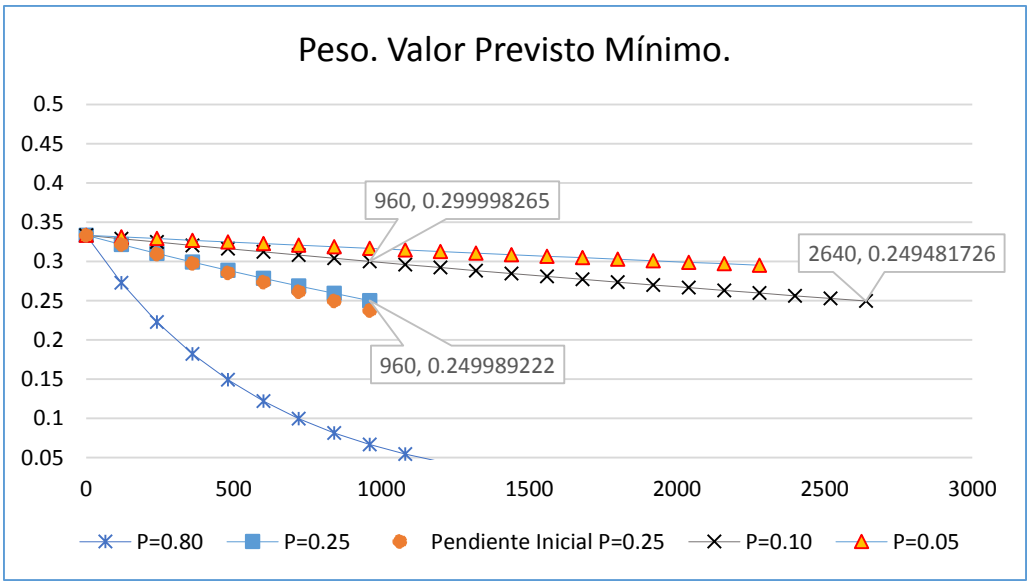


Figura 29. Valor de cambio del peso de una cola, a través del tiempo, con su tendencia al mínimo, con el uso del Método STP.
El peso inicia con un valor de 1/3, cuando en cada intervalo de tiempo de tamaño τ, durante el lapso T = 960 *s*, el peso se marca para perder valor. Se muestran gráficas para diversos valores de P. Se incluye la recta de tendencia con pendiente inicial, para P = 0.25.

En la Figura 29 se observa que en el tiempo T = 960 *s*, para las curvas asociadas a los valores de P, el valor del peso inicial ha disminuido un (P x 100)%. Por ejemplo, para P = 0.25 el valor del peso, en el tiempo T = 960 *s*, ha llegado a 1/3 x (1 – 0.25) = 0.25, y para P = 0.10 el valor del peso ha llegado a 1/3 x (1 – 0.10) = 0.30. Se observa que la curva de P = 0.10 llega "casi" al valor de 0.25 en el tiempo 2640 *s*.

7.3.5 Cálculo de Retrasos Promedio en Nodo c_0

A continuación se hace un análisis teórico del tiempo de espera en el sistema de atención de colas en el nodo c_0. Para esto se considera que el sistema es del tipo $M/D/1$, donde M indica un proceso de llegadas de paquetes tipo poissoniano; D indica un tipo de servicio determinista (el tiempo de atención por paquete es constante); y 1 indica que hay un solo servidor para el sistema de atención de tráfico. Se considera un tiempo e observación desde 0 a 1560 *s*, durante el cual se hacen cálculos en este análisis.

El proceso de llegadas que se maneja en esta tesis no necesariamente es tipo poissoniano. El considerar este tipo de proceso de llegadas permite un desarrollo matemático simplificado para tener resultados que se puedan comparar con los resultados de retrasos medidos en esta tesis.

La nomenclatura que se usa, para los tiempos de espera de un paquete, en el sistema, es:

$$a_i = b_i + c_i \tag{27}$$

Donde a_i es el "intervalo de servicio"; b_i es el "tiempo de espera en cola", y c_i es el "tiempo de servicio". Los parámetros λ y μ son la tasa promedio de llegadas de paquetes y la tasa promedio de atención de paquetes, respectivamente. Se observa que $E\{c_i\} = 1/\mu$. El parámetro ρ es la intensidad de tráfico ofrecido ("Offered Load") que hay en el sistema, que se calcula como λ/μ.

Para este sistema el tiempo de espera, promedio, total es:

$$E\{a_i\} = \frac{1}{2\mu} \cdot \frac{2-\rho}{1-\rho} \tag{28}$$

La tasa de la interfaz de salida es de 30 *Mb*/*s*.

Se considera que al sistema llegan flujos de tres diferentes rutas (sean éstas la ruta 1, la ruta 2 y la ruta 3). Durante la parte inicial de tiempo de observación, que va de 0 a 600 *s*, el sistema recibe 12 flujos por cada ruta, para un total de 36 flujos. Cada flujo tiene un promedio de 0.621 *Mb*/*s*. La gran generalidad de paquetes manejados tiene 1000 *Byte*. A los 600 *s*, la ruta 2 incrementa su tráfico en 4 fuentes (para un total de 40 flujos de llegada), y posteriormente (y hasta los 1560 *s*) ya no hay cambio en la cantidad de tráfico en esa ruta. Las otras 2 rutas no cambian de cantidad de tráfico en todo el tiempo.

Si el sistema tuviese 1 cola en donde se alojara el tráfico que esperase atención, proveniente de las tres rutas, el sistema sería un sistema operando con un Método *q1*. Para este sistema, en el lapso inicial, el valor de λ sería: (36·0.621 *Mb*/s)/(8000 *b*/*paq*) = 2794.5 *paq*/*s*; el valor de μ sería: 30 *Mb*/*s* / 8000 *b*/*paq* = 3750 *paq*/*s*; y el valor de ρ sería: 2794.5/3750 = 0.7452.

Para el caso del Método *q1*, la Tabla 3 muestra los parámetros calculados requeridos para el cálculo del tiempo de servicio $E\{a_i\}$, y el valor de dicho tiempo, calculado mediante la ecuación (28). Se observa que dicho tiempo es menor a 1 *ms* en los renglones de la tabla.

# Colas	**# Flujos de Llegada**	**λ**	**Ancho de banda asegurado**	**μ**	**ρ**	**E{a_i}**	**Comentario**
		paq/s	***Mb/s***	***paq/s***	**--**	***ms***	--
1	36	2794.5	30.0	3750.0	0.7452	0.657	En la cola única
1	40	3105.0	30.0	3750.0	0.8280	0.909	

Tabla 3. Tiempo promedio de servicio calculado en un sistema de atención de tráfico *M*/*D*/1, usando el Método *q1*. Se muestran otros parámetros requeridos para el cálculo.

Una variante del mismo sistema sería aquél que tuviese 3 colas en lugar de 1. Habría una cola para alojar el tráfico de cada ruta, y se usaría un despachador ahorrador de energía para la atención de las colas, de tal forma que repartiese el ancho de banda de la salida del sistema entre el tráfico de las 3 colas, según el tamaño promedio relativo de cada cola. El despachador, por ser ahorrador de energía, permitiría que para la atención de cada cola se utilizara el ancho de banda ocioso de las otras dos colas. La atención de tráfico de este sistema tiene las características del Método STP de atención de tráfico, con excepción de que faltaría definir la conducta del cambio de pesos.

Nuevamente, durante el lapso inicial de observación, de 0 a 600 *s*, la tasa promedio de tráfico de entrada sería de 12 flujos por cada ruta. La operación del despachador debería permitir que la repartición del ancho de banda se estabilizara para ser igual, con un 33.33% del ancho de banda para el tráfico de cada ruta. Para este despachador, este porcentaje sería el mínimo garantizado para cada ruta.

En el lapso inicial, para cualquiera de las tres rutas, λ sería igual a: (12·0.621 *Mb*/s)/(8000 *b*/*paq*) = 931.5 *paq*/*s*. En el lapso subsecuente, para la ruta 2, λ sería: (16·0.621 *Mb*/s)/(8000 *b*/*paq*) = 1242 *paq*/*s*.

Con un despachador que NO FUESE ahorrador de energía, durante el lapso inicial el valor de μ para la atención de cualquiera de las rutas sería: (30/3 *Mb*/*s*)/(8000 *b*/*paq*) = 1250 *paq*/*s* (es importante indicar que el valor de μ sería mayor para un despachador ahorrador de energía).

Al aumentar de tráfico la ruta 2, su cola asociada aumentaría de tamaño, y eso generaría un cambio en los pesos. En la Figura 29 se indicó el mayor cambio de pesos que puede haber, correspondiente a cada valor del parámetro *P*, y con un valor del parámetro *T* de 960 *s*, con el uso del Método STP; sin embargo, el mayor cambio no llegaría a darse, a menos que el tamaño de la cola asociada a la ruta 2 fuese mayor a las otras 2 colas, en cada evaluación que hiciese el Método STP, en cada segundo, desde el tiempo 601 hasta el 1560.

Una opción para calcular el cambio de pesos que habría es considerar que el peso final que tendría la cola asociada a la ruta 2 sería igual a la proporción del tráfico relativo de llegada de dicha ruta. Esa proporción es 16/40 = 0.4. Las otras colas tendrían, cada una, una proporción de 12/40 = 0.3. Considerando que el peso inicial de las colas es de 1/3, entonces el porcentaje de pérdida de tamaño de cada una de las otras dos colas sería de (1 - 0.3 / (1/3))·100 = 10%. Entonces, se puede hacer el cálculo de los cambios de peso de las rutas 1 y 3 considerando un valor de *P* = 0.1 (recordando que con *P* = 0.1 indica el porcentaje de 10% de pérdida del valor de peso), haciendo mención de que este cambio sería forzado, con el uso de la ecuación

(26), y no de manera "natural" del método STP, mediante la comparación de los tamaños de las colas y la aplicación de la ecuación (18)[83].

Para el caso de un despachador que *no fuese* ahorrador de energía, la Tabla 4 muestra los parámetros calculados requeridos para el cálculo del tiempo de servicio $E\{a_i\}$, y el valor de dicho tiempo, calculado mediante la ecuación (28). La Tabla 4 muestra los valores de los pesos calculados con la ecuación (26) (con un parámetro *P* = 0.1 por lo explicado en el párrafo anterior). En la tabla se observa que los pesos se mantienen iguales en los intervalos de tiempo, hasta aquél que concluye en el tiempo 600 *s*.

Desde los 600 *s*, los valores del "ancho de banda asegurado", μ y ρ no se incluyen debido a que los mismos se calculan, por separado, cada segundo que trascurre dentro de cada intervalo. El valor del tiempo de servicio $E\{a_i\}$ proviene del promedio obtenido de su cálculo, segundo a segundo, en cada intervalo. Una porción de la lista de los cálculos segundo a segundo se muestra en la Tabla 5.

Tiempo Inicial	Tiempo Final	Tiempo a partir de aumento	Peso Rutas 1 y 3 (promedio en intervalo)	Peso Ruta 2 (promedio en intervalo)	# Flujos de Llegada Ruta 2	λ Ruta 2	Ancho de banda asegurado Ruta 2	μ Ruta 2	ρ Ruta 2	$E\{a_i\}$ Ruta 2 (promedio en interv9
s					flujos	*paq/s*	*Mb/s*	*paq/s*	--	*ms*
0	120		0.3333	0.3333	12	931.5	10.00	1250.0	0.7452	1.97
120	240		0.3333	0.3333	12	931.5	10.00	1250.0	0.7452	1.97
240	360		0.3333	0.3333	12	931.5	10.00	1250.0	0.7452	1.97
360	480		0.3333	0.3333	12	931.5	10.00	1250.0	0.7452	1.97
480	600	0	0.3333	0.3333	12	931.5	10.00	1250.0	0.7452	1.97
600	720	120	0.3311	0.338	16	1242.0				25.015
720	840	240	0.3268	0.346	16	1242.0				9.184
840	960	360	0.3225	0.355	16	1242.0				6.049
960	1080	480	0.3183	0.363	16	1242.0				4.532
1080	1200	600	0.3141	0.372	16	1242.0				3.660
1200	1320	720	0.3100	0.380	16	1242.0				3.093
1320	1440	840	0.3060	0.388	16	1242.0				2.692
1440	1560	960	0.3020	0.396	16	1242.0				2.395

Tabla 4. Tiempo promedio de servicio calculado en un sistema de atención de tráfico *M*/*D*/1, usando el Método STP, con un despachador ***no*** ahorrador de energía. Se muestran otros parámetros requeridos para el cálculo.

[83] Con la aplicación de los pasos del método, que se dan en el subcapítulo 7.4. Como se puede observar en los resultados del subcapítulo 8.2, en la operación simulada de un sistema de atención de tráfico con el Método STP, esta disminución de peso correspondería a un valor de *P*=0.25.

Tiempo Inicial	Tiempo Final	Tiempo a partir de aumento	Peso Rutas 1 y 3	Peso Ruta 2	# Flujos de Llegada Ruta 2	λ Ruta 2.	Ancho de banda asegurado Ruta 2	μ Ruta 2	ρ	$E\{a_i\}$ Ruta 2
s					flujos	*paq/s*	*Mb/s*	*paq/s*	--	*ms*
600	601	1	0.3333	0.3334	16	1242.0	10.0022	1250.3	0.99338	60.83
601	602	2	0.3333	0.3335	16	1242.0	10.0044	1250.5	0.99316	58.89
602	603	3	0.3332	0.3336	16	1242.0	10.0066	1250.8	0.99295	57.07
603	604	4	0.3332	0.3336	16	1242.0	10.0088	1251.1	0.99273	55.36
604	605	5	0.3332	0.3337	16	1242.0	10.0110	1251.4	0.99251	53.75

Tabla 5. Tabla de complemento a la Tabla 4, con una lista parcial de los tiempos promedio de servicio calculados cada segundo, para obtener los valores promedio en lapsos de 120 *s*.

Los valores del tiempo final por lapso, y el retraso promedio de la ruta 2, mostrado en la Tabla 4, se muestra en la Figura 30. Asimismo, en esa figura se muestran los valores correspondientes al retraso promedio calculado para el tráfico de la ruta 1.

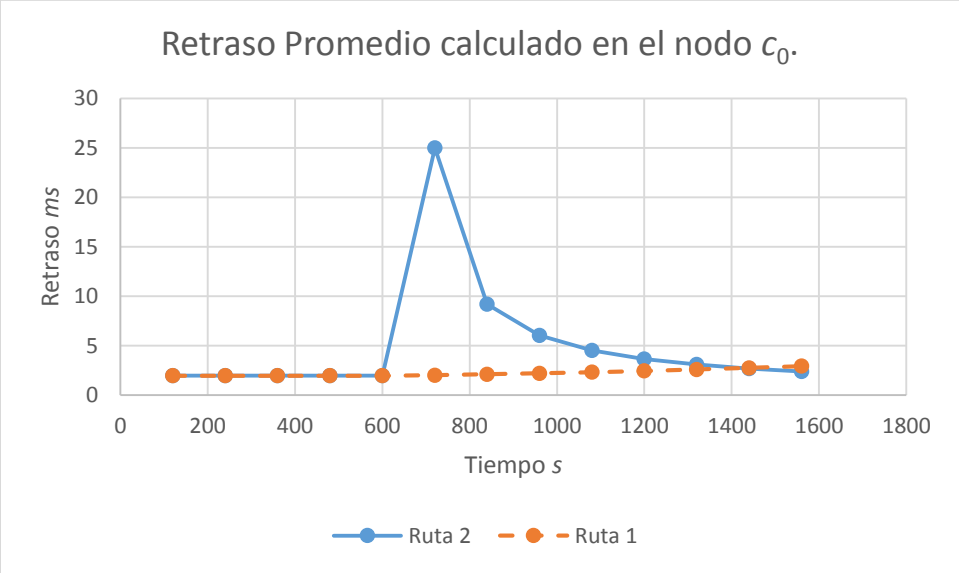


Figura 30. Retrasos teóricos promedio en las colas de un sistema con el uso del Método STP de atención de tráfico, correspondiendo a un sistema *M*/*D*/1, pero con un despachador *no* ahorrador de energía.

Es importante recalcar que estos valores calculados de retraso promedio se refieren a un sistema de colas asiladas una de otra, es decir, donde el despachador NO ES ahorrador de energía.

Para el caso en donde el despachador es ahorrador de energía, los retrasos promedio serían menores, aunque no menores que los retrasos para un sistema atendido con el Método *q1*, cuyos valores de retraso se muestran en la Tabla 3.

7.4 Pasos del Método STP

INICIO.

Requerimiento: En $t = t_0 + r\tau$

Ya sea que los pesos de las colas se dejaron intactos, o los mismos acaban de ser calculados. En el último caso, con los pesos calculados, asegurarse que[84]:

$$\sum_{i=1}^{N} \phi_i(t_0 + r\tau) = 1$$
$$0 \le \phi_i(t_0 + r\tau) \le 1 \qquad (29)$$

Para actualizar los pesos de las colas, en uso, realice los siguientes pasos.

Requerimiento: En $t = (t_0 + (r+1)\tau)^-$

si {todas las colas están vacías} **entonces**

Hacer $t \leftarrow t_0$

Los pesos se dejan intactos.

$r \leftarrow r + 1$

Ir al INICIO[85].

fin si

En esta etapa al menos una de las colas no está vacía, así que el cálculo para obtener un nuevo conjunto de pesos para las colas está a punto de comenzar.

En este intervalo τ, recientemente concluido, obtener las longitudes promedio de las colas, y los valores de los pesos en uso.

para $\{i = 1 \rightarrow N\}$ **hacer**

Calcular el indicador $I_i(t_0 + (r+1)\tau)^-$, como:

[84] Cada uno de los pesos calculados se divide entre la suma de estos pesos. Esto corrige cualquier posible desviación de 1, muy pequeña, que dicha suma pudiese tener.

[85] Una revisión posterior de este algoritmo indica que es suficiente y mejor buscar si la suma del tamaño de las colas es cero. En caso afirmativo solamente requeriría hacer *r* = 0 e ir al INICIO (dejando los pesos intactos).

$$I_i(t_0+(r+1)\tau)^- \leftarrow \frac{\overline{Q_i(t_0+(r+1)\tau)}^-}{\sum_{j=1}^{N}\overline{Q_j(t_0+(r+1)\tau)}^-} - \phi_i(t_0+r\tau) \tag{30}$$

Nota. El Segundo miembro de la expresión (30) tiene dos términos. El primer término es la longitud relativa de la cola i. El Segundo término, $\phi_i(t_0+r\tau)$, es el peso en uso de la cola i. Entonces, este indicador es una comparación de estos dos valores.

si $I_i(t_0+(r+1)\tau)^- < 0$ **entonces**

La cola *i* se marca para perder peso, y se considera que está en el conjunto *L* de colas (donde *L* significa *Loss* –pérdida).

$\Delta_i(t_0+r\tau)$ se calcula con la ecuación (15), que se repite, como una expresión de asignación, en la expresión (31).

$$\Delta_i(t_0+r\tau) \leftarrow ln(1-P)\frac{\tau}{T}\phi_i(t_0+r\tau) \tag{31}$$

fin si $I_i(t_0+(r+1)\tau)^- < 0$

si $I_i(t_0+(r+1)\tau)^- = 0$ **entonces**

Para esta situación, no común, se considera que la cola i está en el conjunto *L* de colas, y a su incremento de peso se le asigna el valor 0:

$$\Delta_i(t_0+r\tau) \leftarrow 0 \tag{32}$$

fin si $I_i(t_0+(r+1)\tau)^- = 0$

si $I_i(t_0+(r+1)\tau)^- > 0$ **entonces**

La cola *i* se marca para ganar peso, y se le considera estar en el conjunto *G* de colas (donde *G* significa *Ganancia*).

No se realiza cálculo de $\Delta_i(t_0+r\tau)$, sino hasta después del presente lazo de programación.

fin si $I_i(t_0+(r+1)\tau)^- > 0$

fin para $\{i=1\rightarrow N\}$

para $\forall i \in G$ **hacer**

$$\Delta_i(t_0 + r\tau) \leftarrow \frac{I_i(t_0 + (r+1)\tau)^-}{\sum_{j \in G} I_j(t_0 + (r+1)\tau)^-} \left[-\sum_{j \in L} \Delta_j(t_0 + r\tau) \right] \quad (33)$$

fin para $\forall i \in G$

$r \leftarrow r + 1$

Ir a INICIO

FIN

Observaciones acerca del Método STP.

Cuando se suman ambos miembros de la expresión (33), $\forall i \in G$, se obtiene la expresión (34):

$$\sum_{i \in G} \Delta_i(t_0 + r\tau) \quad \leftarrow \quad \frac{\sum_{i \in G} I_i(t_0 + (r+1)\tau)^-}{\sum_{j \in G} I_j(t_0 + (r+1)\tau)^-} \left[-\sum_{j \in L} \Delta_j(t_0 + r\tau) \right] \quad (34)$$

De tal forma que:

$$\sum_{i \in G} \Delta_i(t_0 + r\tau) = -\sum_{j \in L} \Delta_j(t_0 + r\tau) \quad (35)$$

Así que la suma de los incrementos de los pesos (los que ganan valor) es igual al negativo de la suma de los decrementos de pesos (los que pierden valor), manteniendo la suma de los pesos en 1, por norma.

Resumen y Conclusiones del Capítulo

Al principio de este capítulo se dan definiciones de conceptos antecedentes importantes para la concepción del Método STP.

Entre los conceptos que se presentan está el que indica que

Se plantea el Método STP como un método que opera dentro de cada interfaz de salida de cada nodo central, donde se opera de tal forma que hay una cola para el tráfico proveniente de cada interfaz de entrada en el nodo, y que el tráfico de cada interfaz de entrada corresponde al tráfico de una ruta.

La protección del método se aplica a las colas de la interfaz de salida, por medio del peso de las colas. El Método STP da protección de las rutas mediante la protección de las colas de las rutas, en los nodos por donde pasa la ruta.

El método se concibe para que una ruta que aumenta de tráfico vaya ganando peso, en cada cola dentro de los nodos por donde pase, con el objeto de que la ruta adquiera protección

para el tráfico que admite; sin embargo, el peso que gana no es abrupto, de tal manera que se pueda proteger a las rutas interferidas contra el aumento de tráfico, pero que “en el largo plazo” los pesos de las colas en los nodos correspondan a la relación de tráfico que se tenga entre las rutas que se interfieren. Lo anterior implicaría que una ruta que demanda más tráfico, adquiriría, en el largo plazo, protección para el tráfico que ha adquirido.

Se obtiene la expresión matemática del comportamiento o conducta de disminución máxima que puede tener el peso de una cola, que se da en la ecuación (26).

Se espera que el método permita una administración simple de la red, con prospecto de ser escalable hacia al menos decenas de nodos, sea sencillo de manejar y se ajuste de acuerdo a las necesidades de ancho de banda de las rutas, y sea útil para los administradores de las rutas de la red, para tener confianza, por lo menos dentro de una ventana de tiempo, sobre la cantidad de ancho de banda con que se cuenta en cada ruta.

Se plantean las hipótesis generales y una hipótesis específica, del Método STP (no se repiten aquí).

En un sistema de atención de tráfico con varias colas, la proporción del tamaño de las colas se utiliza como una aproximación a la proporción del tráfico entrante, para dar equidad de atención a las colas.

Se menciona que la aproximación indicada puede no ser tan buena, dependiendo del tipo de tráfico que se maneje. Por lo anterior, la posibilidad de que en los nodos se lleve cuenta del tráfico promedio entrante a las colas para calcular con eso el peso de las mismas (en lugar de utilizar el tamaño promedio de las colas) se vislumbra como una alternativa deseable para el Método STP.

Capítulo 8. Validación Experimental del Método STP

Introducción

En este capítulo se verifica, por medio de experimentación por simulación, la validez de las hipótesis sobre el comportamiento de los pesos cuando se usa el Método STP, aplicado a la red del escenario de trabajo de esta Tesis. Se observa este comportamiento en función de dos valores seleccionados para *P* y se fundamente el uso de uno u otro valor en el método. El valor del parámetro *T* se fija en 960 *s* (16 minutos).

También se observan los retrasos en las colas del nodo de interferencia, y los retrasos en las rutas, como consecuencia de aumentos de tráfico y de cambio en los pesos de las colas.

Finalmente en el escenario de trabajo de esta Tesis se utiliza un retraso máximo de 19 *ms* para las rutas, con el fin de comparar los resultados de retrasos, en forma cuantitativa, con el uso de un indicador de ganancia en la red, cuando se utiliza el Método STP, y cuando se utiliza el Método q1.

Se plantean las contribuciones de este método. Se dan las conclusiones.

Este método y resultados de experimentos con el mismo se publican en [80] (ver el Anexo A.1).

8.1 Selección de los Valores del Argumento P para el Método STP

Con los siguientes resultados experimentales se dan razones para seleccionar los valores de 0.1 y 0.25 para el parámetro *P* del Método STP.

8.1.1 Uso del Parámetro *P* con Valor 0.25

La Figura 31 muestra los pesos de las colas en la interfaz de salida del nodo c_0 (hacia el nodo c_1) asociadas a las tres rutas, s_1-d_1, s_2-d_2 y s_3-d_3, para diversos aumentos en el tráfico de la ruta s_2-d_2. La Tabla 6 ayuda a interpretar los valores de los parámetros utilizados en los diversos cuadros de la Figura 31. Se muestran estos pesos para los casos correspondientes a los incrementos de tráfico, en la ruta s_2-d_2, de 2, 4, 6 y 10 fuentes, respectivamente, en el tiempo 600 *s*. Las utilizaciones del enlace de salida del nodo c_0, para los diversos cuadros de la Figura 31, después de los 600 *s* de tiempo experimental, son de 0.7866, 0.8280, 0.8694, 0.9522 (el valor de ρ en la interfaz de salida del nodo c_0).

	Aumento de fuentes en flujo s_2-d_2 a 600 *s*			
P = 0.25	A →	2	B →	4
	C →	6	D →	10

Tabla 6. Valores del aumento de tráfico en la ruta s_2-d_2 según el cuadro de la Figura 31.

La Figura 31 también muestra, en cada cuadro, la curva de tendencia mínima que puede tener cada peso, la que se presenta en la Figura 29 (en la página 122) para $P = 0.25$, la cual depende de los valores de T y P, del Método STP, según lo indica la ecuación (26), con la diferencia de que las curvas de esa figura indican su tendencia del tiempo 0 *s* al tiempo 960 *s* (por un lapso de tiempo de $T = 960$ *s*), mientras que la curva de tendencia mínima, en la Figura 31, se recorre en el tiempo, indicando su cambio de los 600 *s* a los 1560 *s* (también por un lapso de tiempo de $T = 960$ *s*).

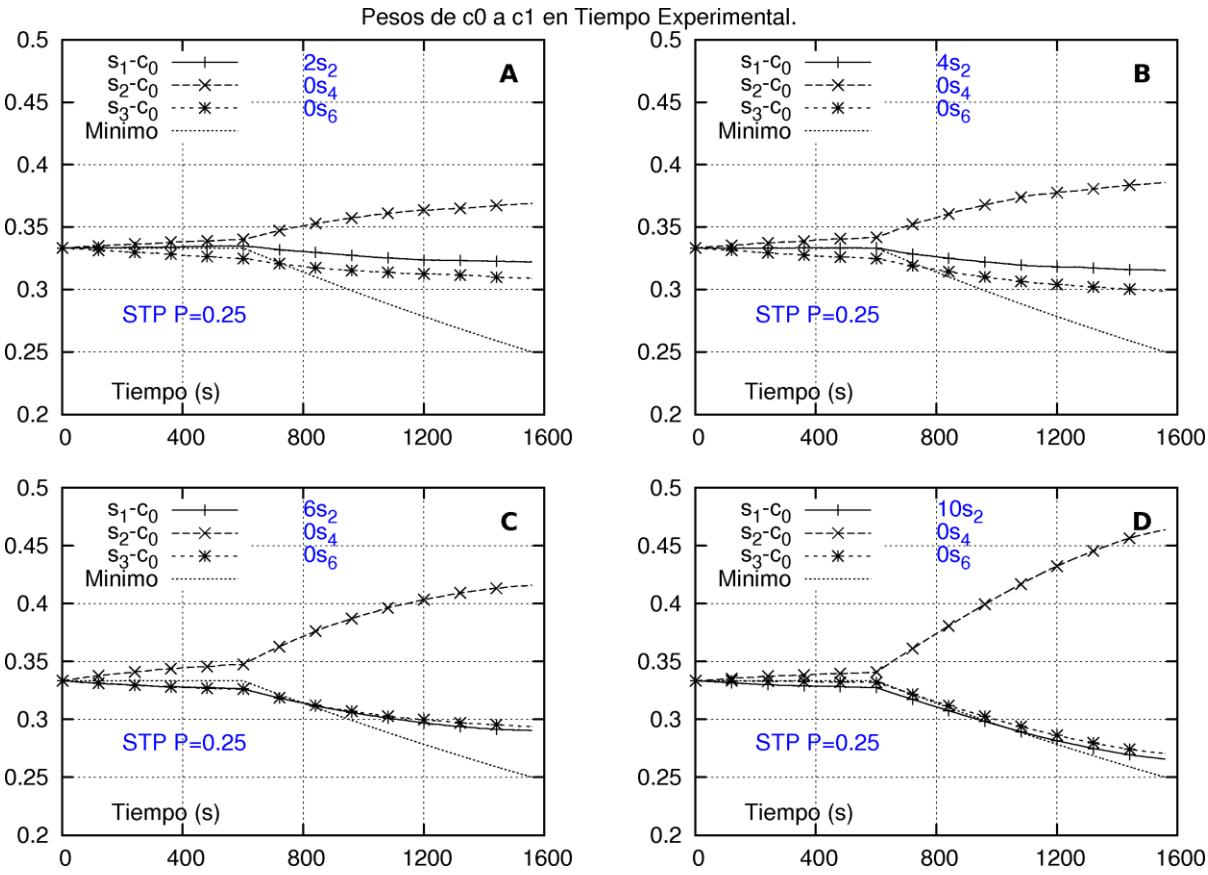


Figura 31. Pesos de las colas de las rutas s_1-d_1, s_2-d_2, s_3-d_3, en la interfaz de salida del nodo c_0, con el uso del Método STP para la atención de tráfico. Se incluye la curva de valor mínimo que pueden tener los pesos.
En los cuadros "A", "B", "C" y "D" la ruta s_2-d_2 aumenta de tráfico en 2, 4, 6 y 10 fuentes, respectivamente, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Note que, por norma, la suma de los pesos de las tres colas es igual 1, para cada punto de abscisa en cualquiera de las gráficas. Asimismo, los pesos para las colas de las rutas s_1-d_1 y s_3-d_3 son los pesos que siguen más cercanamente la tendencia mínima.

Observe que la curva de tendencia mínima de peso llega a un valor de 1/3(1-0.25) = 0.25 en el tiempo 1560 *s* (equivale a 600 *s* + 960 *s*).

En el subcapítulo 7.1.5 se muestran expresiones que comparan el tráfico relativo con los pesos en las rutas.

El tráfico relativo que se tiene de las diversas rutas, correspondiente a los resultados del cuadro "A" de la Figura 31 (con un aumento de 2 fuentes en el flujo de la ruta s_2-d_2 a los 600 *s*), se resume en la Tabla 7.

	0-600 *s*		**600-1560 *s***		
Flujo	**Tráfico**	**Tráfico Relativo**	**Tráfico**	**Tráfico Relativo Equivalente al Valor Objetivo del Peso.**	**Disminución del Tráfico Relativo en %**
s_1	12	12/36 = 0.333	12	12/38 = 0.316	100 x [12/36 – 12/38] / (12/36) = 5.26%
s_2	12	12/36 = 0.333	14	14/38 = 0.368	Aumenta en igual valor que la suma de las disminuciones.
s_3	12	12/36 = 0.333	12	12/38 = 0.316	100 x [12/36 – 12/38] / (12/36) = 5.26%
Suma	36	1	38	1	

Tabla 7. Tráfico relativo de las rutas s_1-d_1, s_2-d_2 y s_3-d_3, aplicado en el cuadro "A" de la Figura 31.

La Tabla 8 muestra la comparación entre los valores del tráfico relativo (que pudiesen llamarse "pesos objetivo") y los pesos observados de las colas, después de 600 *s*, que muestra el cuadro "A" de la Figura 31**.**

	Valores según tráfico relativo.	**Valores observados a los 1560 *s* en el experimento.**
Cola s_1-d_1	0.316	0.322
Cola s_2-d_2	0.368	0.369
Cola s_3-d_3	0.316	0.309

Tabla 8. Valores objetivo de los pesos de las rutas s_1-d_1, s_2-d_2 y s_3-d_3, y valores observados, para esas rutas, en los resultados del cuadro "A" de la Figura 31.

Se observa que el peso de la cola s_2-d_2 prácticamente llega al objetivo, y los pesos de las otras colas quedan, 2.21% arriba y 1.89% abajo del objetivo, respectivamente[86].

Note que el valor objetivo del peso no depende del Método STP, sino únicamente de las condiciones dadas de tráfico en el experimento; sin embargo, el valor de *P* del Método STP es

[86] (0.316-0.309)/0.316 = 0.0221; (0.322-0.316)/0.316=0.01898.

importante para que el Método STP permita la suficiente disminución de los pesos de las colas, cuando el tráfico relativo de las mismas es constantemente menor al promedio de tráfico de todas las colas, en un lapso de tiempo T = 960 s.

		0-600 *s*		**600-1560 *s***				
Cuadro de la Figura 31.	**Flujo**	**Tráfico**	**Tráfico Relativo**	**Tráfico**	**Tráfico Relativo Equivalente al Valor Objetivo del Peso.**	**Disminución del Tráfico Relativo en %**	**Peso observado a los 1560 *s***	**Peso mínimo permitido cuando *P* = 0.25**
A	s_1	12	12/36 = 0.333	12	12/38 = **0.316**	100 x [12/36 – 12/38] / (12/36) = 5.26%	**0.322**	**0.25**
B					12/40 = **0.300**	100 x [12/36 – 12/40] / (12/36) = 10.0%	**0.315**	
C					12/42 = **0.286**	100 x [12/36 – 12/42] / (12/36) = 14.3%	**0.290**	
D					12/46 = **0.261**	100 x [12/36 – 12/46 / (12/36) = 21.73%	**0.266**	

Tabla 9. Valores objetivo de los pesos de la ruta s_1-d_1, y los valores observados, para esa ruta, en los resultados de los cuadros “A” a “D” de la Figura 31.

Los valores observados contra los valores “objetivo” tienen una diferencia, en todos los casos, de menos del 5%.

El uso del valor 0.25 para el parámetro P se justifica por ser un valor suficientemente grande para permitir que los pesos puedan cambiar de valor de manera suficientemente amplia en el curso del tiempo T, como se observa en la Tabla 9. En esta tabla se incluyen los valores “objetivo” de los pesos, y los valores medidos, correspondientes a los cuadros “A” a “D” de la Figura 31. El peso de la ruta s_1-d_1, en un lapso de tiempo T = 960 s, pasa de un valor inicial de 1/3=0.333 a un valor final de 0.315 (cercano a “objetivo” 0.300) cuando hay un aumento de 4 fuentes en la ruta s_2-d_2, al inicio de dicho lapso; y pasa a un valor final de 0.266 (cercano al “objetivo” 0.261) cuando el aumento es de 10 fuentes.

En la Tabla 9 se incluye también el peso mínimo que puede llegarse a obtener con el Método STP para los pesos (que se tendría al disminuir un peso el 25% de su valor). El peso 0.333, al disminuir 25%, llegaría al valor de 0.25[87].

[87] Es decir, 0.333 x (1 – 0.25) = 0.25.

8.1.2 Uso del Parámetro *P* con Valor 0.10

La Figura 32 es similar a la Figura 31, en el sentido de que se muestran los pesos de las colas en la interfaz de salida de c_0 (hacia el nodo c_1) de las rutas s_1-d_1, s_2-d_2 y s_3-d_3; pero la ruta s_2-d_2 aumenta su tráfico "solamente" en 4 fuentes a los 600 *s*. Esta figura también muestra las curvas de tendencia mínima para los pesos.

Los cuadros de la figura muestran, respectivamente, resultados con cambios que permiten sacar conclusiones, cuando el Método STP opera con valores de *P* de 0.1 y 0.25.

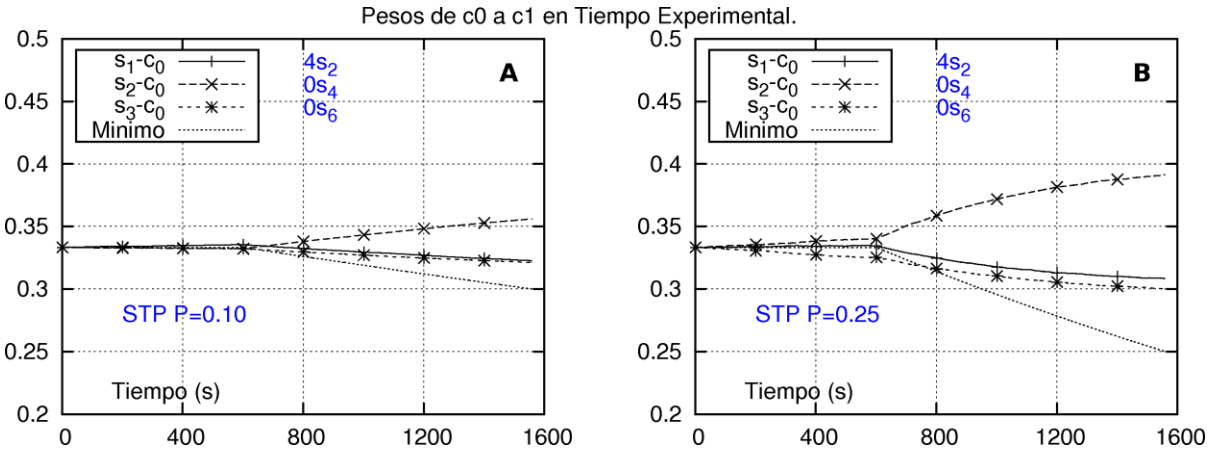


Figura 32. Pesos de las colas de las rutas s_1-d_1, s_2-d_2, s_3-d_3, en la interfaz de salida del nodo c_0, a través del tiempo experimental, con el uso del Método STP para la atención de tráfico. Se incluye la curva de valor mínimo que pueden tener los pesos.
En los cuadros "A" y "B" el valor del parámetro *P* del Método STP es de 0.1 y 0.25, respectivamente. En ambos cuadros la ruta s_2-d_2 aumenta de tráfico en 4 fuentes, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Observe que la curva de tendencia mínima de peso, para $P = 0.25$ (cuadro "B" de la figura) llega a un valor de 1/3(1-0.25) = 0.25 en el tiempo 1560 *s* (equivale a 600 *s* + 960 *s*). Asimismo, la curva de tendencia mínima de peso, para $P = 0.10$ (cuadro "A" de la figura), llega a un valor de 1/3(1-0.10) = 0.30 en el tiempo 1560 *s*.

El uso del valor 0.10 para el parámetro *P* se considera justificado porque con dicho valor la tendencia mínima de cambio de pesos en las rutas interferidas puede pasar de 0.333 a 0.300, en el curso del tiempo *T*, cuando hay un aumento de 4 fuentes en la ruta interferente, y ese aumento es el máximo que se considera en esta Tesis, para el escenario de trabajo[88]. Esto se

[88] En la sección de "Acotamientos de esta Tesis", se indica que el aumento de tráfico que se maneja en la mayor parte de los experimentos en el escenario de trabajo representa hasta el 11.1% del tráfico total inicial (4/36=0.11111).

explica mejor en la Tabla 10, que muestra los valores "objetivo" de los pesos, y los valores observados, correspondientes a los cuadros "A" y "B" de la Figura 32.

En la Tabla 10 se observa que con el uso de $P = 0.10$ el peso de la cola asociada a la ruta s_1-d_1 permite una disminución máxima de dicho peso, en un lapso de tiempo $T = 960$ *s*, desde 0.333 hasta 0.30 (justo el valor objetivo); aunque tal disminución no se observa en los resultados de la experimentación (donde el peso disminuye de 0.333 a 0.322), es decir, con un aumento de 4 fuentes, el valor de $P = 0.10$ es el mínimo valor de P que permite el movimiento completo de un valor de peso de 0.333, al valor objetivo que es 0.3.

Con motivos de comparación, la Tabla 10 incluye el peso mínimo que puede llegarse a obtener para valores de P de 0.1 y de 0.25.

			0-600 *s*		**600-1560 *s***				
Cuadro de la Figura 32.	***P***	**Flujo**	**Tráfico**	**Tráfico Relativo**	**Tráfico**	**Tráfico Relativo Equivalente al Valor Objetivo del Peso**	**Disminución del Tráfico Relativo en %**	**Peso observado a los 1560 *s***	**Peso mínimo permitido según el valor de *P*.**
A	0.10	s1	12	12/36 = 0.333	12	12/40 = **0.300**	100 x [12/36 – 12/40] / (12/36) = 10.0%	**0.322**	0.30
B	0.25							**0.315**	0.25

Tabla 10. Valores "objetivo" de los pesos, y valores observados, para esos pesos, en los resultados de los cuadros "A" y "B" de la Figura 32.

En la Tabla 11 se indican las razones para utilizar el valor de 0.1 o 0.25 para el argumento P, del Método STP.

Palor de P.	**0.1**	**0.25**	**Comentario**
Rapidez de ajuste del peso al valor objetivo.	Menor	Mayor	Con la mayor rapidez de cambio de los pesos de cada cola, hacia su valor objetivo, las rutas que pueden admitir tráfico adquieren peso más rápidamente, y con eso equidad en la competencia por ancho de banda.
Protección contra tráfico cruzado.	Mayor	Menor	Aumenta la protección de las rutas que no aumentan de tráfico.

Tabla 11. Justificaciones para el uso de los valores 0.25 ó 0.10 para el argumento P del Método STP.

8.2 Retrasos

Este subcapítulo muestra las gráficas referentes a los valores medidos de retraso, en los experimentos en el escenario de trabajo, tanto internamente en el nodo c_0, como de extremo a extremo en las rutas.

8.2.1 Retrasos en las Colas en el nodo c_0 según el Valor de P

La Figura 33 muestra los pesos de las colas relativas a las rutas s_1-d_1, s_2-d_2 y s_3-d_3, y los retrasos en el nodo c_0, a través del tiempo experimental. La ruta s_2-d_2 aumenta 4 fuentes a los 600 *s*. En el nodo c_0 se utiliza el método STP.

La Tabla 12 ayuda a interpretar lo que se muestra en la Figura 33, incluyendo los valores de los parámetros usados.

Método STP. La ruta s_2-d_2 aumenta **4** fuentes.		
	Pesos	Retrasos en c_0.
P = 0.10	**A**	**B**
P = 0.25	**C**	**D**

Tabla 12. Valores de los argumentos de *P*, del Método STP, usados en los experimentos reportados en los cuadros de la Figura 33.

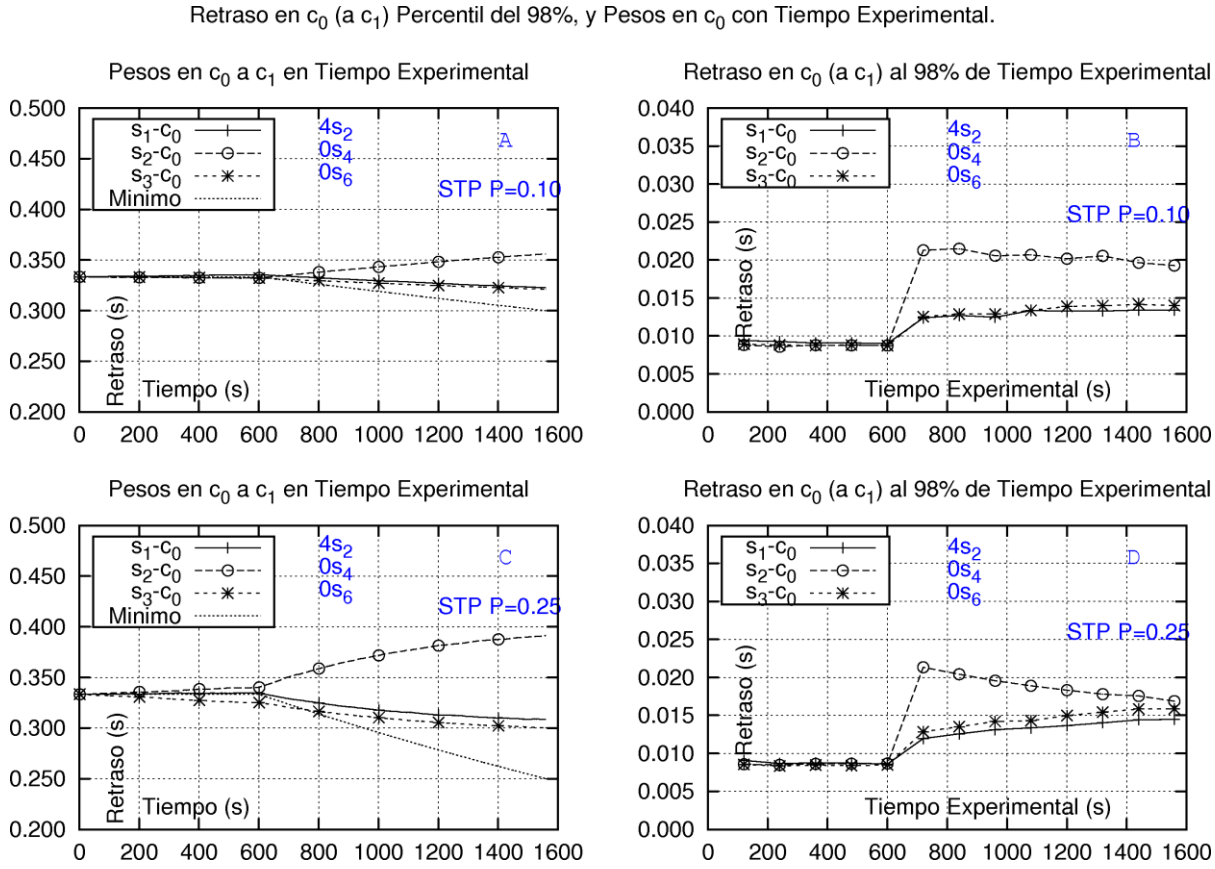


Figura 33. Pesos y retrasos, asociados a las colas de las rutas s_1-d_1, s_2-d_2, s_3-d_3, en la interfaz de salida del nodo c_0, a través del tiempo experimental, con el uso del Método STP para la atención de tráfico. Se incluye la curva de valor mínimo que pueden tener los pesos.
Los cuadros "A" y "B" muestran los pesos, y los retrasos, respectivamente, cuando el valor del parámetro *P* del Método STP es de 0.1. Los cuadros "C" y "D" son similares a los cuadros "A" y "B", pero para un valor de *P* de 0.25. En todos los cuadros la ruta s_2-d_2 aumenta de tráfico en 4 fuentes, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Se observa que a los 720 *s* el retraso en la cola de la ruta s_2-d_2 ha aumentado a su máximo en el experimento. Con el aumento del peso que esta cola va adquiriendo a partir de los 720 *s*, su retraso va disminuyendo.

Las otras colas, van disminuyendo su peso y aumentando su retraso, a partir de los 720 *s*, por tener menos tráfico y menor longitud en sus colas. Cuando $P = 0.25$ los cambios de pesos son suficientemente rápidos para llevar a las colas de las rutas a tener similares retrasos (dentro del 3.5%), a los 1560 *s*.

8.2.2 Comparativo de Retrasos al Percentil del 98%, retrasos máximos y retrasos promedios.

Con motivos netamente comparativos se incluye en esta tesis la Figura 34. Esta figura repite el cuadro "D" de la Figura 33, que muestra los retrasos en las tres colas del nodo c_0, a través

del tiempo experimental, cuando el nodo c_0 se utiliza el método STP, con P = 0.25, y la ruta s_2-d_2 aumenta 4 fuentes a los 600 *s*. Lo especial de esta figura es que la misma no solamente muestra los retrasos medidos del percentil del 98% de mayor retraso, sino que, además, esta figura muestra los retrasos máximos y promedio medidos. Esta es la única figura en esta tesis que muestra estos retrasos máximo y promedio, y lo hace con fines comparativos. Todas las demás gráficas que muestran retrasos en esta tesis, usan el retraso del percentil del 98% de mayor retraso.

El cuadro "A" de la Figura 34 muestra los valores medidos de retraso de la cola asociada a la ruta s_1-d_1. Estos retrasos son: retraso promedio, retraso al percentil de 98%, y retraso máximo. En el tiempo en el tiempo 1560 *s* estos valores son: 3, 14.5 y 31.2 *ms*, respectivamente.

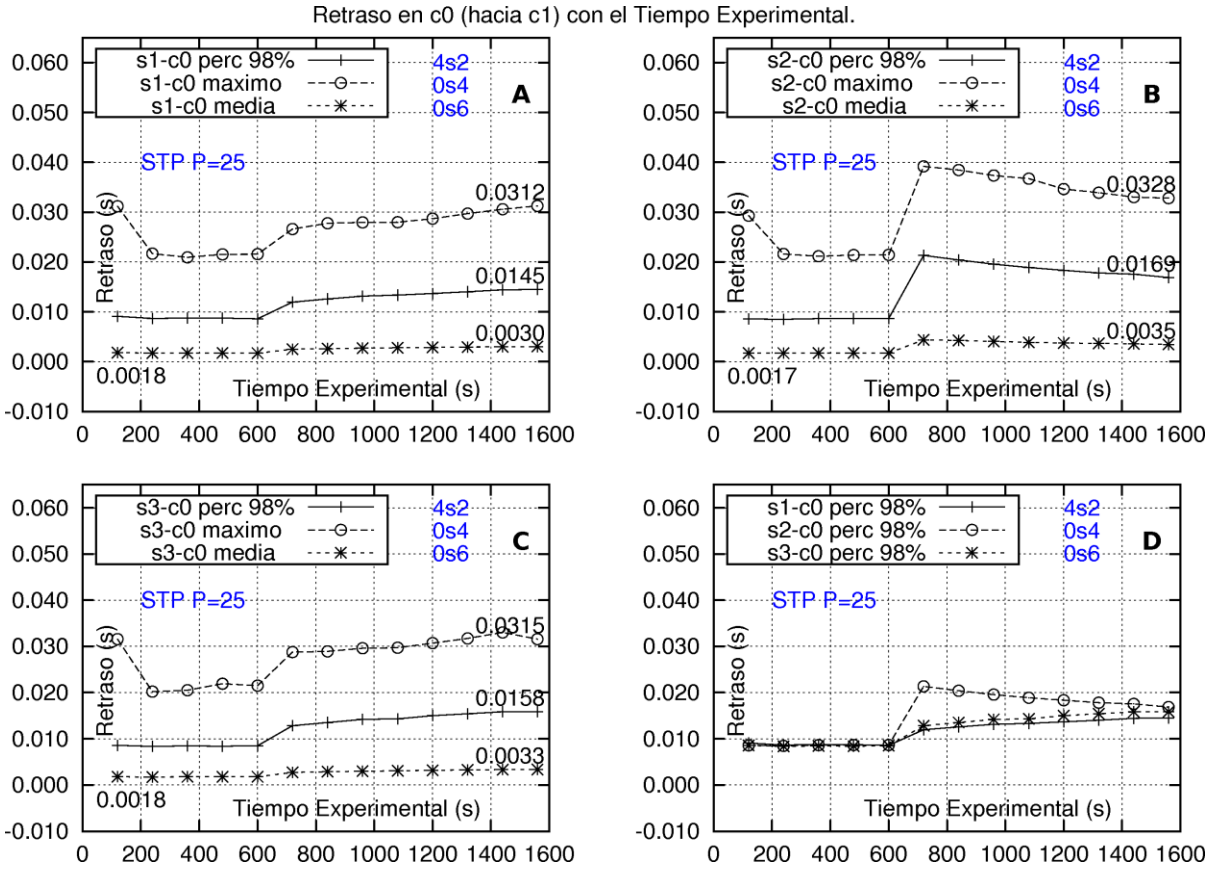


Figura 34. Comparación de los valores de retraso según si son máximo, al percentil del 98%, o promedio, para los retrasos en las colas de las rutas s_1-d_1, s_2-d_2 y s_3-d_3, en el nodo c_0.con la aplicación del Método STP, para la atención del Tráfico.
Para todos los cuadros de esta figura, la ruta s_2-d_2 aumenta 4 fuentes a los 600 *s*.
La etiqueta "s1-c0 perc 98%" significa el retraso al percentil 98% en la cola asociada a la ruta s_1-d1. Las etiquetas "s1-c0 maximo" y "s1-c0 media" significan retraso máximo y promedio, respectivamente.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

8.2.3 Retrasos en las Rutas según el Aumento de Tráfico en la Ruta s_2-d_2

En la Figura 35 se observan los retrasos en las rutas s_1-d_1, s_2-d_2 y s_3-d_3 correspondientes a los experimentos de la Figura 31.

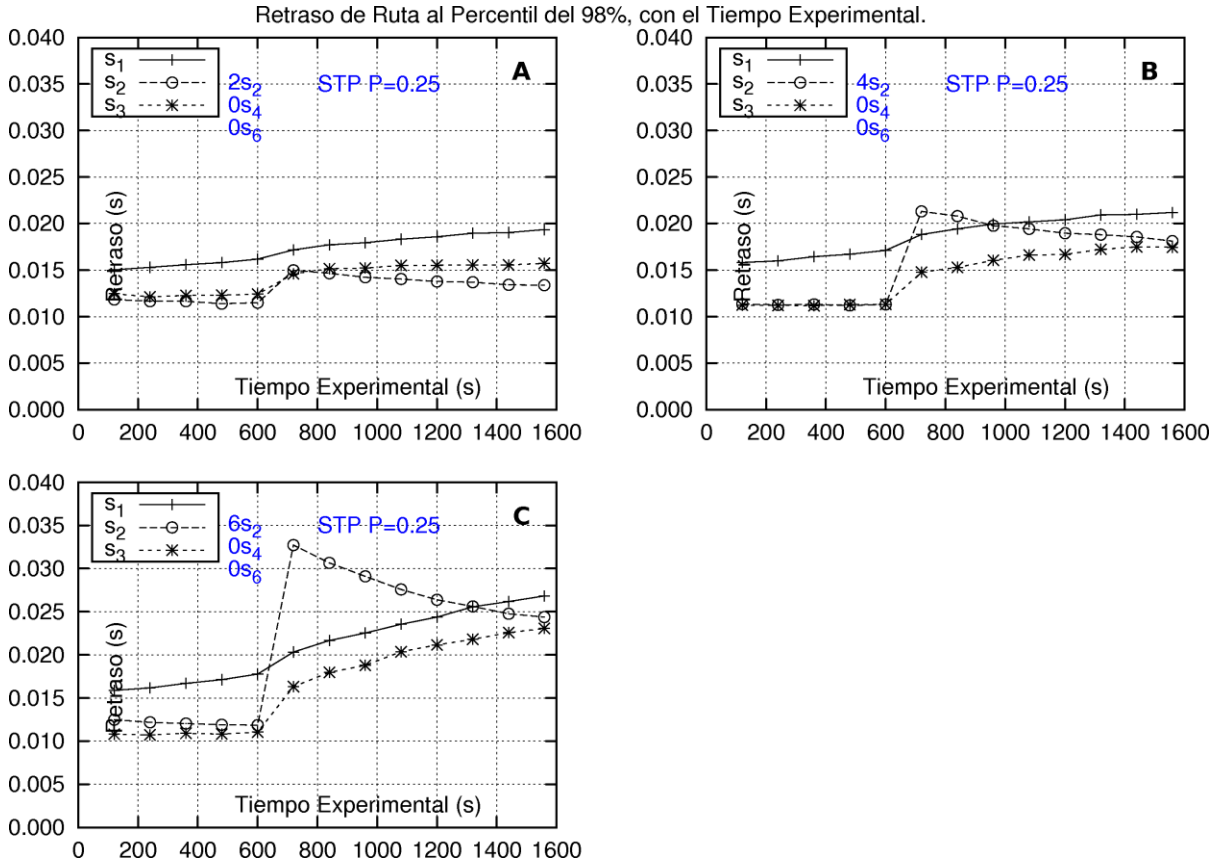


Figura 35. Retrasos en las rutas s_1-d_1, s_2-d_2 y s_3-d_3 correspondientes a los experimentos de la Figura 31 (que muestra los pesos de las colas correspondientes).
En los cuadros "A", "B" y "C" la ruta s_2-d_2 aumenta de tráfico en 2, 4 y 6 fuentes, respectivamente, a los 600 *s* de tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

En el tiempo 600 *s* el retraso en la ruta $s_2 - d_2$ se incrementa como consecuencia del incremento de tráfico en esta misma ruta, en el tiempo 600 *s*. El incremento de retraso en la ruta $s_2 - d_2$ es más pronunciado dependiendo de la cantidad de tráfico incrementada en esta ruta.

A los 720 *s* se observa que el retraso en la ruta $s_2 - d_2$ ha decrecido como resultado del incremento gradual de peso de la cola de esta ruta.

Los retrasos en las rutas $s_1 - d_1$ y $s_3 - d_3$ tienen un incremento desde el tiempo 600 *s*. Estas rutas están protegidas para limitar la rapidez en su incremento de retraso. A partir de los 600 *s*, dentro del tiempo experimental, estos retrasos se incrementan gradualmente.

Es importante notar que, aunque el Método STP está actuando en la interfaz de salida de cada uno de los tres nodos centrales, el método está efectivamente dando protección a la

ruta $s_1 - d_1$ sólo en la interfaz de salida del nodo central c_0, porque ésta es la única interfaz de salida que tiene un cambio en su tráfico promedio.

El comportamiento indicado de los retrasos es lo que se espera del Método STP.

8.2.4 Retrasos en las Rutas según el Número de Rutas que Aumentan su Tráfico

En la Figura 36 se observa una comparación de los retrasos de las rutas s_1-d_1, s_2-d_2 y s_3-d_3, cuando se utiliza el Método STP, con valores de P = 0.10 y P = 0.25, y cuando se utiliza el Método q1 (para comparación).

Los retrasos se muestran para dos casos: 1- Cuando la ruta s_2-d_2 aumenta en 4 fuentes. 2- Cuando las rutas s_2-d_2, s_4-d_4 y s_6-d_6 aumentan simultáneamente en 4 fuentes, ambos casos en el tiempo experimental de 600 *s*. Este último caso es extremo, y en el mismo la ruta s_1-d_1 se afecta simultáneamente por las tres rutas que aumentaron su tráfico (que son cruzadas a la primera).

La Tabla 13 ayuda a interpretar lo que muestra en la Figura 36, indicando los valores de los parámetros usados.

Retrasos en Rutas para el Método STP y para el Método q1			
	Método STP. P = 0.10	**Método STP. P = 0.25**	**q1**
La ruta s_2-d_2 aumenta su tráfico en 4 fuentes, en el tiempo 600 *s*.	**A**	**B**	**C**
Las rutas s_2-d_2, s_4-d_4 aumentan su tráfico en 4 fuentes, simultáneamente, cada una, en el tiempo 600 *s*.	**D**	**E**	**F**
Las rutas s_2-d_2, s_4-d_4 y s_6-d_6 aumentan su tráfico en 4 fuentes, simultáneamente, cada una, en el tiempo 600 *s*.	**G**	**H**	**I**

Tabla 13. Valores de aumento de tráfico, y métodos de atención de tráfico, utilizados para los experimentos reportados en los diferentes los cuadros de la Figura 36.

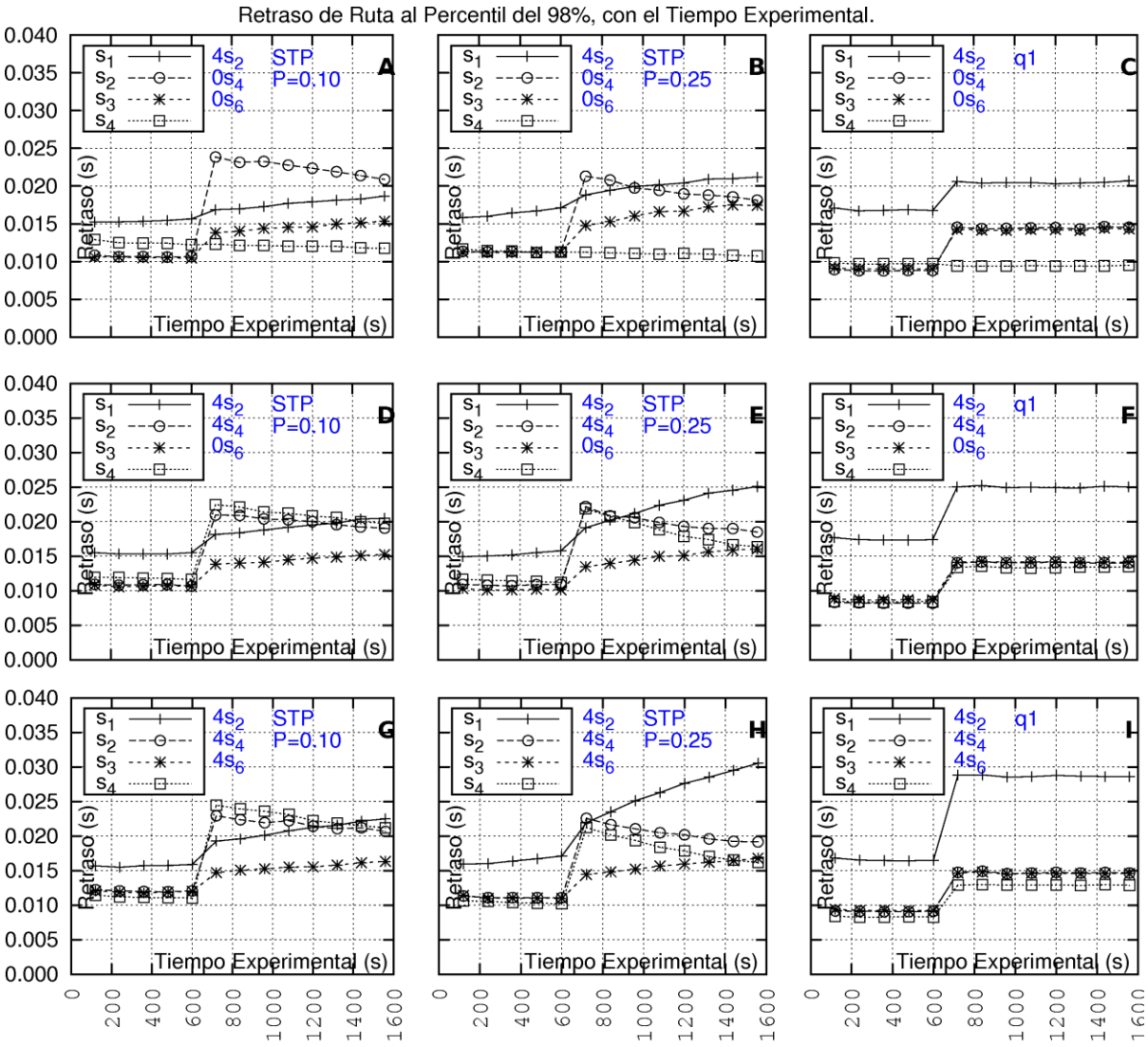


Figura 36. Retrasos en las rutas s_1-d_1, s_2-d_2, s_3-d_3 y s_2-d_4 para la aplicación del Método STP, con P = 0.10, y con P = 0.25, y del Método q1.
La Tabla 13 ayuda a diferenciar los resultados mostrados por cada cuadro de esta figura.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Cuando solamente la ruta s_2-d_2 aumenta su tráfico, se observa que con el Método STP, con P = 0.25, el retraso de la ruta s_2-d_2 logra bajar hasta 18 *ms* a los 1560 *s*, mientras que con P = 0.10 el retraso solamente logra bajar a 20.9 *ms*. Este es un aspecto positivo de trabajar con valores mayores de P, es decir, cuando baja más rápido el retraso de una ruta que ha aumentado su tráfico, como la ruta s_2-d_2, ésta tiene mayor posibilidad de lograr volver a admitir tráfico, en un tiempo menor.

Cuando dos o tres rutas interferentes aumentan simultáneamente su tráfico, en lugar de una sola, la ruta s_1-d_1 sufre un impacto mayor.

Sobre la ruta s_1-d_1, con el Método STP su retraso aumenta gradualmente. Este aumento es más lento con P = 0.10 que con P = 0.25, lo que india la mayor protección con P = 0.10.

Con el uso del Método q1 el retraso de la ruta s_2-d_2 se queda en 15 *ms* a partir de los 720 *s* del tiempo experimental. Esta ruta tiene una mejor posibilidad de competir equitativamente desde el tiempo 600 *s*, con el detrimento de que en la ruta s_1-d_1, cuyo mayor retraso ya se observa desde los 720 *s*.

8.2.5 Aumento de Tráfico en el Método STP posterior al Primer Aumento

Después de que con el Método STP se han admitido dos fuentes en la ruta s_2-d_2, si los pesos a los 1560 *s* llegaran a igualar a los tráficos relativos, 12/38 = 0.316 para las colas de las rutas s_1-d_1 y s_3-d_3, y a 14/38 = 0.368 para la cola asociada a la ruta s_2-d_2, en el nodo c_0, se tendría una situación como la descrita en la Figura 37, en donde se observan los resultados experimentales de retrasos en rutas, y de pesos, cuando las colas en el nodo c_0 inician con los valores indicados. A los 1560 *s* de tiempo experimental, los pesos podrían llegar a un valor objetivo de 12/40 = 0.3, para las colas de la rutas s_1-d_1 y s_3-d_3, y a 16/40 = 0.400, para la cola asociada a la ruta s_2-d_2, en el nodo c_0.

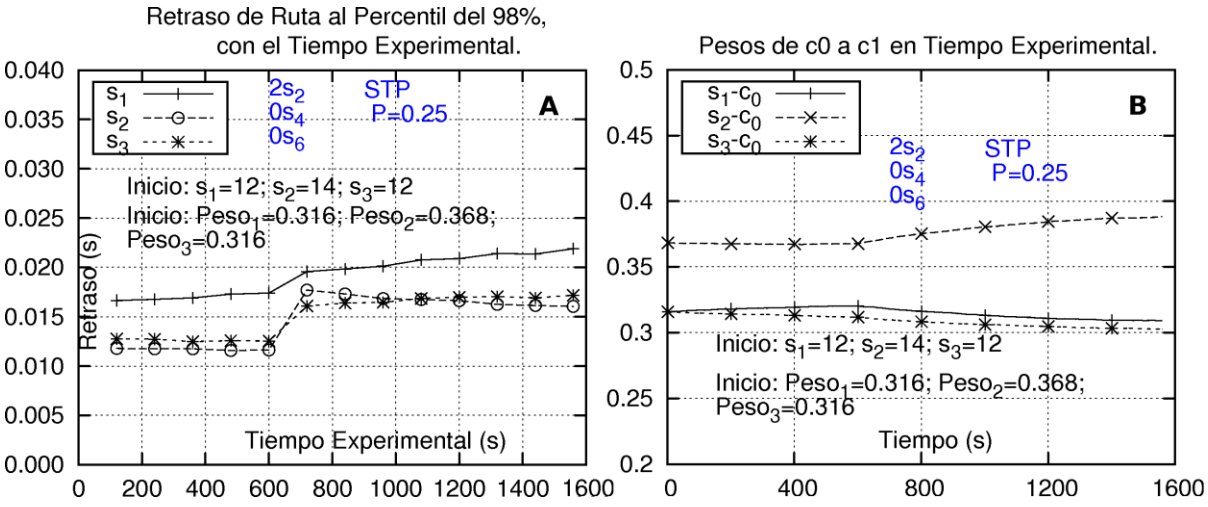


Figura 37. Retrasos en las rutas s_1-d_1, s_2-d_2, s_3-d_3, y pesos en las colas de las mismas, en el nodo c_0, cuando los pesos iniciales de dichas colas son: 0.316, 0.368 y 0.316, respectivamente (los tráficos relativos de llegada a las colas).
En ambos cuadros de la figura la ruta s_2-d_2 aumenta su tráfico en 2 fuentes a los 600 *s* del tiempo experimental.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Se puede observar que gracias a que la ruta s_2-d_2 aumentó de peso, nuevamente es posible para la misma volver a admitir otras dos fuentes a los 600 *s* del experimento, pues después de la admisión, a los 720 *s* se observa que la ruta s_2-d_2 tiene un retraso de 17.7 *ms*. Note también que aquí la ruta s_1-d_1 inicia con un retraso de 16.6 *ms* y no con 19.3 *ms* como en el

experimento anterior. Esto se debe a que los pesos iniciales en este experimento no son exactamente iguales a los pesos con que concluyó el experimento anterior.

En este caso la admisión en la ruta s_2-d_2 llevaría al retraso de la ruta s_1-d_1 a 19.6 *ms* a los 720 *ms*, llegando a 21.9 *ms* a los 1560 *s* de tiempo experimental, así que el Método STP aquí ya no ha conseguido proteger a la ruta s_1-d_1 contra este aumento.

8.2.6 Análisis de Ventajas del Método STP relativas al Retraso

En este subcapítulo se hace un análisis, en el escenario de trabajo de esta Tesis, para cuantificar las ventajas que puede tener el Método STP, y bajo qué condiciones se tiene. Se comparan tres casos: 1- El uso del Método STP para el valor de su argumento P = 0.10; 2- El uso del Método STP para el valor de su argumento P = 0.25, y 3- El uso del Método q1.

Para este análisis se establece un valor de retraso máximo que se puede aceptar en todas las rutas de la red, que es de 19 *ms*.

Este retraso se plantea porque es apenas (por 1 *ms*) sobrepasado por el retraso en la ruta s_1-d_1, cuando se usa el Método q1 y la ruta s_2-d_2 aumenta su tráfico en 4 fuentes (el 33% de su tráfico inicial). Con esto, se van a evaluar las ventajas que pudiese tener el Método STP, sobre el Método q1.

Las condiciones son las mismas ya utilizadas. El tráfico inicial que cursa del nodo c_0 al nodo c_1 es de 36 fuentes (12 por cada ruta s_1-d_1, s_2-d_2 y s_3-d_3), lo que implica un tráfico promedio de 36 x 0.621 *Mb/s* = 22.356 *Mb/s*, equivalente al 74.52% de la capacidad del enlace, que es de 30 *Mb/s*.

En la Figura 38 se muestran los retrasos en las rutas en el tiempo experimental, para los casos del uso del Método STP, con P = 0.10 ó 0.25, y para el caso del uso del Método q1. En los cuadros superiores de la figura el tráfico en la ruta s_2-d_2, a los 600 *s* aumenta en 2 fuentes, y en la parte inferior dicho aumento es de 4 fuentes. En los seis cuadros de la Figura 38 se observa una línea horizontal puesta en los 19 *ms*.

La Tabla 14 ayuda a interpretar los resultados que se muestran en la Figura 38, incluyendo los valores de los parámetros usados.

Retrasos en Rutas para Diversos Métodos			
Número de fuentes aumentadas	Método STP. P = 0.10	Método STP. P = 0.25	q1
2 en s_2-d_2 (16.7% de aumento)	**A**	**B**	**C**
2 en s_2-d_2 (16.7% de aumento) + 2 en s_4-d_4 (16.7% de aumento)	**D**	**E**	**F**
4 (33.3% de aumento)	G	**H**	**I**

Tabla 14. Valores de aumento de tráfico, y métodos de atención de tráfico, utilizados para los experimentos reportados en los diferentes cuadros de la Figura 38.

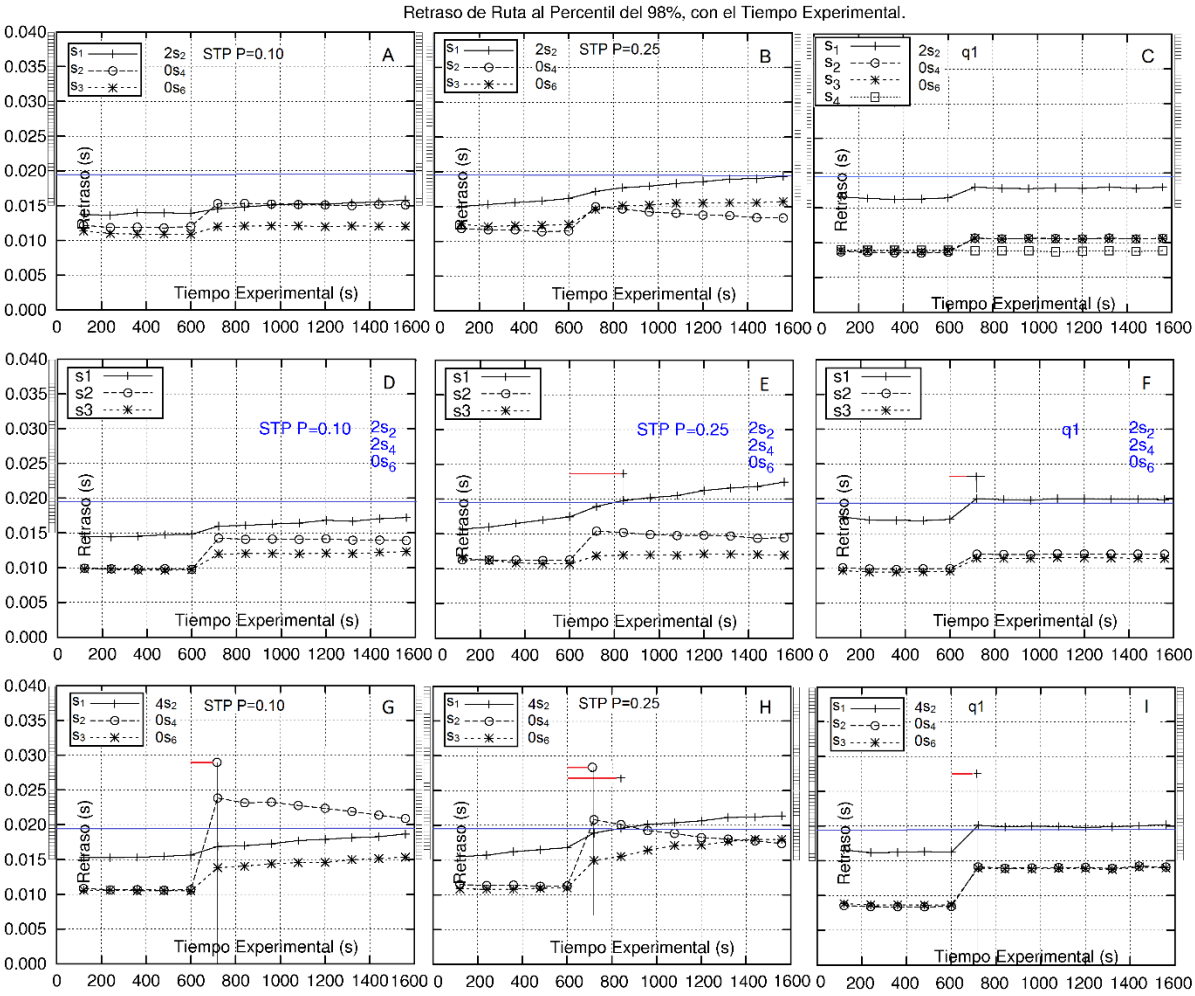


Figura 38. Retrasos en las rutas s_1-d_1, s_2-d_2 y s_3-d_3, con la aplicación del Método STP y del Método q1, para la atención del tráfico.
La Tabla 14 ayuda a interpretar los resultados mostrados por cada cuadro de esta figura.
Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

Se observa que el retraso en s_1-d_1 de 0 a 600 *s* es menor para los resultados con la aplicación del Método STP para ambos valores del parámetro *P*. La razón de esto puede deberse a una compleja interacción entre las colas en los nodos c_0, c_1 y c_2. En el método STP, en los periodos

donde llega más tráfico a la cola de la ruta s_1-d_1, dicha cola crece, hay más retraso de lo que habría si se usara el Método q1, debido a que con este último método sí se puede aprovechar el ancho de banda que pudiese estar disponible de las otras rutas, y el tráfico hacia el nodo c_1 disminuye en sus ráfagas. Esto que en el nodo c_1 haya menor congestión para el tráfico de dicha ruta. También se observa que de 0 a 600 *s* el retraso con el Método STP, en la ruta s_1-d_1 va aumentando ligeramente, lo cual también puede deberse a la misma compleja interacción entre las colas en los nodos c_0, c_1 y c_2. Al tener menor congestión el tráfico de la ruta s_1-d_1 en el nodo c_1, el tamaño promedio de la cola de dicha ruta en dicho nodo decrece, y por la acción del método STP la cola va perdiendo peso y su tráfico va adquiriendo mayor retraso.

Con relación al retraso dentro del nodo en la cola asociada a la ruta s_1-d_1, en el nodo c_0, entre el tiempo 0 y 600 *s*, para la aplicación del Método q1, en el cuadro "A" de la Figura 17 se observa que el retraso fluctúa en 7.5 *ms*; y para la aplicación del Método STP con P = 0.10 y con P = 0.25, en los cuadros "B" y "D" de la Figura 33, se observa que el retraso fluctúa entre 9 *ms*. Es decir, dentro del nodo c_0 el método q1 tiene menor retraso por poder aprovechar un poco mejor el ancho de banda total del nodo.

La Tabla 15 muestra los valores de retraso en tiempos determinantes, para que el algoritmo de admisión de la ruta s_2-d_2 admita, o no, nuevo tráfico. Con una pequeña línea horizontal, pintada a partir de los 600 *s*, se marcan los casos cuando el retraso de la ruta s_1-d_1 y de la ruta s_2-d_2 llegan a 19 *ms*. Dichas líneas horizontales tienen los mismos puntos distintivos que las líneas de retraso graficadas.

Cuadro de la Figura 38	**Método usado**	**Aumento a los 600 *s***	**Retraso observado en s_2-d_2 a los 720 *s***	**Aumento de Tráfico para s_2-d_2 (y s_4-d_4)**	**¿Rebasa s_1-d_1 los 19 *ms*?**
A	STP con *P*=10	$2s_2$ $0s_4$ $0s_6$	15.3 *ms*	Aceptable	No
B	STP con *P*=25	$2s_2$ $0s_4$ $0s_6$	15.0 *ms*	Aceptable	A los 1560 *s*
C	q1	$2s_2$ $0s_4$ $0s_6$	10.7 *ms*	Aceptable	No
D	STP con *P*=10	$2s_2$ $2s_4$ $0s_6$	14.2 *ms*	Aceptable	No
E	STP con *P*=25	$2s_2$ $2s_4$ $0s_6$	15.4 ms	Aceptable	Antes de los 840 *s*
F	q1	$2s_2$ $2s_4$ $0s_6$	12.0 *ms*	Aceptable	A los 720 *s*
G	STP con *P*=10	$4s_2$ $0s_4$ $0s_6$	23.8 *ms*	No aceptable	No
H	STP con *P*=25	$4s_2$ $0s_4$ $0s_6$	21.3 *ms*	No aceptable	Antes de los 840 *s*
I	q1	$4s_2$ $0s_4$ $0s_6$	14.1 *ms*	Aceptable	A los 720 *s*

Tabla 15. Valores de retraso en tiempos determinantes, para que el algoritmo de admisión de la ruta s_2-d_2 admita, o no, nuevo tráfico.

En los resultados de los cuadros "A", "B" y "C" se observa que el aumento de 2 fuentes en la ruta s_2-d_2 no causa que la ruta s_2-d_2 llegue a 19 *ms* de retraso, ni para los dos casos del uso del Método STP, ni para el caso del uso del Método q1, por lo que, para dicha ruta, el aumento es aceptable para esos casos.

En el cuadro B se observa que las rutas s_2-d_2 y s_3-d_3 no están teniendo el mismo retraso en la parte del tiempo final del experimento. En este caso el método STP podría estar dando ventaja a la ruta s_2-d_2 al otorgarle un peso proporcional a los tamaños medios de sus colas asociadas en los nodos centrales, algo que se explica en el Capítulo 2. Esta misma tendencia podría verse en el cuadro H, pero no se alcanza a observar en el tiempo del experimento.

En el cuadro I, que presenta los retrasos para el caso del uso del Método q1, se observa que el aumento de 4 fuentes en la ruta s_2-d_2 es aceptable para dicha ruta, pues el retraso en dicha ruta, observado a los 720 *s*, es de 14.1 *ms*, menor a 19 *ms*. El retraso de la ruta s_1-d_1 en este tiempo ha aumentado a 20 *ms*, que ya no es menor a 19 *ms*, pero eso la ruta s_2-d_2 "no lo sabe".

En el cuadro H, que presenta los retrasos para el caso del uso del Método STP con P = 0.25, se observa que el aumento de 4 fuentes en la ruta s_2-d_2 no es aceptable para dicha ruta, pues el retraso en dicha ruta, observado a los 720 *s*, es de 21.3 *ms*. Con esto, aun cuando el retraso en la ruta s_1-d_1 va creciendo poco a poco, y sobrepasa los 19 *ms* ya a los 840 *s* de tiempo experimental (llegando a 19.6 *ms*), ya no importa porque la ruta s_2-d_2 no va a aceptar un aumento de 4 fuentes. Algo similar a lo observado en el cuadro H pasa en el cuadro G. Aun cuando el retraso en la ruta s_1-d_1 sube más lentamente, esto ya no se aprovecha porque la ruta s_2-d_2 no va a aceptar el nuevo tráfico.

Cuando hay 2 fuentes de aumento, simultáneo, en las rutas s_2-d_2 y s_4-d_4, estos aumentos son aceptables para estas rutas (dado que el retraso en ellas no rebasa los 19 ms a los 720 *s* del tiempo experimental). En este caso, con la aplicación del Método STP con P = 0.25 se observa que el retraso en la ruta s_1-d_1 rebasa los 19 *ms* a los 840 *s*, mientras que con el Método q1 esto sucede a los 720 *s*. El Método STP con P = 0.10 da protección a la ruta s_1-d_1 en todo el tiempo experimental.

Se observa que el Método STP actúa porque:

1- El retraso en la ruta s_1-d_1 aumenta más lentamente que el retraso en dicha ruta con la aplicación del Método q1

2- El retraso en la ruta s_2-d_2, a los 720 *s*, es mayor en comparación con el retraso con la aplicación del Método q1.

Como conclusiones de los resultados de la Figura 38 se puede decir que:

1- Para el efecto de un aumento de 2 fuentes en la ruta s_2-d_2 no se requiere proteger a la ruta s_1-d_1. Quizá el Método STP con P = 0.25 puede fallar a los 1560 *s*.

2- Para el efecto de un aumento de 4 fuentes en la ruta s_2-d_2, la ruta s_1-d_1 sí requiere protección. El Método STP sí protege a la ruta s_1-d_1 al no permitir dicho aumento de tráfico. El Método q1 sí permite el aumento de tráfico y no protege a la s_1-d_1.

3- Para el efecto de un aumento de 2 fuentes en la ruta s_2-d_2 y 2 fuentes en la ruta s_4-d_4, simultáneamente, la ruta s_1-d_1 no tiene protección del Método q1; el Método STP con P = 0.25 le da protección solo hasta antes de los 840 *s*, y el Método STP con P = 0.10 protege a dicha ruta todo el tiempo.

4- Al comparar los retrasos entre las mismas rutas, cerca de 1560 *s*, con el uso del Método q1 y con el uso del Método STP (tanto para el valor de P = 0.25 como para el valor de P = 0.10), se observa que con el Método q1 se tienen menores retrasos. Una razón de esto es que estos pesos son al percentil del 98% de mayores retrasos, y el Método q1 tiene mayor habilidad de acomodar el tráfico para mejorar su desempeño.

5- Si hubiese la forma de contar con una información centralizada que indicase el porcentaje de utilización de la capacidad de los enlaces, el algoritmo de admisión del Método q1 podría beneficiarse para evitar llevar a la utilización del ancho de banda de los enlaces a valores mayores a un límite dado, como podría ser 74.5%.

8.3 Ganancias

Para la cuantificación de las ventajas y desventajas al comparar los métodos STP y q1, se utiliza el límite de retraso máximo para las rutas, de 19 *ms*, añadido en el escenario de trabajo de esta Tesis, y se usa un indicador de "ganancia" con el cual cada paquete que cruza su ruta dentro del límite de tiempo permitido, es decir por debajo de 19 *ms*, gana 1 *punto* para la ruta, y cada paquete que no cumple con ese límite, en su curso por su ruta, pierde 10 *puntos*, es decir, hay una "penalización"[89].

Se prevé que para el tráfico de una ruta que no tiene paquetes que sobrepasan los 19 *ms* de retraso, con 12 fuentes de tráfico, se tenga una ganancia de aproximadamente 12 x 0.621 *Mb/s* x 1 *punto*/*paq* / 8000 *bit*/*paq* = 931.5 *puntos*. Esta ganancia sería mayor pues algunos paquetes tienen menor tamaño de 1000 *Byte*. Si hubiese 14 fuentes de tráfico en la ruta, la ganancia sería mayor a 1086.75 *puntos*. Cuando hubiese en la ruta 14 fuentes, con 2% de paquetes con mayor retraso de 19 *ms*, con 10 *puntos* de penalización por cada uno de estos paquetes, se tendría una ganancia, para la ruta, mayor a 14 x 0.621 *Mb*/*s* / 8000 *bit*/*paq* x (0.98 x 1 – 0.02 x 10) *puntos*/*paq* = 847.665 *puntos*.

La Figura 39 muestra 12 cuadros, en 4 renglones (3 cuadros por renglón), con resultados de ganancias, en donde se pueden comparar los resultados para diversos casos de método de atención de tráfico utilizado, y del aumento de tráfico a los 600 *s* de tiempo experimental, ya

[89] Por ejemplo, en [60], en una función de ganancia y penalización similar, se penaliza 8 veces por paquete perdido o retrasado.

sea en 1 o en 2 rutas simultáneamente. Especialmente interesa la comparación entre las ganancias en los métodos STP, ya sea con $P = 0.10$ o $P = 0.25$, contra el método q1.

Es importante aclarar que aun cuando se presentan los cuadros de penalización para aumentos de 4 fuentes (para los casos $4s_2\ 0s_4\ 0s_6$ y $4s_2\ 4s_4\ 0s_6$) para el método STP, el mismo no aceptaría estos aumentos, como se explicó con relación a la Figura 38, cuadros D y E.

Los subcapítulos siguientes, dentro de este capítulo, explican los diversos cuadros de la Figura 38.

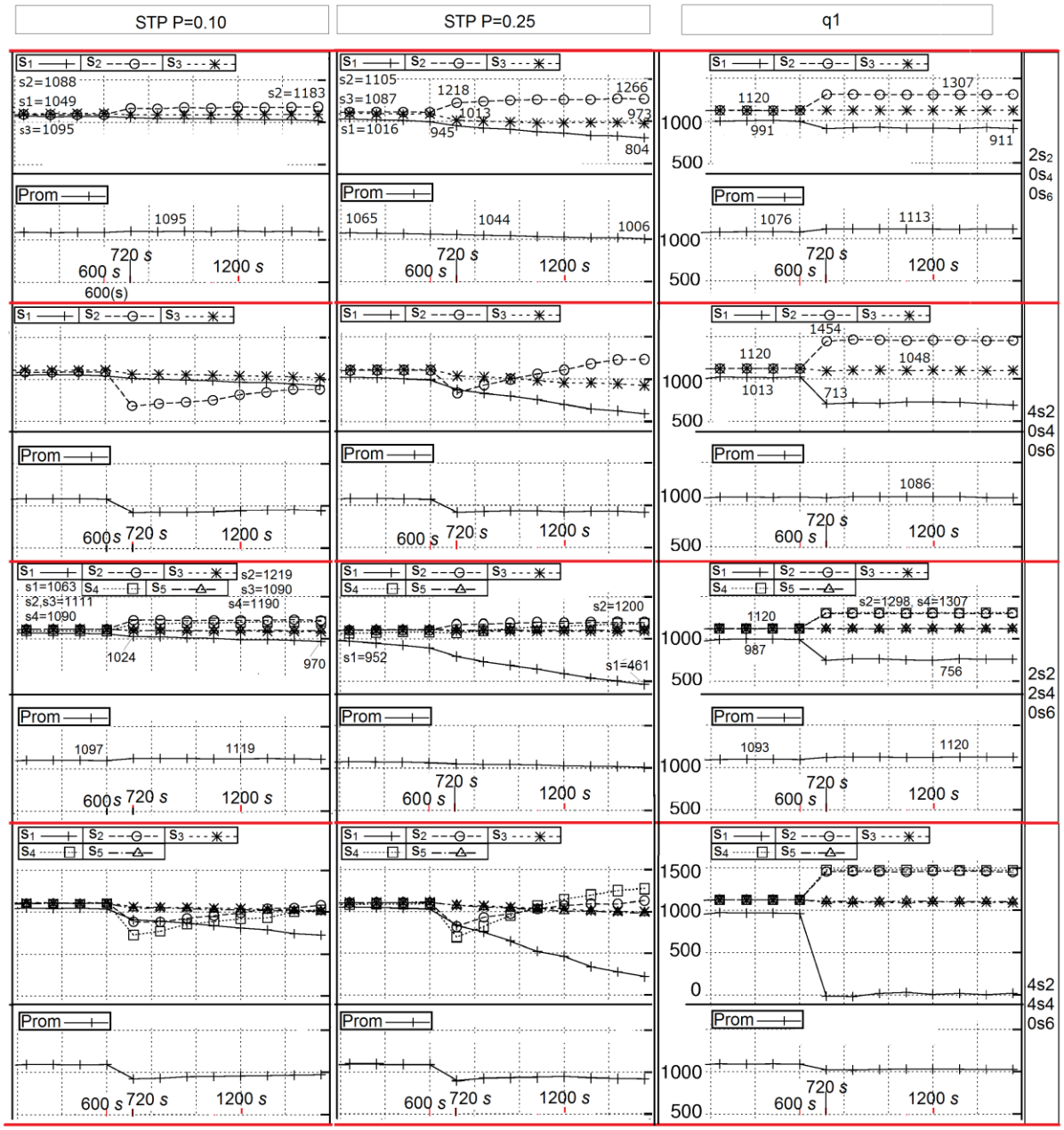


Figura 39. Ganancias en las rutas en el escenario de trabajo. Cada uno de los 12 cuadros de esta gráfica presenta un caso diferente de método de atención de tráfico aplicado, y del aumento de tráfico, a los 600 *s* de tiempo experimental, en las rutas.

La figura tiene 3 columnas. La columna izquierda se refiere a la aplicación del Método STP con P = 0.10, la columna media a la aplicación del método STP con P = 0.25, y la columna derecha a la aplicación del Método q1. Esta figura tiene 4 renglones. En el primero y segundo renglones la ruta s_2-d_2 aumenta su tráfico en 2 y 4 fuentes, respectivamente. En el tercero y cuarto renglones las rutas s_2-d_2 y s_4-d_4 aumentan su tráfico en 2 y 4 fuentes, cada una simultáneamente, respectivamente.

Nota: ver página 77 sobre el significado general de las etiquetas de las figuras como ésta.

8.3.1 La ruta s_2-d_2, Aumenta 2 Fuentes

Cuando una sola ruta, la ruta s_2-d_2, aumenta 2 fuentes a los 600 *s* (situación indicada con $2s_2$ $0s_4$ $0s_6$) no hay incumplimiento de retraso en ninguna rutas en más del 2% de los paquetes enviados, según lo que muestra la Figura 38 en sus cuadros *A*, *B* y *C*, con la aplicación de cualquiera de los métodos. En este caso, por el tamaño del aumento de tráfico, aun no se requiere la protección del Método STP. Aun así, los valores de ganancias obtenidos se indican a continuación.

La Tabla 16 da los valores de las ganancias que se muestran en los tres cuadros del primer renglón de la Figura 39.

Con relación a la ganancia de la ruta s_1-d_1 y en comparación con el uso del Método q1, el Método STP con *P* = 0.25 es mejor de 720 a 840 *s*, y el método STP es mejor de 720 a 1560 *s*. Con el método STP, al usar *P* = 0.10 (con relación al uso de *P* = 0.25) se otorga mayor protección a la ruta s_1-d_1 contra el aumento de tráfico en la ruta s_2-d_2, pero causando que esta última ruta requiera más tiempo para poder admitir más tráfico.

En este caso, las ganancias de todos los métodos son similares en cerca de un 10%.

El Método STP, para ambos valores del parámetro *P*, tiene esta mejora a costa de tener menor ganancia en las rutas s_2-d_2 y s_3-d_3.

$2s_2$ $0s_4$ $0s_6$			
	STP con *P* = 0.10	**STP con *P* = 0.25**	**q1**
s_1-d_1	1050 *puntos* El mejor	945 a 804 *puntos* En 840 s tiene aun 919 *puntos* Mejor que q1 de 720 a 840 *s*	911 *puntos*
s_2-d_2	1100 *puntos*	1218 a 1266 *puntos*	1307 *puntos* El mejor
s_3-d_3	1100. *puntos*	1013 a 973 *puntos*	1120 *puntos* El mejor
Total	1095 *puntos*	1044 a 1006- *puntos*	1113 *puntos*

Tabla 16.Comparativo de ganancia de rutas desde los 720 a 1560 *s*, referente a lo mostrado por la Figura 39, para el caso $2s_2$ $0s_4$ $0s_6$ de aumento de tráfico.

8.3.2 La ruta s_2-d_2, Aumenta 4 Fuentes

Cuando la ruta s_2-d_2 aumenta 4 fuentes a los 600 *s* (situación indicada con $4s_2$ $0s_4$ $0s_6$), con el uso del Método STP se produciría un incumplimiento de retraso en la ruta s_2-d_2, tanto para *P* = 0.10 como para *P* = 0.25, es decir, esta ruta llegaría a tener más del 2% de tráfico sin cumplir con el límite establecido, según lo que muestra la Figura 38 en sus cuadros *D* y *E*. Por lo anterior, el Método STP ya no admitiría este aumento para la ruta s_2-d_2, y los efectos causados

sobre las otras rutas ya no serían importantes. Aun así, en los cuadros del segundo renglón de la Figura 39 muestran los valores de ganancia para esta situación, con el Método STP.

Con el Método q1, este aumento de tráfico sí sería aceptable para la ruta s_2-d_2, como se muestra en la Figura 38 F. Con este aumento de tráfico, la ruta s_2-d_2 aumentaría su ganancia de 1120 a 1454 *puntos* (ganando 334 *puntos*), y la ruta s_1-d_1 bajaría su ganancia de 1013 a 713 *puntos* (perdiendo 300 *puntos* o el 29.61%). La ganancia total se mantendría, muy aproximadamente, después del aumento.

Asimismo, para el Método q1 se observa que la ganancia total es muy similar a aquella obtenida con la situación $2s_2$ $0s_4$ $0s_6$, es decir, en aquella situación la ganancia total era de 1113 *puntos*, y ahora es de 1086 *puntos*.

En este caso la ruta s_1-d_1 ha quedado protegida con el uso del Método STP.

La Tabla 17 muestra los valores de ganancias para el Método q1.

$4s_2$ $0s_4$ $0s_6$			
	STP con *P* = 0.10	**STP con *P* = 0.25**	**q1**
s_1-d_1	Aumento no admitido por la ruta s_2-d_2	Aumento no admitido por la ruta s_2-d_2	713 *puntos*
s_2-d_2			1454 *puntos*
s_3-d_3			1048 *puntos*
Total			1086 *puntos*

Tabla 17.Comparativo de ganancia de rutas desde los 720 a 1560 *s*, referente a lo mostrado por la Figura 39, para el caso $4s_2$ $0s_4$ $0s_6$ de aumento de tráfico.

8.3.3 Las rutas s_2-d_2 y s_4-d_4 Aumentan 2 Fuentes cada Una

Cuando las rutas s_2-d_2 y s_4-d_4 aumentan 2 fuentes cada una, simultáneamente, a los 600 *s* (situación indicada con $2s_2$ $2s_4$ $0s_6$), no hay incumplimiento de retraso en ninguna de las rutas en más del 2%, con excepción de la ruta s_1-d_1 que ahora sufre el aumento de 2 rutas interferentes (la figura que muestra el retraso se omite en esta Tesis).

Para la ruta s_1-d_1, y con relación al uso del Método q1, la ganancia con el uso del Método STP con P = 0.25 es mejor desde 720 hasta antes de 840 *s*, y con P = 0.10 es mejor de 720 a 1560 *s*. Entonces, después del aumento de tráfico, el Método STP con P = 0.10 protege todo el tiempo a la ruta s_1-d_1, y cuando P = 0.25 la protección dura menos de 2 minutos.

En este caso la ruta s_1-d_1 ha quedado protegida con el uso del Método STP, en los tiempos indicados.

El Método STP, para ambos valores del parámetro P, tiene esta mejora a costa de tener menor ganancia en las rutas s_2-d_2 y s_3-d_3.

La Tabla 18 muestra las ganancias para esta situación.

$2s_2$ $2s_4$ $0s_6$			
	STP con *P* = 0.10	**STP con *P* = 0.25**	**q1**
s_1-d_1	Baja de 1024 a 970 *puntos* El mejor	De 792 a 461 *puntos* (en 840 s ya tiene 731 *puntos*) Mejor que q1 de 720 a antes de 840 *s*	756 *puntos*
s_2-d_2	Se mantiene alrededor de 1219 *puntos*	1175 a 1200 *puntos*	1300 *puntos* El mejor
s_3-d_3	Baja de 1094 a 1084 pts.	1092 *puntos*	1120 *puntos* El mejor
s_4-d_4	Sube de 1173 a 1204 *puntos*	1071 a 1179 *puntos*	1307 *puntos* El mejor
s_5-d_5	Se mantiene alrededor de 1098 *puntos*	1102 *puntos*	1119 *puntos* El mejor
Total	1122 a 1114 *puntos*	1047 a 1007 *puntos*	1120 *puntos*

Tabla 18. Comparativo de ganancia de rutas desde los 720 a 1560 *s*, referente a lo mostrado por la Figura 39, para el caso $2s_2$ $2s_4$ $0s_6$ de aumento de tráfico.

8.3.4 Las rutas s_2-d_2 y s_4-d_4 Aumentan 4 Fuentes cada Una

En la situación en que dos rutas, las rutas s_2-d_2 y s_4-d_4, aumentan 4 fuentes cada una, a los 600 *s* (situación indicada con $4s_2$ $4s_4$ $0s_6$), con el Método STP, tanto para *P* = 0.10 como para *P* = 0.25, estas rutas ya no admitirían este aumento pues ambas rebasarían el 2% de retraso mayor a 19 *ms*.

Con el Método q1, este aumento de tráfico sí sería aceptable para la ruta s_2-d_2, como se muestra en la Figura 36 F. Ahora la ganancia de la ruta s_1-d_1 bajaría de 961 a 7 *puntos* (es decir se deteriora esta ganancia en 99.27%). La ganancia de la ruta s_2-d_2 sube de 1120 a 1455 *puntos* (sube 335 puntos). Las ganancias de las otras rutas se mantienen muy aproximadamente. La ganancia total se mantiene también, aproximadamente, pasando de 1088 a 1026 *puntos*. Esta ganancia total no es muy diferente a los valores de ganancia total en otras situaciones, por la compensación de ganancias en las diversas rutas.

En este caso la ruta s_1-d_1 ha quedado protegida con el uso del Método STP.

La Tabla 19 muestra los valores de ganancias para el Método q1 en esta situación.

$4s_2$ $4s_4$ $0s_6$			
	STP con *P* = 0.10	**STP con *P* = 0.25**	**q1**
s_1-d_1	Aumento no admitido por la ruta s_2-d_2	Aumento no admitido por la ruta s_2-d_2	7 *puntos*
s_2-d_2			1455 *puntos*
s_3-d_3			1091 *puntos*
s_4-d_4			1474 *puntos*
s_5-d_5			1102 *puntos*
Total			1026 *puntos*

Tabla 19.Comparativo de ganancia de rutas desde los 720 a 1560 *s*, referente a lo mostrado por la Figura 39, para el caso $4s_2$ $4s_4$ $0s_6$ de aumento de tráfico.

Resumen y Conclusiones del Capítulo

Con el propósito de validar el método propuesto, STP, en este capítulo se evaluó el desempeño Método STP y del Método q1, aplicados al escenario de trabajo de esta Tesis, y se comparan para generar conclusiones. La evaluación se hace mediante experimentación por simulación.

En cuanto a las hipótesis del Método STP, se observa que se cumplen sus hipótesis generales sobre que es posible realizar un comportamiento de protección, en el corto plazo, a las rutas, en una red de decenas de nodos, sin requerir reservación en las mismas, mediante una operación autónoma en cada nodo central; y que es posible proteger a una ruta mediante la protección individualizada, en cada interfaz de salida por donde pasa la ruta, de la cola asociada a dicha ruta. Se observa que se cumple la hipótesis específica del método sobre el comportamiento o conducta de disminución máxima de los pesos (de las colas), que sigue una expresión teórica planteada (la ecuación (26)).

También se observa que en el largo plazo la proporción del tamaño de las colas se acerca a la proporción del tráfico entrante, aspecto que no se indica que necesariamente se vaya a cumplir en escenarios diferentes al escenario de trabajo de esta Tesis, con tráfico de diferentes características.

Se observan gráficas que presentan el retraso en las rutas, cuando se aplica el Método STP y cuando se aplica el Método q1.

Los retrasos en las rutas cuando se aplica el Método STP son como se esperaba: una ruta que aumenta su tráfico ve un aumento máximo en la siguiente evaluación de retraso, y a partir de ahí el retraso decrece gradualmente, lo que implica que esta ruta va ganando protección y posibilidad de competir con equidad en el largo plazo. Al contrario, una ruta que no aumenta su tráfico y que resiente el aumento de tráfico en rutas interferentes, ve un aumento de retraso gradual, lo que implica que va perdiendo protección.

Posterior a cuando aumenta el tráfico de una ruta, "parece", por lo que se observa, que el tiempo de espera "en las colas" asociadas a las rutas, tanto la que aumenta de tráfico como las interferidas, tiende a igualarse, en el largo plazo. Esta observación requiere aun mayor validación y no se plantea en esta Tesis como un resultado necesario del Método STP.

Estas gráficas tienen parecido, en su forma, a las gráficas que se plantearon, como una aproximación teórica (en la Figura 29 y Figura 30 del subcapítulo 7.3.5 en donde se está presentando el método STP). También existe una relación coherente entre los valores esperados en la aproximación teórica y los valores encontrados en la evaluación en este capítulo[90].

Con relación a los retrasos observados en las rutas de la red del escenario de trabajo, se observan los efectos del retraso en las rutas, al haber un aumento en una, o simultáneamente en dos, o en tres, rutas interferentes a la ruta larga. Se puede decir que el aumento de retraso en la ruta larga, para el caso q1, es abrupto. Con el uso del Método q1 las rutas en donde se tienen los aumentos de tráfico resienten menos (comparativamente con el uso del Método STP) el impacto, en términos de aumento de retraso, porque este método es aquél que permite la utilización más libre del ancho de banda disponible de los nodos por una ruta cuando la misma ha aumentado su tráfico.

Esta situación está a favor del Método STP, en cuanto a la protección a la ruta larga por la interferencia de las rutas cortas, pues con este método las rutas interferentes que aumentan su tráfico resienten más el aumento de tráfico, en términos de aumento de retraso, y es probable que no permitan aumentos de tráfico, sino pequeños.

Con relación al indicador, adicionado al escenario de trabajo de la Tesis, llamado Ganancia, este indicador se añade para cuantificar las ventajas y desventajas al comparar los métodos STP y q1. Para esto se añade a dicho escenario un límite de retraso, único para todas las rutas de la red del escenario. Para calcular la ganancia se califica con 1 *punto* por cada paquete que cumpla con cursar su ruta debajo de un límite de retraso establecido, y se califica con –10 *puntos* (como una penalización) por cada paquete que no curse su ruta por debajo de dicho límite.

En términos de ganancias, se observa que el Método STP protege a la ruta larga contra la interferencia de sus rutas cruzadas. Esta protección se da a costa de afectar a las otras rutas cruzadas, tanto a la que aumenta su tráfico como a las otras rutas cortas.

[90] En la aproximación teórica los valores manejados son valores promedio y en los resultados de este capítulo los valores son al percentil del 98%. Asimismo, en la aproximación teórica, las colas en el nodo en donde se aplica el Método STP están aisladas, lo que produce que el aumento de retraso, en la cola asociada a la ruta que aumenta de tráfico, sea considerablemente mayor a lo que se observa en los valores obtenidos en las evaluaciones en este capítulo.

Para los aumentos de 2 fuentes, en una ruta o en forma simultánea en 2 rutas, que interfieran a la ruta larga de la red del escenario de trabajo indicado, el Método STP con P = 0.10 da la mayor protección entre todos los métodos, y el método STP con P = 0.25 solamente da mayor protección que el método q1 durante los dos primeros minutos posteriores al aumento de tráfico. Con el uso del Método STP, la ruta que aumentó de tráfico puede estar en condiciones de volver a aumentar de tráfico si gana más rápidamente peso después del aumento de tráfico, y eso se tiene cuando P = 0.25.

Para los aumentos de 4 fuentes, en una ruta o en forma simultánea en 2 rutas, que interfieren a la mencionada ruta larga, el Método q1 admite ese aumento de tráfico, lo que ya no permite el Método STP. En estos casos es más notoria la protección a la ruta larga por parte del Método STP, pues con el uso del Método q1 la ruta larga pierde cerca de 30% de su ganancia, cuando solamente una ruta interferente aumenta de tráfico, o 100% de su ganancia cuando dos rutas interferentes aumentan de tráfico.

En términos de la ganancia total (el promedio de la ganancia de todas las rutas), ambos métodos tienen un desempeño similar (ya sea para el parámetro P = 0.25 o el parámetro P = 0.10 con el Método STP), inclusive cuando más de una ruta que interfiere a la ruta larga aumenta su tráfico. Asimismo, la ganancia antes y después de los aumentos, en todos los casos, se mantiene, aproximadamente. Esto se debe a que los aumentos de ganancia de las rutas que aumentan su tráfico, interferentes a la ruta larga, se compensan con la disminución de ganancia de la ruta larga, como consecuencia del aumento de tráfico de dichas rutas interferentes.

Conclusión Metodológica.

El Método STP se puede utilizar en redes mayores a la presentada en el escenario de trabajo de esta Tesis debido a su característica de poder operar de manera autónoma en cada nodo central de la red. Es importante aceptar que mientras mayor sea una red mayor cantidad de interferencias tenderán a existir en sus rutas, y mayor retraso tenderá a haber en las mismas. El Método STP no se concibió para disminuir estas tendencias.

Para la protección de rutas largas interferidas, en una red, el Método STP con P = 0.10 tiene un mejor desempeño que el Método q1. Este desempeño se logra por la operación independiente en cada nodo central de la red, por lo que en redes de diverso tamaño se puede generar el efecto de resguardar a rutas interferidas largas contra rutas interferentes que aumentan su tráfico.

Conforme P crece, una ruta que aumenta su tráfico tarda más en adquirir equidad para competir por el ancho de banda, después de que aumenta de tráfico. El impacto de este efecto

para lograr una mayor ganancia por admisión de tráfico en la red es algo aun abierto para estudio.

Aportación de este Capítulo.

En este capítulo se presenta un método llamado STP para la protección de rutas que se interfieren en redes fijas en donde los flujos están restringidos a cursar por dichas rutas, donde la admisión se realiza de forma distribuida, por ruta, y donde la red ofrece QoS orientada a limitar el retraso en el que los flujos cruzan a las rutas.

En un escenario de trabajo seleccionado en esta Tesis se dan las condiciones de tráfico, el valor seleccionado del parámetro *P* del Método STP, y las razones de su selección. En dicho escenario se observa que el Método STP tiene ventaja sobre el Método q1 para proteger a la ruta larga contra aumentos de tráfico de rutas interferentes, siendo mayor su ventaja si más de una ruta interferente aumenta su tráfico simultáneamente. El Método STP mejora su protección de la ruta interferida conforme el valor de *P* decrece, aunque eso implica mayor tiempo para que las rutas interferentes que aumentan su tráfico readquieran equidad en su competencia por ancho de banda en los nodos.

Se hace la observación que el Método q1 podría verse beneficiado al ser complementado con una administración centralizada con la que se pudiese conocer y limitar el uso del ancho de banda de los enlaces, en el escenario de trabajo de esta Tesis. Esta observación es una hipótesis que no se estudia en este Tesis.

Una situación inesperada fue que el Método STP también protege a la ruta larga debido a que la ruta corta no admite tanto tráfico como lo haría con el uso del Método q1. Esto se debe a que con el Método STP un aumento de tráfico en una ruta causa mayor impacto en el retraso en esa ruta. El control de admisión de la ruta debería prevenir la admisión del tráfico.

Capítulo 9. Escenarios de Aplicación del Método STP

Introducción

En este capítulo se presentan diversos escenarios de aplicación para el Método STP.

9.1 Escenarios para la Aplicación del Método STP

La Internet tiene, básicamente, un diseño como una red para servicio “Del Mejor Esfuerzo” (*Best Effort*), que se compone de miles de redes diferentes [81]. En la Internet de “media milla”, la comunicación entre redes se ve afectada por la congestión entre puntos de conexión entre pares (*Peering Points*) [43], entre redes que posiblemente compiten; por las limitaciones de desempeño de los protocolos de enrutamiento exterior (BGP), y por la falta de confiabilidad de las redes.

Un enfoque para usar la Internet, sin cambios, para tratar los aspectos de calidad, es el uso de las “redes de entrega” (*Delivery Networks*). Una red de entrega es una red virtual, sobre la Internet, que no requiere cambios en la Internet, y que ofrece una confiabilidad mejorada y un desempeño mejor para la entrega. Un objetivo principal de una red de entrega [44] puede ser la minimización de las comunicaciones de larga distancia en la Internet, en donde existen cuellos de botella (como los puntos de conexión entre pares). Esto se hace a través del uso de múltiples servidores de distribución, de servicios de multidifusión (*Multicast*) en la capa de aplicación, y de varias rutas alternativas a través de la media milla de la Internet.

Un escenario para la operación del Método STP puede tomar ideas de operación de una red de entrega. El escenario podría ser una red dedicada a la entrega de tráfico sensible al tiempo, como el de video-conferencia. Esta red podría abarcar varias redes subyacentes específicas de proveedores de servicio de Internet (ISP), con puntos de interconexión que no representen cuellos de botella. Los proveedores de servicio de Internet, en forma conjunta, abarcarían grandes zonas de negocio. La red tendría rutas fijas para conectar a las redes de los clientes. Los nodos de los proveedores de servicio de Internet usarían el Método STP para dar el servicio requerido a la red de entrega. Los procesos de admisión, en cada ruta, serían responsables de verificar el cumplimiento del límite de retraso previsto en la ruta. No habrá reserva de ancho de banda en las rutas, y las interfaces de salida de los nodos deberían tener memorias (*Buffers*) suficientemente grandes para limitar la cantidad de paquetes perdidos a causa de posibles situaciones de sobrecarga.

Con respecto al protocolo de transporte sugerido, habría dos variantes del escenario. Para la primera variante, es interesante observar las tendencias de uso de la UDP y TCP en la Internet. Para el año 2009, las sesiones TCP seguían siendo responsables de la mayoría de los paquetes y Bytes en la Internet, pero en términos de flujos, UDP fue el protocolo de transporte dominante [82] (principalmente proveniente de las aplicaciones P2P las cuales utilizan UDP para

su tráfico de señalización). Hay un informe sobre el tráfico UDP, en la conexión de un campus de la Internet, que afirma que el tráfico UDP ha crecido de manera constante, entre los años 2008 a 2011 [83], para llegar a tener una participación del 22% del tráfico total.

El uso de UDP como el protocolo de transporte se sugiere para la primera variante del escenario. La razón de esto es que la red del escenario se dedicaría a tráfico sensible al retraso, y funcionaría con control de admisión. El uso principal de la red sería el tráfico estrictamente sensible al retardo (como el de video-conferencia), aunque tráfico menos estrictamente sensible al retardo podría admitirse (como el de tráfico "*Streaming*" proveniente de contenido almacenado, o en vivo). Los nodos subyacentes de los proveedores de servicio de Internet podrían aun usar el ancho de banda disponible para enviar tráfico de baja prioridad, como tráfico TCP, en otra clase, utilizando el servicio *Best Effort*.

Para la segunda variante de este escenario, el protocolo de transporte seleccionado sería un protocolo amigable con TCP (TCP-*Friendly*), como un tipo de DCCP [84] (adecuado para los flujos con limitaciones de tiempo, tales como de "*Streaming-Media*" o flujos de juegos, en línea, entre varios jugadores, o de telefonía y video-conferencia). En esta variante, tráfico TCP podría también utilizar la red de distribución, en la misma clase, pero sujeto a un proceso de admisión.

Con respecto a la capacidad de transmisión, ya que se considera que los despachadores son conservadores de energía, el Método STP no debe imponer una disminución de la capacidad de transmisión en la red.

9.2 El Método STP y las tendencias de QoS

Algunos enfoques de ingeniería de tráfico (Ingeniería de Tráfico Basada en IP [35] [36] [37] [38]) tratan de efectuar una distribución de tráfico óptima en las redes, sin el uso de redes superpuestas (en redes como las que tienen rutas fijas), las cuales se consideran administrativamente costosas dado que pueden requerir que los nodos establezcan muchas conexiones lógicas. Las funciones de distribución óptima utilizadas en estos métodos no observan el concepto de QoS, por lo que, para ellos, la adopción de un método como el Método STP no parece ser adecuado, o sería una vertiente de aplicación del Método STP sin rutas preestablecidas, y con enrutamiento basado en el peso de los enlaces. Parece que las cuestiones de calidad en estos enfoques deben ser tratadas por soluciones en las capas de aplicación.

Por otro lado, otro enfoque a la QoS, llamado de QoS de Enrutamiento [69] [67] [68], trata de identificar, en una topología fija, una posible ruta fuente-a-destino que satisface un conjunto de restricciones de QoS, traducidos en pesos en los enlaces. Las restricciones pueden

ser retraso, fluctuación de retraso (Jitter), costo, confiabilidad, rendimiento (Throughput). En este enfoque los pesos de los enlaces representan las características de QoS. Cada enlace tiene diversos pesos que se deben combinar adecuadamente[91]. Este enfoque no incluye el concepto de admisión, sino que más bien encuentra una ruta para cada "solicitud de enrutamiento". Las rutas en estas redes tienen interferencias, así que, el Método STP, aplicado a estas redes, podría ser adecuado.

[91] Los artículos referidos no indican cómo se realiza la asociación de la característica de QoS contra el valor de peso del enlace. Asimismo no se indica cómo se sujetaría a un flujo a seguir una ruta determinada.

Capítulo 10. Resumen, Conclusiones, Contribuciones y Trabajo de Investigación Futura

Resumen de la Tesis

Esta Tesis presenta conceptos generales sobre el desempeño de las redes, con énfasis en redes donde se ofrece QoS. Para presentar el problema que se aborda, que es el de la interferencia entre rutas en redes, se presentan resúmenes de trabajos relacionados que pretenden mejorar el desempeño, o solucionar problemas en redes, y en cuyos escenarios de trabajo para experimentación se observa el problema de la interferencia entre rutas, que dejan por lo general abierto, y que da oportunidad para el planteamiento del objeto de investigación de esta Tesis.

10.1 Aspectos Generales

En una red en donde los flujos están restringidos a cursar por rutas, dos rutas se interfieren cuando las mismas confluyen en un nodo por diferentes interfaces de entrada, para continuar su curso por una misma interfaz de salida del nodo (comparten al menos una arista). La interferencia se da porque ambas rutas compiten por los recursos de ancho de banda de esa interfaz de salida. Cuando una ruta aumenta su tráfico, sufre, o puede sufrir, por sí misma una afectación, y afecta, o puede afectar, a las rutas que interfiere, en aspectos como aumento de retraso, o la mayor pérdida de paquetes por sobrecupo en colas de espera, aspectos que se consideran “parámetros” de QoS en una red.

En esta Tesis se evalúa el impacto en las redes debido al problema de interferencia entre rutas, aplicado a redes fijas, donde los flujos están restringidos a seguir rutas específicas, en las cuales se ofrece QoS. Ese impacto se mide en términos del retraso, donde el proceso de admisión a flujos es distribuido, por ruta. No se maneja el concepto de “clases” lo que significa que en una red todo el tráfico tiene las mismas características y es valorado de la misma forma.

En este tipo de redes se pretende simplificar el proceso de admisión, por lo que se presenta como deseable que el proceso sea distribuido, por rutas (una ruta “no sabe” las condiciones de tráfico en las otras rutas), pero esto genera la situación de que al admitir tráfico en una ruta se puedan afectar a otras rutas que están en interferencia con la primera, a tal grado de llevar la operación de las rutas interferidas a condiciones no aceptables en términos de la QoS.

En una red en donde opera el Método q1, el tráfico perteneciente a una sola clase comparte siempre una misma cola en cada interfaz de salida de los nodos de la red. El tráfico de las rutas compite por el ancho de banda en la red. Los flujos que cursan por las rutas más largas (las que cursas más nodos), tienden a sufrir más retraso en la red. Las rutas más cortas, que interfieren a las largas, al realizar admisiones con el uso del Método q1, pueden generar los problemas de afectación de QoS mencionados.

En esta Tesis se propone un método, llamado STP (Protección a Corto Plazo), que ofrece protección de corto plazo a las rutas largas, contra la interferencia de rutas cortas. El corto plazo del método no es un término que implica un tiempo específico. El método tiene dos parámetros con los cuales el tiempo de protección puede ampliarse, o reducirse. La idea de proteger a corto plazo quiere decir que una ruta va entrando en un proceso de competición por el ancho de banda de los enlaces por donde cursa. En el largo plazo, la ruta debe competir por el ancho de banda de la red (de los enlaces de la red), como sucede en una red en donde se utiliza el Método q1. El tiempo del paso entre la protección y la apertura a la competencia, sin protección, puede fluctuar dependiendo de los parámetros indicados (en el escenario de trabajo de esta Tesis el tiempo puede fluctuar entre 2 y 16 minutos).

La protección de cada ruta toma la idea de cuando existen rutas, en una red, con protección permanente. Una forma de lograr esta protección en una ruta es asegurarle un ancho de banda mínimo en cada interfaz de salida de los nodos por donde curse. Si la ruta no usa ese ancho de banda, o solamente usa una porción, el sobrante se puede utilizar para los flujos de otras rutas. El ancho de banda asegurado se implementa, en cada interfaz de salida, mediante la asignación de una cola exclusiva para esa ruta, y un peso de atención para la misma, que sea utilizado por el despachador de la interfaz. El peso significa una porción de ancho de banda, que es otorgado necesariamente a la cola si el tráfico de la cola lo requiere. El despachador reparte su atención de acuerdo a la proporción de los valores de los pesos de cada cola que atiende.

Con la protección permanente a una ruta no se permite a las otras rutas competir, contra la primera, por el ancho de banda mínimo que tiene asegurado. La competencia se considera una característica de equidad para la concepción del Método STP. La idea del método es dar protección a corto plazo, en donde inicialmente la protección de una ruta se asemeja a una protección permanente, pero con el curso del tiempo, en el largo plazo, todas las rutas están en situación de competir por el ancho de banda de las interfaces de salida. La manera de equidad de competencia en el Método STP es dar a cada cola un peso proporcional al tamaño promedio de la misma.

La necesidad de protección se debe a que una ruta interferente aumenta su tráfico causando impacto en la QoS, tanto en ella misma como en las rutas que interfiere. Las rutas largas, las cuales pueden tener más rutas interferentes, suelen ser las más vulnerables contra el impacto por la interferencia de rutas.

El Método STP actúa de manera autónoma en cada interfaz de salida, en los nodos de la red donde opera. El método cambia el peso de cada cola en la interfaz, en proporción al tamaño promedio de la cola, pero lo hace lentamente con un algoritmo que no permite que una cola

pierda más porcentaje de peso que *P*% en un tiempo *T*, siendo los anteriores los dos parámetros del método. Una cola podría perder hasta este porcentaje de peso si de manera continua, durante todas las evaluaciones del tamaño de las colas, en el curso del tiempo *T*, resultase que tiene un tamaño menor que las demás. El tiempo *T* puede durar 900 *s*, por ejemplo, y el periodo entre cada evaluación puede ser de 1 *s*, por ejemplo.

Para evitar que en un nodo se deba tener conocimiento de las rutas existentes en la red, y conocimiento de qué paquetes pertenecen a cada ruta, la operación del método consiste en generar, en cada interfaz de salida de los nodos, una cola por cada interfaz de entrada. Así, dos rutas que se interfieren en un nodo dado, cuentan, para cada una, con una cola en la interfaz de salida del nodo. Esta forma de operación implica que dos rutas que confluyesen en un nodo anterior, no serían identificadas como diferentes en un mismo nodo posterior por donde ambas rutas continuaran su curso (si lo hicieren).

Se pretende que el Método STP ayude a la mejor y más fácil administración del tráfico en una ruta, al limitar la rapidez con que pueda cambiar la disponibilidad de ancho de banda para la ruta, y por consiguiente al reducir la rapidez de deterioro en el retraso observado en la ruta, cuando la misma *no* ha modificado su promedio de tráfico. Al operar el método de manera autónoma en los nodos, se pretende que el mismo pueda ser escalable hacia redes de al menos decenas de nodos. Esta Tesis presenta formalmente la operación del método.

En esta Tesis se realizan evaluaciones experimentales, con el uso de un escenario de trabajo que incluye una topología de red que tiene rutas predefinidas, donde una ruta es más larga que las demás, con un número de rutas que le interfieren que permite obtener conclusiones de interés. Sobre este escenario se realizan experimentos por simulación, para la aplicación de diversos métodos de atención de tráfico, con la intención de evaluar y comparar el desempeño de los mismos. Los métodos son: el método convencional (llamado q1) donde hay una cola por cada interfaz de salida en los nodos centrales de la red, el método en donde hay protección permanente a las rutas de la red (llamado q3f), un método alternativo obtenido de un trabajo relacionado (llamado CMS) que pretender proteger a los flujos en las rutas, y el Método STP.

10.2 Resultados de la Evaluación de los Métodos Competidores

Con el uso del Método q1 se observa que el algoritmo de admisión ayuda, por sí mismo, a la protección de rutas cruzadas cuando las mismas tienen un tamaño similar, pues la afectación que una ruta genera a sí misma, al aumentar de tráfico, inhibe su mayor admisión, y eso ayuda a prevenir afectación a rutas interferidas.

En el escenario de trabajo de esta Tesis, con el uso del Método q1 se generó una afectación poco importante (de menos de 3 *ms*) en rutas cruzadas al aumentar el tráfico de una ruta interferente, mientras se trabaje con cantidades de tráfico bajos (menos del 50% del ancho de banda de los enlaces).

Con el uso del Método q3f se observa que las rutas tienen una protección permanente contra el aumento de tráfico de rutas interferentes. Este método tiene falta de equidad en la competencia por el ancho de banda en la red.

Con el uso del Método CMS, en su aplicación para proteger rutas que se interfieren en una red, el hecho de dar protección a un flujo, en su curso por una ruta en una red, causa que el mismo adquiera una alta prioridad que causa detrimento a los flujos interferentes en nodos posteriores, en la ruta, formándose una cadena de afectación a los siguientes flujos cuya magnitud no es clara. Por lo anterior, los métodos q3f y CMS no se continuaron evaluando en esta Tesis.

10.3 Resultados de la Evaluación del Método STP

En cuanto a las hipótesis del Método STP, se observa que se cumplen sus hipótesis generales: 1- es posible tener un comportamiento de protección, en el corto plazo, a las rutas, sin requerir reservación en las mismas, mediante una operación autónoma en cada nodo central; 2- Es posible proteger a una ruta mediante la protección individualizada, en cada interfaz de salida por donde pasa la ruta, de la cola asociada a dicha ruta.

Se cumple la hipótesis específica del método: 1- El comportamiento, o conducta, de disminución máxima de los pesos (de las colas) sigue una expresión planteada (ecuación (25)).

En el largo plazo la proporción del tamaño de las colas se acerca a la proporción del tráfico entrante, aspecto que no se indica que necesariamente se vaya a cumplir en otros escenarios de trabajo, con tráficos diferentes en sus características.

Los retrasos en las rutas son como se esperaba. Una ruta que aumenta su tráfico ve un aumento máximo en la siguiente evaluación de retraso, y a partir de ahí el retraso decrece gradualmente. Una ruta que no aumenta su tráfico y que resiente el aumento de tráfico en rutas interferentes, experimenta un aumento de retraso gradual.

10.4 Sobre las Ventajas del Método STP contra el Método q1

En términos de ganancias por ruta, el Método STP protege a la ruta larga contra la interferencia de sus rutas cruzadas y es mejor, en este aspecto, que el Método q1, es decir, el Método q1 pierde habilidad para proteger a la ruta larga. El Método STP aumenta su ventaja observada de protección a la ruta larga cuando más de una ruta interferente aumenta su tráfico.

El Método STP también puede limitar más la admisión en las rutas, porque un aumento de tráfico en una ruta causa mayor impacto en sí misma, con relación a lo que sucede con el uso del Método q1. Con esto, el Método STP restringe la admisión y puede promover la mayor protección a rutas interferidas.

10.5 Sobre las Desventajas del Método STP

La protección que da el Método STP a una ruta se da a costa de afectar a las otras rutas cruzadas. Es por esto que, en términos de la ganancia total, cuando se promedian las ganancias de las rutas, *para la misma cantidad de tráfico admitido*, el Método STP no se desempeña mejor que el Método q1 cuando ambos métodos manejan la misma cantidad de tráfico, con ninguno de sus dos valores usados para el parámetro *P*, ni cuando más de una ruta interfiere a la ruta larga.

10.6 Interrogantes sobre el Método STP

El Método STP mejora su protección a la ruta interferida conforme el valor de *P* decrece, aunque eso implica mayor tiempo para que las rutas interferentes que aumentan su tráfico readquieran equidad en su competencia por ancho de banda en los nodos. Este aspecto aun deja terreno de evaluación, sobre el Método STP.

10.7 Conclusión Metodológica sobre el Método STP

El Método STP se puede utilizar en redes mayores a la presentada en el escenario de trabajo de esta Tesis, por su operación autónoma en cada nodo central de la red. Para la protección de rutas largas interferidas en una red, el Método STP tiene un mejor desempeño que el Método q1.

Al ser mayor una red es mayor el retraso que se puede observar, al menos en las rutas largas. Asimismo, las rutas pueden tener más interferencias. El Método STP no se concibió para disminuir estas tendencias.

10.8 Conclusiones Generales

El Método STP resulta eficaz para proteger a las rutas largas interferidas en una red. En este sentido, el método cumple su cometido, teniendo mejor desempeño que el Método q1.

El Método STP tiene efectos secundarios: la protección de una (o más) ruta interferida causa impacto negativo en las rutas interferentes. Esto se puede medir en términos de una ganancia por ruta y una ganancia total (promedio de las ganancias de las rutas), que equivalen a una ganancia económica para la red, por su trabajo de cursar flujos manteniendo una QoS (en este caso de guardar un mínimo retraso en el curso de los flujos de extremo a extremos por sus rutas). Por los efectos secundarios, la ganancia total del Método STP no es mejor que la ganancia total del Método q1, cuando ambos se comparan con un tráfico igual.

El Método q1 no protege tan bien como el Método STP a la ruta larga, pero "trata" mejor a las rutas interferentes.

El Método STP restringe más la admisión que el Método q1, cuando se llega a un límite muy cercano a tener el retraso máximo. Esto es un factor a favor para el cometido de protección del Método STP.

El Método STP no es infalible en su protección a rutas largas, es decir, es posible que se admita tráfico en una ruta corta, que impacte inadmisiblemente a una ruta larga. Esto, desde luego, sucede también con el Método q1.

Una característica del Método STP es que cuando el mismo aumenta su protección a las rutas interferidas (disminuyendo el valor de su parámetro *P*), se disminuye la rapidez con la que las rutas interferentes obtienen equidad para competir por el ancho de banda en la red. Esta situación genera interrogantes sobre la habilidad que puede llegar a tener una red que usa el Método STP, para maximizar sus ganancias.

El Método STP es más complejo que el Método q1.

Por su sencillez, el Método q1 resulta atractivo si se puede combinar con una forma de limitar el uso del ancho de banda de los enlaces de la red, según se requiera para mantener el límite máximo de retraso en la red, y dependiendo del tamaño de la red. Asimismo, cuando los tamaños de las rutas que se interfieren en una red son iguales, el Método q1 puede proteger a las rutas interferidas por el mismo efecto negativo que se tiene en las rutas interferentes, cuando éstas aumentan su tráfico.

Un escenario para la operación del Método STP puede tomar ideas de operación de una red de entrega, en donde se curse tráfico sensible a retraso.

10.9 Caminos de Investigación Futura

El Método STP puede adaptarse para trabajar de la siguiente forma. El valor del parámetro *P* puede aumentarse cuando en la interfaz de salida se tiene una utilización de ancho de banda bajo (del 50%), e ir gradualmente disminuyendo cuando dicha utilización aumenta (por ejemplo a valores de operación del 70%). Los valores de *P* podrían fluctuar entre 0.25 y 0.10.

Cuando el tamaño relativo de una cola disminuya porque el tamaño absoluto de esa cola ha disminuido, el Método STP podría disminuir el peso de esa cola de una forma más intensa que en el caso en que esa cola disminuya de tamaño relativo por el aumento de los tamaños absolutos de las otras colas.

Es deseable evaluar el Método STP con diferentes escenarios de trabajo, lo que incluye mayor diversidad en tamaños y formas de redes y de condiciones y tipo de tráfico.

El Método STP se pude evaluar cuando en las redes el tráfico no está restringido a cursar por rutas. Por su operación, el Método STP no requiere de la existencia de rutas.

Referencias

[1] N. Wang, K. H. Ho, G. Pavlou y M. Howarth, «An Overview of Routing Optimization for Internet Traffic Engineering,» *IEEE Communications Surveys & Tutorials,* vol. 10, nº 1, pp. 36-56, 2008.

[2] E. Rosen, A. Viswanathan y R. Callon, « Multiprotocol Label Switching Architecture. RFC 3031,» IETF. The Internet Society., 2001.

[3] Defense Advanced Research Projects Agency, «RFC 791. Internet Protocol. DARPA Internet Program Protocol Specification,» IETF, 1981.

[4] K. Nichols y B. E. Carpenter, «Differentiated services in the Internet,» *Proceedings of the IEEE,* vol. 90, nº 9, pp. 1479- 1494, Sep. 2002.

[5] B. Carpenter y K. Nichols, «Differentiated Services in the Internet,» Febrero 2002. [En línea]. Available: http://domino.watson.ibm.com/library/Cyberdig.nsf/398c93678b87a12d8525656200797aca/8efe46a5de7e3b8f85256b5e003c5883?OpenDocument. [Último acceso: 7 Noviembre 2012].

[6] S. G. Chaudhuri, C. S. Kumar y R. RajaKumar, «Validation of a DiffServ Based QoS Model Implementation for Real-Time Traffic in a Test Bed,» de *Communications (NCC), 2012 National Conference on*, Kharagpur. Bengala Occidental. India, 2012.

[7] C.-L. Chen, «A Proposal of Next Generation Network: QoS Mapping for MPLS-DiffServ and Label Forwarding,» de *5th International Conference on Biomedical Engineering and Informatics (BMEI 2012)*, Chongqing, China, 2012.

[8] S. Subramanian, J. Curtis, E. Pasiliao, J. Shea y W. Dixon, «Continuous Congestion Control for Differentiated-Services Networks,» de *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, Maui, Hawaii, USA, 2012.

[9] M. Pourvali, S. Seno, M. Yaghmaee y A. Mohajerzadeh, «Differentiating Services with Dynamic, QoS-Aware Queuing,» de *Computer and Knowledge Engineering (ICCKE), 2012 2nd International eConference on*, Mashhad, Jorasán Razaví, Irán, 2012.

[10] X. Sun, «Research on QoS of Next Generation Network Based on MPLS,» de *Information Science and Technology (ICIST), 2012 International Conference on*, Hubei, China, 2012.

[11] S. Gang, P. T. Oanh y S. He, «QoS Guarantee for IPTV Using Low Latency Queuing with Various Dropping Schemes,» de *Systems and Informatics (ICSAI), 2012 International Conference on*, Yantái, Shandong, China, 2012.

[12] H. Tarasiuk, S. Hanczewski, A. Kaliszan, R. Szuman, L. Ogrodowczyk, I. Olszewski, M. Giertych y P. Wisniewski, «A Proposal of the IPv6 QoS System Implementation in Virtual Infrastructure,» de *Telecommunications Network Strategy and Planning Symposium (NETWORKS), 2012 XVth International*, Rome, Italy, 2012.

[13] P. Flavius y P. Ferdi, «6th International Conference on Signal Processing and Communication Systems, ICSPCS'2012,» de *A QoS enabled two-stage service differentiation model for the Internet*, Gold Coast, Australia, 2012.

[14] ITU-T, «Recommendation Y.3001. Infrastructure, Internet Protocol Aspects and Next-Generation Networks,» Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks, Suiza, 2011.

[15] C. Cetinkaya, V. Kanodia y E. Knightly, «Scalable Services via Egress Admission Control,» *IEEE Transactions on Multimedia: Special Issue on Multimedia over IP,* vol. 3, nº 1, pp. 69-81, 2001.

[16] C. Cetinkaya y E. Knightly, «Egress Admission Control,» de *19th IEEE International Conference on Computer Communications, INFOCOM 2000*, Tel Aviv, Israel, 2000.

[17] T. Hui y C. Tham, «Adaptive Provisioning of Differentiated Services Networks Based on Reinforcement Learning,» *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 33, nº 4, pp. 492-501, 2003.

[18] K. Kar, M. Kodialam y T. V. Lakshman, «Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications,» *IEEE Journal on Selected Areas in Communications,* vol. 18, nº 12, pp. 2566-2579, Dic. 2000.

[19] M. Kodialam y T. V. Lakshman, «Minimum Interference Routing with Applications to MPLS Traffic Engineering,» de *IEEE INFOCOM 2000*, 2000.

[20] P. Aukia, M. Kodialam, P. V. Koppol, T. V. Lakshman, H. Sarin y B. Suter, «RATES: A Server for MPLS Traffic Engineering,» *IEEE Network,* vol. 14, nº 2, pp. 34-41, Mar/Abr. 2000.

[21] K. Kalamani, S. Palaniswami y B. Kaarthick, «Delay Minimization and Link Failure Management in MPLS Networks Incorporating RRATE Algorithm for Traffic Engineering under Real Time Traffic Conditions,» *European Journal of Scientific Research,* vol. 77, nº 3, pp. 402-416, 2012.

[22] C. Li y E. Knightly, «Schedulability Criterion and Performance Analysis of Coordinated Schedulers,» *IEEE/ACM Transactions on Networking,* vol. 13, nº 2, pp. 276-287, 2005.

[23] B. Krithikaivasan, K. Deka y D. Medhi, «Adaptive Bandwidth Provisioning Envelope based on Discrete Temporal Network Measurements,» de *Proceedings of IEEE Infocom 2004*, Hong Kong, 2004.

[24] M. Baldi y O. Yoram, End-to-end Delay of Videoconferencing Over Packet Switched Networks, International Business Machines Corporation. Research Division, 1996, p. 122.

[25] M. Baldi y Y. Ofek, «End-to-end delay analysis of videoconferencing over packet-switched networks,» *IEEE/ACM Transactions on Networking,* vol. 8, nº 4, pp. 479-492, Aug 2000.

[26] Cisco Systems, Inc., «Implementing QoS for IPv6,» 2011. [En línea]. Available: http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-qos.html. [Último acceso: 29 Ago 2013].

[27] Cisco Systems, Inc., «IP QoS Policy Parameters (Service Level IP QoS Parameters),» [En línea]. Available: http://www.cisco.com/en/US/docs/net_mgmt/ip_solution_center/3.2_security_management_suite/qos/user/guide/06qosec.html. [Último acceso: 29 Ago. 2013].

[28] Cisco Systems Inc., «The Cisco Learning network. Routing Protocol Overhead Negligible?,» 2010. [En línea]. Available: https://learningnetwork.cisco.com/thread/20682. [Último acceso: 29 Ago. 2013].

[29] IEEE Computer Society., «802.1D IEEE Standard for Local and Metropolitan Area Networks. Media Access Control (MAC) Bridges,» IEEE, New York, NY, 2004.

[30] Y. Lim, H. Yu, S. Das, S. S. Lee, M. Mario Gerla y M. Gerla, «QoS-aware Multiple Spanning Tree Mechanism over a Bridged LAN Environment,» de *2003 Global Communications Conference (Globecom'03)*, San Francisco, 2003.

[31] Cisco Systems Inc., «User Guide for Campus Manager 4.0. Managing Network Spanning Trees,» [En línea]. Available: http://www.cisco.com/en/US/docs/net_mgmt/ciscoworks_campus_manager/4.0/user/guide/stp.html. [Último acceso: 29 Ago. 2013].

[32] A. Meddeb, «On building multiple spanning trees and VLAN assignment in metro ethernet networks,» *Networks, Wiley,* vol. 61, nº 3, p. 263–280, 2013.

[33] V. Kumar, T. Lakshman y D. Stiliadis, «Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet,» *IEEE Communications Magazine,* vol. 36, nº 5, pp. 152-164, 1998.

[34] Youngseok Lee y B. Mukherjee, «Traffic Engineering in Next-Generation Optical Networks,» *IEEE Communications Surveys & Tutorials,* vol. 6, nº 3, pp. 16-33, 2004.

[35] D. Xu, M. Chiang y J. Rexford, «Link-State Routing with Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering,» *IEEE/ACM Transactions on Networking,* vol. 19, nº 6, pp. 1717-1730, Dic 2011.

[36] D. Xu, M. Chiang y J. Rexford, «DEFT: Distributed Exponentially-Weighted Flow Splitting,» de *INFOCOM*, Anchorage, AK, May 2007.

[37] B. Fortz y M. Thorup, «Increasing Internet Capacity Using Local Search.,» *Computational Optimization and Applications,* vol. 29, nº 1, pp. 13-48, 2004.

[38] B. Fortz y M. Thorup, «Internet Traffic Engineering by Optimizing OSPF Weights,» de *IEEE INFOCOM*, 2000.

[39] M. Neely, «EE 549 Webpage --- Queueing Theory,» 1996. [En línea]. Available: http://www-bcf.usc.edu/~mjneely/ee549notes/EE549_Supplementary_Lecture_Notes_01.pdf. [Último acceso: 9 Nov 2012].

[40] A. K. Parekh y R. G. Gallager, «A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case,» *IEEE/ACM Transactions on Networking,* vol. 1, nº 3, pp. 346-357, Jun. 1993.

[41] S. Jamin, P. B. Danzig, S. J. Shenker y L. Zhang, «A Measurement-Based Admisión Control Algorithm for Integrated Services Packet Networks,» *Networking, IEEE/ACM Transactions on,* vol. 5, nº 1, pp. 56-70, Feb. 1997.

[42] E. W. Knightly y N. B. Shroff, «Admission Control for Statistical QoS: Theory and Practice,» *IEEE Network,* vol. 13, nº 2, p. 20 – 29, Mar. 1999.

[43] S. Secci, M. Huaiyuan, B. E. Helvik y J. -L. Rougier, «Resilient Inter-Carrier Traffic Engineering for Internet Peering Interconnections,» *Network and Service Management, IEEE Transactions on,* vol. 8, nº 4, pp. pp.274-284, Dic. 2011.

[44] E. Nygren, R. Sitaraman y J. Sun, «The Akamai Network: A Platform for High-Performance Internet Applications,» *ACM SIGOPS Operating Systems Review,* vol. 44, nº 3, pp. 2-19, 2010.

[45] R. Braden, D. Clark y S. Shenker, «RFC 1633. Integrated Services in the Internet Architecture: an Overview,» IETF, 1994.

[46] S. Shenker, C. Partridge y R. Guerin., «RFC 2212. Specification of Guaranteed Quality of Service,» IETF, 1997.

[47] J. Wroclawski, «RFC 2211. Specification of the Controlled-Load Network Element Service,» IETF, 1997.

[48] R. Ogier y N. Taft-Plotkin, «Neural Network Methods for Call Admission Control,» de *Computational Intelligence in Telecommunications Networks*, W. Pedrycz, Ed., CRC Press, 2001, p. 3.

[49] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang y W. Weiss, «RFC 2475. An Architecture for Differentiated Services,» IETF, 1998.

[50] K. Nichols, S. Blake, F. Baker y D. Black, «RFC 2474. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,» IETF, 1998.

[51] K. Nichols, V. Jacobson y L. Zhang, «RFC 2638. A Two Bit Differentaited Services Architecure for the Internet,» IETF, Jul. 1999.

[52] K. Nichols y B. Carpenter, «RFC 3086. Definition of Differentiated Services Per Domain Behaviors and Rules for Their Specification,» IETF, 2001.

[53] O. Medina y L. Toutain, «State of the Art in DiffServ. Technical Report ITEA Project 99011 "RTIPA",» ITEA, Feb. 2001.

[54] A. P. Mateos Papis y J. Incera Diéguez, «Control de Admisión en Redes con Arquitecturas de Diferenciación de Servicios: Algunas Tendencias,» de *Decimoquinta Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición industrial ROC&C'2004*, Acapulco, Guerrero, México, 2004.

[55] H. Wang, C. Shen y S. K. G. , «Adaptive-Weighted Packet Scheduling for Premium Service,» de *IEEE International Conference on Communications (ICC 2001)*, Helsinki, Finland, 2001.

[56] A. Pereira y E. Monteiro, «Admission Control in IntServ to DiffServ Mapping,» de *IEEE International conference on Networking and Services, ICNS '06.*, 2006.

[57] E. Mykoniati, C. Charalampous, P. Georgatsos, T. Damilatis, D. Goderis, P. Trimintzios, G. Pavlou y D. Griffin, «Admission Control for Providing QoS in DiffServ IP Networks: The TEQUILA Approach,» *Communications Magazine, IEEE,* vol. 41, nº 1, pp. 38- 44, Jan. 2003.

[58] I. Cisco Systems, «Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2,» 2007. [En línea]. Available: http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/fqos_c.html. [Último acceso: 12 Nov. 2012].

[59] T. I. Society, «Traffic Engineering for Quality of Service in the Internet, at Large Scale,» [En línea]. Available: http://www.ist-tequila.org/. [Último acceso: 12 Nov. 2012].

[60] M. Rajaei y S. Noferesti, «Browse Conference Publications > Computational Intelligence an ... Help Back to Results,» de *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, Wuhan, 2009.

[61] M.-F. Horng, W.-T. Lee, K.-R. Lee y Y.-H. Kuo, «An Adaptive Approach to Weighted Fair Queueing with QoS Enhanced on IP Network,» de *IEEE Region 10 International*

Conference on Electrical and Electronic Technology (TENCON 2001), Singapore, Singapore, 2001.

[62] V. Jacobson, K. Nichols y K. Poduri, «RFC 2598. An Expedited Forwarding PHB,» IETF, Jun. 1999.

[63] D. Hang, H.-R. Shao, W. Zhu y Y.-Q. Zhang, «TD2FQ: An Integrated Traffic Scheduling and Shaping Scheme for DiffServ Networks,» de *IEEE Workshop on High Performance Switching and Routing*, Dallas, Texas, USA, 2001.

[64] R. R.-F. Liao y A. T. Campbell, «Dynamic Core Provisioning for Quantitative Differentiated Services,» *IEEE/ACM Transactions on Networking,* vol. 12, nº 3, pp. 429-442, 2004.

[65] C.-C. Li, S.-L. Tsao, M. C. Chen, Y. Sun y Y.-M. Huang, «Proportional Delay Differentiation Service Based on Weighted Fair Queuing,» de *IEEE Ninth International Conference on Computer Communications and Networks (ICCCN 2000)*, Las Vegas, Nevada, USA, 2000.

[66] B. A. Forouzan, TCP / IP Protocol Suite. Second Edition, New York, NY, EEUU: Mcgraw-Hill, 2003.

[67] G. Xue, A. Sen, W. Zhang, J. Tang y K. Thulasiraman, «Finding a Path Subject to Many Additive QoS Constraints,» *IEEE/ACM Transactions on Networking,* vol. 15, nº 1, pp. 201-211, Feb. 2007.

[68] Y. Xiao, K. Thulasiraman, X. Fang, D. Yang y G. Xue, «Computing a Most Probable Delay Constrained Path: NP-Hardness and Approximation Schemes,» *IEEE Transactions on Computers,* vol. 61, nº 5, pp. 738-744, May. 2012.

[69] J. Huang, X. Huang y Y. Ma, «An Effective Approximation Scheme for Multiconstrained Qualiy-of-Service Routing,» de *IEEE Global Telecommunications Conference (GLOBECOM 2010)*, Dec. 2010.

[70] A. Matrawy, L. Lambadaris y C. Huang, «MPEG4 Traffic Modeling Using the Transform Expand Sample Methodology,» de *IEEE 4th International Workshop on Networked Appliances*, Gaithersburg, Maryland, USA, 2002.

[71] G. Salmon, M. Ghobadi, Y. Ganjali, M. Labrecque y J. Gregory Steffan, «NetFPGA-Based Precise Traffic Generation,» de *NetFPGA Developers Workshop'09*, California, USA, 2009.

[72] «ns-2 Network Simulator,» 2011. [En línea]. Available: http://www.isi.edu/nsnam/ns/.

[73] E. Altman y T. Jimenez, NS Simulator for Beginners, Lecture Notes, 2003-2004, M. V. a. E. S. F. Univ. de Los Andes, Ed., Disponible: Nov 2012, from: http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/COURS-NS/n3.pdf.

[74] The VINT Project. Kevin Fall, Kannan Varadhan, Editors., «The ns Manual,» 4 Nov. 2011. [En línea]. Available: http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf. [Último acceso: 20 Feb. 2013].

[75] P. Pieda, J. Ethridge, M. Baines y F. Shallwani, «A Network Simulator Differentiated Services Implementation,» Open IP, Nortel Networks, 2000.

[76] A. Mrkaic, «Porting a WFQ Scheduler into ns-2's Diffserv Environment. Student Thesis.,» Zürich, 2001.

[77] J. D. C. Little, «Little's Law as Viewed on Its 50th Anniversary,» *Operations Research,* vol. 59, nº 30, pp. 536-549, 2011.

[78] S. Floyd y V. Jacobson, «Random Early Detection Gateways for Congestion Avoidance,» *IEEE/ACM Transactions on Networking,* vol. 1, nº 4, pp. 397-413, Aug. 1993.

[79] L. Mesa y F. Yesid, Interés Simple, Interés Compuesto y Tasas de Interés., Universidad de Santo Tomás de Aquino, 2009.

[80] A. P. Mateos Papis, «A Method with Node Autonomy to Preserve QoS in Networks with per-Route Admission,» *Journal of Applied Research and Technology (JART),* vol. 11, nº Feb., pp. 42-64, 2013.

[81] «CIDR REPORT, 2012,» [En línea]. Available: http://www.cidr-report.org/as2.0/.

[82] T. C. A. f. I. D. (CAIDA), «Analyzing UDP usage in Internet traffic,» 2012. [En línea]. Available: http://www.caida.org/research/traffic-analysis/tcpudpratio/. [Último acceso: 2012].

[83] C. Lee, D. Lee y S. Moon, «Unmasking the growing UDP traffic in a Campus Network,» de *Passive and Active Measurement (PAM) Conference*, Vienna, Austria, 2012.

[84] E. Kohler y S. Floyd, «Datagram Congestion Control Protocol (DCCP) Overview,» ICSI Center for Internet Research, Berkeley, CA, USA, 2003.

[85] A. P. Mateos Papis, «A Bandwidth Sharing Method under Node Autonomy and Short-Term Protection for QoS,» *Research in Computing Science. Special Issue on Advances in Computer Science and Electronic Systems.,* nº Abr., pp. 155-168, 2011.

[86] A. P. Mateos Papis, «Arquitectura de Servicios Diferenciados,» 2011. [En línea]. Available: http://ccd.cua.uam.mx/~amateos/. [Último acceso: 7 Noviembre 2012].

[87] R. Braden, L. Zhang, S. Berson, S. Herzog y S. Jamin, «RFC 2205. Resource ReSerVation Protocol (RSVP). Version 1 Functional Specification.,» IETF, 1997.

[88] J. Heinanen, F. Baker, W. Weiss y J. Wroclawski, «RFC 2597. Assured Forwarding PHB Group,» IETF, Jun. 1999.

[89] «NS-2 Trace Formats,» 2010. [En línea]. Available: http://nsnam.isi.edu/nsnam/index.php/NS-2_Trace_Formats. [Último acceso: 2013].

[90] J. Qiu, C. Cetinkaya, C. Li y E. W. Knightly, «Inter-Class Resource Sharing Using Statistical Service Envelopes,» de *IEEE INFOCOM*, New York, NY, 1999.

[91] J. Qiu y E. W. Knightly, «Measurement-Based Admission Control with Aggregate Traffic Envelopes,» *IEEE / ACM Transactions on Networking,* vol. 9, nº 2, Abr. 2001.

[92] S. Salman y H. Schulzrinne, «An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol.,» 2004. [En línea]. Available: http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf. [Último acceso: 2011].

[93] J. L. Devore, Probability and Statistics for Engineering and Sciences, 6 ed., Brooks Cole, 2003.

[94] «ns-2 Contributed Code.,» [En línea]. Available: http://nsnam.isi.edu/nsnam/index.php/Contributed_Code. [Último acceso: Mar. 2013].

[95] Z. Wang, Internet QoS Architectures and Mechanisms for Quality of Service, Morgan Kaufmann Publishers, 2001.

[96] A. P. Mateos Papis, «Algorithm to Implement the WFQ Scheduler with Changing Weights,» 2011. [En línea]. Available: http://ccd.cua.uam.mx/~amateos/REP-TEC/Rep-Tec-3-WFQ-Changing-Weights.pdf. [Último acceso: 7 Noviembre 2012].

[97] S. Georgoulas, P. Trimintzios, G. Pavlou y K. H. Ho, «Measurement Based Admission Control for Real-time Traffic in IP Differentiated Services Networks,» de *ICT 2005*, Cape Town, South Africa, 2005.

[98] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja y X. Xiao, «Overview and Principles of Internet Traffic Engineering. RFC 3272,» 2002..

[99] Cisco Systems, Inc., «Bandwidth, Packets Per Second, and Other Network Performance Metrics.,» [En línea]. Available: www.cisco.com/web/about/security/intelligence/network_performance_metrics.html. [Último acceso: Feb. 2013].

[100] Skype, «How Much Bandwidth Does Skype Need,» 2011. [En línea]. Available: https://support.skype.com/en/faq/FA1417/How-much-bandwidth-does-Skype-need;jsessionid=5A4C03A96FCF4A2AE9CE5D72D15093CD?frompage=search&q=how+much+bandwidth+for+a+call&fromSearchFirstPage=false. [Último acceso: 2011].

[101] Cisconinja's Blog, «CBWFQ, Routing Protocols, and max-reserved-bandwidth,» 18 Feb. 2009. [En línea]. Available: http://cisconinja.wordpress.com/2009/02/18/cbwfq-routing-protocols-and-max-reserved-bandwidth/. [Último acceso: 29 Ago. 2013].

[102] B. A. Forouzan, TCP/IP Protocol Suite., New York.: McGraw-Hill Companies, Inc., 2003.

Índice de Acrónimos

STP: Short-Term Protection 1
Tspec: IntServ Traffic SPECification part 239
WFQ: Weighted Fair Queuing 29
WRR: Weighted Round Robin 29

Índice de Términos

Apéndice A. Trabajos Emanados por el Trabajo de esta Tesis

En el curso de la investigación relacionada con esta Tesis, los siguientes trabajos fueron presentados, o publicados, o aceptados para su publicación, o puestos en sitios accesibles para su consulta, según se detalla a continuación.

Conferencias.

A. P. Mateos Papis y J. Incera Diéguez, «Control de Admisión en Redes con Arquitecturas de Diferenciación de Servicios: Algunas Tendencias,» de Decimoquinta Reunión de Otoño de Comunicaciones, Computación, Electrónica y Exposición industrial ROC&C'2004, Acapulco, Guerrero, México, 2004 (ver [54]).

A. P. Mateos Papis, «A Bandwidth Sharing Method under Node Autonomy and Short-Term Protection for QoS», de Conferencia Iberoamericana en Ingeniería Electrónica y Ciencias Computacionales 2011 (CIIEC'11), Aguascalientes, Aguascalientes, México, Abr. 2011.

Revistas (Journals).

A. P. Mateos Papis, «A Bandwidth Sharing Method under Node Autonomy and Short-Term Protection for QoS», Research in Computing Science, Instituto Politécnico Nacional, México (ISSN: 1870-4069), Special Issue on Advances in Computer Science and Electronic Systems, No. pp. 155-168, Abr. 2011 (Latinindex) (ver [85] y Apéndice A.1).

A. P. Mateos Papis, «A Method with Node Autonomy to Preserve QoS in Networks with per-Route Admission», Accepted for publication (submitted 2012 –accepted: June 2012) *Journal of Applied Research and Technology (JART),* vol. 11, nº Feb., pp. 42-64, 2013, Centro de Ciencias Aplicadas y Desarrollo Tecnológico (CCADET), Universidad Nacional Autónoma de México (ISI Thomson) (ver [80] y Apéndice A.2).

Sitios WEB Accesibles para su Consulta.

A. P. Mateos Papis, «Arquitectura de Servicios Diferenciados,» 2011. [En línea]. Available: http://ccd.cua.uam.mx/~amateos/REP-TEC/Rep-Tec-3-WFQ-Changing-Weights.pdf. [Último acceso: 7 Noviembre 2012] (ver [86]).

A.1 Artículo “A Bandwidth Sharing Method under Node Autonomy and Short-Term Protection for QoS” [85].

A Bandwidth Sharing Method under Node Autonomy and Short-Term Protection for QoS

Alfredo P. Mateos-Papis (a).

(a) Division of Sciences of Communication and Design. Universidad Autónoma Metropolitana. Cuajimalpa. Mexico DF.
amateos@correo.cua.uam.mx

Abstract. This paper proposes a method to share bandwidth between intersecting routes in networks subject to per-route distributed admission processes. With this method the current bandwidth share of a flow is guaranteed in the short-term against a possible bandwidth decrease caused by flows recently admitted in intersecting routes. However, this share-protection is decreased slowly such that, in the long term, bandwidth is distributed according to the intersecting route's observed requirements. With this method every node takes its own decisions, having an impact on the bandwidth sharing between routes and attaining, with this, global behaviors in the network. This method is expected to help preserve the Quality of Service in the routes while, because of its simplicity, allowing a simple bandwidth management in the network and possibly permitting its implementation in bigger-sized networks. The method proposed is termed: the Short Term Protection (STP) method.

Keywords. Quality of Service (QoS); Bandwidth Sharing; Distributed Traffic Control.

1. Introduction.

This article addresses in a novel way the problem bandwidth-sharing for the long-time studied QoS-aware-networks having per-route distributed admission processes, using a novel uncomplicated method where the nodes operate autonomously offering the possibility to attain global behaviors.

In bandwidth-limited data networks which offer Quality of Service (QoS), a flow-admission-process is used to foresee if the traffic increase caused by the entrance new flows to the network will originate an unacceptable deterioration of the QoS offered.

An admission process to a network can be centralized or distributed. In the centralized approach a single entity (like a bandwidth broker) makes admission decisions based on its global view of the network. This global-view requirement makes this approach less scalable compared with that of the distributed admission approach wherein there is an admission entity for each one of different parts of a network. The admission to each part is done based on a partial view of the network, with the potential risk that an admission decision could inadvertently cause an unacceptable deterioration of the QoS in other parts of the network.

The bandwidth sharing method introduced in this paper, which is called Short-Term Protection method (STP method), is aimed at networks with per-route distri-

© C. Delgado, C. Gutiérrez, R. Velázquez, H. Sossa (Eds.)
Advances in Computer Science and Electronic Systems
Research in Computing Science 52, 2011, pp. 155-168

Received 23/01/11
Accepted 20/02/11
Final version 07/03/11

buted admission processes, as those proposed in [1] [2], where there is no bandwidth reservation considered for any route. This method is intended to protect routes against bandwidth exhaustion derived from sudden increments in traffic in intersecting routes while still allowing the network to have a simple distributed management scheme which adjusts in accordance to the bandwidth necessities of the routes. With this method each node acts autonomously and there is no central admission authority considered. The nodes are aware neither of the existence of routes nor of the existence of flows in the network. Fig. 1(a) is a schematic representation of a network with two routes that intersect at node x (more specifically at its output interface) and share the bandwidth of the node's output link. In this paper these routes are called *intersecting routes* and node x is called an *intersection node*.

An intersection node has, at every one of its output interfaces, one queue for every one of its input interfaces. As an example, Fig. 1(b) represents a node with three input interfaces: A, B and C, from where three routes converge at the node and leave the node through the output interface D which has three queues.

Inspired by the Weighted Fair Queuing[1] (WFQ) scheduling algorithm [3], the STP method gives each queue a weight, however, each weight may change slowly according to an updating algorithm that is periodically evaluated. In this paper the STP method uses a WFQ scheduler, but the scheduler could be other kind of work-conserving scheduler [4] which could adequately operate with changing weights.

In the STP method traffic coming from the different input interfaces competes for bandwidth in a special way: the weights change slowly in favor of the queues which have bigger average lengths. For example, a route could lose up 20% of its weight in a 30-minute interval, against an intersecting route, if its queue length were constantly smaller –within this interval– compared to that of the intersecting route.

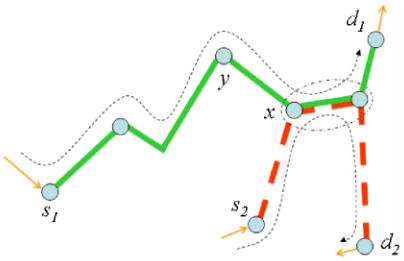
The STP method offers a simple and novel option for data networks to offer scalable QoS in comparison with the DiffServ model, studied since 13 years ago, which is oriented for scalability and flexibility in bandwidth-sharing (see section 2) but which poses a native difficulty in the evaluation of the traffic conditions in the networks.

2. Background and Related Works.

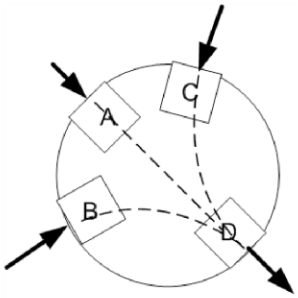
The STP method has been motivated by some characteristics of the nodes used in the DiffServ[2] model [5] [6], and in the network conditions proposed in [1] which deals with networks with per-route distributed admission where the admission decisions are taken at the edge of the routes. In DiffServ the packets are classified when entering to the so-called DiffServ domain (so its admission is per-class oriented instead of per-flow oriented), and the central nodes (also called core nodes) of that domain follow a specific behavior when treating the packets of a specific class.

It is reasonable to consider that the behaviors provided by a node are implemented at its output interfaces [7]. Typically, an output interface of a node keeps one queue (buffer) for every one of the classes to attend, which share the total bandwidth of the output interface. Any output interface with more than one queue in operation needs a

[1] Also called PGPS (Packet-by-Packet Generalized Processor Sharing).
[2] See technical report *Rep-Tec-4-Arquitectura DS.pdf* at http://ccd.cua.uam.mx/~amateos/.



(a) Routes $s_1 \rightarrow d_1$ and $s_2 \rightarrow d_2$ in a network intersect at the output interface of node x.



(b) Representation of an intersection node with three input interfaces, A, B, and C, and one output interface, D, in which the incoming traffic converges.

Fig. 1. The concepts of two intersecting routes (subfigure a), and the concept of an intersection node (subfigure b).

scheduler to choose the next packet to be dispatched over the output interface, thus creating a bandwidth sharing process between classes.

There are methods for optimization of a global indicator in a network by obtaining the most effective bandwidth-share between classes. For example [8] presents a bandwidth-sharing method described as proactive (in contrast with *reactive* algorithms) which uses a central bandwidth manager. Another bandwidth-sharing method between classes [9] adaptively adjusts the weights of a weighted round robin scheduler in every core node and does not depend on a central bandwidth manager. It looks like the nodes can effectively operate autonomously and cooperate to obtain an expected end-to-end QoS indicator.

Centikaya et al [1] [2] propose a method of admission to a network that has predefined routes. This method is defined as *distributed* since the admissions are made per-route, not per-flow oriented, without a central management, where a route can cause a

negative impact in intersecting routes when admitting a flow. The main research goal of the paper is the admission process, not the route-intersection problem.

The STP method is oriented towards handling the negative interaction between intersecting routes with per-route admission but it is not aware of any admission process.

In [10] a *Coordinated-Schedulers* method is presented to provide delay-bounds in a network, which is not per-flow oriented. In it, the core nodes modify each arriving-packet's priority index depending on whether the packet was serviced late or early at the upstream node as a consequence of cross-traffic. The method provides natural coordination between nodes but each node operates autonomously. This method can work in the context of a single class and it seems that it could have the benefit of protecting a route against intersecting routes. The STP method is simpler although it is not in the scope of this paper to study if it can attain the benefits of the method presented by [10].

3. The STP Method.

3.1 Remarks.

The STP method was conceived to operate within one class of traffic in a network, so this paper considers traffic in a single class. The possible interaction of a class using the STP method with schemes doing bandwidth provisioning between classes, to optimize a global indicator in a network, is out of the scope of this paper.

The evaluation of the STP method is made with the results obtained from simulations with traffic generated using a mixture of constant bit rate and Pareto sources. This traffic does not represent the diversity of traffic that might exist in a network; nevertheless these results are presented as a starting point for the evaluation of this method. In the simulations a change in the amount of traffic is made through the change of the number of sources.

The evaluation of the method is limited to a single direction in the routes as it is considered that the traffic in one direction is independent of the traffic in the opposite direction.

The topology used for the simulations is rather small. It has one central node (see section 2). The STP method is used in just the central node as this paper is aimed to compare the results in a network when a single central node uses the STP method with the results when that central node is a *normal* node, that is, the node uses a single output queue for each output interface. The author has made simulations with two nodes capable of using the STP method, which will be presented in a subsequent paper.

3.2 Condensed Explanation of the STP Method.

The QoS parameter of interest in this paper is the delay. This paper evaluates the STP method based on the end-to-end delay in the routes. This delay depends on several factors, including the queuing delay in the intersection nodes which will be the

factor of interest in this paper. This paper defines the term *delay-limit* as the maximum permissible delay that a packet can experiment while traversing a route.

Consider an output interface of a node working with the STP method, where there are N queues being attended by the scheduler (so there are N routes intersecting at that interface). The weight of each queue should tend to be proportional to its relative average length compared with the sum of the lengths of the queues. The rate of weight-change should be slow, that is, the bandwidth required to satisfy a bandwidth-greedy route should be released slowly in order to protect, temporarily, the routes which would loose bandwidth.

The method starts at time t_0 where the all the weighs have the same value. At the ending of every time-interval of duration τ, the method computes an indicator for every queue of the output interface, to compare the current weight of the queue with its current relative length with regard to the lengths of the other queues. The indicator is represented with $I\Delta$ in equation (1). This time-interval should be sufficiently big as to be able to observe several arrivals and departures of packets (in this paper $\tau = 1s$). After computing this indicator the method computes the new weights of the queues and substitutes those in operation. All this computation should take place in a time much smaller than τ. In (1), the computation of this indicator is made at time $t_0 + (r+1)\tau$ for queue i, where r is an integer such that $r \geq 0$, $\phi_i^{Act}(t_0 + r\tau)$ represents the weight of queue i, which has been valid from $t_0 + r\tau$ to $(t_0 + (r+1)\tau)^-$), and the average lengths of the queues, obtained at time $t_0 + (r+1)\tau$, are represented with $Q_i(t_0 + (r+1)\tau)$.

$$I\Delta\left(\phi_i^{Act}(t_0 + r\tau)\right) = \frac{Q_i(t_0 + (r+1)\tau)}{\sum_{j=1}^{N} Q_j(t_0 + (r+1)\tau)} - \phi_i^{Act}(t_0 + r\tau) \quad \textbf{(1)}$$

If, from equation (1), the indicator results negative for queue i, then this queue should lose weight so that the result of its new weight, $\phi_i^{Act}(t_0 + (r+1)\tau)$, minus its weight in operation, $\phi_i^{Act}(t_0 + r\tau)$, should reflect a negative increment (or a decrement). Equation (2) proposes a form of calculation for this negative increment, which is represented with $\Delta\left(\phi_i^{Act}(t_0 + r\tau)\right)$.

$$\Delta\left(\phi_i^{Act}(t_0 + r\tau)\right) = \phi_i^{Act}(t_0 + (r+1)\tau) - \phi_i^{Act}(t_0 + r\tau) = \ln(1-P)\frac{\tau}{T}\phi_i^{Act}(t_0 + r\tau) < 0 \quad \textbf{(2)}$$

The two important parameters of (2) are P and T. Parameter P is called the *loss factor*, which is a positive constant much smaller than 1 (with values near to 0.1). When P is very small then $-\ln(1-P) \approx P$. The parameter T represents a *time span* and it might be on the order of $1200s$. The ratio T/τ should be a big integer called K.

From (2) it can be seen that, as τ is much smaller than T, the decrement of the queue, from one interval to the next one, should be very small.

It can be shown that if queue i decreased its weight for K consecutive intervals of size τ within time-span T, then the ratio of its initial and ending weights, with regard to that time span, would be equal to $(1-P)$, that is: $1-P = \phi_i^{Act}\left(t_0+(l+1)T\right) / \phi_i^{Act}\left(t_0+lT\right)$. This formula can be written as: $\phi_i^{Act}\left(t_0+(l+1)T\right) - \phi_i^{Act}\left(t_0+lT\right) = -P\,\phi_i^{Act}\left(t_0+lT\right)$. With this it can be stated that with this method the maximum weight deterioration for a queue in a time T is P%. This result is demonstrated using the fact, which can be obtained from (1) and (2), that $\phi_i^{Act}\left(t_0+(m+K)\tau\right) = \phi_i^{Act}\left(t_0+m\tau\right)\left(1-f(P)/(T/\tau)\right)^{T/\tau}$. The variables l and m represent integers greater than, or equal to 0, and $f(P) = -\ln(1-P)$. Because of the limitation of space of this paper, all the details of this method are not developed here[3].

The STP method also proposes a calculation for the weight of every queue that should not decrease its weight, as indicated by (1), after every interval of duration τ, in such a way that the sum of the weights of all queues be always equal to 1.

Following the method proposed in [11], the average length Q_i of queue i is computed as $Q_i = (1-w_q)Q_i + w_q q_i$, where w_q is the averaging parameter and q_i is the instantaneous queue-length. This equation acts as a low-pass filter. In this equation, the smallest the value of w_q the smoother the output will be. The value for w_q is set to 0.002, to cope with the possibly burst behavior of the instantaneous queue-lengths.

4. Experimental Results

The performance evaluation of the STP method was done through a series of simulations which uses the topology shown in Fig. 2, a bounded network (or domain) with one central node, *c1*, and three edge nodes: *e1, e2* and *e3*. Outside the network boundaries there are three source nodes: *s1, s2, s3*, and three destination nodes: *d1*, *d2* and *d3*. Inside the network there are three routes: *Route1*, *Route2*, *Route3*. See that node *c1* acts as an exit node for *Route2*. Links inside the network have a bandwidth of 3*Mb/s*, for the operating Class. The links connecting the bounded network with the outside nodes have a bandwidth of 100*Mb/s*. The propagation delay of the links is 0.05ms. *Route1* and *Route3* intersect at the output interface of *c1* (going to *e2*). *Route2* intersects with *Route1* at the exit interface of *e1* (going to *c1*). *Route1*, then, suffers from two intersections which puts it in disadvantage against *Route3* which is not affected by *Route2*. In the experiments the situation is that *Route3* increases its traffic so that *Route1* is affected with this increase.

Node *e1* was selected to be a *normal* node in every experiment, and node *c1*, in a group of experiments uses the STP method and in other group of experiments it does not, as it is explained in the next paragraphs (remember that this paper is aimed at

[3] See technical report *Rep-Tec-2-Equations-STP-Method.pdf* at http://ccd.cua.uam.mx/~amateos/.

evaluating the effect of the STP method in just one node in the network). With the STP method *Route1*-traffic passes through one queue and *Route3*-traffic passes through the other queue of the output interface of *c1* going to *e2*. *For the sake of simplicity-of-explanation the first queue of c1 can be referred to as the queue for Route1 and the second queue as the queue for Route3*.

The results of these experiments are compared to evaluate the STP method. Because of the random nature of experiments the results are taken from the mean values of 40 to 60 experiments, for each result.

4.1. Experimental Setting

The experiments were carried out with the ns-2 simulator [12], version 2.33, also using the DS tools included in the simulator [13]. In the STP method these tools were modified to add the WFQ packet-scheduling operation, with the implementation proposed in [14], which uses the WFQ scheduler [3]. Additional modifications were made to incorporate the possibility to change the weights of the scheduler in a dynamic form, in accordance with the STP method[4].

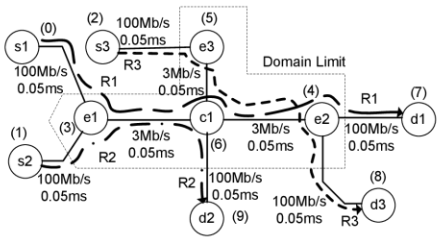


Fig. 2. The topology for the experiments has two routes: *Route1* traversing nodes *s1–e1–c1–e2–d1*, and *Route3* traversing nodes *s3–e3–c1–e2–d3*. The routes intersect at the output interface of node *c1* and separate at node *e2*. *Route1* and *Route2* intersect at the output interface of node *e1*.

The Pareto sources are *On/Off* with 250*ms* on/off duration, with 68 Kb/s during *On* periods and 95-byte packets [15][5] with Pareto shape parameter of 1.7 (infinite variance). The CBR sources have 1500-byte packets with 256Kb/s[6]. It should be clear that these rates are small compared with those actually used in the Internet, but they

[4] See technical report *Rep-Tec-3-WFQ-Changing-Weights.pdf* at http://ccd.cua.uam.mx/~amateos/.

[5] A voice packet can have 67 bytes which would make a 95-byte IP packet adding 8 bytes of UDP header and 20 bytes of IP header. A reasonable voice mean-rate of 3.0 Kbytes/s in one direction makes a traffic mean rate of 34Kb/s (3 Kbyte/s x 8 bits/byte x 95 / 67 = 34 Kb/s – considering the headers' overhead), making 68 Kb/s during the On period of On/Off sources.

[6] The rate value is taken from information from [16].

are used for the purpose of the evaluation of the STP method, within the scope of this article, as indicated in section 3.1.

It is supposed that for a long time before the beginning of each experiment the routes have had the same traffic. At the beginning of every experiment *Route3* increases its number of sources (without knowing about the possible deterioration of the QoS inflicted on *Route1*).

Three different groups of experiments were carried out, depending on the buffer used in the output interface of *c1*. In the first group, called "case *q1*", there is just one queue at the output interface of node *c1* (going to *e2*), which handles the traffic of the two routes. The second group, called "case *q2*", uses the STP method and utilizes two queues at the output interface of node *c1*, one corresponding to each one of *Route1* and *Route3*. The parameters employed for case *q2* are: $T = 1200s$, $P = 0.25$. The time τ is $1s$. The last groput, called "case *q2f*", also utilizes two queues at the output interface of node *c1*, one corresponding to each one of the routes, but the weight of each queue is fixed throughout the duration of the experiment.

For a P parameter value greater than 0, a *q2* case with parameter T being close to 0 should offer results similar to those of the *q1* case. A *q2* case with parameter T being close to infinite should offer results similar to those of a *q2f* case. All the other queues in all the nodes are Droptail. The queue lengths are big enough such that the total observed packet loss rate is always less than 2% in every experiment. In every experiment *Route1* and *Route2* have almost the same number of sources. *Route1* has 3 CBR sources + 17 Pareto sources, and *Route2* has 3 CBR sources + 16 Pareto sources. These numbers of sources are big enough to cause average packet-delays near to $3ms$ at the exit interface of node *e1*, without causing packet congestion losses.

Route3 initiates each experiment with 3 CBR sources + 8 Pareto sources. The initial traffic-share of *Route1* and *Route3*, at the exit of node *c1*, is: 0.5641 and 0.4359, respectively. These values are also the initial weights used for the queues of *c1* going to *e2*, for the *q2* case, which correspond respectively to *Route1* and *Rotue3*. Remember that the weights given by the STP method to intersecting routes in an output interface of a node (node *c1* in this case) tend to be the same as the bandwidth proportion that the routes take at that interface, in the long term. For the *q2f* case these values are taken as the fixed weights for the two queues of *c1* going to *e2*.

Each row of Table 1 shows the number of sources of *Route1* and *Route3*, their rates and the bandwidth percentage share at the exit interface of node *c1* going *e2* for every experiment. +PAR means the number of Pareto sources increased in *Route3* at the beginning of the experiment, for example, in the second row the total number of Pareto sources of *Route3* is 8 + 1 = 9). The heading –%R1 indicates, for a given row, the bandwidth percentage deterioration of *Route1* with regard to that which this route has in the first row. The queue-weight deterioration for a route with the STP method can not be bigger than the traffic-share deterioration of that route within a time-duration T). For example in the second row %R1 = 55.62 so that 1–55.62/56.41 = 0.01405 (1.41%). The maximum traffic-share deterioration for *Route1* shown in the table is 13.55%.

Table 1. Number of sources of *Route1* and *Route3*, their rates and the bandwidth-percentage share at the exit interface of node *c1*, going to *e2*, for every experiment.

CBR	CBR	PAR	PAR	+PAR	Kb/s	Kb/s	Kb/s	%	%	–%
R1	R3	R1	R3	R3	R1	R3	R3+R1	R3	R1	R1
3	3	17	8	0	1346	1040	2386	43.59	56.41	0.00
3	3	17	9	1	1346	1074	2420	44.38	55.62	1.41
3	3	17	10	2	1346	1108	2454	45.15	54.85	2.77
..	..	..	..							
3	3	17	16	8	1346	1312	2658	49.36	50.64	10.23
3	3	17	17	9	1346	1346	2692	50.00	50.00	11.37
3	3	17	18	10	1346	1380	2726	50.62	49.38	12.47
3	3	17	19	11	1346	1414	2760	51.23	48.77	13.55

4.2. Initial Results.

Fig. 3 presents the weight values for the *q2* case of the output interface in node *c1*, for four different values of *P*, through a period of time $T = 1200s$ in a special group of experiments where the traffic of *Route3* is bigger than that of *Route1*[7].

In the case of $P = 0.05$, at $t = T = 1200$ the weight of the queue attending *Route1* should be $0.564 * 0.95 = 0.5358$, and the experiments offered a weight value of 0.5291. For $P = 0.15$ this weight should be of $0.564 * 0.85 = 0.4794$, and the experiments offered a weight value of 0.4661. For $P = 0.25$ the weight obtained should be $0.564 * 0.75 = 0.4230$ and the experiments offered a weight value of 0.41. All these results are close to the projected ones.

4.3 Gain Results.

In order to assess the benefits of the STP method, this paper uses a figure of merit which consists of an overall gain for each route, for each experiment. This gain rewards with one point for every packet that traverses its route within the delay-limit of the route, and penalizes with ten points otherwise.

The first scenario considers experiments of short duration, 120*s*, for cases *q1* and *q2f*, and a delay limit is 18*ms*. The reason to make this scenario is to observe the behavior of these two cases to compare their results with those of the *q2* case, made in the next scenario. The left side of Fig. 4 shows the gains obtained for *Route1* and *Route3* (*Gain1* and *Gain3*) as a function of the number of sources added to *Route3*. Every point in the figure shows the gain obtained in each 120*s*-experiment. The upper abscissa axis shows the number of sources added to *Route3* in the experiment, and the lower abscissa axis shows the resulting rate in *Route3*. The gain in each experiment is normalized dividing it by the duration of the experiment in the simulation.

[7] *Route3* augments 30 Pareto sources at the beginning of the experiment time. Regarding Table 1, *Route1* has 40% of the traffic and *Route3* has 60% (only for this experiments the bandwidth of *c1*-*e2* is 4[Mb/s]).

For the *q1* case *Gain3* has a maximum value at 9, decreasing afterwards. The admission process of *Route3* should avoid allowing an increase of more than 9 sources. *Gain1* deteriorates with the increase of sources in *Route3*. For the *q2f* case it is also clear that the admission process of *Route3* should neither allow more than 9 sources. From 0 to 9 sources, for the *q1* case, *Gain1* goes from 704 a 537 points, representing a loss of (704-537) / 704 = 27.7%. For the *q2f* case *Gain1* goes from 825 to 742 points, representing a loss of (825 – 742) / 825 = 10%. It is clear that the *q2f* case protects *Route1*. It is observed that the *q1* case is a good option when the available bandwidth is big compared to the traffic in the routes[8].

The right side of Fig. 4 shows the gains and the delays of the routes for the *q1* case (the delays of the *q2f* case are not shown because of space limitations in this paper). The delays are computed as the 98th percentile of the delay observed by the packets traversing their corresponding route. In other words, the delay of the route is the time exceeded by the delay of only the 2% most delayed packets. The *q2f* case is not appropriate for networks where the traffic can change in the routes and where there is no reason to give fixed protection to routes.

The second scenario is one using the *q2* case with $P = 0.25$ and $T = 1200s$, with experiments of 1200*s* duration, where it is supposed that the change in traffic at *Route3* is at $t = 0s$ of the experiment, with no further change in traffic during those 1200*s*.

It is also supposed that the traffic of the routes has been the same for a long time before time 0*s* of the experiments, so that as the STP method tends to give to the weight of each queue the same value as its relative traffic-share, then the initial traffic-share values of *Route1* and *Route3* are reflected in the initial values of the weights of the queues of *c1*, which are: 0.5641 and 0.4359, for *Route1* and *Route3* respectively. The gains are evaluated at the ending part of each subsequent interval of 120*s* duration (that is the gain at time 1200*s* is evaluated from 1080 to 1200*s*).

The left side of Fig. 5 shows how the delay in the *q2* case augments with time in every experiment for *Route1* (and decreases for *Route3*). The right side shows how the *Gain1* and *Gain3* of *q2* case tend to be equal to case *q1* at the ending of each experiment (at 1200*s*) and tend to be equal to *q2f* case at the beginning of each experiment (at 120*s*). With this, it can be seen that in the first part of each experiment the *q2* case protects *Route1* as case *q2f* does, and at the ending part of each experiment the *q2* case tend to protect *Route1* as case *q1* does.

It can also be shown, from graphics not presented in this paper, that the total gain, that is the sum of gains (*Gain1* + *Gain3*), is approximately the same for the three cases. The reason for this is that the schedulers for *q2* and *q2f* cases are work conserving, that is, the schedulers serve at full transmission rate whenever there is data to be served, ideally having a service, for the *q2* and *q2f* cases, as big as that of the *q1* case. This is clear as the STP method is not intended to maximize the total gain indicator.

The result of other experiments for *q2* scenarios with parameter $T = 120s$ (instead of 1200*s*), not shown in this paper, show that the gains and delays behave very similar to that of Fig. 5, but they change 10 times faster. This results could give an insight of how to select the parameter *T*, which could be on the order of the mean time between

[8] The standard deviation obtained for the gain was big (for most cases of increased-sources it was bigger than the mean value times 0.5. The reason of this is that the sources are of Pareto type.

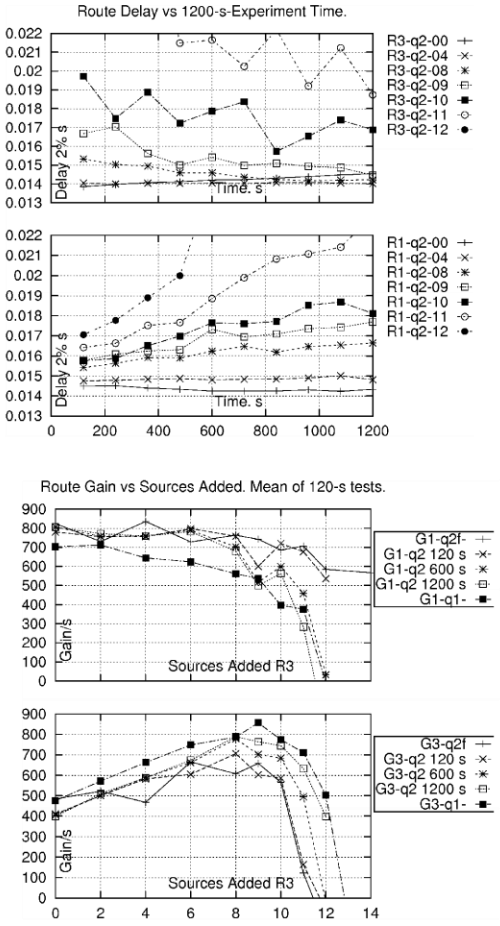


Fig. 5. The left side of the figure shows the delays of routes for *q2* case, throughout the experiment time. Label "R1-q2-04" stands for delay of *Route3* for 4 sources added in *Route3*. The right side of the figure shows the gains of routes for the *q2* case. Label "G1-q2 120 s" stands for *Gain1* of *q2* case at 120*s* of experiment. The other labels have similar meanings.

admission of flows, or the mean time of the life (duration) of the flows, in the network.

An example of a possible disadvantage of the *q2* case is, if at a certain moment *Route3* had a big weight and suddenly it decreased its traffic allowing for *Route1* to increase its traffic, and then, before *Route3* began to lose, substantially, its weight-share, it increased again its traffic, then *Route1* could suffer from this increase, even more than what it would do in a similar situation with a *q1* case.

5 Conclusions and Future Work.

This article addresses the mature problem bandwidth-sharing for QoS-aware networks with per-route distributed admission processes proposing a method which should help the network administrators to have confidence, at least within a time-window, about the amount of bandwidth they count with.

The method allows the nodes to operate autonomously, offering the possibility to attain global behaviors and being scalable. The objectives of the method are: 1- To protect, in the short term, the routes against traffic increments in intersecting routes, and 2- To dynamically assign, in the long term, a bandwidth share proportional to the average measured demands of the routes traversing congested links. As its scheduler is work-conserving this method should not impose a decrease of the transmission capacity in the network. The method's computations are separated by relatively long intervals of time (1*s*) so it should not cause noticeable performance-degradation in actual routers.

Under situation of little traffic the method is not better than the case of using single-queue configurations as there is no need to protect flows against each other.

Bandwidth reservation is still an appreciated form of service which can be a source of important economic compensations; however its overall performance depends heavily on the proper weight assignments which have to be setup by a central authority. On the contrary, the STP method is a simple method that dynamically adapts to the changing network conditions without the need of a central administration.

An analysis about how the STP parameters affect its performance is an important line of research.

References.

1 Centikaya, C., Knightly, E.: Egress Admission Control. In: IEEE INFOCOM 2000, vol. 3, pp. 1471-1480. (2000).

2 Cetinkaya, C., Kanodia, V., Knightly, E.: Scalable Services via Egress Admission Control. IEEE Transactions on Multimedia: Special Issue on Multimedia over IP, vol. 3(1), pp. 69-81. (2001).

3 Parekh, A., Gallager, R.: A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case. IEEE / ACM Transactions on Networking, vol. 1(3), pp. 346-357. (1993).

4 Neely, M.: Lecture Notes. EE 549 (from lecture 1 to 5). University of Southern California, http://www-rcf.usc.edu/~mjneely/ee549notes/EE549_Supplementary_Lecture_Notes_01.pdf

5 Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: RFC 2475. An Architecture for Differentiated Services. The Internet Society (1998).
6 Nichols, K., Blake, S., Baker, F., Black, D.: RFC 2474. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. The Internet Society (1998).
7 Kumar, V., Lakshman, T., Stiliadis, D.: Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet. IEEE Communications Magazine, vol. 36(5), pp. 152-164. (1998).
8 Hui, T., Tham, C.: Adaptive Provisioning of Differentiated Services Networks Based on Reinforcement Learning. IEEE Transactions on Systems, Man, and Cybernetics, vol. 33(4), pp. 492-501. (2003).
9 Wang, H., Shen, C., Shin, K.: Adaptive-Weighted Packet Scheduling for Premium Service. In: IEEE International Conference on Communications'2001, pp. 1846-1850. (2001).
10 Li, C., Knightly, E.: Schedulability Criterion and Performance Analysis of Coordinated Schedulers. IEEE/ACM Transactions on Networking, vol. 13(2), pp. 276 – 287. ISSN: 1063-6692. (2005).
11 Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestions Avoidance. IEEE/ACM Transactions on Networking, vol. 1(4), pp. 397-413. (1993).
12 **ns-2** network simulator, http://nsnam.isi.edu/nsnam/index.php/Main_Page
13 Pieda, P., Ethridge, J., Baines, M., Shallwani, F.: A Network Simulator Differentiated services Implementation. Open IP, Nortel Networks, http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/DOC/DSnortel.pdf
14 Mrkaic, A. (tutor: U. Fiedler, supervisor: Prof. Dr. B. Plattner). Porting a WFQ Scheduler into ns-2's Diffserv Environment. Computer Engineering and Networks Laboratory (Insitut für Technische Informatik und Kommunikationsnetze). Swiss Federal Institute of Technology. Eidgenössische Technische Hochschule Zürich (2001).
15 Salman, S., Schulzrinne, H.: An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. Department of Computer Science. Columbia University. 2004, http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf
16 Skype, https://support.skype.com/en/faq/FA1417/How-much-bandwidth-does-Skype-need;jsessionid=5A4C03A96FCF4A2AE9CE5D72D15093CD?frompage=search&q=how+much+bandwidth+for+a+call&fromSearchFirstPage=false

A.2 Artículo “A Method with Node Autonomy to Preserve QoS in Networks with per-Route Admission” [80]

A Method with Node Autonomy to Preserve QoS in Networks with Per-Route Admission

A. Mateos-Papis

División de Ciencias de la Comunicación y Diseño.
Universidad Autónoma Metropolitana. Unidad Cuajimalpa.
México, DF., México

ABSTRACT
This paper presents a method called the Short Term Protection (STP) which is applied to fixed networks where flows[1] are constrained to follow specific routes, where the admission-process is distributed, per-route, and where the traffic is sensitive to delay (like MPEG4 traffic). The share of bandwidth of any route is protected, in the short-term, against the traffic-increment of its intersecting-routes. In the long term, in every congested link, the share of bandwidth between the routes that intersect at that link is proportional to the average relative measured demands of those routes. The nodes act autonomously, without central administration. This method is expected to: 1) help network administrators to have confidence, within a time-frame , about the amount of bandwidth they count on for every route; 2) allow a simple bandwidth management in the network, with prospect to be scalable to at least tens of nodes. This paper presents the general and detailed operation of the method, the evaluation of the method, by simulations, including a comparison with other method, a discussion of a possible scenario of application, conclusions and possible paths for further research.

Keywords: quality of service (QoS); bandwidth sharing; distributed traffic control.

RESUMEN
Este artículo presenta un método llamado short term protection -STP- (Protección a Corto Plazo) que se aplica a redes fijas donde los flujos[1] están restringidos a seguir rutas específicas, donde el proceso de admisión es distribuido, por ruta, y donde el tráfico es sensible a retrasos (como el tráfico MPEG4). La porción de ancho de banda de cada ruta está protegida, en el corto plazo, contra incremento de tráfico en rutas de intersección. En el largo plazo, en cada enlace congestionado, la proporción de ancho de banda entre las rutas que se interceptan en ese enlace es proporcional a las demandas relativas promedio medidas de esas rutas. Los nodos actúan de manera autónoma, sin administración central. Se espera que este método: 1) Ayude a los administradores de la red a tener confianza, por un periodo de tiempo, acerca de la cantidad de ancho de banda con la que cuentan para cada ruta; 2) Permita una administración simple en la red, con prospecto de ser escalable hacia al menos decenas de nodos. Este artículo presenta la operación general y detallada del método, la evaluación del método, por simulaciones, incluyendo una comparación con otro método, un diálogo sobre un posible escenario de aplicación, conclusiones y posibles rutas para investigación futura.

1. Introduction

In contrast with other works which, for optimization purposes, propose schemes to share bandwidth between the diverse classes in networks which implement Quality of Service (QoS), this paper proposes a method which manages the available bandwidth inside a single class of a QoS-implementing network, to help simplify the admission-process to this class. So, in order to facilitate the explanations in this paper, the concept of class is not used; the concept of available bandwidth of a class is expressed, simply, as the available bandwidth of a network, and the concept of flow-admission, to a class, is expressed as the flow admission to a network.

Networks with limited resources, which offer satisfactory QoS to their admitted flows, may

[1] A flow is a sequence of related packets that enter a network through the same source-node and leave the network through the same egress-node.

suffer from deterioration of the QoS offered, after increasing their traffic (maybe as a consequence of admitting new flows). This deterioration may be small, and consequently permissible, while there are still plenty of available resources in the network; but along with the increase of more and more traffic in the network this deterioration may grow to the point where it becomes intolerable.

The admission-process for a flow, to enter a network, implies the operation of a predictive evaluation to avoid granting admissions which could turn the offered QoS, in the network, unacceptable. A network where every flow is admitted, with the restriction of traversing the network through a specific route, may be called: a network with per-route admission. The QoS offered to flows, in this kind of networks, is expected to be easier to maintain, compared to the QoS offered in networks where flows may take different paths at different moments.

The admission-process to a network with per-route admission may be called a per-route admission-process. In this case, every admission decision is made for a specific route. A network with per-route admission, which does not implement bandwidth reservation in any one of its routes, allows for more flexibility in the use of its bandwidth, with the drawback that these routes may intersect one another.

For example, a portion of a network is represented in Figure 1 with three routes, $s_1 - d_1$, $s_2 - d_2$ and $s_3 - d_3$, that converge at node x, arriving, each one, to the node's input interface A, B and C, respectively, and sharing the bandwidth of the node's output-interface D. In this paper, this node and the routes are referred to as *intersection node* x and *intersecting-routes*, respectively. These routes converge, specifically, at the output-interface D of node x (it can be said that the routes converge at the link coming out of node x through the output-interface D). For the sake of clarity, the output-interface D, of node x, is called the *intersection output-interface D*.

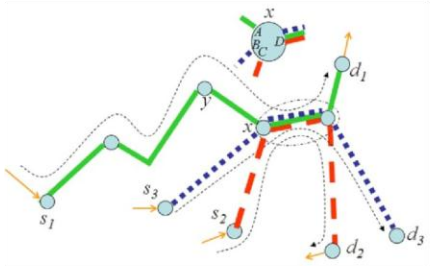


Figure 1. Routes $s_1 - d_1$, $s_2 - d_2$ and $s_3 - d_3$ intersect at the output-interface of node x, the intersection-node. This node has three input interfaces: A, B and C, as well as one output-interface D (which is an intersection output-interface) at which the traffic of the three intersecting-routes converge.

Whenever one route increases its traffic it may cause a decrease in the available bandwidth of its intersectingroutes, with a possible consequential deterioration of the QoS offered in those routes.

A per-route admission-process can be centralized or distributed. It is centralized if it is aware, at all times, of the traffic conditions in all the routes of the network, so that, before granting an admission to a route, it considers the possible effects of the admission in all the routes of the network.

A per-route admission-process is distributed if there is a separate admission-process for every route on the network, meaning that every admission-process is aware, only of the traffic conditions in its route, without considering the possible effects in the intersecting-routes when granting an admission. When evaluating if an admission to a route is granted or not, it is easier to consider the traffic conditions in just that route. This simplicity makes the distributed admission-processes to be more scalable, compared with its centralized-admission counterpart, but also, this simplicity is concomitant with the possible, inadvertent, deterioration of the QoS in intersecting-routes.

Within the type of networks which implement QoS without bandwidth reservation, those which implement per-route, distributed, admissions represent an important subset of networks, where the difficulty of the admission-process is alleviated.

This paper introduces a method called short-term protection (STP) method, which is intended to protect the routes of a network with per-route, distributed, admission, without bandwidth reservation, against the bandwidth decrease resulting from sudden increments in traffic in intersecting-routes. This method is easy to manage and adjusts in accordance to the bandwidth necessities of the routes. This method is prospected to be scalable to, at least, tens of nodes, and it is projected to be helpful for the network's route-administrators to have confidence, at least within a time- frame, about the amount of bandwidth they can count on.

The STP method is implemented at every output-interface of a core-node, on a network. Each node working with the method acts autonomously, without needing to be aware of the existence of routes, or of the existence of flows on the network, and without marking any packet. This method does not require any central admission authority on the network; still, this method is expected to attain a single extensive behavior on the networks where it is implemented.

The STP method is aimed to protect the bandwidth of routes, so the QoS parameter to evaluate this method is associated with the routes. To understand the operation of the STP method, think that a copy of the STP method operates in every output-interface of every node. At an output-interface, this method protects every queue at that interface, against the increase traffic in the other queues. Each queue represents a route intersecting at that node. So, the protection of the queue has an extending effect to protect the associated route (the input interfaces of nodes are considered to have no queuing delay [1]). At the ending, the protection offered by this method is reflected in the protection of the delay in the routes.

The QoS parameter of interest, in this paper, is the end-to-end delay in every route, so, the terms “route-delay” and “delay in route” refer to this end-to-end delay. Thus, the protection of the method is evaluated with regard to this delay. The delay in a given route depends on the available bandwidth at every node of the route, including those nodes where the route intersects with other routes.

To the best of the author's knowledge, there is no method which is oriented to work inside a class, which assigns, in the output-interfaces of every core-node, a different queue for every input interface of the node, as the STP method does (as it is explained in section 3).

The rest of the paper is organized as follows: Section 2 presents the motivation to do this work, and presents related work. Section 3 presents the conceptual framework of the STP method. Section 4 presents experimental results. Section 5 explores the scaling possibilities of the STP method. Section 6 presents a discussion of the scenarios where the STP method could be used; some application-considerations of the method; and a discussion of the STP method with regard to tendencies of QoS. Section 7 presents the conclusions derived from this paper, as well as some possible paths for further research. Finally, the demonstration of a theorem, related with the STP method, is given in Appendix A.

2. Background and related work

The STP method is motivated by some characteristics of the types of nodes used by the DiffServ model [2, 3] and by the network conditions proposed in [4, 5], where the networks considered have predefined routes, the admission-processes is per-route, distributed, and where the admission decisions are taken at the edges of the routes. These last two works highlight the problem of the traffic interactions between intersecting-routes, and they make reference to a method in which the routes can get information about the traffic of the other routes, to take admission decisions which do not unacceptably impact the intersecting-routes, with the cost of affecting the simplicity of their proposed per-route admission-process. However, the main research goal of these work is the admission-process itself, rather than the route intersection problem.

There are methods which obtain the most effective share of bandwidth between classes in a network, in terms of a global indicator to be optimized; however, these methods do not address the route-

intersection problem inside a single class. A method like this is presented in [6], where the authors describe a method referred to as a Reinforcement-Learning algorithm, to attain the most effective share of bandwidth between different classes in a network, through the maximization of a global gain-figure (resulting from the addition of the gains of all the classes in that network).

This method uses a central bandwidth manager which uses a feedback scheme to periodically propose a new bandwidth allocation, which is expected to be better than the one being used. The new setting is applied and tested and the algorithm learns. The prospective characteristic of this algorithm is the reason why the authors identify it as "proactive", in contrast with other reactive algorithms which are based on control-feedback principles. This method considers that there is contention between classes for bandwidth. The allocation of bandwidth for the diverse classes is done through the adjustment of the weight of each class, used by the scheduler located at each output-interface, in every node. There is a central bandwidth-manager, in charge of running the algorithm and evaluating the global gain.

Comparatively, the STP method considers the contention for bandwidth, not between classes but between intersecting-routes within a class. This method allocates bandwidth through dynamic queue-weight adjustments, at every output-interface of every node where two or more routes intersect; however, there is no central authority which allocates bandwidth or which evaluates a global gain: the nodes operate autonomously when allocating bandwidth.

A Coordinated-Schedulers method is presented in [7], which provides guaranteed service, in terms of end-to-end delay, without per-flow state in the network core. In this method, the nodes modify the value of each arriving-packet's priority-index (stored in its header), which represents the eligible time for the packet to be serviced. Any packet that was serviced late, at an upstream node, as a consequence of the excess of traffic in other classes contending for bandwidth in that node, arrives to the downstream node with a favorable priority-index value, in comparison with packets of other flows contending for service, and which might have been serviced early at their respective upstream nodes. This method provides natural coordination between the nodes to attain a global behavior and with the nodes operating autonomously, with the complexity that it requires the marking of every packet. This method is of special interest in the present paper (see subsection 4.5), since it can be used, for comparison purposes, applied to protect a route against intersecting-routes within a single class.

In [8], a scheduling algorithm is proposed, which intends to deliver fair bandwidth between classes and enhance the bandwidth utilization on a network offering QoS. In the scheduler, this method adjusts the weight of the queue of every class, based on the queue-length. A class gets more bandwidth if the size of its queue is bigger than the size of the other queues. Each node operates autonomously.

In [9], a method is proposed, which adaptively adjusts the weights of a weighted packet-scheduler, such as a weighted-round-robin scheduler, in every core-node of a network, to protect the Premium Service against the transient burstiness of the Expedited-Forwarding per-hop behavior (PHB), in the differentiated service architecture. The weights adjust according to the average queue-sizes. The nodes operate autonomously and cooperate to obtain QoS indicators, achieving low loss rate, low delay and delay jitter for the premium service.

In [10], a scheme with two parts is presented. The first part is an algorithm which operates autonomously at every core-node. The algorithm changes the weights of the per-class work-conserving WFQ (weighted fair queuing) scheduler, at the time of every packet-arrival. With a PGPS-like (packet-by-packet generalized processor sharing) method, the algorithm calculates the packet attention-time ($Next(i)$), and if the waiting-time is longer than that calculated for the last packet arrived to the queue, then the queue-weight is increased. The purpose of the algorithm is to maintain dynamic fairness according to the delay status of every queue. The second part of the scheme is a flow shaping algorithm operating at every class-queue of the edge-nodes, to maintain fairness between different flows. This algorithm has an excessive work-load calculating weights at the time of every packet arrival.

Finally, in [11], a sophisticated scheme for operation in a DiffServ (differentiated services) network is presented. This scheme includes a self-adaptive algorithm operating autonomously at every core-node, which controls local delay and loss values, by adjusting buffer sizes, queue-dropping thresholds and weights of the per-class weighted schedulers of the node, with the objective of maintaining delay bounds, differential loss bounds, and bandwidth priority assurances, across the service classes throughout the core network.

In order to attain global QoS objectives, all these methods may allow sudden impacts on flows "already admitted" which have not increased their traffic. Besides, all these methods focus on the share of bandwidth between classes.

3. The STP method

This section presents the operation of the STP method. The general scenario for this method is presented in Figure 1. Let us consider that node x works with the STP method. A node like this has, at every one of its output-interfaces, one queue for every one of its input interfaces (considering that the queues in a node are formed only at its output-interfaces [1]). Specifically, node x would have three queues at its output-interface D, one for every one of its three input interfaces. Each queue of the intersection output-interface has a weight which may change, slowly, according to an updating algorithm which periodically compares the traffic coming from each input interface, through the evaluation of the average length of each queue. With this algorithm, the queues at the output-interface compete for bandwidth.

3.1 Initial assumptions, general procedure and nomenclature of the method

This subsection explains the operation of the STP method, implemented in one intersection output-interface of a node, where it is supposed that there are N queues, where $N \geq 2$. At the beginning of the operation of the method the queues are empty and all the weights have the same value.

The method calculates the average length of every queue each time a packet arrives or departs from that queue. This average length (*avg*) is computed with the use of Equation 1 (as it is done in [12]), where w_q is the averaging parameter, and q is the instantaneous queue-length.

$$avg = (1 - w_q)\, avg + w_q\, q \qquad (1)$$

This equation acts as a low-pass filter. The smallest the value of w_q the smoother the output will be. The value for w_q is set to 0.002, to cope with the burst behavior of the instantaneous queue-length.

Besides the calculations of the average-lengths of the queues, the method makes its main bulk of calculations at the ending part of every time-interval of size τ (for simplicity it is called "a time interval τ") the first time-interval beginning at time t_0. The time interval τ should be sufficiently big as to be able to observe many arrivals and departures of packets. In this paper $\tau = 1(s)$.

These calculations have the objective to obtain a new set of weights for the queues. The beginning of the ending part of each time interval τ is represented with $(t_0 + (r+1)\tau)^-$ (meaning: just before time $(t_0 + (r+1)\tau)$), where r is an integer such that $r \geq 0$.

The time-interval required to complete this bulk of calculations is very small, compared with τ, so, in the algorithm of the method it is considered that these calculations take no time[2] and begin and finish at time $(t_0 + (r+1)\tau)^-$.

Only at time $(t_0 + (r+1)\tau)$, the new set of values, recently computed for the queue-weights, replace the set of current queue-weights (those that were in use in the time interval τ that just ended). There may be a small time to wait before doing this updating in the scheduler, as required by the normal operation of the scheduler. So, the weights used by the scheduler are fixed from time $(t_0 + r\tau)$ to time $(t_0 + (r+1)\tau)^-$, and the new weights take effect at time $(t_0 + (r+1)\tau)$. The weight of queue i, at time t, is written as $\phi_i(t)$.

[2] The STP method should take less than 200 floating-point operations, which are calculated in several nanoseconds by modern processors.

In this paper the STP method uses a WFQ scheduler [13] (also called packet-by-packet generalized processor sharing), but the method is not intended to be restricted to this kind of work-conserving scheduler.

At the beginning of every bulk of calculations, the method checks if all the queues are empty. If so, the time resets to t_0, leaving the queue-weights as they were. If at least one of the queues is not empty, then the calculation of a new set of values, for the queue-weights, takes place.

In the calculations for every queue i, the method computes an indicator (represented with I_i –see Equation 9), to compare the current weight of the queue with its current relative average-length. The term "relative" means that the average length of the queue is divided by the sum of the average lengths of all the queues of the intersection output-interface. The weight of every queue is decreased if its relative average length results smaller than its current weight; otherwise, that weight is increased. In this method, every weight-change is always very small, and always all the queue-weights add up to 1. The weight-increment of queue i, from time $(t_0 + r\tau)$ to time $(t_0 + (r + 1)\tau)$, is written as $\Delta_i(t_0 + r\tau)$, such that:

$$\Delta_i(t_0 + r\tau) = \phi_i(t_0 + (r+1)\tau) - \phi_i(t_0 + r\tau) \quad (2)$$

The average length of queue i, at time t, is written as $Q_i(t)$.

Whenever, after a calculation for updating the weights, the indicator I_i indicates that queue i has to decrease its weight, the value for its decrement is calculated as:

$$\Delta_i(t_0 + r\tau) = f\frac{\tau}{T}\phi_i(t_0 + r\tau) \quad (3)$$

In Equation 3. T is a period of time such that T/τ is an integer, $T/\tau >> 1$, and f is a negative factor which causes the value of $\Delta_i(t_0 + r\tau)$ to be negative too. If from time $(t_0 + r\tau)$, queue i obtained T/τ consecutive results indicating that it has to decrease its weight, then it would hold that:

$$\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) - \phi_i(t_0 + r\tau) \approx \phi_i(t_0 + r\tau)\left(e^f - 1\right) \quad (4)$$

The proof of Equation 4 is given in Theorem 1, in Appendix A, which makes use of Equations 2 and 3.

In Equation 4, it is useful to make f to be equal to $f(P) = ln(1 - P)$, where P is a small positive constant, such that $P > 0$ and $P << 1$. So, given that $1 - P < 1$, then $f(P) = ln(1 - P) < 0$, such that the value of $\Delta_i(t_0 + r\tau)$ is negative. Then Equation 4 turns into Equation 5.

$$\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) - \phi_i(t_0 + r\tau) \approx \phi_i(t_0 + r\tau)\left(e^{ln(1-P)} - 1\right) = -P\phi_i(t_0 + r\tau) \quad (5)$$

Equation 5 can be rewritten as:

$$\frac{\phi_i(t_0 + r\tau + T)}{\phi_i(t_0 + r\tau)} \approx (1 - P) \quad (6)$$

This ratio means that the weight of queue i has lost (P x 100)% of its value, in the big interval of length T. For example if $P = 0.2$, this ratio is equal to 0.8 and the weight loss is 20% of the weight value.

That is why the argument P is called "the loss factor".

With the above results, it is observed that the STP method enforces the weight-loss of the queue to be limited in relation with the factor P, in a big interval of length T (in the evaluation part –see Section 4– of this paper the value of T is 960(s)). Equation 3, then turns into Equation 7.

$$\Delta_i(t_0 + r\tau) = ln(1 - P)\frac{\tau}{T}\phi_i(t_0 + r\tau) \quad (7)$$

Also, it is important to notice that, as $\tau/T << 1$, then $|\Delta_i(t_0 + r\tau)| << 1$, and $\sum_{\forall i}|\Delta_i(t_0 + r\tau)| << 1$.

Note. For the calculation of Equation 7, when the value of P is very small, then $ln(1 - P) \approx -P$, and the calculation of the logarithm could be considered to be useless, but when the value of P is not too small (0.2 for example), then the calculation of the logarithm is necessary.

STP method steps

BEGINNING.

Require: At $t = t_0 + r\tau$

The values of the queue-weights were left intact, or they have just been calculated. In the last case, with the calculated weights, make sure[3] that:

$$\sum_{i=1}^{N} \phi_i(t_0 + r\tau) = 1 \qquad (8)$$
$$0 \le \phi_i(t_0 + r\tau) \le 1$$

To update the weights of the queues in use, with the new calculated weights, follow the next steps.

Require: At $t = (t_0 + (r+1)\tau)^-$

if all the queues are empty **then**

Make $t \leftarrow t_0$

The values of the weights are kept intact.

$r \leftarrow r + 1$

Go to the BEGINNING.

end if

At this stage at least one of the queues is not empty, so a calculation to obtain a new set of values, for the queue-weights, is about to commence.

In this recently finished interval of length τ, obtain the average lengths of the queues, and the values of the weights in use.

for $\{ i = 1 \rightarrow N \}$ **do**

[3] Each one of the calculated-weighs is divided by the sum of these weights. This corrects any possible, very small, deviation from 1, which the sum of the weights could have.

Calculate the indicator $I_i(t_0 + (r+1)\tau)^-$, as:

$$I_i(t_0 + (r+1)\tau)^- \leftarrow \frac{Q_i(t_0 + (r+1)\tau)^-}{\sum_{j=1}^{N} Q_j(t_0 + (r+1)\tau)^-} - \phi_i(t_0 + r\tau) \qquad (9)$$

Note. The second member of the assignment-expression 9 has two terms. The first term is the relative average length of queue *i*. The second term, $\phi_i(t_0 + r\tau)$, is the weight in use for queue *i*. So this indicator is a comparison of these two values.

if $I_i(t_0 + (r+1)\tau)^- < 0$ **then**

Queue i is marked to lose weight and it is considered to be in the set L of queues (where L stands for *Loss*).

$\Delta_i(t_0 + r\tau)$ is calculated with Equation 7, repeated, as an assignment expression, in expression 10.

$$\Delta_i(t_0 + r\tau) \leftarrow ln(1 - P)\frac{\tau}{T}\phi_i(t_0 + r\tau) \qquad (10)$$

end if $I_i(t_0 + (r+1)\tau)^- < 0$

if $I_i(t_0 + (r+1)\tau)^- = 0$ **then**

For this uncommon situation, queue *i* is considered to be in the set L of queues, and its weight-increment is assigned the zero value.

$$\Delta_i(t_0 + r\tau) \leftarrow 0 \qquad (11)$$

end if $I_i(t_0 + (r+1)\tau)^- = 0$

if $I_i(t_0 + (r+1)\tau)^- > 0$ **then**

Queue i is marked to gain weight, and it is considered to be in the set G of queues (where G stands for *Gain*).

No calculation of $\Delta_i(t_0 + r\tau)$ is made, but after the ending of this loop.

end if $I_i(t_0+(r+1)\tau)^- > 0$

end for $i = 1 \rightarrow N$

for $\forall i \in G$ **do**

$$\Delta_i(t_0+r\tau) \leftarrow \frac{I_i(t_0+(r+1)\tau)^-}{\sum_{j\in G} I_j(t_0+(r+1)\tau)^-}\left[-\sum_{j\in L}\Delta_j(t_0+r\tau)\right] \quad (12)$$

end for $\forall i \in G$

$r \leftarrow r+1$

Go to the BEGINNING

END

Remarks of the STP method.

When adding both members of expression 12, $\forall i \in G$, expression 13 is obtained:

$$\sum_{i\in G}\Delta_i(t_0+r\tau) \leftarrow \frac{\sum_{i\in G} I_i(t_0+(r+1)\tau)^-}{\sum_{j\in G} I_j(t_0+(r+1)\tau)^-}\left[-\sum_{j\in L}\Delta_j(t_0+r\tau)\right] \quad (13)$$

Such that:

$$\sum_{i\in G}\Delta_i(t_0+r\tau) = -\sum_{j\in L}\Delta_j(t_0+r\tau) \quad (14)$$

So the sum of the weight-increments is equal to the negative of the sum of the weight-decrements.

4. Evaluation

This section presents the evaluation of the STP method with the use of the ns-2 simulator [14, 15]. To evaluate the STP method, the DS tools for ns-2 [16] were modified to add the WFQ packet-scheduling proposed in [17], with some amendments and also several additions, made in order to dynamically change the weights of the scheduler according to the proposed STP method.

Note that on the network used for the experiments, every input interface of the intersection nodes of the network carries the traffic of only one route which intersects at that node. That is why each queue of each one of the intersection output-interfaces is referred to as "the queue of" a specific route. It must be clear that these naming simplifications may not be valid in other general cases, for the STP-method application.

4.1 Construction of the network topology

The network topology used in the experiments is shown in Figure 2. This topology is a bounded network, similar to that found in the Figure 4 in [7].

The network has three core-nodes (c_0, c_1 and c_2), and eight edge-nodes (e_1, ..., e_8). In this network, the core-nodes are the only nodes where the routes intersect, so the core-nodes are the intersection-nodes. More specifically, the intersection output-interfaces of the core-nodes are those at the links: $c_0 - c_1$, $c_1 - c_2$ and $c_2 - e_8$. All the other interfaces of the core-nodes and all the output-interfaces of the edge-nodes use a single queue.

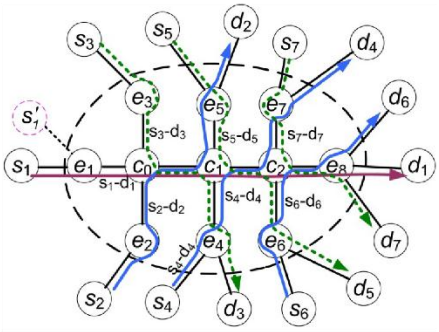


Figure 2. Topology of the bounded network used in all the experiments. The network has three core-nodes c_i, and eight edge-nodes e_i. Outside the network there are source nodes s_i, and destination nodes d_i. The network is traversed by seven routes, which intersect in groups of three routes. Even though the STP method is presented with its benefits, a route $s'_1 - d_1$ is used to observe a weakness of the STP method.

Outside the network boundaries there are seven source nodes, s_1, ..., s_7, and seven destination nodes, d_1, ..., d_7. All the output-interfaces of these nodes have one queue.

In this network topology, there are seven routes ($s_1 - d_1$, $s_2 - d_2$, ... $s_7 - d_7$). Route $s_1 - d_1$, which is the longest one, traverses 7 nodes: s_1, e_1, c_0, c_1, c_2, e_8 and d_1. The other routes are smaller, in terms of traversed nodes. For example, Route $s_2 - d_2$ traverses 6 nodes: s_2, e_2, c_0, c_1, e_5 and d_2, and Route $s_3 - d_3$ traverses 6 nodes: s_3, e_3, c_0, c_1, e_4 and d_3. These three routes intersect at the intersection output-interface of core-node c_0 (going to core-node c_1).

Routes $s_1 - d_1$, $s_4 - d_4$ and $s_5 - d_5$ intersect at the intersection output-interface of core-node c_1 (going to core-node c_2). Finally, routes $s_1 - d_1$, $s_6 - d_6$ and $s_7 - d_7$ intersect at the intersection output-interface of core-node c_2 (going to core-node e_8). There are no other intersections on this network.

Only route $s_1 - d_1$ intersects with two routes three times, that is, at the intersection output-interface of each one of the three core-nodes. All the other routes intersect with two routes at the intersection output-interface of just one core-node.

The links inside the network have a bandwidth of 30(*Mb*/*s*). The links connecting the edge-nodes with the outside nodes have a bandwidth of 100(*Mb*/*s*). Every link has 0.05(*ms*) of propagation delay.

Figure 2, also shows route $s_1' - d_1$. If this route were in use, it could not be distinguished apart from route $s_1 - d_1$ by the STP method, at nodes c_0, c_1 and c_2, as both routes enter the network through node e_1. If the STP method were also used at the intersection output-interface of node e_1, then these two routes could be distinguished by the method at that node.

The conditions for the experiments are:

- As it has been highlighted, in every experiment of this paper, throughout all the experiment-time, each flow follows the same route. That is, there is just one flow considered for every route, therefore the packets of a flow are referred to as the packets of the corresponding route.
- The result of each experiment is the calculated-delay of the packets to pass through their respective routes (so this is an end-to-end packet-delay). The delay obtained for every route is the 98-percentile-delay, that is, this delay is the delay of the packet in the limit of the 2% most delayed packets to go across the route. This delay is calculated for every interval of 120(*s*), inside the experiment time-duration.
- The delay-calculations use the trace-files obtained from the tests done.
- The reported result of each experiment is the average of the results of 10 simulations, which is referred to as the result of the experiment[4].
- In the experiments of this paper, a flow is created from the traffic generated by the "traffic sources" (or simply the "sources"). A source is a place where traffic is generated in ns-2. A source is connected to a "source node" (like any one of the source-nodes, s_i, of Figure 2).
- This network was tested with UDP-based MPEG4 traffic (see subsection 6.2 for a discussion about source traffic). For this purpose, the algorithm to generate this kind of traffic [19] was added to ns-2. The average rate, with the selected settings for this algorithm, turned out to be 0.621(*Mb*/*s*) for each source. The lengths of the generated packets

[4] The result of every experiment is the mean value of the results of 10 simulations. This is an estimated mean. In order to locate the confidence interval to indicate the reliability of the estimate, the Bootstrap Percentile Confidence Interval method was used, using 1000 different re-samples of the 10 simulation-results. This method has been successful with a broad range of probability distributions [18]. The method indicated that, with a 90% probability, the real mean value lies within a window located from 6.67% below to 6.99% above the estimated mean value.

varied, but generally that length was of 1000(*bytes*).

- In comparison to the newer traffic-generation methods, which use test beds [20], the traffic generation method in this paper could be considered to be restrictive in terms of its lack of precision in its inter-packet generation times, and also in terms of its lack of responsiveness with regard to changes on the network conditions, including the network traffic; nevertheless, the following points may be noted:
  - As the buffers, of the nodes, used in the experiments of this paper are large, to avoid the loss of packets, as a result of queue saturation, the results in these experiments should not be sensitive to slight deviations in packet-generation patterns. The time-scales used in the experiments of this paper are, rather, capacity-planning scales.
  - The limitation of the lack of responsiveness indicated is handled, in this paper, by doing diverse experiments, all of them beginning with the same traffic conditions. Each experiment has 1560(*s*) of experiment time-duration, and at time 600(*s*), there is a change in the traffic conditions, to test the reaction of the STP method, and the reaction of other methods also tested in this paper, for comparison purposes. This change in the traffic conditions is caused when route $s_2 - d_2$ increases its traffic at time 600(*s*).
- In the experiments, the initial traffic in every route is big enough to make the waiting times of the packets, in the queues of the intersection output-interfaces, considerable. In this way, the delay-increment in the routes, caused by the traffic-increment in route $s_2 - d_2$, can be observed in the results of the experiments.
  - In every experiment, from 0(*s*) to 600(*s*) of the experiment-time, the flow of every route has 12 sources. Then, at time 600(*s*), there is a change in the traffic conditions, as the flow of route $s_2 - d_2$ increases its number of sources at that time. As indicated, the purpose of this increment is to observe the delay-impact on the other routes of the network.
- In this paper, diverse solutions to handle traffic are tested, besides the STP method which is presented in subsection 4.4. All these solutions are presented to compare their results.
- The experimental settings for the tested-solutions, that is, the topology and traffic situations of the experiments are the same for all the solutions, where four routes are selected to show their results. These routes are: $s_1 - d_1$, $s_2 - d_2$, $s_3 - d_3$ and $s_4 - d_4$.
- The experimental settings of the experiment of subsection 4.6 are an exception. In this subsection the experimental settings are a different from those of the other experiments.

4.2 Solution with one queue at the intersecting output-interfaces of the core nodes

In this solution all the intersection output-interfaces of the core-nodes have just one output-queue, a situation which could be referred to as "the traditional solution" to handle the traffic.

Figure 3, shows the results of the experiments for this solution. Subfigures 3-A through 3-D show the results of four different experiments. The difference, from one experiment to the following one, is the number of sources increased in route $s_2 - d_2$ at time 600(*s*). The increments are: 2, 4, 6 and 10 sources, for each experiment, at time 600(*s*). Label "**s1 02**" stands for "Route-delay in route $s_1 - d_1$, where, at time 600(*s*), route $s_2 - d_2$ increases its traffic in 2 sources". The other labels have similar meaning.

In all subfigures it is observed that, at time 600(*s*), there is an increase of delay in every one of the routes $s_1 - d_1$, $s_2 - d_2$ and $s_3 - d_3$. These increments depend on the traffic-increase in route $s_2 - d_2$. In every subfigure the delay in route $s_1 - d_1$ is bigger than the delay in each one of the other routes, as route $s_1 - d_1$ is the longest one. Every route-delay results, mainly, from the delay at the intersection output-interface of core-node c_0

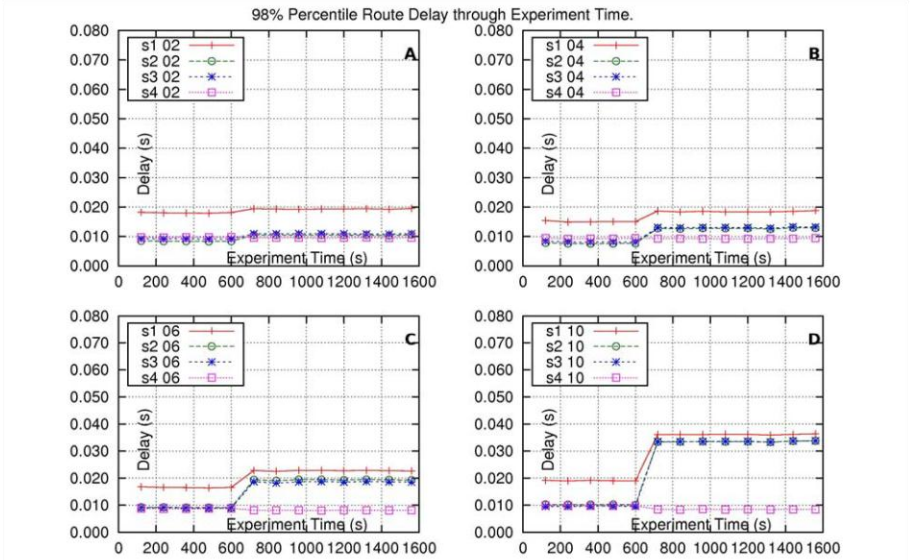


Figure 3. Delay-results of the experiments where each core-node has just one queue at its intersection output-interface. Label "**s1 02**" stands for "Route-delay in route $s_1 - d_1$" where at time 600(*s*) the route $s_2 - d_2$ increases its traffic in 2 sources". The other labels have similar meaning.

(going to core-node c_1). All these three routes suffer the same delay at this output-interface because the packets of these routes are placed in the same single queue of the interface. Note that the two lines corresponding to the delay in route $s_2 - d_2$ and the delay in route $s_3 - d_3$ are, practically, overlapped.

The delay in route $s_4 - d_4$ is not affected by the increment of traffic in route $s_2 - d_2$ because these two routes do not intersect.

In this, and in all the tested solutions, it is the route $s_2 - d_2$ (its administrators) itself which would have to evaluate, within its admission procedure, the amount of delay it can tolerate before admitting new traffic.

4.3. Solution with three queues, with fixed weights, at the intersecting output-interfaces of the core nodes

In this solution, at the intersection output-interface of every core-node, c_0, c_1 and c_2, there are three queues, one for each one of the three input interfaces of the node. Each queue has a fixed weight, equal to the weight of each one of the other two queues. This is another solution which can be referred to as "traditional".

The subfigures of Figure 4, show the results of the experiments for this solution. The meaning of the labels of these subfigures is similar to those of Figure 3. In all subfigures it is observed that, as a result of the increase of traffic in route $s_2 - d_2$, at time 600(*s*), there is a delay-increment in this route; the delay in route $s_1 - d_1$ and the delay in

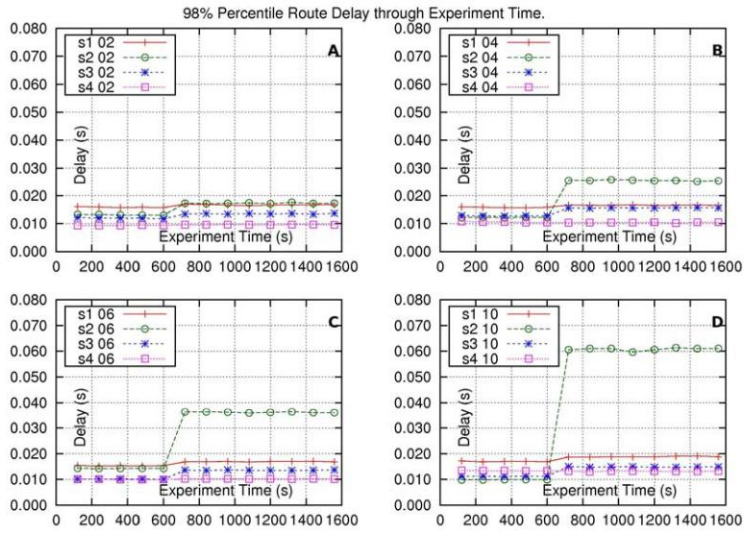


Figure 4. Delay-results of the experiments where each core-node has three queues at its intersection output-interface. The three queues have equal, fixed, weights. The meaning of the labels of these subfigures is similar to those of Figure 3.

route $s_3 - d_3$ are both barely affected; and the delay in route $s_4 - d_4$ is not affected.

In this case, the fixed weights of the queues of routes $s_1 - d_1$ and $s_3 - d_3$, at the intersection output-interface of core-node c_0, have protected these routes from the traffic-increase in route $s_2 - d_2$.

4.4 Solution with the STP method

In this solution, at the intersection output-interface of every core-node c_0, c_1, and c_2, the STP method operates with parameters T = 960(*s*) and P = 0.25.

For this solution, at the intersection output-interface of every core-node, there are three queues, each queue corresponding to each one of the three input interfaces of the node.

The subfigures of Figure 5 show the results of experiments for this solution. The meaning of the labels of these subfigures is similar to those of Figures 3 and 4.

It has been observed that, at time 600(*s*), the delay in route $s_2 - d_2$ increases as a consequence of the increase of traffic in this route, also at time 600(*s*). The delay-increment in route $s_2 - d_2$ is more pronounced depending on the amount of traffic increased in this route.

After time 600(*s*), the delay in route $s_2 - d_2$ decreases as a result of the gradual weight-increase of the queue of this route (remember that this queue is located at the intersection output-interface of core-node c_0). This weight increases as a result of the bigger average length of the queue.

The delays in routes $s_1 - d_1$ and $s_3 - d_3$ have a slight increase, at time 600(*s*), as these routes

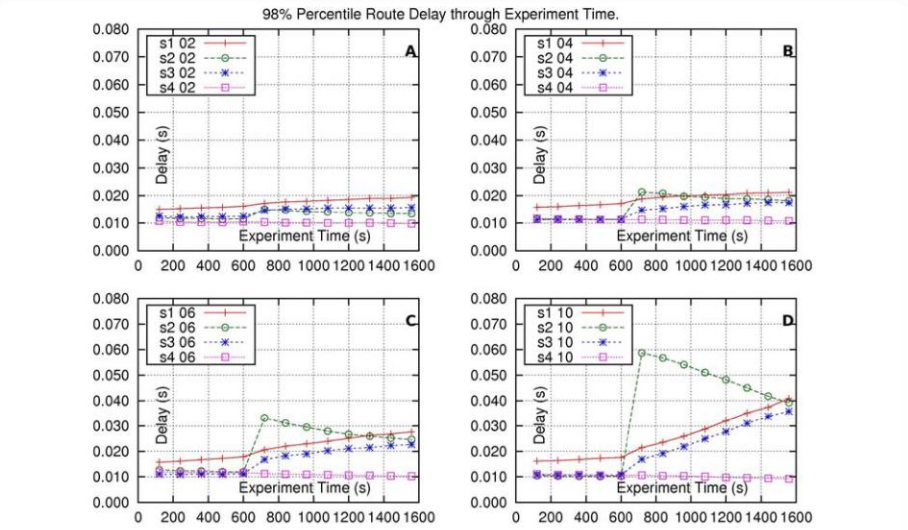


Figure 5. Delay-results of the experiments using the STP method at the intersection output-interfaces of the core-nodes. The meaning of the labels of this figure is similar to those in Figures 3 and 4.

are protected, to a void having a sudden loss of bandwidth and, consequently, a sudden increase in delay. From this time on, within the experiment-time, these delays increase slowly and steadily. The delay in route $s_4 - d_4$ is not affected, as this route does not intersect with route $s_2 - d_2$.

It is important to note that, although the STP method is acting at the intersection output-interface of each one of the three core-nodes, the method is effectively giving protection to route $s_1 - d_1$, only at the intersection output-interface of core-node c_0, because that is the only output-interface having a change in its average traffic (Section 5 shows results of other evaluations of the STP method where route $s_1 - d_1$ has more intersecting-routes increasing its traffic).

Figure 6, shows the weights of the three queues at the intersection output-interface of core-node c_0. Subfigures 6-A through 6-D show these weights for the cases corresponding to traffic-increments, in route $s_2 - d_2$, of 2, 4, 6 and 8 sources, respectively, at time 600(s). Note that the addition of the weights of the three queues is equal to 1, for every abscissa point of any one of the graphs. Notice also that in no case, a weight (every weight has a beginning value of 0.333) decreases its value in more than 25% at the ending of the experiment; in other words, no weight becomes as small as 0.333 x 0.75 = 0.25, at the ending of the experiment.

Before time 600(s), the tendency of the weight may vary, even though these results come from the average of several simulations; but from time 600(s) on, the tendencies are clear: the weight, corresponding to the queue of route $s_2 - d_2$, increases, and the weights, corresponding to the queues of the other two routes, decrease.

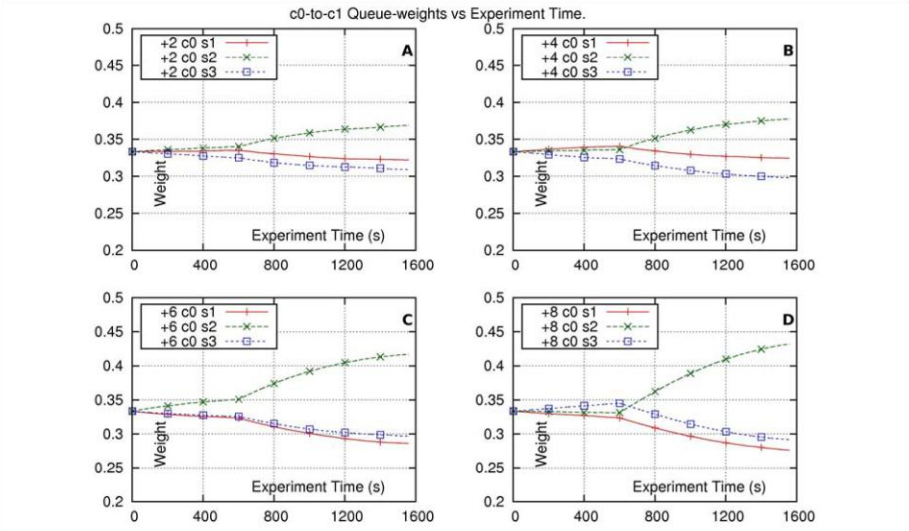


Figure 6. Weights of the queues, at the intersection output-interface of core-node c_0, in the experiments using the STP method at the intersection output-interfaces of the core-nodes. The label "**+2 c0 s1**" stands for "Weight of queue of route $s_1 - d_1$, at the intersection output-interface of core-node c_0, when route $s_2 - d_2$ increases its traffic in 2 sources, at time 600(*s*)". The other labels have similar meaning.

4.5 Solution with the method of coordinated multi-hop scheduling service

In the explanation of this solution the term "delay of flow" is preferred over the term "delay in route" (even though there is still one flow per route) due to the fact that this method regards very specifically to the flows on the network.

The method presented in this section is a special case of the Coordinated Multi-hop Scheduling Service (CMS): a work-conserving variant of the core-stateless jitter virtual clock (CJVC). This method was presented in [7] to protect flows of one class against the traffic increase of flows of other classes in a network. In the case where a flow is delayed as a consequence of the increase of traffic of another flow, which shares the same output-interface of a core-node, then the method helps the affected flow giving it a higher priority in the next core-node, where the flow goes through, allowing it to "catch up" (as [7] indicates). The purpose of this method is to try to maintain a low value in the overall delay of the flow, across its whole route.

In this paper, this method is applied on a network to protect flows in the same class, against the increase of traffic of flows in intersecting-routes. A general explanation of the form of operation implemented is as follows. At the ingress node, the method stores, in the header of each packet, a priority-index which is used by the output scheduler of the node, to serve the packet (the smaller the priority-index value the higher the priority). The value of this priority-index depends on the time the packet arrives at the ingress node (the smaller the time the smaller the index value). Then, every time the packet arrives at a downstream core-node, that node uses the priority-index value of the packet to recalculate a new priority-index value for that packet (basically

adding the maximum time that may be needed to retransmit the packet), which is again stored in the packet. Then, the packet waits to be served at a specific queue corresponding to the packet's flow, located at the output-interface of the node. The scheduler of the output-interface serves that packet which has the smallest priority-index value (highest priority), among those packets which wait in front of each queue of the interface.

Specifically, Equation 15 shows the calculation of the priority-index of packet *k*, of flow *i*, at node *j*, where l_i^k is the packet size, r_i is the reserved bandwidth for flow *I* (l_i^k / r_i is the maximum time to retransmit the packet), t_i^k is the time of arrival of the packet at the first hop, and ξ_i^k is a slack-variable assigned to the packet. The unit of the priority-index is given in seconds.

$$d_{i,j}^k = \begin{cases} max\{t_i^k, d_{i,1}^{k-1}\} + \frac{l_i^k}{r_i} & j = 1 \\ d_{i,j-1}^k + \frac{l_i^k}{r_i} + \xi_i^k & j > 1 \end{cases} \quad (15)$$

For this method, in this subsection, a set of experiments is presented where the flow which increases traffic is not able to harm the flows of intersecting-routes, whereas in subsection 4.6 another single experiment is presented where the flow which increases traffic is able to harm other flows.

For the first set of experiments r_i is equal to 2.4(*Mb/s*), meaning that the flow of every one of the three intersecting routes, at every core-node, has a reserved bandwidth of 8% of the capacity of the output link of the core node (of 30(*Mb/s*)); ξ_i^k is equal to 0.0003(*s*) for the flow which follows the long-sized route, $s_1 - d_1$; ξ_i^k is equal to 0.000375(*s*) for the flow which follows any one of the middle-sized routes, $s_2 - d_2$, $s_3 - d_3$, $s_4 - d_4$, $s_5 - d_5$; and ξ_i^k is equal to 0.0005(*s*) for the flow which follows any one of the small-sized routes, $s_6 - d_6$ and $s_7 - d_7$.

Subfigures 7-A through 7-D show the results of four different experiments, where the flow of route $s_2 - d_2$ increases its number of sources in 2, 4, 6 and 10, respectively, at time 600(*s*). As a result of these increments, it can be observed that, in terms of delay, the flow of route $s_2 - d_2$ has a sudden impact at 600(*s*); the flows of routes $s_1 - d_1$ and $s_3 - d_3$ are both barely affected; and the delay of the flow of route $s_4 - d_4$ is not affected. The reason for this "self-inflicted" delay-impact in the flow of route $s_2 - d_2$ is that the packet arrival-rate of this flow is bigger than the packet-arrival rate of the other flows. As each packet of a flow has a bigger priority-index value (representing less priority) compared to that of the previous packet, of the same flow, then, the priority-index values of the packets of the flow of route $s_2 - d_2$, queued at the intersection output-interface of core-node c_0, are bigger (representing less priority) than the priority-index values of the packets of the flows of routes $s_1 - d_1$ and $s_3 - d_3$, queued at the same interface. So the packets of these last two flows have advantage, when evaluated for service, with respect to the packets of the flow of route $s_2 - d_2$.

In this case, the flow packets of route $s_1 - d_1$ arrive at core-node c_1 having priority-index values not necessarily smaller or bigger than those of the packets of the flows of routes $s_4 - d_4$ and $s_5 - d_5$, so the method does not favor any one of these flows, at the output-interface of node c_1.

4.6 CMS-method solution where flow $s_2 - d_2$ has advantage over the other flows

This is the only solution which has an experiment with traffic conditions different from those of the experiments of the other solutions . The reason for this is to give the flow of route $s_2 - d_2$ the biggest advantage in this experiment. In this experiment, the results of a single simulation are observed.

The number of initial sources are 13, 13, 8, 14, 14, 0, 0, for the flows of routes $s_1 - d_1$ through $s_7 - d_7$,

respectively. The flow of route $s_2 - d_2$ increases its traffic, at time 600(s), in 6 flows. The reserved bandwidth for all flows is 6(Mb/s), except for the route flow of $s_2 - d_2$, which has 9(Mb/s).

In this experiment, the values for ξ_i^k are smaller to obtain smaller priority-index values. These index values are equal to 0.00012(s), for the flow of the long-sized route $s_1 - d_1$; and equal to 0.00015(s), for the flow of each one of the middle-sized routes $s_2 - d_2$, $s_3 - d_3$, $s_4 - d_4$ and $s_5 - d_5$; and equal to 0.0002(s), for the flow of each one of the small-sized routes $s_6 - d_6$ and $s_7 - d_7$.

CMS is very sensible to the amount of bandwidth reservation for flows. In this experiment, the flow of route $s_2 - d_2$ has more reserved-bandwidth than the bandwidth reserved for each one of the other flows (the flows of routes $s_1 - d_1$ and $s_3 - d_3$), at the output-interface of node c_0. Looking at Equation 15, the priority-index values of the packets, of the flow of route $s_2 - d_2$, have a smaller growing rate, from one packet to the next one, meaning that the tendency is to give smaller priority-index values (higher priority) to these packets, and thus, contributing to cause an important impact to the other two flows (the flows of routes $s_1 - d_1$ and $s_3 - d_3$), at the output interface of node c_0 when the flow of route $s_2 - d_2$ increases its traffic.

In this case, it can be seen how CMS protects the flow of route $s_1 - d_1$ against the increase of traffic of the flow of route $s_2 - d_2$, by giving the flow of route $s_1 - d_1$ a higher priority at the output-interface of node c_1, but, affecting, as a consequence, other flow, in an immediate form (the flow of route $s_5 - d_5$ is the one which is affected -as it is shown subsequently).

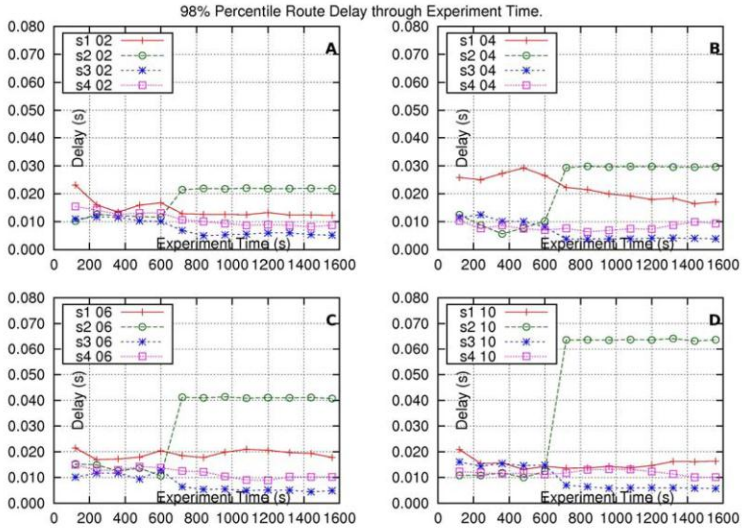


Figure 7. Delay-results of the CMS-method experiments. CMS being in its special case, the work-conserving variant of the CJVC method. This method is applied to a network with intersecting-routes, working with a single class. Label "**s1 02**" stands for "Delay of the flow of route $s_1 - d_1$ where, at time 600(s), the flow of route $s_2 - d_2$ increases its traffic in 2 sources". The other labels have similar meaning.

Subfigure 8-A shows the delay of the flow of each one of the routes $s_1 - d_1$, $s_2 - d_2$, $s_3 - d_3$ and $s_4 - d_4$. It can be seen that the flow of route $s_1 - d_1$, at time 600(s), suffer of an increase of delay, and the flow of the route $s_2 - d_2$ barely increases its delay as a consequence of its increase of traffic. The flow of route $s_3 - d_3$ is almost not affected, in terms of its delay, because it has less traffic, and so the priority-index values of its packets are smaller (higher priority). The flow of route $s_4 - d_4$ decreases its delay, and the reason is explored in the explanation of subfigure 8-C.

Subfigure 8-B shows the results for the delay of the flow of each one of the routes: $s_1 - d_1$, $s_2 - d_2$ and $s_3 - d_3$, at the intersection output-interface of core-node c_0.

Subfigure 8-C shows results for the delay of the flow of each route: $s_1 - d_1$, $s_4 - d_4$ and $s_5 - d_5$, at the intersection output-interface of core-node c_1. It can be seen that the flow of route $s_1 - d_1$ has almost no delay throughout all the time of the experiment. The delay of the flow of route $s_4 - d_4$ is roughly 43(ms), before time 600(ms), and it decreases to roughly 9.3(ms) after 600(ms). The delay of the flow of route $c_5 - d_5$ is the one affected as it increases its delay from around 5(ms) to around 43(ms), at time 600(ms).

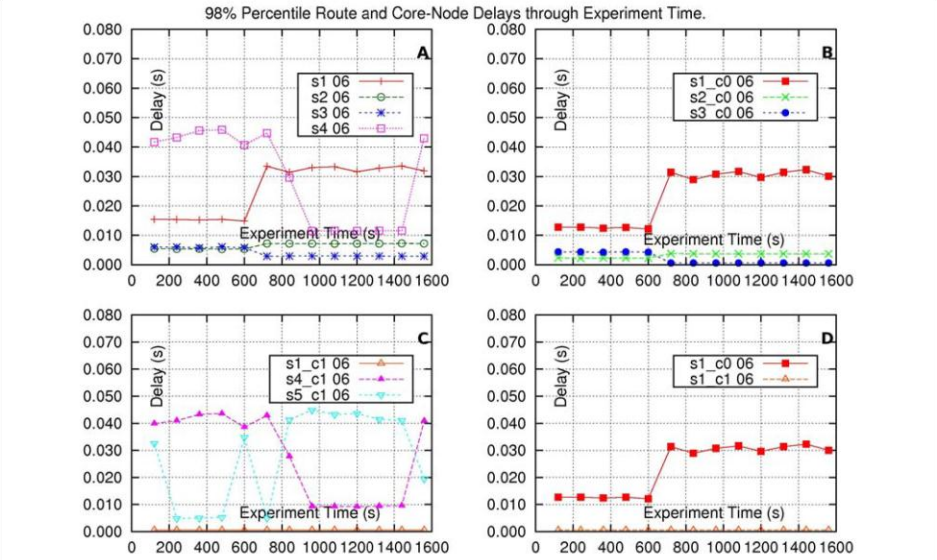


Figure 8. Delay-results of the CMS-method experiments with the special case of the work-conserving variant of the CJVC method. This method applied to a network with intersecting-routes, working within a single class. The flow of route $s_2 - d_2$ has advantage over the other flows. Label "**s1 06**" stands for "End-to-end delay (route-delay) of the flow of route $s_1 - d_1$", and Label "**s1_c0 06**" stands for "Delay, at the intersection output-interface of core-node c_0, of the flow of route $s_1 - d_1$". Both labels also indicate that at time 600(s), the flow of route $s_2 - d_2$ increases its traffic in 6 sources. The other labels have similar meaning.

This affectation is caused by: 1- the increase of traffic of the flow or route $s_2 - d_2$, 2- the protection of the flow of route $s_1 - d_1$[5], and 3- because even though the flows of routes $s_4 - d_4$ and $s_5 - d_5$ have the same number of sources, in this experiment the flow of route $s_4 - d_4$ has less traffic than that of the flow of route $s_5 - d_5$.

Subfigure *8-D* shows the delay-results for the flow of route $s_1 - d_1$, at the intersection output-interface of core-node c_0, and at the intersection output-interface of core-node c_1. The flow of route $s_1 - d_1$ has approximately 12.5(*ms*) of delay at the output-interface of core-node c_0 (to go to core-node c_1) before 600(*ms*), and after that time, the delay at that output-interface increases to approximately 30(*ms*). At the intersection output-interface of core-node c_1 (to go to core-node c_2) the delay of the flow of route $s_1 - d_1$ is at all times very close to 0.57(*ms*). So, the delay of the flow of route $s_1 - d_1$ is compensated by this method.

5. Scaling possibilities of the STP method

In order to explore the scaling possibilities of the STP method, six additional experiments were done, with mainly the same settings used for the evaluation of the solution with the STP method (subsection 4.4), but with a change in the value of parameter *P*.

The settings of these experiments are shown in Table 1.

[5] From subfigures 8-A and 8-D, it is important to notice that the result of adding the delay observed at the output-interface of node c_0, plus the delay observed at the output-interface of node c_1, for the packets of the flow of route $s_1 - d_1$, is not necessarily smaller than the delay observed for the packets of this flow to go end-to-end through this route. The reason for this is that the 2% most delayed packets to go across the intersection output-interface of core-node c_0 are not necessarily the same 2% most delayed packets to go across the intersection output-interface of core-node c_1.

In the experiments route $s_1 - d_1$ is subject to the increment of traffic, in 10 sources, at time 600(*s*), in one, two or three of its intersecting routes: $s_2 - d_2$, $s_4 - d_4$ and $s_6 - d_6$, as Table 1 indicates. In this Table, for every experiment, a row with an "x" indicates that there is an increment of traffic in the corresponding route, for example, in experiment #2, routes $s_2 - d_2$ and $s_4 - d_4$ increment their traffic, each one, in 10 sources, at time 600(*s*).

The first group of three experiments corresponds to parameter *P* = 0.25, and the results are given in Figures 9-A and 9-B. The second group of three experiments corresponds to parameter *P* = 0.10, and the results are given in Figures 9-C and 9-D.

Subfigure 9-A presents the delay in route $s_1 - d_1$ for experiments 1, 2 and 3 (corresponding to *P* = 0.25). It can be observed that, from time 600(*s*), the delay in this route grows, as more intersecting-routes increase their traffic.

Exper.	1	2	3	4	5	6
P	0.25			0.1		
$s_2 - d_2$	x	x	x	x	x	x
$s_4 - d_4$		x	x		x	x
$s_6 - d_6$			x			x
	Figs. 9-A and 9-B			Figs. 9-C and 9-D		

Table 1. Settings of the six experiments, to explore the scaling possibilities of the STP method.

Subfigure 9-B presents the delay in route $s_2 - d_2$ for experiments 1, 2 and 3 (corresponding to *P* = 0.25). It can be observed that, as routes $s_2 - d_2$, $s_4 - d_4$ and $s_6 - d_6$ do not intersect each other, the delay of route $s_2 - d_2$ is not affected if routes $s_4 - d_4$ and $s_6 - d_6$ increase their traffic.

Subfigures 9-C and 9-D are similar, respectively, to subfigures 9-A and 9-B, but the value of parameter *P* is 0.1 (10%). It can be observed that the delay-changes in routes $s_1 - d_1$ and $s_2 - d_2$ are smaller than those observed in subfigures 9-A and 9-B.

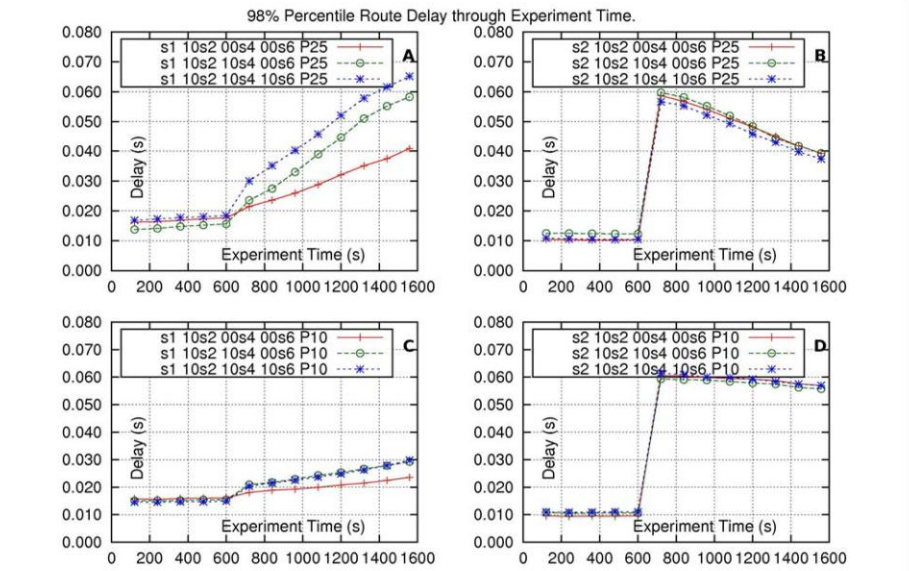


Figure 9. Delay-results of the experiments using the STP method at the intersection output-interfaces of the core-nodes. Label "**s1 10s2 10s4 00s6 P25**" stands for "Route-delay in route $s_1 - d_1$, where at time 600(s), each one of route $s_2 - d_2$, and route, $s_4 - d_4$ increase their traffic in 10 sources, and where P = 0.25". The other labels have similar meaning.

Wrapping up, when a route has tenths of intersecting-routes, and a worst-case occurs in which many of those intersecting-routes synchronize increasing their traffic within a small window of time then, the impact on the intersected route could be severe. To limit this possible impact, the selection of the values for parameters P and T should be either: 1- A smaller value for parameter P, 2- A bigger value for parameter T, 3- A combination of both possibilities.

The relation about the time increment, of the delay in a route, as a function of the values of parameters P and T, and as a function of the number of the intersecting-routes, is still a matter of study .

6. Discussion

6.1. A scenario where the STP method could be used

The Internet has a basic design as a Best-Effort network, composed of thousands of different networks [21]. In the "middle-mile" Internet, internetwork data communication is affected by peering-point congestion between, possibly competing, networks; performance limitations of routing protocols (BGP); and unreliable networks.

One approach to use the Internet, without changes, to address quality issues, is the Delivery

Network. A delivery network is a virtual network over the Internet, which needs no changes to the underlying Internet, offering enhanced reliability and performance of delivery. One main objective of a delivery network [22] can be to minimize the long-haul communications in the Internet, where middle-mile bottlenecks are found (like peering points between networks). This is done through the use of many distribution servers, the use of application-layer multicast services, and the use of multiple alternate-paths across the middle mile of the Internet.

A scenario for the operation of the STP method can take ideas of operation of the delivery network. One scenario could be a delivery network devoted to the delivery of time-sensitive traffic, like that of video-conference. This network could span several underlying specific networks of Internet Service Providers (ISPs), with interconnection points which did not represent bottlenecks. The ISPs would jointly cover big business areas.

The network would have fixed-routes to connect to the networks of the clients. The nodes of the ISPs would run the STP method to give the required service to the delivery network. The admission-processes would be per-route, which would be in charge of verifying the fulfillment of the delay-limit expected in the route. There would be no bandwidth reservation in the routes and the output-interfaces of the nodes should have big-enough buffers to limit the amount of lost packets because of possible overload-situations.

Regarding the transport protocol suggested, there would be two variants of the scenario. For the first variant, it is interesting to observe the tendencies of use of UDP and TCP in the Internet. By the year 2009, TCP sessions were still responsible for most packets and bytes in the Internet, but in terms of flows, UDP was the dominant transport protocol [23] (mainly from P2P applications using UDP for their overlay signaling traffic). There is a report about UDP traffic, in a campus connection to the Internet, which states that UDP traffic has grown steadily, in the years 2008 to 2011 [24], to reach a share of 22% of the total traffic.

The use of UDP as the transport protocol is suggested for the first variant of the scenario. The reason for this is that the network of the scenario is devoted to delay-sensitive traffic, and it operates with admission-control. The main use of the network would be for the strict delay-sensitive traffic (like that of video-conference), although less strict delay-sensitive traffic could also be admitted (like that of streaming traffic originated from stored or live content distribution). The nodes of the underlying ISPs might still use the spare bandwidth to send low-priority traffic, like TCP traffic, in other class, using the best effort service.

For the second variant of the scenario, the transport protocol selected would be a TCP-Friendly protocol, like a form of DCCP [25] (suitable for flows with timing constraints such as streaming media or flows from or multi-player on-line games or telephony and video-conference). In this variant, TCP traffic could also use the delivery network, in the same class, but subject to the admission-process.

Regarding transmission capacity, as the considered schedulers are work-conserving, the STP method should not impose a decrease of the transmission capacity in the network.

6.2 STP-method application considerations

The first consideration, to be in mind, is that, under situations of little traffic on a network, the STP method is not better than the case of using a single queue at every intersection output-interface of the network.

Another consideration is that a possible problem of the method can be the situation where, in an output-interface, the queue of a given route "A" has the biggest weight of them all. This route may decrease its traffic, allowing the other intersecting-routes, at that output-interface, to increase their traffic. As the weight decrement of a queue is slow, route "A" could suddenly increase its traffic, while its queue still has the biggest weight. This could cause important problems to the other routes.

6.3. The STP method and the tendencies of QoS

Some modern traffic engineering approaches try to achieve near-optimal traffic distribution in the networks, without the use of overlay networks (networks like those with fixed routes), which are considered to be administratively costly as they

may require the nodes to establish many logical connections. The optimal distribution functions used in these approaches do not observe the concept of QoS so, for these approaches, the adoption of a method like the STP method does not seem to be suitable. It seems that the issues of quality in these approaches should be addressed by application-layered solutions.

On the other side, other approach to QoS, called Quality-of-Service Routing [26, 27, 28], tries to identify, on a fixed topology, a feasible source-to-destination path that satisfies a set of QoS constraints, i.e., a path on which all links have sufficient resources to satisfy the diverse constraints, like delay, delay-jitter, cost, reliability, throughput. This approach does not include the concept of admission, but it rather tries to find a route for every "routing request". The routes on these networks have intersections, so, the STP method, applied to these networks, seems to be suitable.

7. Conclusions and future work

This paper addressed the problem of the intersection between routes, on networks admitting delay-sensitive flows, where each one of these flows is constrained to follow a specific route. The main contribution of this paper was to present an innovative method, called short term protection method (STP method), operating independently in every node of the network, oriented to be easily managed, with a prospect to be scalable to at least tenths of nodes, and oriented to attain a single extensive behavior in the network. The goal of the method is to help maintain the QoS in the routes of these networks, while these routes are subject to the increasing traffic in intersecting-routes.

In the results of the tests done, this method proved to be effective and with promising possibilities with regard to its scaling orientation. The STP method is compared to other method, oriented to protect flows against the increment of traffic in other flows, on a network. The benefits of the STP method are observed.

Future lines of research are: 1- Make the STP method become proactive (instead of reactive); 2- Study the way in which the STP-method's parameters affect its performance; 3 Integrate an additive increase/multiplicative decrease (AIMD) mechanism to the STP method, observing if the weight-loss of a route is due to the own decrease of traffic in the route, or to the increment of traffic in the intersecting-routes; 4- Study the STP method with more types of traffic.

Acknowledgements

The author appreciates the helpful discussions with Francisco García-Ugalde, José Incera-Diéguez and Fabián García-Nocetti. Also, the author appreciates the valuable comments of Francisco López-Fuentes and of the blind reviewers, with regard to the previous versions of this paper.

References

[1] V. Kumar et al, "Router Architectures for the Differentiated Services of Tomorrows Internet", IEEE Communications Magazine, Vol. 36, No. 5, pp. 152-164, 1998.

[2] S. Blake et al, "An Architecture for Differentiated Services", RFC 2475, IETF, 1998.

[3] K. Nichols et al, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, IETF, 1998.

[4] C. Cetinkaya and E. Knightly, "Egress Admission Control", in: 19th IEEE International Conference on Computer Communications, INFOCOM 2000, The Conference on Computer Communications, Tel Aviv, Israel, 2000, pp. 1471-1480.

[5] C. Cetinkaya et al. "Scalable Services via Egress Admission Control", IEEE Transactions on Multimedia: Special Issue on Multimedia over IP, Vol. 3, No. 1, pp. 69-81, 2001.

[6] T. Hui and C. Tham, "Adaptive Provisioning of Differentiated Services Networks Based on Reinforcement Learning", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 33, No. 4, pp. 492-501, 2003.

[7] C. Li and E. Knightly, "Schedulability Criterion and Performance Analysis of Coordinated Schedulers", IEEE/ACM Transactions on Networking, Vol. 13, No. 2, pp. 276-287, 2005.

[8] M. F. Horng et al, "An Adaptive Approach to Weighted Fair Queue with QoS Enhanced on IP Network", in: IEEE Region 10 International Conference on Electrical and Electronic Technology, (TENCON 2001), Singapore, Singapore, 2001, pp. 181-186.

[9] H. Wang et al, "Adaptive-Weighted Packet Scheduling for Premium Service", in: IEEE International Conference on Communications, ICC 2001, Helsinki, Finland, 2001, pp. 1846-1850.

[10] D. Hang et al, "TD2FQ: An Integrated Traffic Scheduling and Shaping Scheme for DiffServ Networks", in: IEEE Workshop on High Performance Switching and Routing", Dallas, Texas, USA, 2001, pp. 78-82.

[11] C. C. Li et al, "Proportional Delay Differentiation Service Based on Weighted Fair Queuing", in: IEEE Ninth International Conference on Computer Communications and Networks, ICCCN 2000, Las Vegas, Nevada, USA, 2000, pp. 418-423.

[12] S. Floyd and V. Jacobson V., "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp. 397-413, 1993.

[13] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", IEEE/ACM Transactions on Networking, Vol. 1, No. 3, pp. 346-357, 1993.

[14] ns-2 Network Simulator, Information Science Institute, 2011, Marina del Rey, California, USA, Available from: http://www.isi.edu/nsnam/ns/

[15] E. Altman and T. Jimenez, "NS Simulator for Beginners, Lecture Notes, 2003-2004", Univ. de Los Andes, Merida, Venezuela and ESSI, Sophia-Antipolis, France. Available from: http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/COURS-NS/n3.pdf.

[16] P. Pieda et al, "A Network Simulator Differentiated Services Implementation", Open IP, Nortel Networks. 2000. Available from: http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/DOC/DSnortel.pdf.

[17] A. Mrkaic, "Porting a WFQ Scheduler into ns-2's Diffserv Environment", Student Thesis, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, 2001.

[18] J. L. Devore, "Probability and Statistics for Engineering and Sciences". The 6th Edition. Brooks Cole, ISBN 0534-39933-9. ISBN 13: 978-0534-39933-7. 2003.

[19] A. Matrawy et al, "MPEG4 Traffic Modeling Using the Transform Expand Sample Methodology", in: IEEE 4th International Workshop on Networked Appliances, Gaithersburg, Maryland, USA, 2002, pp. 249-256. Also available, 2012, from: "ns-2 contributed-code", http://nsnam.isi.edu/nsnam/index.php/Contributed_Code #Topology_and_Traffic_Generation

[20] G. Salmon et al, "NetFPGA-Based Precise Traffic Generation", in: NetFPGA Developers Workshop'09, Standford University, California, USA, 2009.

[21] CIDR REPORT, 2012. Available from: http://www.cidr-report.org/as2.0/

[22] E. Nygren et al, "The Akamai Network: A Platform for High-Performance Internet Applications", ACM SIGOPS Operating Systems Review, Vol. 44(3), pp. 2-19, 2010.

[23] Analyzing UDP usage in Internet traffic. The Cooperative Association for Internet Data Analysis. 2012. Available from: http://www.caida.org/research/traffic-analysis/tcpudpratio/

[24] C. Lee et al, "Unmasking the growing UDP traffic in a Campus Network", in: Passive and Active Measurement (PAM) Conference, Vienna, Austria, 2012.

[25] E. Kohler and S. Floyd, "Datagram Congestion Control Protocol (DCCP) Overview", ICSI Center for Internet Research, Berkeley, CA, USA, 2003.

[26] J. Huang et al, "An Effective Approximation Scheme for Multiconstrained Qualiy-of-Service Routing", in: IEEE Global Telecommunications Conference (GLOBECOM 2010), pp. 1-6, 6-10, 2010.

[27] G. Xue et al, "Finding a Path Subject to Many Additive QoS Constraints", IEEE/ACM Transactions on Networking, Vol. 15, No. 1, pp. 201-211, 2007.

[28] Y. Xiao et al. "Computing a Most Probable Delay Constrained Path: NP-Hardness and Approximation Schemes", IEEE Transactions on Computers, Vol. 61, No. 5, pp. 738-744, 2012.

Appendix A

Theorem 1. Proof of Equation 4.

Consider that queue i obtains T/τ consecutive results indicating that it has to lose weight, where, as already stated, τ and T are a small and a big intervals of time, respectively, such that T/τ is a big integer.

Equations 2 and 3 are repeated here, for convenience, as Equations. A-1 and A-2.

$$\Delta_i(t_0 + r\tau) = \phi_i(t_0 + (r+1)\tau) - \phi_i(t_0 + r\tau) \qquad \text{(A-1)}$$

$$\Delta_i(t_0 + r\tau) \leftarrow f\frac{\tau}{T}\phi_i(t_0 + r\tau) \qquad \text{(A-2)}$$

Mixing these equations the next one is obtained:

$$\phi_i(t_0 + (r+1)\tau) = \phi_i(t_0 + r\tau)\left(1 + \frac{f}{T/\tau}\right) \quad \text{(A-3)}$$

This was the result of the weight of queue *i* after the first interval, as a function of its initial weight. Similarly, the weight for the next interval would be:

$$\phi_i(t_0 + (r+2)\tau) = \phi_i(t_0 + (r+1)\tau)\left(1 + \frac{f}{T/\tau}\right) \quad \text{(A-4)}$$

Mixing these last two equations it is obtained that:

$$\phi_i(t_0 + (r+2)\tau) = \phi_i(t_0 + r\tau)\left(1 + \frac{f}{T/\tau}\right)^2 \quad \text{(A-5)}$$

So, for the T/τ consecutive increments, the following equation is obtained:

$$\phi_i(t_0 + (r+T/\tau)\tau) = \phi_i(t_0 + r\tau)\left(1 + \frac{f}{T/\tau}\right)^{T/\tau} \quad \text{(A-6)}$$

If T/τ is big, then the factor with the exponent can be approximated to:

$$\left(1 + \frac{f}{T/\tau}\right)^{T/\tau} \approx e^f \quad \text{(A-7)}$$

And with this, it is obtained that:

$$\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) \approx \phi_i(t_0 + r\tau)\, e^f \quad \text{(A-8)}$$

And,

$$\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) - \phi_i(t_0 + r\tau) \approx \phi_i(t_0 + r\tau)\left(e^f - 1\right) \quad \text{(A-9)}$$

And with this the theorem is demonstrated.

Apéndice B. IntServ y DiffServ. Comportamientos y Servicios

B.1 IntServ

B.1.1 Servicio Garantizado

El Servicio Garantizado [46] provee a los flujos un servicio de transporte por una ruta en una red, con una cota máximas de retrasos por encolamiento de extremo a extremo, y garantiza que sus paquetes no sean descartados por sobrecupos en las colas de los nodos en el curso de la ruta, siempre que los flujos se mantengan dentro de parámetros de tráfico especificados.

El servicio se provee por cada nodo a lo largo de la ruta. En la interfaz de salida del nodo se debe asegurar que el retraso por encolamiento local no pase de un máximo. Este servicio se aproxima cercanamente a aquél que sería provisto por un hilo dedicado de ancho de banda R entre la fuente y el receptor. Este servicio se invoca bajo dos especificaciones de flujo: *RSpec* y *TSpec*. RSpec provee la especificación deseada de servicio (las garantías buscadas), que asegura a cada flujo una parte *R* del ancho de banda de la interfaz (del enlace), y una parte *B* del buffer de la interfaz. TSpec define características de tráfico que implican el comportamiento que debe seguir el tráfico. Esto incluye, de alguna forma, qué tanto puede variar su tasa con relación a su tasa promedio (el comportamiento de sus ráfagas de tráfico). La tasa de RSpec pude ser mayor que la tasa de TSpec. Si un flujo no mantiene sus parámetros de tráfico el servicio puede someterlo a cumplimiento (bajo el proceso llamado vigilancia–“Policing"). El mecanismo de comunicación con los nodos para establecer este servicio puede ser manual o puede utilizar protocolos como SNMP (Simple Network Management Protocol) o RSVP (Resource ReSerVation Protocol) [87].

B.1.2 Servicio de Carga Controlada

Con el Servicio de Carga Controlada [47], el comportamiento de extremo a extremo provisto por una serie de elementos de una red que proporcionan este servicio estrechamente se aproxima a la conducta visible del servicio que se recibe por Best Effort en condiciones “sin carga”, por la misma serie de elementos de esa red.

Este servicio no ofrece garantías de máximo retraso sino que indica que habría un retraso "un poco mayor" con relación a lo que se tendría en el caso de haber una ráfaga manejada a la tasa solicitada por el flujo. Los flujos deben proveer a los elementos de red una estimación de su tráfico (TSpec). Cada elemento de red debe asegurarse que cuenta con recursos (como ancho de banda y espacio de Buffer y capacidad computacional de despacho) para dar servicio al flujo, antes de admitirlo, y puede emplear enfoques estadísticos para calcular sus capacidades (por ejemplo considerar las características de comportamiento de los flujos manejados con base en el comportamiento pasado para sobre asignar sus recursos en cierta medida reduciendo en un margen su "nivel" de servicio provisto).

B.2 Diffserv

La formación de un servicio en un dominio Diffserv se hace a partir del uso de un PHB, que significa: "Comportamiento por Salto" (del inglés Per-Hop Behavior), y que es el comportamiento que cada nodo tiene con relación a un flujo que pertenezca a un Agregado de Comportamiento (a una Clase) determinado. El servicio a un flujo incluye la aplicación de vigilancia y acondicionamiento a la entrada del dominio DS, y posibles acondicionamientos al interior del dominio. En cada nodo puede operar un PHB por cada Agregado de Comportamiento.

B.2.1 Grupo de PHBs Assured Forwarding (AF)

El servicio AF [88] intenta entregar los paquetes del flujo, o de los flujos a los que da servicio, a través del dominio, con alta probabilidad de que no se desechen los paquetes debido a saturación en los nodos del dominio. Esto siempre que los flujos cumplan con su perfil de tráfico convenido. El servicio también trata de que, para los flujos que no cumplan con su perfil, sea posible la entrega de sus paquetes, aunque con una menor probabilidad.

Sobre si debe o no haber reordenamiento de paquetes de un mismo flujo, aun si éstos perteneciesen a un tráfico fuera de perfil, no está indicado.

El PHB referente a AF no es uno solo, sino es un conjunto de PHBs que se definen en conjunto. El Grupo de PHBs AF maneja *N* PHB, o clases, independientes, y *M* diferentes niveles de precedencia de desecho ("Drop Precedence") por clase, donde la precedencia indica la propensión a ser desechado en una cola debido a situaciones de proximidad a saturación en la misma.

Los paquetes de flujos que vayan a recibir servicios provistos con el grupo de PHB AF se marcan, ya sea por el cliente, o por el dominio DS. El marcado indica la clase AF y el nivel de precedencia de desechado, conceptos que se explican a continuación.

Cada paquete IP de un agregado servicio, bajo un grupo de PHBs AF, se marca (en su campo llamado DiffServ Codepoint) con una marca equivalente a AF_{ij}, donde "*i*" corresponde a la clase y "*j*" a la precedencia. Actualmente se han definido 4 clases, con tres niveles de precedencia para cada clase (aunque para una utilización particular se podrían utilizar más clases y niveles de precedencia).

Un número de precedencia más bajo equivale a mayor precedencia (menor probabilidad de desechado), es decir, AF_{x1} debe tener la probabilidad más baja de desecho para la clase *x*. No siempre, por haber tres niveles de precedencia, se manejan tres probabilidades de desecho, a veces AF_{x2} y AF_{x3} se manejan con la misma probabilidad de desecho, y AF_{x1} se maneja con otra probabilidad de desecho (menor).

Un nodo DS debe implantar las cuatro clases AF. Los paquetes en una clase AF no se agregan con paquetes de otra clase. Para la operación de un grupo de PHB AF, un nodo DiffServ debe considerar un mínimo de recursos exclusivos (espacio de memoria –Buffer— y ancho de banda) para la operación de cada clase AF, en largo y corto plazo. Desde luego una clase puede recibir, en periodos de tiempo limitados, más recursos, si existen recursos ociosos de otras clases o de otros PHB.

Diffserv asigna, a cada clase AF en cada nodo, una cantidad de recursos (espacio de memoria –Buffer— y ancho de banda), de tal forma que se desempeñe la tasa configurada de servicio, tanto para escalas de tiempo pequeñas como grandes.

En operación normal otros PHB no deben invadir (Preempt) los recursos del grupo de PHB AF. En casos emergentes, por ejemplo el tráfico de control de red, sí se puede invadir a los recursos del grupo de PHB AF.

El comportamiento de encolado y desecho en una clase AF debe minimizar la congestión de largo plazo, permitiendo la congestión de corto plazo resultante de ráfagas. Así, el algoritmo para implantar el Grupo de PHBs AF en los nodos debe permitir que flujos con diferentes formas de ráfagas, pero con idénticas tasas de largo plazo, se traten de la misma manera, lo que se logra con la introducción de un elemento aleatorio en el desechado de paquetes. Esto se hace con algún algoritmo de administración de colas (como RED –Random Early Discard), aunque no se recomienda un algoritmo específico.

Para un “nivel de congestión suavizado” (Smoothed Congestion Level) la tasa de desecho de paquetes de un flujo particular, dentro de un nivel de precedencia determinado, será proporcional al porcentaje que ese flujo tenga del total de tráfico pasando por ese nivel de precedencia.

En los límites de los dominios DS, los procesos de acondicionamiento pueden implicar que a los paquetes de tráfico agregado AF se les incremente o disminuya su precedencia de desecho, o se les reasigne a otras clases AF. Se debe considerar que cierto tipo de reasignaciones pueden causar el desorden los paquetes de los flujos, lo que no es admisible para algunas aplicaciones.

Servicio Olímpico.

Con relación a los servicios que se puedan construir con el Grupo de PHBs AF, se puede implantar el *Servicio Olímpico*, donde hay tres clases de servicio: oro, plata y bronce, que pueden corresponder a diferentes clases AF, y dentro de cada clase también se pueden implantar niveles de precedencia. Por ejemplo, los paquetes en la clase oro tendrían mayor probabilidad de ser entregados a tiempo, con relación a los paquetes en la clase plata, y bronce.

B.2.2 El PHB EF

El PHB EF [62] se puede utilizar para producir servicios de punta-a-punta, a lo largo de dominios DiffServ que tengan características de baja pérdida, baja latencia, bajo Jitter, y ancho de banda asegurado. Servicios así darían la impresión, ante los puntos terminales, como si se tuviese una conexión hecha con una línea privada, por eso al tipo de servicio lo llaman "servicio de línea privada virtual". Para proveer las características estipuladas en el PHB EF es necesario que los paquetes del agregado, en el curso que lleven, solo encuentren nodos en donde las colas de espera en las que se alojen sean de nulo tamaño (no haya congestión), o sean colas de tamaño "muy pequeño". Las colas se forman, en un nodo, cuando la tasa de corto plazo de llegada de tráfico excede la tasa de salida (que es la tasa efectiva a la que se está saliendo).

El PHB EF se define como un tratamiento de envío, en un nodo y para un agregado de comportamiento particular, en donde la tasa de salida de los paquetes del agregado en cualquier nodo debe igualar o exceder una tasa configurable (una tasa objetivo). El agregado debe contar con un ancho de banda, en la interfaz de salida de un nodo, independiente de lo que tenga asignado cualquier otro agregado en dicha interfaz. Este ancho de banda debe promediar un valor de al menos la tasa configurable al medirse en cualquier intervalo de tiempo de duración igual o mayor que el tiempo que toma enviar un paquete de tamaño MTU (unidad máxima de transmisión) a la tasa configurable.

Se deben acondicionar los agregados en un dominio para que su tasa de llegada a cualquier nodo del dominio sea siempre menor que la mínima tasa de salida configurada en el nodo.

El PHB EF se puede producir por medio de más de una estrategia de atención de colas (tipo de despachador), más acondicionamiento. Se dan algunos ejemplos a continuación:

• Despachador "Priority Queue (Cola con Prioridad), siempre que no haya una cola de mayor prioridad que pueda aniquilar o invadir (Preempt) a la cola EF, por más de una paquete a la vez, operando a la tasa configurada (esto se puede lograr teniendo un limitador). La implantación debe incluir previsiones para limitar el daño que el tráfico EF pueda ocasionar en otro tráfico, de tal forma que el tráfico EF que exceda un límite debe descartarse.

- Despachador "Round Robin con Pesos" (Weighted Round Robin Scheduler), que sirva a varias colas simples, entre ellas, a una cola simple para EF, en donde la parte de ancho de banda de salida para EF corresponda a la tasa configurada para EF.

- Despachador "Encolado Basado en Clases" (CBQ –Class Based Queueing), donde la cola EF tuviese una prioridad correspondiente a la tasa configurada para EF.

Es importante darse cuenta que en todas las estrategias de atención de colas anteriores, el ancho de banda de EF se puede utilizar por otras clases mientras no haya paquetes que enviar del agregado EF.

El ancho de banda "reservado" para el agregado EF no debe rebasar la suma de los anchos de banda requeridos por los flujos miembros del mismo. Con esto se logra que el ancho de banda del nodo no tenga que ser enorme para poder tener en operación una cola EF.

Algunos ejemplos de servicios que se pueden construir con el PHB EF son:

Servicio Premium

El Servicio Premium [55] cercanamente garantiza que un paquete de un flujo con este servicio no tenga retraso por congestión en los nodos del dominio, siempre que la tasa del flujo no sea mayor que un valor máximo permitido. Con este servicio sí se pueden perder paquetes en el proceso de acondicionamiento cuando el flujo no cumpla con su perfil.

El costo económico del servicio Premium sería alto pues con este servicio se tiene "una especie" de reservación, aunque el despachador es conservador de energía.

El servicio Premium es adecuado para aplicaciones donde se requieren tasas constantes de paquetes (poco Jitter), aplicaciones tales como transmisión de señales de video o de audio en tiempo real. A estas aplicaciones no les debe afectar la pérdida de uno que otro paquete, y sin embargo las retransmisiones de paquetes debido a pérdidas sí les podrían ser perjudiciales.

El acondicionamiento de un flujo que reciba Servicio Premium no debe permitir que el flujo rebase la tasa máxima promedio definida para el mismo, aunque para hacerlo tenga que tirar paquetes del flujo.

Los agregados de flujo Premium no deben aniquilar [Preempt] la capacidad para otros agregados de comportamiento en un nodo.

B.3 Formación de Diversos Servicios con base en el mismo PHB en DiffServ

Un solo PHB se puede utilizar para formar diversos servicios. Esto es tan fácil como considerar que un servicio puede ofrecer un “alto” aseguramiento de que los paquetes del flujo llegarán a su destino, si se cumple que el flujo no pase de una tasa promedio “X”. Otro servicio puede ofrecer lo mismo que el primero, pero con relación a una tasa promedio “Y”. Estos dos servicios se pueden lograr con el mismo PHB EF, pero haciendo acondicionamientos diferentes a los flujos.

Las características de los flujos deben mantenerse, al integrarse a algún agregado de comportamiento. Por ejemplo, muchos flujos Premium, ya en el interior del dominio, que tengan tasas diferentes entre sí, pueden viajar en un solo agregado de comportamiento, pero las tasas de cada flujo deben mantenerse, y no debe haber Jitter en flujo alguno. Esto se puede lograr si los nodos prácticamente no hacen esperar a los paquetes que los cursan.

Apéndice C. Operación de ns-2

C.1 Generalidades

El simulador ns-2 es un simulador escrito en C++ y en el lenguaje traductor ampliado de Tcl, con funcionalidad orientada a objetos, OTcl. Al escribir “ns” en la terminal de la computadora (al correr ns), se crean muchas clases de C++, que representan piezas importantes para la operación de ns, como lo son clases para crear paquetes, nodos, enlaces, colas. Es decir, y como ejemplo, cada paquete que se maneja, en el tiempo de simulación, es un objeto que se va relacionando con otros diversos objetos como nodos y colas, en su curso por la red que se ha generado (de hecho en el simulador ns-2 las colas está asociadas con los enlaces y no con los nodos).

Asimismo se genera una “instancia” de Otcl, escrita en C++, que se presenta como una interfaz interactiva al usuario, es decir, se instala OTcl como ambiente de interfaz para comunicación con el usuario. Se crean clases y rutinas en OTcl para comunicación con los objetos y rutinas de C++.

Normalmente, el usuario al escribir “ns” acompaña a dicha instrucción con el nombre de un archivo de código en OTcl. Es decir, el archivo es un “Script” en OTcl. Este Script contiene instrucciones para la descripción de redes, según las reglas de ns. Con estas instrucciones se describe la configuración de la red y se los parámetros relacionados con los elementos de la misma. Este Script lo lee e interpreta la instancia interactiva de OTcl.

Las instrucciones, en OTcl, pueden hacer que se generen objetos en el entorno de C++, a partir de las clases previamente establecidas en C++. Asimismo, se pueden generar objetos “compuestos” que se conforman meramente de objetos C++ interconectados (con apuntadores), por ejemplo un nodo.

La última instrucción del Script es la instrucción “run” (correr), que inicia la ejecución de un lazo “while”. Este lazo sirve para ir repasando los eventos puestos en una lista para ser leída por el Scheduler del simulador, con el fin de atenderlos. Los primeros eventos puestos en el Scheduler emanan de las mismas líneas de código del Script en OTcl, que son las primeras instrucciones de ejecución de alguna acción, para el simulador. Estos eventos se ponen en la lista mientras van ejecutando las líneas de código del Script, y antes de ejecutar la instrucción final: “run”.

Este Scheduler es un objeto único en el simulador, que lee los elementos llamados eventos, que están puestos, y ordenados por tiempo ascendente de inicio de ejecución, en la lista del Scheduler. La lectura es en orden ascendente de tiempo. El orden se tiene según el tiempo “real” marcado en cada elemento. Cada uno de estos eventos es un objeto también, y contiene algunos parámetros, como el tiempo de inicio de atención, un identificador único, una

referencia al siguiente y al anterior evento, y una referencia para un objeto tipo “Handler”, con el que se puede saber qué función se debe ejecutar, para la realización del evento.

Cada vez que se atiende un evento del Scheduler del simulador, se ejecuta una rutina apuntada por el evento. Esta rutina se llama mediante otra rutina llamada “handle”, la cual se ejecuta hasta su terminación. Una vez concluida la atención de un evento se regresa el control al lazo “while” para que se busque, con el Scheduler, el siguiente evento de la lista. Al atenderse un evento, el tiempo actual del Scheduler toma el valor del tiempo marcado para ejecución de dicho evento (así va avanzando el tiempo del Scheduler).

Las rutinas ejecutadas por un evento pueden poner uno o más eventos adicionales en el Scheduler, mediante la rutina Schedule, la cual toma como parámetros: 1- un valor de “delay” (siempre positivo), que es el tiempo que se agregará al tiempo actual del Scheduler, para ponerlo como el tiempo de ejecución del evento; 2- una referencia a un objeto (tipo Handler) que contiene siempre una rutina llamada “Handle” (manejador), la cual, a su vez, contiene referencia a la rutina específica que se corre al ejecutarse el evento; 3- una referencia a un nuevo objeto tipo Event, que es el evento en sí.

Cada vez que se pone (que se inserta) un evento en la lista del Scheduler, este evento se ordena, mediante un procedimiento de funciones Hash, en la lista, de tal forma de mantener los eventos ordenados según el tiempo de ejecución marcado en los eventos.

Con la instrucción “AT”, en el Script, se pueden poner en el Scheduler eventos específicos. Es necesario poner al menos un evento inicial en el Scheduler. Aquí se supone que dicho evento es la generación del primer paquete que va a generar la fuente. Este primer evento generará uno o más eventos subsecuentes en el Scheduler.

La última instrucción “run” del Script es la instrucción con la que se empieza a ejecutar el lazo “while” que va a ir repasando los eventos en el Scheduler.

C.2 Ejemplo de Operación

Considere un ejemplo muy simple en donde hay una sola fuente de paquetes (fuente), un nodo, y un receptor, o destino, de paquetes (Sink). La fuente genera paquetes y no tiene que manejar colas.

La cola de los nodos se maneja en los enlaces. En este caso, la cola del nodo para la salida al destino se maneja en el enlace #2.

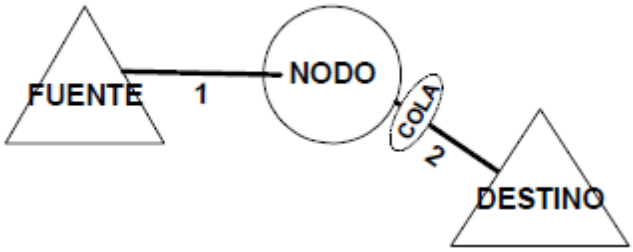


Figura 40**.** Una fuente, un nodo, un destino y dos enlaces.

Considere que se ejecuta un primer evento, que es la generación y despacho de un paquete por parte de la fuente. Para esto, se ejecutan los siguientes pasos:

1- Se genera un objeto tipo paquete (tipo Packet).

2- Se coloca en el Scheduler un evento de recepción de paquete[92] para hacer el encolamiento del paquete en la cola del enlace #2. Este evento está marcado para ejecutarse después de que haya pasado el tiempo de transmisión del paquete y el tiempo de propagación, por el enlace #1.

Nota. Si el paquete proviniese de un nodo que se ubicase “corriente arriba” (o sea que fuese anterior en el curso del paquete), en lugar de provenir de la fuente indicada en la Figura 40, entonces el evento de la recepción del paquete en la cola del enlace #2 implicaría un inmediato desbloqueo de la cola de salida del nodo “corriente arriba”. El desbloqueo querría decir que esa cola “corriente arriba” ya no estaría enviando el presente paquete, y por tanto quedaría libre para enviar otro paquete. Entonces, en este caso, sería el evento de desbloqueo el primero en ponerse en el Scheduler, y luego se pondría el evento de recepción de paquete en la cola del enlace #2[93].

3- Se coloca en el Scheduler el evento de generación y despacho del siguiente paquete que ha de emitir la fuente. El tiempo de ejecución de este evento depende de los algoritmos de generación de paquetes propios de la fuente y es un tiempo que debe ser mayor al tiempo en que ha salido de la fuente el paquete anteriormente enviado por la fuente.

[92] Las rutinas asociadas son “send” y “recv” de la clase “LinkDelay” que se define en el archivo “delay.cc” ([74] Cap. 8). El evento puesto en el Scheduler debe ser una rutina “recv” tipo “Queue”, que contiene inicialmente la rutina “enque”.

[93] Esto se hace mediante una rutina “recv” de LinkDelay, que pone en el Scheduler un evento de recepción del paquete.

Los siguientes eventos pueden ser cualquiera de los eventos recientemente generados.

El evento de recepción de paquete en la cola del enlace #2 implica un encolamiento del paquete en la cola del enlace #2 (realizado con la rutina “enque” y otras derivadas que “anotan” la referencia del objeto del paquete en la cola).

Si la cola del enlace #2 se encontrase bloqueada al momento del evento de encolamiento del paquete en esta cola, entonces este evento no generaría otro evento[94]. Si esta cola no estuviese bloqueada al momento del encolamiento se generaría un evento de salida de paquete y otro un evento de recepción de paquete en el destino (en un caso más general dicho destino podría ser la cola del próximo nodo corriente abajo). Este último evento estaría marcado para ejecutarse después de que hubiese pasado el tiempo de transmisión del paquete por el enlace #2 y el tiempo de propagación por dicho enlace. Finalmente se bloquea la cola del enlace #2 para indicar que se está en estado de transmisión de este último paquete.

C.3 Notas sobre las Colas

Los procedimientos anteriores ven las colas como objetos simples, sin embargo, dichas colas pueden estar compuestas. Por ejemplo, se puede tener un conjunto de colas DS (cola tipo dsREDQueue para ambientes DS) que en realidad es un conjunto de colas red (tipo redQueue) que estos procedimientos ven como una cola. Entonces, cuando uno de estos procedimientos pone en cola o quita de cola, un paquete, “no sabe” lo que está pasando en los procesos que manejan las colas. Por ejemplo, si la cola es tipo DS (realmente un conjunto de colas), uno de los procedimientos para manejo de esta cola “compuesta” es la selección de una de sus colas componentes, para sacar un paquete. Este procedimiento operaría según el tipo de despachador de colas que se utilice.

Toda cola es un objeto de alguna clase. La clase dsREDQueue (que es un conjunto de colas simples DS) tiene como clase superior jerárquica a la clase Queue que agrega varias rutinas, pocas pero importantes. Las colas simples pueden ser de varios tipos[95].

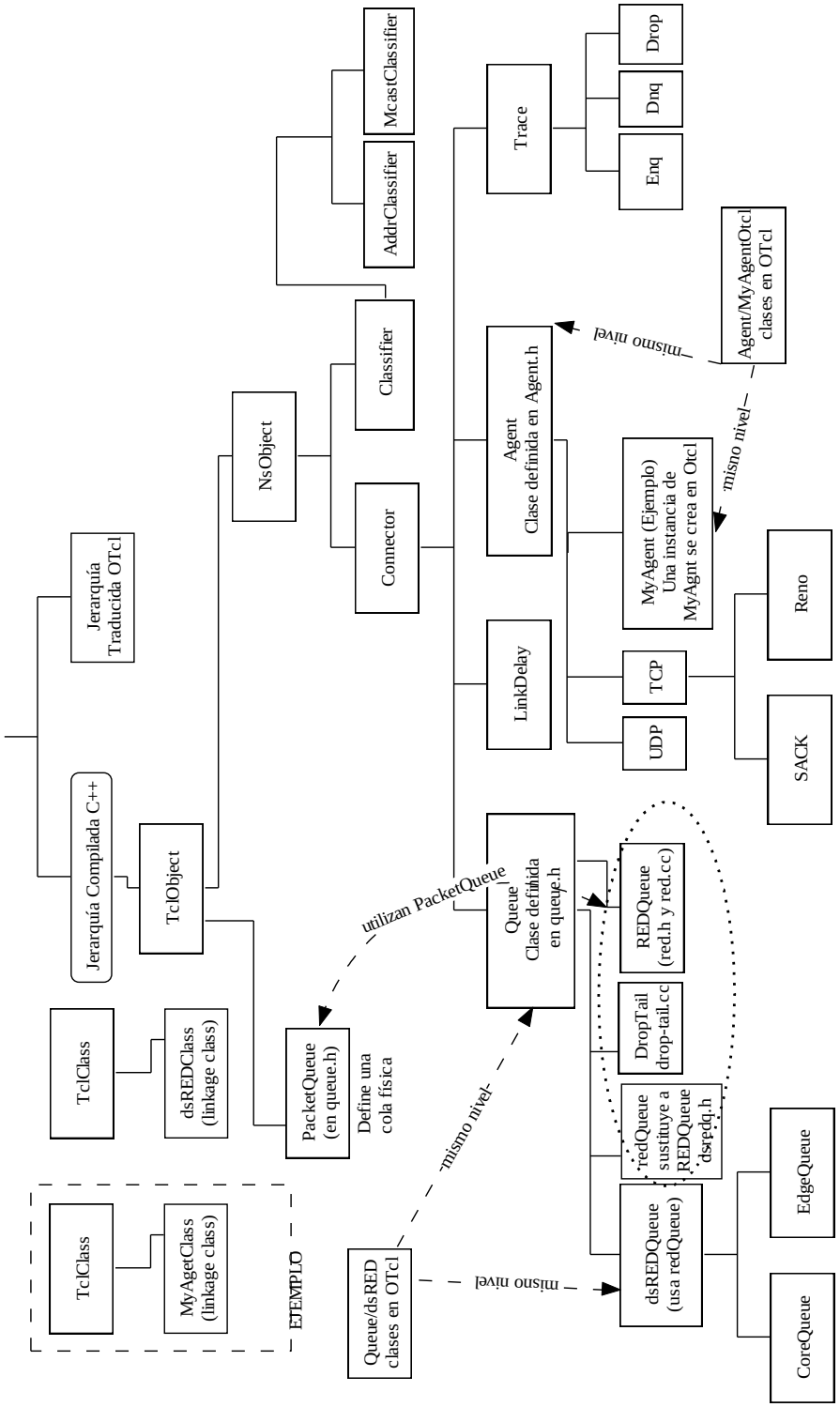
---

[94] En este caso debe haber en el Scheduler un evento que indica cuándo se acabaría de enviar el paquete. Al llegar a dicho evento en el Scheduler se habrá de correr qh_.handle(event *) de clase Queue, que corre, a su vez: *this.resume, que a su vez corre Packet *p = deque( ) de clase Queue. Si de esta última corrida se recibe un objeto tipo paquete no nulo, entonces se corre recv(p, &qh_) del objeto (cola) que está corriente abajo. Este recv corre primeramente enque(p), que es una rutina virtual que es sustituida por la rutina correspondiente al objeto de la clase derivada (como podría ser dsREDQueue).

[95] Toda cola simple, ya sea DropTail, redQueue o REDQueue, tiene como base a una cola física FIFO (tipo PacketQueue). Esta cola física es un objeto que contiene paquetes ordenados (cada paquete también es un objeto). Cada paquete apunta al siguiente formándose una estructura.

C.4 Estructura de las Clases en la Programación del Simulador ns-2

En el siguiente diagrama se puede observar una jerarquía y clasificación de las “clases”, dada la programación orientada a objetos utilizada para construir el Simulador ns.



C.5 Configuraciones Adicionales

Notas Sobre la Configuración del simulador ns-2 en Linux.

INICIALMENTE, al hacer modificaciones en los archivos en el directorio /usr/local/ns-allinone-2.27/ns-2.27/diffserv, desde el directorio /usr/local/ns-allinone-2.2.7/ns-2.27, se estaba corriendo la rutina ./configure.

Lo anterior no funcionaba bien así que nuevamente se instaló todo el simulador ns-2 corriendo la instrucción ./install nuevamente, estando en el directorio /usr/local/ns-allinone-2.27, y todo se instaló correctamente nuevamente. Al final de la instalación se tuvieron las siguientes notas (las anotaciones para la versión 2.33 son muy similares).

```
Please compile your gt-itm & sgb2ns separately.
Ns-allinone package has been installed successfully.
Here are the installation places:
tcl8.4.5:       /usr/local/ns-allinone-2.27/{bin,include,lib}
tk8.4.5:               /usr/local/ns-allinone-2.27/{bin,include,lib}
otcl:           /usr/local/ns-allinone-2.27/otcl-1.8
tclcl:          /usr/local/ns-allinone-2.27/tclcl-1.15
ns:             /usr/local/ns-allinone-2.27/ns-2.27/ns
nam:    /usr/local/ns-allinone-2.27/nam-1.10/nam
xgraph: /usr/local/ns-allinone-2.27/xgraph-12.1
Please  put  /usr/local/ns-allinone-2.27/bin:/usr/local/ns-allinone-
2.27/tcl8.4.5/unix:/usr/local/ns-allinone-2.27/tk8.4.5/unix     into
your   PATH   environment;   so   that   you'll   be   able   to   run
itm/tclsh/wish/xgraph.
 IMPORTANT NOTICES:
 (1)  You  MUST  put  /usr/local/ns-allinone-2.27/otcl-1.8,  /usr/lo-
cal/ns-allinone-2.27/lib, into your LD_LIBRARY_PATH environment var-
iable. If it complains about X libraries, add path to your X libraries
into LD_LIBRARY_PATH.
    If you are using csh, you can set it like:
                setenv LD_LIBRARY_PATH <paths>
    If you are using sh, you can set it like:
                export LD_LIBRARY_PATH=<paths>
 (2) You MUST put /usr/local/ns-allinone-2.27/tcl8.4.5/library into
your TCL_LIBRARY environmental variable. Otherwise ns/nam will com-
plain during startup.
 (3) [OPTIONAL] To save disk space, you can now delete directories
tcl8.4.5 and tk8.4.5. They are now installed under /usr/local/ns-
allinone-2.27/{bin,include,lib}
After these steps, you can now run the ns validation suite with
```

```
 cd ns-2.27; ./validate
For troubleshooting, please first read ns problems page
http://www.isi.edu/nsnam/ns/ns-problems.html. Also search the ns
mailing list archive for related posts.
[root@localhost ns-allinone-2.27]#
```

Con lo anterior, se hicieron las modificaciones al archivo etc/profile (que sirve para la clave "root") de la siguiente forma:

```
pathmunge /sbin
      pathmunge /usr/sbin
      pathmunge /usr/local/sbin
         pathmunge /usr/local/ns-allinone-2.27/bin
         pathmunge /usr/local/ns-allinone-2.27/tcl8.4.5/unix
         pathmunge /usr/local/ns-allinone-2.27/tk8.4.5/unix
         pathmunge /usr/local/ns-allinone-2.27/otcl-1.8
         pathmunge /usr/local/ns-allinone-2.27/lib
```

Para modificar el PATH del usuario individual apiero se modificó el archivo .bash_profile en el directorio /home/apiero (normalmente invisible si en la terminal se pone ls, se debe usar ls - a). Así que para editarlo se debe escribir en la Terminal: gedit .bash_profile

Se modificó este archivo para que tuviese las líneas:

```
PATH=$PATH:$HOME/bin
PATH=$PATH:"/usr/local/ns-allinone-2.27/bin":"."
PATH=$PATH:"/usr/local/ns-allinone-2.27/tcl8.4.5/unix"
PATH=$PATH:"/usr/local/ns-allinone-2.27/tk8.4.5/unix"
PATH=$PATH:"/usr/local/ns-allinone-2.27/otcl-1.8"
PATH=$PATH:"/usr/local/ns-allinone-2.27/lib"
```

C.6 Instalación de ns-2 en el Ambiente Fedora 3.12

Instalación de 19 julio de 2011[96]. Para la instalación de ns-2.34 en el ambiente Fedora 12 hubieron problemas con gcc en su versión 4.5 y se debió a que el código c++ original de ns-2.34 tenía problemas. Estos problemas se encontraron en los archivos:

```
nakagami.cc, en folder alfredo\ns-allione-2.34\mobile
ranvar.cc, en folder alfredo\ns-allione-2.34\tools
```

En estos problemas los encontró Jeremie Decouchant (en las líneas en donde los había).

En ranvar.cc la línea modificada fue:

```
// SIGUIENTE LINEA ARREGLADA POR JEREMIE - UAM CUA MEXICO 19 JULIO 2011
```

[96] Documento en: /home/alfredo/LOG-INSTALACION

```
//return    GammaRandomVariable::GammaRandomVariable(1.0    +    alpha_,
beta_).value() * pow (u, 1.0 / alpha_);
return GammaRandomVariable(1.0 + alpha_, beta_).value() * pow (u, 1.0 /
alpha_);
```

En nakagami.cc las líneas modificadas fueron:

```
// SIGUIENTE LINEA ARREGLADA POR JEREMIE - UAM CUA MEXICO 19 JULIO 2011
//   resultPower   =   ErlangRandomVariable::ErlangRandomVariable(Pr/m,
int_m).value();
resultPower = ErlangRandomVariable(Pr/m, int_m).value();
           } else {
// SIGUIENTE LINEA ARREGLADA POR JEREMIE - UAM CUA MEXICO 19 JULIO 2011
//     resultPower     =     GammaRandomVariable::GammaRandomVariable(m,
Pr/m).value();
resultPower = GammaRandomVariable(m, Pr/m).value();
```

Luego se modificaron las variabes PATH, según lo indicado por el archivo README, en el archivo de inicio usando vi .bash_profile

Al final de la validación, el mensaje fue el siguiente (corrido con cd ns-2.34; ./validate)

```
validate overall report: all portable tests passed but some non-portable
tests failed:
./test-all-srm ./test-all-smac-multihop ./test-all-hier-routing ./test-
all-algo-routing ./test-all-mcast ./test-all-vc ./test-all-session
./test-all-mixmode ./test-all-webcache ./test-all-mcache ./test-all-plm
./test-all-wireless-tdma to re-run a specific test, cd tcl/test;
./test-all-TEST-NAME

Note that the non-portable test list includes files that are known to
pass on Linux x86 machines but that are known to have some portability
problems to other platforms or operating systems.   Failure of one of
these tests does not necessarily mean that one will encounter a problem
in your simulations, but that one must be careful of using that specific
model on the system on which the validate was run.
[alfredo@saturn ns-2.34]$
```

C.7 Archivos Empleados para Hacer las Simulaciones y Obtener los Resultados, en Fedora 12

Los archivos utilizados para correr las simulaciones fueron:

- “topología7-5-3.cmd”. Este es un archivo de comandos "bash" de Linux que define los parámetros de operación, y corría, a su vez, un archivo “Script” de ns-2, llamado “topologia7-3.ns”.
- “topología7-3.ns”. Este archivo toma parámetros que le pasa el programa de comandos. Cada vez que se corre este programa se obtiene un archivo de trazas de los resultados de la simulación. Para un conjunto de iguales parámetros de simulación, el archivo Script se podía correr varias veces para obtener resultados con los cuales se pudiesen obtener promedios. Con los

archivos de trazas, el archivo de comandos también corría un programa para obtener el retraso en rutas, o en nodos centrales, hecho en AWK, llamado “retraso13rc.awk”.

- “retraso13rc.awk”. Programa para obtener el retraso en rutas o en nodos centrales. La respuesta del programa depende de los argumentos con que se llame. El programa se corre con: LC_ALL=DA gawk -f retraso13rc.awk. Este programa obtiene promedios para lapsos de cada 120 *s* de la simulación.

- “ret-completo-promedia.tcl”. Con este programa, hecho en tcl, y también corrido desde el archivo de comandos, se obtienen los promedios de los retrasos, a partir delos resultados emitidos por el programa “retraso13rc.awk”. Este programa se corría como: “tclsh8.4 ret-completo-promedia.tcl”.

- “tasa-mejorado-1.awk”. Programa empleado en situaciones especiales para medir la tasa promedio en el tráfico en alguna ruta.

- Adicionalmente se emplearon algunos programas Srcipt para gnuplot, para graficas de retrasos, pesos, ganancias. Los resultados se daban en “ps” (Postscript), por lo que se usaban algunos programas de conversión, incluidos en el archivo de comandos “grafica-convertir”. La conversión resultaba en archivos del tipo “jpg” o “png”. Los programas se corrían en el ambiente interactivo de gnuplot, como: “gnuplot> load 'graficar-ganancias.cmd'”

C.8 Explicación de las Líneas del archivo de Trazas de ns-2

Sobre trazas[97], en [89] se puede obtener el formato de los diversos archivos de trazas. Sobre los formatos normales, el formato es el siguiente:

Tipo.	**Núm. de columna o campo.**	**Valor.**
string	1	Tipo de evento. Significados: “r”. Recepción. El paquete ya llegó completo al nodo destino en el enlace (se pone inmediatamente en la cola de la interfaz de salida del nodo destino). “-”. Desencolado. El paquete empezó a salir de cola del nodo fuente del enlace. “+”. Encolado. El paquete llegó completo a cola del nodo fuente en el enlace. d: Caída (“Drop”). e: Error. Se separa del siguiente campo con un espacio “ ”.
double	2	Tiempo de ocurrencia del evento. Se separa del siguiente campo con un espacio “ ”.
int	3	(Enlace de datos). Número del nodo fuente en el enlace. Se separa del siguiente campo con un espacio “ ”.
int	4	(Enlace de datos). Número del nodo destino en el enlace. Se separa del siguiente campo con un espacio “ ”.

[97] Algunos de los archivos de trazas llegaban a tener cerca de 1 *GByte*.

Tipo.	Núm. de columna o campo.	Valor.
string	5	Nombre del Paquete. El nombre indica el tipo de fuente (por ejemplo cbr indica que la fuente es de tasa constante: Constant Bit Rate). Se separa del siguiente campo con un espacio " ".
int	6	Tamaño del Paquete, en Bytes. Se separa del siguiente campo con un espacio " ".
string	7	Conjunto de banderas. Se separa del siguiente campo con un espacio " ".
int	8	ID del flujo del paquete. Se separa del siguiente campo con un espacio " ".
int	9	(Capa de Red). Número del nodo en donde está la fuente del paquete. Se separa del siguiente campo con un punto ".".
int	10	Número de Puerto en el nodo fuente del paquete (corresponde a una fuente). Se separa del siguiente campo con un espacio " ".
int	11	(Capa de Red). Número del nodo destino del paquete. Se separa del siguiente campo con un punto ".".
int	12	Número de Puerto en el nodo destino del paquete (e n nuestro caso corresponde al número de puerto del nodo fuente del paquete). Se separa del siguiente campo con un espacio " ".
int	13	Número de Secuencia. Número del paquete por su secuencia de emisión en cada fuente. Se separa del siguiente campo con un espacio " ".
int	14	Identificador único de un paquete. Campo final.

Como ejemplo, con relación a los campos 9, 10, 11 y 12, si una ruta iniciase en un nodo s_1 (con número de nodo #0) y concluyese en el nodo d_1 (con número de nodo #2), se tendrían los valores siguientes en dichos campos (se dan los valores de dos renglones de ejemplo).

```
0.0 2.0 indica del nodo 0 fuente 0 al nodo 2 fuente 0.
0.1 2.1 indica del nodo 0 fuente 1 al nodo 2 fuente 1.
```

Apéndice D. El Despachador GPS y el Despachador PGPS (WFQ)

D.1 El Despachador GPS

El despachador GPS (General Processor Sharing) [40] es un tipo "ideal" de despachador que sirve como referencia para análisis y comparación. En el despachador GPS se considera que dos o más paquetes pueden atenderse simultáneamente, como si su contenido fuese un fluido, donde los fluidos de cada paquete atendido simultáneamente comparten la salida del despachador. El enlace de salida sería sustituido por *N* tuberías por donde pasaría el tráfico, en forma de fluido, de cada una de las *N* colas. Si alguna cola quedase vacía, en algún intervalo de tiempo, en ese intervalo las demás colas podrían aprovechar la capacidad de la tubería no utilizada, de tal forma que nunca tubería alguna quedaría en desuso, mientras hubiese tráfico en espera a ser atendido por el despachador. Un despachador así no desperdicia nunca sus recursos de transmisión disponibles, cuando tiene tráfico a atender. El despachador GPS atiende a los paquetes sin hacerlos esperar, en absoluto, cuando éstos llegan a la cabeza de su cola correspondiente. Es decir, el tiempo de espera de los paquetes, cuando éstos llegan a la cabeza de su cola, es de cero.

El despachador GPS está caracterizado por valores positivos reales ϕ_1, ϕ_2, ϕ_N, que se llaman "pesos". Hay un peso para cada una de sus *N* colas. Estos pesos se usan para establecer la mínima tasa con que se atiende el tráfico de una tubería, en cualquier intervalo en que haya tráfico en espera para dicha tubería (se dice mínima tasa porque es posible que la tubería pudiese aprovechar en dicho intervalo la capacidad no utilizadas de otra u otras tuberías). Dígase que la tasa total de salida del despachador es *R* (en el mundo de las redes a esta cantidad se le refiere también como "ancho de banda").

El despachador GPS no es realizable pero sirve como referencia para la operación de otro despachador llamado PGPS[98], que es un despachador que opera con paquetes. Cuando opera PGPS, GPS opera en paralelo. GPS opera como si atendiese a los paquetes que van llegando a las colas de la interfaz. Un despachador PGPS se basa en la operación de un despachador GPS, pero el primero solo puede atender a un paquete a la vez, pues es un despachador *real*, no de fluido, y para saber qué paquete es el siguiente a atender depende de la información de GPS. Ya que PGPS atiende paquete por paquete, sus tiempos de atención varían ligeramente con los tiempos de atención de GPS. En [40] se ofrece detalle sobre la diferencia de tiempo en que se atiende a paquetes del despachador PGPS con relación al despachador GPS, conforme ambos operan en paralelo. Ahí se explica por qué el despachador PGPS, comparado con otros despachadores, es el que menor tiempo de espera, en promedio, impone a los paquetes encolados.

[98] A PGPS se le llama también WFQ (Weighted Fair Queueing).

La operación de estos despachadores causa mucho trabajo de cálculo por lo que no son preferidos para ambientes de operación productivos (en donde los enrutadores requieren atender a muchos miles de paquetes por segundo[99]). El uso de estos despachadores se hace con motivos de estudio, por ejemplo con simuladores.

D.1.1 Ecuaciones Básicas de GPS

Esta parte del presente apéndice ofrece una explicación detallada sobre la operación de GPS. No se incluye el análisis de la diferencia entre tiempos de atención entre GPS y PGPS (para lo cual referirse a [40]).

El Apéndice E ofrece un análisis de operación y algoritmo de implantación para el despachador GPS operando con pesos variables. Como se ha indicado, este análisis y propuesta es original de esta Tesis, pero no se considera una parte fundamental de la misma porque existen ya trabajos que usan despachadores de este tipo y que trabajan con pesos variables, por lo que alguna propuesta de operación similar podría estar ya reportada.

Sea $S_i(\tau, t)$ la cantidad de tráfico de la cola *i*, a ser atendida en el lapso $(\tau, t]$. Si la cola *i* tiene continuamente tráfico por atender en dicho lapso[100], entonces se debe cumplir, según la operación de GPS, que:

$$\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\phi_i}{\phi_j}, \quad i,j = 1,2,...,N \tag{36}$$

Y se debe cumplir que:

$$\sum_j S_j(\tau,t) = (t-\tau)R \tag{37}$$

Donde *R* es el "ancho de banda" del enlace de salida del despachador. Lo anterior quiere decir que el tráfico atendido en el lapso $(\tau, t]$ es tan grande como el tráfico máximo que puede atender el despachador en ese lapso, que es igual a $(t-\tau)R$.

De (36) se obtiene: $\frac{\phi_j}{\phi_i} S_i(\tau, t) \geq S_j(\tau, t), i, j = 1,2,...,N$. Sustituyendo en (37) se obtiene:

[99] El enrutador Cisco 7600 está indicado para proveer 16 *Gb/s* o 6.5 millones de paquetes por segundo, de salida por cada módulo [99].

[100] Cuando hay tráfico por atender en la cola *i* se utiliza un término en inglés para indicar que la cola está retrasada en su trabajo por hacer: "Backlogged".

$\frac{S_i(\tau, t)}{\phi_i}\sum_j \phi_j \geq \sum_j S_j(\tau,t) = (t-\tau)R\phi_j$, y despejando se obtiene:

$$\frac{S_i(\tau,t)}{(t-\tau)} \geq \frac{\phi_i}{\sum_j \phi_j} R \tag{38}$$

Donde $\frac{S_i(\tau,t)}{(t-\tau)}$ es la tasa de atención a la cola $S_i(\tau,t)$ durante el lapso $(t-\tau]$. De lo anterior se deriva que la tasa garantizada mínima $R_i(\tau, t)$, para la cola *i*, durante dicho lapso, es:

$$R_i(\tau,t) = \frac{\phi_i}{\sum_j \phi_j} R \tag{39}$$

Pudiendo la tasa para la cola *i* ser mayor cuando en una o más de las otras colas no hay trabajo por hacer.

Considérese que hay un intervalo (τ_1, t_1] tan pequeño como se requiera para que las colas mantengan una tónica, es decir, o tienen trabajo por hacer continuamente, o no tienen trabajo por hacer en absoluto. Entonces, la tasa que tendrá la cola *i* sería.

$$R_i(\tau,t) = \begin{cases} \dfrac{\phi_i(\tau,t)}{\sum\limits_{\phi_j \in B(\tau,t)} \phi_j} R, & \phi_i \in B(\tau,t) \\ 0, & \phi_i \notin B(\tau,t) \end{cases} \tag{40}$$

Donde $B(\tau,t)$ es el conjunto de colas con trabajo por hacer (ocupadas) en el intervalo (τ,t).

D.2 Selección del Próximo Paquete a Enviar de parte de GPS, para PGPS

Mientras un paquete esté siendo atendido por el despachador se considera que el paquete sigue estando en su cola de espera, y mientras haya al menos un paquete en una cola se dice que la cola está ocupada. La finalización de atención de un paquete se da cuando el mismo ha salido completamente del despachador (y de la cola en donde estaba). Justo cuando se finaliza la atención de un paquete, de una cola *i*, se puede iniciar la atención del que siga en dicha cola, si lo hubiere.

Considere la situación en que, en un momento dado, un despachador GPS está atendiendo, simultáneamente, a los flujos de cada uno de los paquetes que encabezan las diversas colas ocupadas. Entonces, con la operación de GPS se puede saber cuánto tráfico de cada paquete se ha atendido hasta el momento. También, en ese momento, se puede identificar al primer

paquete que ha de llegar a ser completamente atendido, y el tiempo de ese evento; sin embargo, si antes de realizarse dicho evento llegase un nuevo paquete a una de las colas desocupadas, si las hubiese, se cambiaría la distribución de la atención de las colas, y sería necesario volver a hacer cálculos para identificar, nuevamente, el primer paquete que llegaría a ser completamente atendido, y el tiempo de ese evento (el paquete recién llegado podría tomar ese lugar).

Recordar que el despachador GPS puede atender a varios paquetes a la vez, mientras que el despachador PGPS no puede, y que ambos despachadores operan a la vez. El despachador GPS opera ficticiamente (no despacha realmente los paquetes), mientras que PGPS sí los despacha. Es el despachador GPS el que va indicando al despachador PGPS el paquete siguiente que debe atender. Este paquete es aquél, en la operación del despachador GPS, cuya conclusión de atención es la más próxima.

A continuación se presentan expresiones matemáticas con las que se plantea la operación del despachador GPS.

Llámese evento al inicio o finalización de atención de un paquete de cualquiera de las colas, por parte del despachador GPS[101], o a la llegada de un paquete a alguna de las colas (se considera que un paquete ha llegado cuando el mismo ha llegado por completo).

Divídase el tiempo en intervalos entre eventos. Cada intervalo concluye cuando hay un siguiente evento, quedando este último evento situado al inicio del siguiente intervalo.

Sea t_{b-1} el tiempo en que ocurre un evento, y sea t_b el tiempo en que ocurre el siguiente evento. El intervalo $\left[t_{b-1}, t_b\right)$ corresponde al evento t_{b-1}. Entonces el conjunto de colas ocupadas es fijo durante ese intervalo, y se denota como $B\left[t_{b-1}, t_b\right)$. Entre un intervalo y el siguiente el conjunto de colas ocupadas puede cambiar, o permanecer igual.

Denótese el tiempo en que un paquete #*k* llega a la cola #*i* como a_i^k, a la longitud de ese paquete como L_i^k, y al paquete mismo como p_i^k.

Considérese que se está atendiendo a p_i^k. Durante el tiempo de atención de este paquete podrían llegar cero, uno o más paquetes a alguna de las colas. El análisis es importante cuando llegan uno o más paquetes, porque cada llegada puede cambiar la selección del próximo paquete que se indicaría a PGPS para enviarse a continuación.

[101] Se dice que un paquete ha sido atendido cuando el mismo ha sido atendido completamente (ha salido completamente).

Considérese que t_1 es el tiempo de inicio de operación del sistema de colas (se toma entonces como que $t_1 = 0$), y que la atención al paquete p_i^k se da en un lapso de tres eventos consecutivos: s, $s + 1$ y $s + 2$, $s \geq 1$, en los tiempos: t_s, t_{s+1} y t_{s+2}, iniciando la atención del paquete en t_s, donde obviamente $a_i^k \leq t_s$. Si justo antes de t_s se hubiese estado atendiendo a un de la cola i, éste estaría saliendo de dicha cola justo en t_s (dicho paquete podría haber sido p_i^{k-1} en caso de que éste no hubiese sido extraído de la cola antes de ser atendido de esa cola).

Considérese que el tiempo de atención a p_i^k cruza el evento $s + 1$, que consiste en la llegada de un paquete a alguna cola diferente a la i, o en la conclusión de la atención de un paquete en otra cola diferente a la i, y que se denota con t_{s+1}. El evento de la conclusión de la atención del paquete p_i^k es el evento $s + 2$, en el tiempo t_{s+2}.

Supóngase que entre los tiempos t_s y t_{s+1} se atiende la fracción del paquete llamada $\mathit{Fracción}_{\lfloor t_s, t_{s+1})}\left(L_i^k\right)$, y supóngase que entre los tiempos t_{s+1} y t_{s+2} se atiende la fracción final del paquete, llamada $\mathit{Fracción}_{\lfloor t_{s+1}, t_{s+2})}\left(L_i^k\right)$. Entonces, se cumple que:

$$L_i^k = \mathit{Fracción}_{\lfloor t_s, t_{s+1})}\left(L_i^k\right) + \mathit{Fracción}_{\lfloor t_{s+1}, t_{s+2})}\left(L_i^k\right) \tag{41}$$

Con esto, las expresiones siguientes son inmediatas:

$$t_{s+1} = t_s + \frac{\mathit{Fracción}_{\lfloor t_s, t_{s+1})}\left(L_i^k\right)}{R\phi_i \Big/ \sum_{\phi_j \in B\lfloor t_s, t_{s+1})} \phi_j} \tag{42}$$

Que equivale a:

$$\frac{t_{s+1} - t_s}{\sum_{\phi_j \in B\lfloor t_s, t_{s+1})} \phi_j} = \frac{\mathit{Fracción}_{\lfloor t_s, t_{s+1})}\left(L_i^k\right)}{R\phi_i} \tag{43}$$

Y similarmente se puede obtener:

$$\frac{t_{s+2} - t_{s+1}}{\sum_{\phi_j \in B\lfloor t_{s+1}, t_{s+2})} \phi_j} = \frac{\mathit{Fracción}_{\lfloor t_{s+1}, t_{s+2})}\left(L_i^k\right)}{R\phi_i} \tag{44}$$

Combinando las expresiones anteriores se obtiene:

$$\frac{t_{s+1}-t_s}{\sum_{\phi_j \in B[t_s,t_{s+1})} \phi_j} + \frac{t_{s+2}-t_{s+1}}{\sum_{\phi_j \in B[t_{s+1},t_{s+2})} \phi_j} = \frac{Fracción_{[t_s,t_{s+1})}\left(L_i^k\right) + Fracción_{[t_{s+1},t_{s+2})}\left(L_i^k\right)}{R\phi_i} = \frac{L_i^k}{R\phi_i} \tag{45}$$

La anterior expresión se puede escribir como:

$$V\left(t_{s+2}-t_{s+1}\right)+V\left(t_{s+1}-t_s\right)=\frac{L_i^k}{R\phi_i} \tag{46}$$

Donde $\frac{t_{s+2}-t_{s+1}}{\sum_{\phi_j \in B[t_{s+1},t_{s+2})} \phi_j}$ se representa con $V\left(t_{s+2}-t_{s+1}\right)$, y $\frac{t_{s+1}-t_s}{\sum_{\phi_j \in B[t_s,t_{s+1})} \phi_j}$ se representa con $V\left(t_{s+1}-t_s\right)$. $V(\tau)$ se define como el lapso de **tiempo virtual** correspondiente al lapso de tiempo real τ. Es importante recalcar que durante cada lapso de tiempo virtual el conjunto de colas ocupadas debe permanecer constante.

El tiempo virtual es un concepto que ayuda a formular matemáticamente el comportamiento del sistema GPS. En el sistema el tiempo real va avanzando y el tiempo virtual también, pero el avance del tiempo virtual depende de las colas que vayan estando ocupadas en los diversos lapsos de tiempo virtual.

Se observa que los dos lapsos de tiempo virtuales abarcan toda la duración de atención del paquete. Se observa también que la suma de dichos dos lapsos de tiempo virtuales es igual a $L_i^k / R\phi_i$ que es una duración virtual completa de la atención al paquete (desde que se empieza a atender hasta que se acaba de atender).

Curiosamente, resulta que esta duración virtual de atención al paquete se puede calcular sin requerir saber qué colas estarán ocupadas o vacías durante la atención del paquete. Esto es como decir que desde la llegada del paquete se puede calcular la duración virtual de su atención. Se establece entonces que:

$$Duración\ virtual\ de\ atención\ al\ paquete\ i = L_i^k / R\phi_i \tag{47}$$

Asimismo, con las ecuaciones anteriores se puede obtener una expresión para poder llevar la cuenta del avance del tiempo virtual.

Considérese que se conoce el tiempo virtual específico $V(t_{s-1})$ (aún no se ha dicho cómo se conoce), entonces el avance del tiempo virtual correspondiente al tiempo real $t = t_{s-1} + \tau$, $0 < \tau < t_s - t_{s-1}$, que corresponda a un tiempo virtual anterior al siguiente tiempo virtual $V(t_s)$ sería[102]:

$$V(t_{s-1}+\tau) = V(t_{s-1}) + \frac{\tau}{\sum_{\phi_j \in B[t_{s-1}, t_{s-1}+\tau]} \phi_j} \quad , \quad 0 < \tau < t_s - t_{s-1}, s = 2,3,... \tag{48}$$

Al estar todas las colas vacías, el primer evento sería la llegada de un paquete. Se define el tiempo real de este primer evento como $t_1 = 0$, y se define $V(t_1) = V(0) = 0$.

Todos los tiempos virtuales subsecuentes correspondientes a los eventos se pueden calcular, de manera secuencial.

La ecuación (48), justo cuando t alcanza el valor t_s se pasa a un nuevo lapso de tiempo virtual $[t_s, t_{s+1})$ y el valor de $V(t_s)$ pasa a ser:

$$V(t_s) = \left[V(t_{s-1}) + \frac{t_s - t_{s-1}}{\sum_{\phi_j \in B[t_{s-1}, t_s)} \phi_j} \right] + \frac{0}{\sum_{\phi_j \in B[t_s, t_{s+1})} \phi_j} \; , s = 2, 3..., \quad t_1 = 0, V(t_1) = 0. \tag{49}$$

Que es igual a:

$$V(t_s) = V(t_{s-1}) + \frac{t_s - t_{s-1}}{\sum_{\phi_j \in B[t_{s-1}, t_s)} \phi_j} \; , s = 2, 3..., \quad t_1 = 0, V(t_1) = 0. \tag{50}$$

Es justo la ecuación (50) la que indica cómo ir calculando el tiempo virtual. Es suficiente actualizar el tiempo virtual cada vez que sucede un evento y no continuamente conforme pasa el tiempo real. Esto es porque la ocurrencia de los eventos es lo que puede cambiar el conjunto *B*, y solo dicho cambio altera el ritmo del tiempo virtual: un lapso de tiempo real en donde hay más colas con trabajo pendiente se convierte en un lapso de tiempo virtual mayor (que transcurre más lentamente).

Si los tiempos virtuales de inicio de atención y de finalización de atención de un paquete que llegó en el tiempo real a_i^k se denotan con S_i^k y F_i^k, respectivamente, de las formulaciones anteriores se puede indicar:

[102] Ecuación (10) de [40].

$$S_i^k = \max\left(F_i^{k-1}, V\left(a_i^k\right)\right) \tag{51}$$

y

$$F_i^k = S_i^k + \frac{L_i^k}{R\phi_i} \tag{52}$$

Se puede establecer la siguiente regla. Al llegar p_i^k a la cola *i*, si se conoce ya el tiempo virtual F_i^{k-1}, se puede calcular en ese momento los valores S_i^k y F_i^k.

La ecuación (50) puede generalizarse para tiempos diferentes a los tiempos de eventos. Para esto, se repite la ecuación (48), haciendo $t = t_{s-1} + \tau$, donde: $0 < \tau < t_s - t_{s-1}$ (equivalente a: $t_{s-1} < t < t_s$), se obtiene que $V(t) = V(t_{s-1}) + \frac{t - t_{s-1}}{\sum_{\phi_j \in B[t_{s-1}, t_{s-1}+\tau]} \phi_j}$. Despejando $V(t_{s-1})$ se obtiene: $V(t_{s-1}) = V(t) + \frac{t_{s-1} - t}{\sum_{\phi_j \in B[t_{s-1}, t_{s-1}+\tau]} \phi_j}$. Sustituyendo $V(t_{s-1})$ en la ecuación (50) se obtiene:

$$V(t_s) = V(t) + \frac{t_{s-1} - t}{\sum_{\phi_j \in B[t_{s-1}, t_{s-1}+\tau]} \phi_j} + \frac{t_s - t_{s-1}}{\sum_{\phi_j \in B[t_{s-1}, t_s)} \phi_j}, s = 2, 3..., \quad t_1 = 0, V(t_1) = 0. \tag{53}$$

Como $0 < \tau < t_s - t_{s-1}$, entonces: $B\left[t_{s-1}, t_{s-1} + \tau\right] = B\left[t_{s-1}, t_s\right)$, y de (53) se obtiene:

$$V(t_s) = V(t) + \frac{t_s - t}{\sum_{\phi_j \in B[t_{s-1}, t_s)} \phi_j}, s = 2, 3..., \quad t_1 = 0, V(t_1) = 0. \tag{54}$$

Asimismo, como $t = t_{s-1} + \tau$, lo que implica que $t_{s-1} < t < t_s$, entonces $B\left[t_{s-1}, t_s\right) = B\left[t, t_s\right)$, obteniendo:

$$V(t_s) = V(t) + \frac{t_s - t}{\sum_{\phi_j \in B[t, t_s)} \phi_j}, s = 2, 3..., \quad t_{s-1} < t < t_s, \quad t_1 = 0, V(t_1) = 0. \tag{55}$$

La ecuación (55) se parece a la ecuación (50), pero cambiando a t_{s-1} por *t*. Dado que solamente interesan, para esta tesis, los tiempos en que suceden eventos, dado que es en ellos en donde se va calculando el avance del tiempo virtual, considérese nuevamente la ecuación (50) (en lugar de la ecuación (55)), y sustitúyase t_{s-1} por *t*, considerando que ahora *t* es un tiempo de un evento. La ecuación 31 se convierte en:

$$V(t_s) = V(t) + \frac{t_s - t}{\sum_{\phi_j \in B[t, t_s)} \phi_j}, s = 2, 3..., \quad t_{s-1} = t, \quad t_1 = 0, V(t_1) = 0. \tag{56}$$

Considérese que *t* es el tiempo presente, en que acaba de ocurrir un evento (de entrada o salida de paquete), y no se conoce cuándo llegaría un nuevo paquete. En ese momento el despachador está atendiendo una o varias colas. El registro que se puede tener hasta ese momento *t*, sobre el siguiente evento, aquél que se da en t_s, necesariamente es un evento de salida del paquete, que justamente es el del paquete más próximo a salir. Desde luego, en ese momento aún no se sabe si llegarían uno o más paquetes antes del tiempo t_s. Si entre *t* y t_s hubiese una llegada de paquete, el tiempo de ese evento se debería llamar ahora t_s, y el tiempo de salida del siguiente paquete a salir se debería recalcular porque el conjunto *B* posiblemente cambiaría con la llegada del nuevo paquete, y además porque ese paquete, recién llegado, podría ahora ser el paquete con el tiempo más próximo a salir.

De cualquier forma, es claro que en el tiempo *t* no se conoce aún qué iría a pasar, desde ese momento hasta antes del tiempo t_s. Denótese al tiempo $V(t_s)$ como F_{min}, que sería el tiempo virtual del tiempo de salida del paquete más próximo a salir, según se conoce hasta el tiempo real *t*. Denótese a t_s como *Next*(*t*). Entonces, la ecuación (55) puede escribirse como:

$$F_{\min} = V(t) + \frac{Next(t) - t}{\sum_{\phi_j \in B[t, Next(t))} \phi_j} \tag{57}$$

Donde *t* es el tiempo real del evento presente, *Next*(*t*) es el tiempo real del evento de salida de paquete del paquete más próximo a salir, y *V*(*t*) y F_{min} son los correspondientes tiempos virtuales.

Se puede despejar *Next*(*t*), quedando,

$$Next(t) = t + \left(F_{\min} - V(t)\right) \sum_{\phi_j \in B[t, Next(t))} \phi_j \tag{58}$$

D.3 El Scheduler del Simulador y su Relación con los Eventos de los Despachadores GPS y PGPS (WFQ)

D.3.1 Sobre el Scheduler del Simulador.

Para la operación del simulador ns-2 se requiere de un Despachador (o Scheduler) del simulador, el cual tiene una lista de programación de los eventos que han de ocurrir en su opera-

ción. Es importante recalcar que el nombre de este elemento del simulador también es "despachador", al que se le refiere desde ahora, con su palabra en inglés, "Scheduler" (del Simulador), para evitar confusiones.

Dicha lista de programación de eventos contiene referencias a los eventos que van ocurriendo en la simulación. Estos eventos incluyen aquellos de llegada de paquetes, inicio de atención de paquetes, o salida de paquetes (recordar que salida es igual a la finalización de atención). Los eventos se colocan en orden creciente de tiempo (de tiempo real), para su atención, por parte del Scheduler del Simulador, también en orden creciente de tiempo (real). Conforme se atiende algún evento, el mismo puede causar la generación de otros eventos, posteriores, que se deben intercalar, según cronología de tiempo de evento, en la lista de eventos del Scheduler del simulador.

En el Scheduler del simulador se coloca la referencia al evento correspondiente a la salida del paquete del despachador GPS que sea el más próximo a salir, que corresponde al tiempo real *Next*(*t*). Cuando el simulador atiende este evento se aplican los cálculos, ya indicados, correspondientes a la salida de un paquete, en el despachador GPS. Es importante notar que solamente hay una referencia de evento de salida de paquete del despachador GPS, puesta en el Scheduler del simulador, a la vez. Con cada llegada o salida de paquete, al o del despachador GPS, se va actualizando el tiempo virtual, llevado por dicho despachador.

Note que el evento de salida de un paquete del despachador GPS no es igual al evento de salida de un paquete del despachador PGPS. La salida de los paquetes del despachador PGPS se hace conforme la dinámica normal de movimiento de paquetes en el simulador.

Cuando llega un paquete a una de las colas de la interfaz, esta llegada incide, simultáneamente, en los despachadores GPS y PGPS.

En el caso de que hubiese una referencia a un evento, correspondiente a la salida de un paquete del despachador GPS (del más próximo a salir), puesto en la lista de eventos del Scheduler del simulador, para su atención, y que antes de la atención de dicho evento llegase algún paquete a una de las colas de la interfaz, entonces se tendría que recalcular el tiempo *Next*(*t*), y actualizar la entrada de la referencia del evento de salida de paquete más próximo a salir, de dicha lista.

En el caso en que la atención del Scheduler del simulador llegase a la atención del evento correspondiente al tiempo *Next*(*t*), entonces se atendería la salida del paquete del despachador GPS, se quitaría dicho evento de la lista de eventos del Scheduler del simulador. A continuación se tendría que calcular el tiempo *Next*(*t*) del siguiente evento de salida de un paquete, para poner a ese evento en dicha lista.

Sobre el procedimiento planteado, se hacen las siguientes observaciones:

- Durante cualquier lapso con trabajo en el despachador, el tiempo virtual es una función creciente estrictamente del tiempo real.

- Como se ha visto, al llegar un paquete se puede calcular, con las ecuaciones (51) y (52), su tiempo virtual de finalización de atención (de salida de cola). Se puede componer una lista ordenada de los tiempos virtuales de finalización de atención de los paquetes, de tiempo menor a tiempo mayor, del despachador GPS. Como hay una relación creciente entre los tiempos virtuales y los tiempos reales, el tiempo virtual menor correspondería al tiempo real menor de atención de paquete. Ese tiempo real se puede calcular con la ecuación (58). Se puede componer una lista para el despachador GPS y otra lista para el despachador PGPS. Estas listas servirían para llevar una cuenta de los paquetes que ya han empezado a ser atendidos, en cada despachador. Una vez que un paquete ha empezado a ser atendido, este será atendido hasta el final de su atención.

- Si llegase un paquete nuevo a una de las colas de la interfaz, se debería calcular su tiempo virtual de finalización de atención, y se colocaría el evento de finalización de atención de este paquete, en orden del tiempo virtual, en cada lista de ambos despachadores, GPS y PGPS, y si dicho paquete llegara a ocupar el primer lugar de la lista del despachador GPS, se debería actualizar el mismo en la lista de eventos del Scheduler del simulador, para poner el nuevo evento de la salida más próxima de un paquete del despachador GPS.

- Mientras mayor peso tiene la cola en donde llega un paquete menor es el tiempo de espera para la atención del mismo.

D.4 Recapitulación

PGPS (WFQ) es una forma de despacho que conserva la energía, en donde se atiende a un paquete a la vez con una tasa *R* (tasa total del enlace), y para seleccionar el próximo paquete a atender, que estuviese a la cabeza de una de las colas de espera, se sigue la pauta que indique un despachador GPS, que maneja la misma tasa total *R* del despachador PGPS, y el mismo tráfico de entrada. Dicha pauta se puede incorporar en las listas ordenada de los tiempos virtuales de finalización de atención de los paquetes. Si el despachador PGPS operase en una interfaz de salida de un dispositivo de red “en producción”, dentro del mismo dispositivo se debería correr una versión simulada de GPS que diese la pauta mencionada.

El despachador PGPS no conoce de pesos sino depende de la lista de tiempos virtuales del sistema paralelo GPS. La información de la composición de pesos en las colas se encuentra

incorporada en los tiempos virtuales asignados a los paquetes, en la lista de tiempos virtuales del despachador GPS.

Los tiempos entre actividad e inactividad total, entre un despachador GPS y su despachador PGPS derivado, son siempre iguales. La relación entre estos tiempos está estudiada con detalle en [40].

Apéndice E. Pesos Variables en Despachadores GPS y PGPS

E.1 Ilustración de las Colas

El ejemplo del estado de las colas del despachador GPS de la Figura 41 sirve de apoyo para la comprensión del procedimiento que sigue.

Debe quedar claro que aquí no se propone cómo calcular los valores de los pesos nuevos. Este método de cálculo es precisamente lo que arrojan los algoritmos propuestos de cambio de pesos, como el algoritmo del Método STP. En el Método STP, el criterio de cambio de pesos es precisamente cuando se ha pasado de la finalización de un periodo de tiempo tamaño *T* (que se ha tomado como de 1 *s*), al siguiente. Entonces se aplica el Método STP para el cambio de pesos.

En la Figura 41 se representan 5 colas con paquetes en proceso de ser atendidos y paquetes en espera para ser atendidos. En cada una de las colas 1 y 2 hay un paquete en proceso de atención y uno en espera para ser atendido. En la cola 3 hay un paquete en proceso de atención y dos en espera para ser atendidos. En cada una de las colas 4 y 5 hay un paquete en proceso de atención y ninguno en espera para ser atendido.

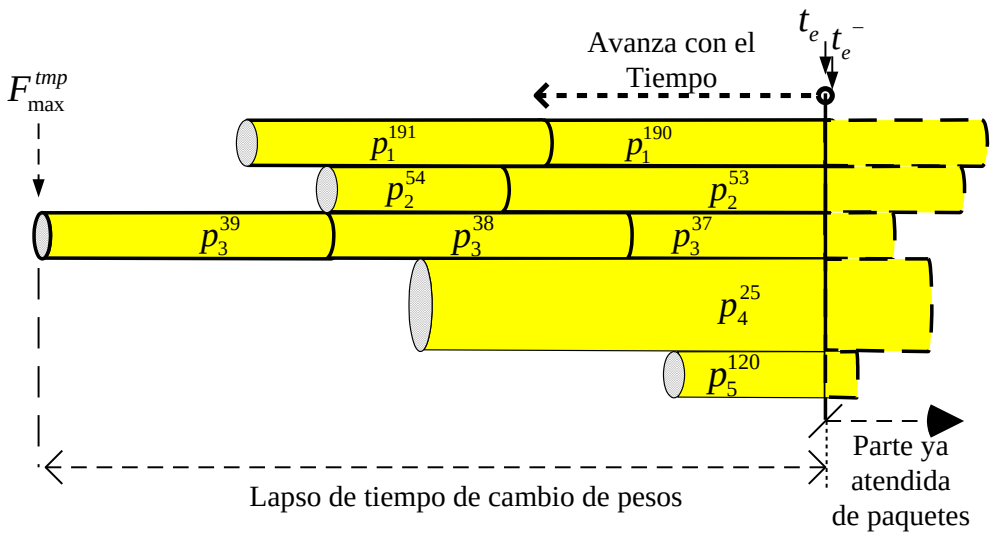


Figura 41. Representación de paquetes en espera de ser atendidos en un despachador GPS.

Se supone que se lleva el registro de la variable $F_{\max}$ como el tiempo virtual máximo de salida de un paquete, el cual se actualiza en el momento de la llegada o la salida de un paquete.

El siguiente algoritmo indica la forma de implantar el procedimiento de cambio de pesos con el despachador GPS. Se incluyen detalles de implantación en el código de C++, producido en [76], para el simulador ns-2. Todo el código del despachador GPS indicado, incluyendo el algoritmo para modificar su operación y que opere con cambio de pesos, ha sido instalado en dicho simulador, para el trabajo de esta Tesis.

E.2 Algoritmo

En la siguiente explicación del algoritmo se incluyen, en pies de página, detalles de su codificación, en C++, en el simulador ns-2. Estos detalles son para el lector que ha revisado la codificación del algoritmo y que quiere apoyarse de la presente explicación, para su comprensión. Estos detalles pueden saltarse para el lector que solamente quiere entender los pasos del algoritmo. La codificación, en las rutinas C++ del simulador ns-2, se dan en el Apéndice J.

Considérese que el número de colas es N[103]. Durante la operación del algoritmo, se tienen dos estados: 1- NO hay requerimiento de cambio de pesos, 2- SI se debe cambiar de pesos en la siguiente oportunidad en que el cambio sea viable.

En el algoritmo se hacen las siguientes declaraciones iniciales[104]:

- $F_{\max}^{tmp}$ es el tiempo máximo virtual de salida de un paquete, que se tiene registrado en el momento en que se cambia el estado, de NO a SÍ se debe cambiar de pesos. Este tiempo virtual indica el tiempo en que se ha de cambiar de pesos, desde la implantación del nuevo estado: SI[105].
- $F_{\max}$ es el tiempo máximo virtual de salida de un paquete que se actualiza en cada llegada o salida de paquete[106].
- ϕ_j, $j=1,...,N$ son los pesos presentes[107].
- ϕ_j^{New}, $j=1,...,N$ son los pesos nuevos[108] que va a sustituir a los pesos presentes. Existen también unos pesos iniciales ϕ_j', $j=1,...,N$, que son fijos[109].

---

[103] En la codificación, el número de colas máximo que se puede manejar en ns-2 está indicado en la variable fija MAX_QUEUES, que se designa igual a 8 en el archivo dsred.h. Posteriormente se declara la variable numQueues_, cuyo valor puede cambiarse, desde OTcl, y que debe ser menor o igual a MAX_QUEUES. El valor N que se utiliza es el de numQueues_, y las colas están numeradas de 0 a numQueues_ - 1. En [75], Págs. 9 y 34, se explica cómo cambiar dicho valor. Este valor se cambia en un objeto después de creado el mismo, pero, como se dice en el Manual de ns-2 ([74] Pág. 27), ns-2 garantiza que los valores actuales de las variables, tanto en los objetos interpretados como en los compilados, mantengan valores idénticos en todo tiempo. Ver también, de dicho manual, las indicaciones sobre inicialización de las variables compiladas.

[104] Declaración hecha en la codificación con inserción clave "a", en el archivo dsred.h, en la definición de la clase dsREDQueue. Los valores iniciales de estas variables se dan en el constructor de dsREDQueue en la inserción clave "b" en el archivo dsred.cc. La definición de estas rutinas está en el archivo dsred.cc en la inserción clave "m".

[105] La variable correspondiente en la codificación es: double Fmaxtmp.

[106] La variable correspondiente en la codificación es: double Fmax.

[107] Las variables correspondientes en la codificación son: double queueWeightpr[MAX_QUEUES] (donde MAX_QUEUES es una variable preexistente que indica el número máximo de colas).

[108] Las variables correspondientes en la codificación son: double queueWeightnw[MAX_QUEUES].

[109] Los pesos fijos iniciales ya existían y eran int queueWeight[MAX_QUEUE] (declarados como protected en la clase dsREDQueue, en el archivo dsred.h).

- wchpr ("Weight-change Process") es el indicador para saber si se está o no en un proceso de cambio de pesos[110].
- Se declaran rutinas varias para calcular las longitudes de las colas, los nuevos pesos.

También, $F_{\min}$ es el tiempo virtual del próximo paquete a concluir su atención.

E.2.1 Caso de la Llegada de un Paquete

Nota. Si se está en un proceso de cambio de pesos la llegada de un paquete no puede cancelar dicho estado.

Comentario	Paso	Acciones a Realizar.
CONDICION DE ESPERA INICIAL.	0	Si el sistema esta ocioso, de forma inicial (antes de cualquier llegada de paquete), los valores de los pesos: ϕ_j , j = 1, ..., N, y de los nuevos pesos, ϕ_j^{New} , j = 1, ..., N, toman, y se mantienen con, el valor de los pesos iniciales ϕ_j' , j = 1, ..., N, pero si éstos últimos no sumasen 1, entonces los mismos pasarían su valor proporcionalmente ajustado para sumar 1. Tanto $F_{\max}^{tmp}$ y $F_{\max}$ toman el valor de cero[111]. Cuando llegue un paquete se va al paso A.
INICIO.	A[112]	En un tiempo dado $t = t_e$, se da un evento de llegada de paquete (controlado por el Scheduler del simulador)[113]. Se va al paso B.

[110] Si la variable wchpr = 1, entonces se está en proceso de cambio de peso. Si wchpr = 0 no se está en proceso de cambio de peso. En la actual implantación en ns-2, el evento que ocurre, justo después de cada segundo transcurrido de la simulación, marca el momento en que se cambia el estado, de NO a SI, dentro del proceso de cambio de pesos. Después de que se hace el cambio de pesos dicho estado permanece en NO hasta el evento posterior al siguiente segundo transcurrido. Inicialmente existían otras formas de indicar cuándo se debería entrar en un estado de SI cambio de pesos. La forma actual utilizada se obtuvo con la guía de resultados de pruebas y evaluaciones, en simulaciones. Las longitudes de las colas se evalúan con base de rutinas de promediado de colas. Los nuevos pesos se obtienen con el algoritmo de cambio de pesos, del Método STP. En ns-2, se programaron herramientas para que desde el ambiente OTcl se pueda tener la opción para que ns-2 opere con cambio de pesos o sin cambio de pesos (ver la inserción "o" en el archivo dsred.cc). Por ejemplo, si el objeto (la cola ds tipo dsREDQueue) tiene el nombre, en OTcl, de "dsredq", con la instrucción: $dsredq setChgWghtOpt 1 se puede cambiar la opción para que ns-2 opere con cambios de pesos.

[111] Estos valores iniciales se definen al crearse la cola (ver inserción clave "b" en el constructor de dsREDQueue e inserción clave "g" en la rutina reset de la clase dsREDQueue en el archivo dsred.cc), y NO se reinician cuando se llega a una situación de ociosidad (ver inserción clave "z" en WFQdequeueGPS de dsREDQueue en archivo dsred.cc).

[112] Esta parte del algoritmo ya existía en WFQenqueue(Packet *p, int queueid) de dsREDQueue (ver dsred.cc).

[113] Los procedimientos de llegada de paquete se ven en rutinas enque(Packet* pkt) y WFQenqueue(Packet *p, int queueid) ambas de dsREDQueue (en el archivo dsred.cc).

Viene de A.	B[114]	Parte existente en el algoritmo GPS original. Se va a pasos C o D, dependiendo si el sistema estaba ocioso o no.

Viene de interrogante B.	C[115]	Parte existente en el algoritmo GPS original. Caso de que el sistema SÍ estaba ocioso. Se asigna el valor del tiempo virtual presente $V(t_e) = 0$. Se marca el sistema como no ocioso. Se va al paso E.

Viene de interrogante B.	D	Parte existente en el algoritmo GPS original. Caso en que el sistema NO estaba ocioso (la siguiente formulación funciona para cuando SÍ o cuando NO se está en un proceso de cambio de pesos). Se calcula $V(t_e)$, con el uso de la ecuación (55), sustituyendo a t_s por t_e y a t_{s-1} por t_{e-1}: $$V(t_e) = V(t_{e-1}) + \frac{t_e - t_{e-1}}{\sum_{\forall \ell \ni Cola_\ell(t_{e-1}) \in B[t_{e-1}, t_e)} \phi_\ell}$$ Donde $\frac{t_e - t_{e-1}}{\sum_{\forall \ell \ni Cola_\ell(t_{e-1}) \in B[t_{e-1}, t_e)} \phi_\ell}$ es el incremento de tiempo virtual. En caso de estar en un proceso de cambio de pesos, $V(t_e)$ debe ser **menor que** $F_{\max}^{tmp}$, dado que al estar en este estado el evento correspondiente al tiempo virtual $F_{\max}^{tmp}$ no ha sucedido. Si $V(t_e)$ **fuese mayor o igual** a $F_{\max}^{tmp}$ **habría un error** y se debería detener el avance del algoritmo[116]. Notar que $V(t_e)$ se calcula usando los pesos actuales. **Cuidado.** Puede ser que por casualidad $V(t_e) = F_{\max}^{tmp}$ y aunque nunca $V(t_e)$ debería ser mayor a $F_{\max}^{tmp}$ (o habría un error), la computadora, por errores de "precisión", podría tomar a $V(t_e)$ como mayor y enviar un mensaje de error, parando el programa (indebidamente). Esto se debe considerar en la programación del algoritmo[117]. Se va al paso E.

[114] Se usa la variable GPS_idle existente. GPS_idle = 1 quiere decir que el sistema sí estaba ocioso (ver WFQenqueue(Packet *P, int queueid) de dsREDQueue en el archivo dsred.cc).
[115] Se asignan otras variables para control del algoritmo. Sobre los valores iniciales ver pie de página 114.
[116] Ver inserción clave "t", en dsREDQueue::WFQenqueue, en el archivo dsred.cc.
[117] Esto se programó en el algoritmo en el simulador ns.

Viene de C y D.	E	Parte existente en el algoritmo GPS original. La siguiente formulación funciona cuando SÍ se está o cuando NO se está en un proceso de cambio de pesos. Se calculan los tiempos de inicio y de terminación de atención del paquete (para efectos de explicación se considera que llega el paquete #*k* a la cola #*i*, de tamaño L_i^k *bit*). Se evalúan S_i^k y F_i^k (notar que F_i^{k-1} y $V(t_e)$ son iguales a cero si el sistema estaba ocioso). Se calcula[118]: $S_i^k = \max\left(F_i^{k-1}, V(t_e)\right)$ $F_i^k = S_i^k + \frac{L_i^k}{\phi_i R}$ Para el caso de sí estar en un proceso de cambio de pesos, evidentemente $F_i^{k-1} \leq F_{\max}^{tmp}$, pero hay que ver, todavía, si F_i^k es mayor o menor a $F_{\max}^{tmp}$. Se va al paso F.

Viene de E.	F	Parte nueva en el algoritmo[119]. ¿Se está o no en proceso de cambio de pesos? (si no se está en este proceso NO se va a cambiar el procedimiento existente). Se va al paso G ó I, dependiendo.

Viene de interrogante F.	G	Parte nueva en el algoritmo. En caso de que SÍ se está en un proceso de cambio de pesos se pregunta: ¿Se cumple que $F_i^k > F_{\max}^{tmp}$? Se va al paso H ó I, dependiendo.

[118] En la rutina dsREDQueue::WFQenqueue, en el archivo dsred.cc, la fórmula original que se tenía se modifica para emplear, en lugar de los pesos originales queueWeight[ ], los pesos presentes queueWeightpr[ ]. Ver código de modificación "f".

[119] Ver inserción clave "k" en WFQenqueue, en el archivo dsred.cc, para las líneas nuevas en el paso actual y en los dos subsiguientes de este algoritmo.

Viene de interrogante G.	H	Parte nueva en el algoritmo. En caso de que $F_i^k > F_{\max}^{tmp}$, entonces no se acepta el valor de F_i^k. Este se debe recalcular. $Fracción\left(L_i^k\right)_{\left[S_i^k, F_{\max}^{tmp}\right)} = \left(F_{\max}^{tmp} - S_i^k\right)\phi_i R$ $Fracción\left(L_i^k\right)_{\left[F_{\max}^{tmp}, F_i^k\right)} = L_i^k - Fracción\left(L_i^k\right)_{\left[S_i^k, F_{\max}^{tmp}\right)}$ El nuevo peso de la cola *i* entre los tiempos virtuales $F_{\max}^{tmp}$ y F_i^k es ϕ_i^{New}. $F_i^k = F_{\max}^{tmp} + \frac{Fracción\left(L_i^k\right)_{\left[F_{\max}^{tmp}, F_i^k\right)}}{\phi_i^{New} R}$ Combinando, se obtiene la ecuación completa: $F_i^k = F_{\max}^{tmp} + \frac{L_i^k - \left(F_{\max}^{tmp} - S_i^k\right)\phi_i R}{\phi_i^{New} R}$. Se va al paso I.

Viene de interrogante F, de interrogante G, y de H.	I	Parte existente en el algoritmo GPS original. Se actualiza el conjunto *B*, y la sumatoria $\sum_B \phi_j$. Esto se hace con los pesos actuales, y no con los nuevos[120]. El conjunto *B* tiene validez a partir del evento de llegada que acaba de suceder. Se actualiza $F_{\max}$ (que necesariamente es mayor a $F_{\max}^{tmp}$). Se va al paso J.

Viene de I.	J	Parte existente en el algoritmo GPS[121]. Se obtiene $F_{\min}$ y se recalcula $Next(t)$ [122]. Aun en el caso de estar en un proceso de cambio de pesos, ya que $F_{\min} \leq F_{\max}^{tmp}$, entonces es válido hacer este cálculo con los pesos actuales. $Next(t) = t_e + \left(F_{\min} - V(t_e)\right) \sum_{j \in B\left[t_e, Next(t)\right)} \phi_j$ Se va al paso K.

[120] Se pone la información del evento de llegada en las tablas GPS y PGPS. Se guarda el par (número de cola, F_j^k), ordenando según el tiempo virtual de finalización. Con estas tablas y con la rutina get_key_max insertada nueva, para obtener el tiempo de atención virtual máximo que se lleva, se puede obtener fácilmente el valor de $F_{\max}$ (ver inserción "c" en el archivo wfq-list.h).

[121] En WFQscheduleGPS en el archivo dsred.cc. *F*min se obtiene con una consulta a la lista GPS.

[122] Se actualiza *Next*(*t*) en el Scheduler, recordando que dicho tiempo es único en su tipo en el Scheduler.

Viene de J.	K	Parte nueva en el algoritmo[123]. ¿Se está o no en proceso de cambio de pesos? Va a paso L o N, dependiendo.

Viene de interrogante K.	L	Parte nueva en el algoritmo. En el caso de NO estar en un proceso de cambio de pesos, se evalúa el criterio de cambio de pesos[124] (ya con la nueva llegada del paquete). Se debe hacer la evaluación aun cuando el sistema estaba ocioso. ¿El criterio de cambio de pesos indica que se deben cambiar los pesos? Se va a paso M o N, dependiendo.

Viene de la interrogante L.	M	Parte nueva en el algoritmo. En el caso de que el criterio de cambio de pesos indicase que SÍ se deben cambiar los pesos: ○ Se registra que se está en un proceso de cambio de pesos (para el próximo evento). ○ Se evalúan los nuevos pesos $\phi_j(t_e)$, los que se llaman ϕ_j^{New}, $j=1,...,N$ (a los pesos presentes se les sigue llamando ϕ_j, $j=1,...,N$). ○ Se asigna $F_{\max}^{tmp} = F_{\max}$. Se va a N.

Viene de interrogante K, de interrogante L, y de M.	N	Se queda en ESPERA DE SIGUIENTE EVENTO.

E.2.2 Caso de la Salida de un Paquete. Sistema de Despacho GPS

INICIO.	A'	Parte existente en el algoritmo GPS. En un tiempo dado t_e se da un evento de salida de paquete[125]. El sistema no puede haber estado ocioso. Se va a B'.

[123] Ver inserción "l" en WFQenqueue(Packet *p, int queueid) en el archivo dsred.cc, para las líneas nuevas en el paso actual y en los dos subsiguientes en este algoritmo.

[124] Con el Método STP el criterio de cambio de pesos es precisamente cuando se ha pasado de la finalización de 1 *s* al siguiente, en el trascurso del tiempo (que en una correspondería al tiempo de corrida del simulador).

[125] El Scheduler llega a un evento de salida del despachador GPS. Existen otros eventos de salida de paquete para el despachador PGPS (manejo por paquete) en el Scheduler, que llevan un tiempo muy cercano pero no igual al del despachador GPS (manejo por fluido). El llevar las salidas en el despachador GPS sirve para actualizar los tiempos virtuales. El despachador PGPS no utiliza los tiempos virtuales de forma directa. Para los eventos relativos al despachador GPS el Scheduler corre la instrucción this->WFQdequeueGPS(e) de dsREDQueue, y para los eventos relativos al despachador PGPS el Scheduler corre la instrucción this-> deque( ) de dsREDQueue, que

Viene de A'.	B'	Parte existente en el algoritmo GPS[126]. Se calcula $V(t_e)$: $V(t_e) = V(t_{e-1}) + \frac{t_e - t_{e-1}}{\sum_{\forall \ell \ni Cola_\ell(t_{e-1}) \in B[t_{e-1}, t_e)} \phi_\ell}$ El valor de $V(t_e)$ siempre se acepta, por lo siguiente: si NO se está en proceso de cambio de pesos el valor de $V(t_e)$ se acepta sin problema. Si SÍ se está en proceso de cambio de pesos el valor de $V(t_e)$ no podría ser mayor a $F_{\max}^{tmp}$, pues de lo contrario, en algún evento anterior, se debía haber detectado esto (caso en que $V(t_e)$ hubiese sido igual a $F_{\max}^{tmp}$), y se habría anulado el estado de SÍ estar en cambio de pesos. Por lo anterior, es válido usar los pesos presentes en la anterior ecuación (en este caso $V(t_e) \leq F_{\max}^{tmp}$). Se actualiza, en la lista de eventos, el tiempo real del próximo paquete a salir y su número de cola. Va a paso C'.

Viene de B'.	C'	Parte existente en el algoritmo GPS. Justamente en el tiempo real t_e se acaba de ir un paquete. Se actualiza entonces la suma de pesos, $\sum_{j \in B[t_e, t_{e+1})} \phi_j$, a partir de t_e hasta el siguiente evento, t_{e+1} (no se sabe aún el valor de t_{e+1}). ¿El sistema se torna ocioso? Esto se puede probar revisando si $V(t_e) = F_{\max}$, o si $\sum_{j \in B[t_e, t_{e+1})} \phi_j = 0$, recordando que $B[t_e, t_{e+1})$ es el conjunto de colas que tienen paquetes, al menos uno, en el lapso de tiempo real $[t_e, t_{e+1})$. Se va a D' o E', dependiendo si el sistema se torna, o no, ocioso.

incluye la línea para correr la rutina selectQueueToDeque, que a su vez incluye la línea para correr la rutina WFQdequeuePGPS( ). Ver archivo dsred.cc. Recordar que la razón para llevar un despachador GPS, operando en paralelo con el despachador PGPS, es que con el despachador GPS se lleva la cuenta de los tiempos virtuales, y de su relación con los tiempos reales.

[126] En WFQdequeueGPS(Event *e), de dsREDQueue. Se designan dichas líneas con la clave "d" (ver archivo dsred.cc). Event *e es un objeto del tipo de objetos ordenados por el Scheduler (ver archivo Scheduler.cc).

Viene de interrogante C'.	D'	Parte existente en el algoritmo GPS[127] y parte nueva[128]. En caso de que el sistema SÍ se torne ocioso, NO SE TOCAN los pesos actuales ni los pesos nuevos. Se asigna $F_{\max}^{tmp} = 0$ y $F_{\max} = 0$, y se desactiva la indicación de que el sistema está en proceso de cambio de pesos (si lo estaba). Todos los tiempos virtuales se hacen cero. Se concluye la rutina de salida de paquete en el sistema de despacho GPS. Se va a paso 0 CONDICION DE ESPERA INICIAL.

Viene de interrogante C'.	E'	Parte nueva en el algoritmo[129]. Caso de que el sistema NO se torne ocioso. Se pregunta si SÍ se está, o NO, en proceso de cambio de pesos, y si $V(t_e) = F_{\max}^{tmp}$. Esto es un paso muy crucial. Considerar que en el programa $V(t_e)$ puede ser muy cercano a $F_{\max}^{tmp}$, sin ser igual. La computadora podría considerar iguales ambas cantidades si se usa un valor de tolerancia. Si se cumple que SÍ se está en un proceso de cambio de pesos y que $V(t_e) = F_{\max}^{tmp}$, entonces se va a F'. Se va a G' en otro caso.

Viene de Interrogante E'.	F'	Parte nueva en el algoritmo. Caso de que SÍ se está en un proceso de cambio de pesos y que $V(t_e) = F_{\max}^{tmp}$, entonces se sabe que **se acabó** el lapso de cambio de pesos (lo que quiere decir que los pesos actuales o presentes ya se pueden sustituir con los nuevos pesos) así que: • Se actualizan los pesos actuales ϕ_j, $j = 1,...,N$, con los pesos nuevos $\phi_j(t_e) = \phi_j^{New}$, $j = 1,...,N$. • Se prende la indicación de NO estar en proceso de cambio de pesos. • No importa cómo quede $F_{\max}^{tmp}$, éste se hace igual a cero por razones de orden. NO se toca $F_{\max}$. Se debe actualizar la suma de pesos que se tenía: $\sum_{j \in B[t_{e-1}, t_e)} \phi_j$, con la suma hecha ahora con los nuevos pesos que hay a partir del tiempo real presente t_e. Se va a G'.

[127] En WFQdequeueGPS(Event *e) (ver archivo dsred.cc).
[128] Ver inserción con la clave "z" en WFQdequeueGPS(Event *e) de dsREDQueue (ver archivo dsred.cc).
[129] Todos los pasos siguientes se ponen en WFQdequeueGPS(Event *e). Ver inserción "e" en archivo dsred.cc.

Viene de interrogante E' y de F'.	G'	Parte nueva en el algoritmo. Al llegar a este paso se tienen posibles condiciones: 1- Se ha llegado aquí desde F' (de ahí se sale sin estar en un proceso de cambio de pesos). 2- Se ha llegado de E' y aunque SÍ se está en un proceso de cambio de pesos, $V(t_e) < F_{\max}^{tmp}$. 3- Se ha llegado de E' y NO se está en un proceso de cambio de pesos (no importa entonces comparar los valores de $V(t_e)$ y $F_{\max}^{tmp}$). Entonces se pregunta si: SI o NO se está en un proceso de cambio de pesos. En caso de SI estar en un proceso de cambio de pesos se va al paso I'. En caso de NO estar en un proceso de cambio de pesos se evalúa el criterio de cambio de pesos (ya considerando que salió el paquete en t_e). Se va al paso H' o I', dependiendo el resultado del criterio.

Viene de segunda interrogante G'.	H'	Parte nueva en el algoritmo. Si el criterio de cambio de pesos indica que SÍ se debe cambiar de pesos, entonces: • Se registra que SÍ se está en un proceso de cambio de pesos (para el próximo evento) • Se evalúan los nuevos pesos $\phi_i^{New} = \phi_i(t_e)$, $i = 1,...,N$ .Los nuevos pesos, anteriores, habían quedado ya asignados como los pesos presentes, con los cuales se trabaja, en el paso F'. • Se asigna $F_{\max}^{tmp} = F_{\max}$.

Viene de H'.	I'	Parte existente en el algoritmo GPS[130]. Se obtiene el tiempo virtual de ocurrencia, $F_{\min}$, del evento de salida de paquete futuro más próximo, y se recalcula el tiempo real correspondiente a dicho evento, $Next(t)$, el cual es igual al tiempo actual, t_e, más la adición de un "retraso" o "delay": $$Next(t) = t_e + \left(F_{\min} - V(t_e)\right) \sum_{j \in B\left[t_e, Next(t)\right)} \phi_j$$ Se actualiza dicho tiempo en el Scheduler del simulador (recordando que dicho tiempo es único en su tipo en dicho Scheduler). **Fijarse** que el conjunto de pesos $B\left[t_e, Next(t)\right)$ pudo haberse actualizado si resultó, en la reciente salida de paquete, en el tiempo presente t_e, que $F_{\max}^{tmp} = V(t_e)$. Esto sucedió en el paso F'.

Viene de I'.	J'	Se queda en ESPERA DE SIGUIENTE EVENTO.

[130] En WFQscheduleGPS( ) (ver archivo dsred.cc modificado por el autor de esta Tesis).

Apéndice F. Experimentos Preliminares para llegar al Método STP

Evolución Hacia la Obtención del Método STP

En este apéndice se presentan diversas propuestas planteadas y evaluadas en el curso de la investigación para esta Tesis, de las que se evolucionó hasta llegar a plantear el Método STP.

Desde la primera propuesta se planteaban ya las ideas de operación pretendidas: que en la interfaz de salida, de algún nodo, en donde se planteara la solución, hubiese una cola por cada interfaz de entrada del nodo. Se buscaba que el peso de atención de cada cola se ajuste al tamaño de su tráfico, de forma lenta (no inmediata), observando la longitud de la cola. Se pretendía que no fuese necesario el marcar los paquetes, y que los nodos trabajasen sin la intervención de un administrador central.

Primera Propuesta. Cambio de Peso Directamente Proporcional a la Longitud Relativa de la Cola

En esta propuesta se considera al número *N* como el número de colas que hay. Los pesos de las colas se llaman “pesos vigentes” o “pesos presentes”, y se representan con ϕ_k, $k = 1, ..., N$. Estos pesos dependen de los pesos iniciales fijos que se representan con ϕ'_k, $k = 1, ..., N$. Al inicio de la operación del despachador los pesos vigentes son iguales a los pesos iniciales.

En esta propuesta se considera que cada vez que se da un evento, ya sea de llegada o de salida de un paquete, con relación a alguna de las colas, en el tiempo $t = t_e$ (tiempo del evento), se realiza la evaluación del promedio de longitud de las colas (ver ecuación (9)), y se evalúa un criterio sobre la necesidad de hacer un cambio de pesos. En caso de que la evaluación sea positiva, es decir, que se indique que se debe cambiar de pesos a las colas, se realizaría un cambio de pesos a todas las colas a la vez. Esta evaluación se pensó con el fin de disminuir la cantidad de cambios de pesos en el tiempo. La misma planteaba “disparar” el cambio de peso una vez que la diferencia de los tamaños promedio de la cola de mayor tamaño y de la cola de menor tamaño fuese mayor a una función predefinida dependiente del número de colas que hubiese. La formulación de esta evaluación no se presenta aquí ya que se empleó solamente en esta Primera Propuesta de Cambio de Pesos, quedando, posteriormente, desechada.

En esta propuesta el peso de una cola se planteó para cambiar conforme la longitud relativa de dicha cola (con relación a la suma de las longitudes de las demás colas), esto es:

$$\phi_i\ t\ = \frac{\overline{Q_i\ t}}{\sum_{k=1}^{N} \overline{Q_k\ t}\ \Big/ N}\phi_i', \quad i = 1, \ldots, N \tag{59}$$

En la ecuación (5), repetida aquí en la ecuación (59), $\phi_i\ t$ representa el peso propuesto para la cola i, y $\overline{Q_i\ t}$ representa la longitud total promedio de la cola i, ambos tomados en el tiempo *t*. Se tomaron previsiones para el caso en el que el denominador fuese cero[131].

Para esta propuesta, y para todas las que le siguen, se calculó el promedio de la longitud de una cola en cada evento de llegada o de salida de un paquete, en la cola. Esta longitud promedio se calculó con el uso de la ecuación (9) (como se hace en [78]), donde w_q es el parámetro para promediar, $q(t_e)$ es la longitud instantánea de la cola en el tiempo del evento, $t = t_e$,

$$\overline{Q_i(t_e)} = (1 - w_q)\,\overline{Q_i(t_{e-1})} + w_q\, q(t_e) \tag{60}$$

Esta ecuación actúa como un filtro paso bajas. Mientras más pequeño sea el valor de w_q, más suave es la salida. Empíricamente el valor para w_q se fija en 0.002 [78], para hacer frente al comportamiento de cambios abruptos de la longitud instantánea de la cola.

El factor $\overline{Q_i\ t}\ \Big/ \left[\sum_{k=1}^{N} \overline{Q_k\ t}\ \Big/ N\right]$ es la relación de la longitud de la cola i entre el promedio de la longitud de las colas, cuyo valor es positivo, y puede ser menor, igual o mayor que 1, dependiendo de la relación entre dichas longitudes. Esta comparación haría crecer o disminuir al peso $\phi_i\ t_e$ con relación al peso inicial ϕ_i'. Una vez obtenidos los pesos nuevos de las colas, éstos podrían escalarse para que sumaran 1.

Un inconveniente de esta propuesta era que en cada llegada o salida de paquete se tenía, también, la necesidad de evaluar el criterio mencionado. Otro inconveniente era que cada nuevo peso tenía una expresión que variaba alrededor de un peso inicial, que forzaba a que dicho peso inicial fuese como un ancla alrededor de la cual variaba el valor del peso. Aun así, para ampliar las posibilidades de esta propuesta se modificó la ecuación (5) para controlar más la tasa de cambio en los pesos, añadiendo un factor *m*. Cuando *m* fuese igual a 1 la ecuación (61) se convertiría en la ecuación (5). El factor *m* se podía considerar como la pendiente

[131] Además, los pesos propuestos reemplazarían a los pesos presentes cuando fuese apropiado, según la operación del despachador usado. Un despachador puede requerir del paso de alguna cantidad de eventos de salida de paquetes, a partir de que se haya indicado, con algún criterio, que se debe hacerse un cambio de pesos.

de una recta que vincula a dos variables, $\phi_i\ t_e$ y $\overline{Q_i\ t_e}\Big/\left[\sum_{k=1}^{N}\overline{Q_k\ t_e}\ \Big/N\right]$, y al igual que el caso de la ecuación **(5)**, cuando esta segunda variable es igual a 1, entonces la primera variable se hace igual a ϕ_i' .

$$\phi_i\ t_e\ = m\phi_i'\left(\frac{\overline{Q_i\ t_e}}{\sum_{k=1}^{N}\overline{Q_k\ t_e}\ \Big/N}-1\right)+\phi_i',\quad i=1,...,N \tag{61}$$

Evaluación Experimental de la Primera Propuesta de Cambio de Pesos

La evaluación para esta Primera Propuesta de Cambio de Pesos se realizó de tal forma que el algoritmo de la misma operase en la interfaz de salida de un solo nodo (ver Figura 42). Este es el nodo c_1 de la red empleada para la experimentación. En ese nodo, más específicamente en su interfaz de salida, se tenía la interferencia de rutas. Ahí se incorporó la posibilidad de modificar el peso de las colas, dinámicamente, según lo indicaba esta propuesta.

Se presentan resultados de la aplicación de esta primera propuesta y se comparan con la solución convencional de tener una cola por cada interfaz de salida en los nodos, y con la solución de tener dos colas, con pesos fijos cada una.

La imagen de la red empleada[132] se da en la Figura 42 donde el único nodo central de la red es el llamado c_1. Los nodos de frontera (de ingreso y egreso) de la red son e_1, e_2 y e_3. Los nodos de fuente son s_1, s_2 y s_3, y los nodos destino son d_1 y d_3. Hay tres rutas de flujo de datos en el dominio, denominadas con: s_1-d_1, s_2-d_1, y s_3-d_3. Las siglas de la ruta s_1-d_1 indican la entrada de flujos por el nodo fuente (Source) s_1, pasando una ruta definida (que se representa en la Figura 42 con una línea rayada), hasta llegar al nodo destino d_1. En el nodo e_1 se cruzan las rutas s_1-d_1 y s_2-d_1 (en su interfaz de salida yendo hacia el nodo c_1).

[132] Sobre la Figura 5, en el Apéndice J se dan más detalles con relación a la configuración de los parámetros en el simulador, para implantar el comportamiento en que una interfaz de salida tenga una cola por cada interfaz de entrada.

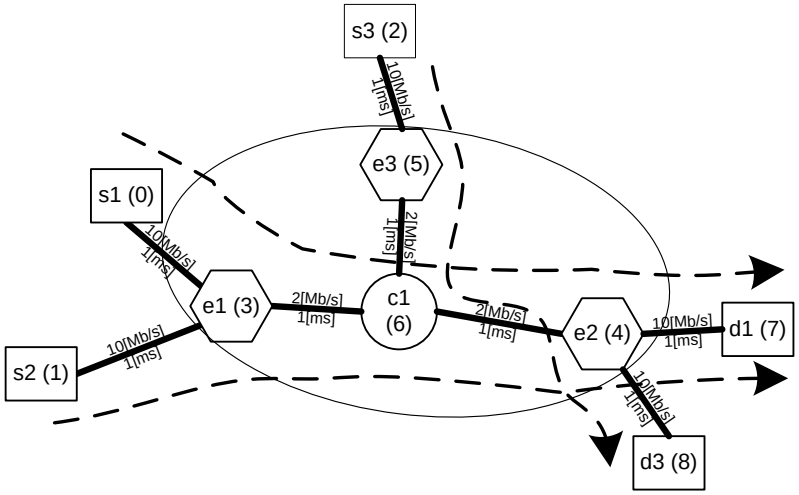


Figura 42. Topología de la red empleada para la evaluación, por simulación, de la Primera Propuesta de Cambio de Pesos.

En esta parte de este capítulo, los resultados de la evaluación de las soluciones utilizan un indicador llamado *Envolvente de Servicio Mínimo* medido [90] [16] [91], que indica el tiempo máximo para servir (para que crucen por una ruta) una cantidad dada de bits, lo que era equivalente a la mínima cantidad de bits servidos en un tiempo dado. El uso de este envolvente, como indicador del retraso del tráfico en las rutas, se desechó en las siguientes propuestas de cambio de peso, de esta Tesis, y se utilizó otra forma de evaluar el retraso.

Para la evaluación de esta Primera Propuesta de Cambio de Pesos se incluyó, en los resultados, la observación de los pesos en las dos colas del nodo c_1.

En esta evaluación específica cada experimento resulta de una sola simulación. Para obtener los resultados de cada simulación se analizó el archivo de trazas (“Trace File”) obtenido al final de la simulación.

Las condiciones para los experimentos fueron las siguientes:

- En cada experimento cada flujo sigue la misma ruta.
- En los experimentos, el tráfico inicial en cada ruta es lo suficientemente grande como para que el incremento de retraso en las rutas, causado por el incremento de tráfico en la ruta que cruza se puede observar en los resultados.
- En la evaluación se usan colas con suficiente capacidad para que los paquetes perdidos por congestión en las mismas no rebasaren el 2% de los paquetes cursados.
- El tráfico de las rutas s_1-d_1 y s_2-d_1, que se cruzan en el nodo e_1, permanece constante en el tiempo experimental, en un total que no satura la salida del dicho nodo. Este cruce

no se considera en la evaluación de la solución con la Primera Propuesta de Cambio de Pesos. En la interfaz de salida del nodo e_1, se utiliza una sola cola.

- En el nodo central c_1 se cruza la combinación de las rutas s_1-d_1 y s_2-d_1 con la ruta s_3-d_3, y en este nodo sí se utiliza la operación de esta propuesta de cambio de pesos, cuando se evalúa la solución empleando dicha propuesta, y por consiguiente, sí se utilizan dos colas en la interfaz de salida de este nodo (yendo hacia el nodo e_2), siendo una de estas colas para el tráfico combinado de las rutas s_1-d_1 y s_2-d_1, y la otra cola para el tráfico de la ruta s_3-d_3.
- En todos los experimentos de esta Tesis, para todas las interfaces en que hay una sola cola de salida, se maneja, en el simulador, un tipo de cola llamada “Droptail”, que significa que cuando la cola se ve desbordada se tiran los paquetes que recién llegan a la cola.
- En los experimentos, el flujo se crea a partir del tráfico generado por las "Fuentes de tráfico" ("fuentes"). Una fuente es un lugar donde el tráfico se genera, en ns-2. Una fuente está conectada a un "nodo fuente".
- Al principio de cada experimento, para las soluciones que utilizan dos colas en la interfaz de salida del nodo c_1, el peso de las colas es igual (1 y 1). En el caso de la solución que usa pesos fijos, estos pesos se conservan en todo el experimento. En el caso de la solución con la Primera Propuesta de Cambio de Pesos, los pesos podrían variar a lo largo del tiempo experimental. Estos pesos iniciales implican, para esta solución, que antes del inicio del experimento se han tenido tráficos, en promedio iguales, alimentando a las dos colas.
- La ruta s_1-d_1 se mantiene con 8 fuentes Pareto (PAR) *On*/*Off*, de 128 *Kb/s* (en los intervalos ON), con parámetro θ = 1.7 (varianza infinita –ver Apéndice G). Este tráfico se mantiene en la duración de los experimentos. La ruta s_2-d_2 tiene 1 fuente de tráfico constante (CBR[133]) de 128 *Kb/s*, en la duración de los experimentos. La anterior implica que el tráfico combinado de estas dos rutas, se mantiene en promedio igual.
- La ruta s_3-d_3 tiene *N* fuentes Pareto *On*/*Off*, de iguales características que las fuentes de la ruta s_1-d_1., En cada experimento el valor de *N* es constante, es decir, en cada experimento el número de fuentes en las diversas rutas se mantiene constante. De un experimento al siguiente, el valor de *N* es mayor. Los valores de *N* son: de 6, 8, 10, 12, 14, 16, 18 y 20 fuentes. Con esto se evaluaría el impacto del tráfico de la ruta s_3-d_3 en su cruce con la ruta s_1-d_1.
- Todas las fuentes usan paquetes de 40 *Byte*.

[133] Del inglés, Constant Bit Rate.

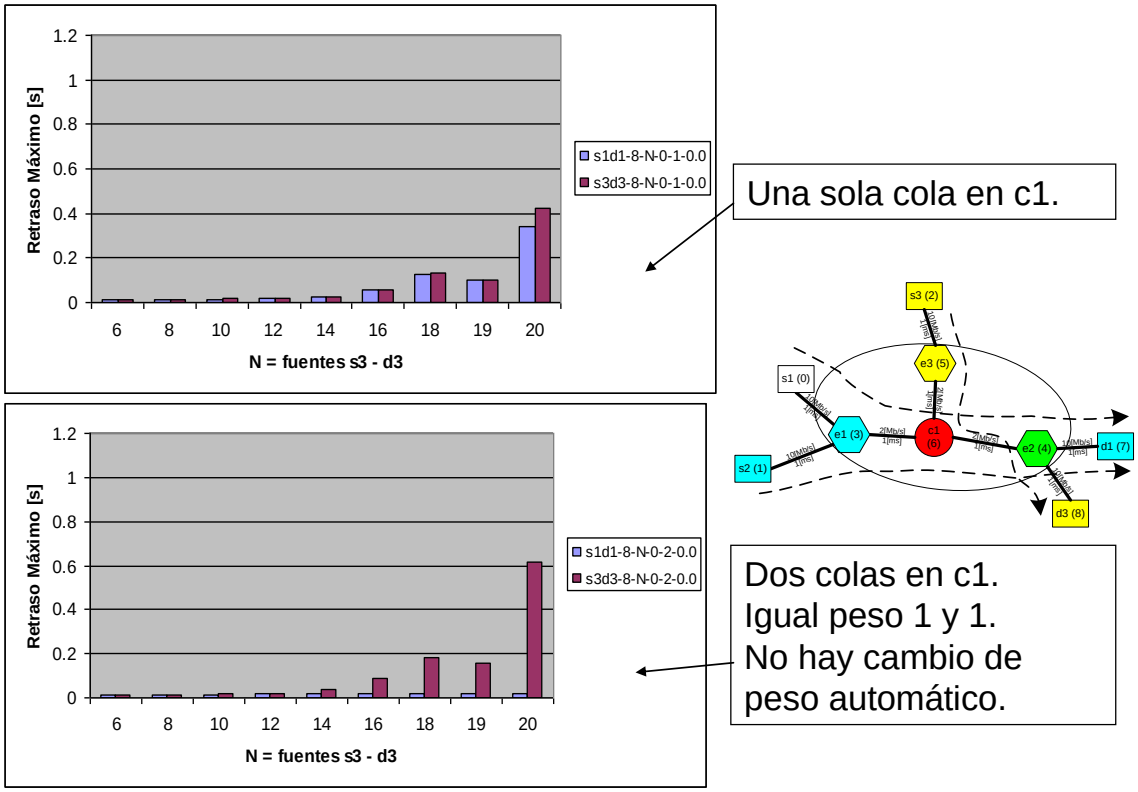
- Cada experimento dura 30 *s*. Este envolvente se calculó en toda esta duración.

En el Apéndice H se observan detalles de la implantación experimental de la Figura 42.

- En resumen, las tres soluciones que se evalúan son:

  - Solución 1. Donde se tiene una sola cola en la interfaz de salida del nodo c_1, y en todas las demás interfaces.
  - Solución 2. Donde se tienen dos colas en la interfaz de salida del nodo c_1, una para el tráfico de la ruta s_3-d_3 y otra para el tráfico que incluye al tráfico de la ruta s_1-d_1. El peso de cada cola es constante a lo largo del tiempo experimental, con peso igual 1 y 1 (0.5 y 0.5 si se escalan los pesos).
  - Solución 3. Caso similar al segundo, pero el peso de las colas varía con el tiempo experimental, de una forma acorde a la propuesta de cambios de pesos.

La Figura 43 muestra los resultados del valor obtenido en cada Envolvente de Servicio Mínimo Medido, en las rutas s_1-d_1 y s_3-d_3, para diversos valores de *N*, comparando los resultados de las dos primeras soluciones.



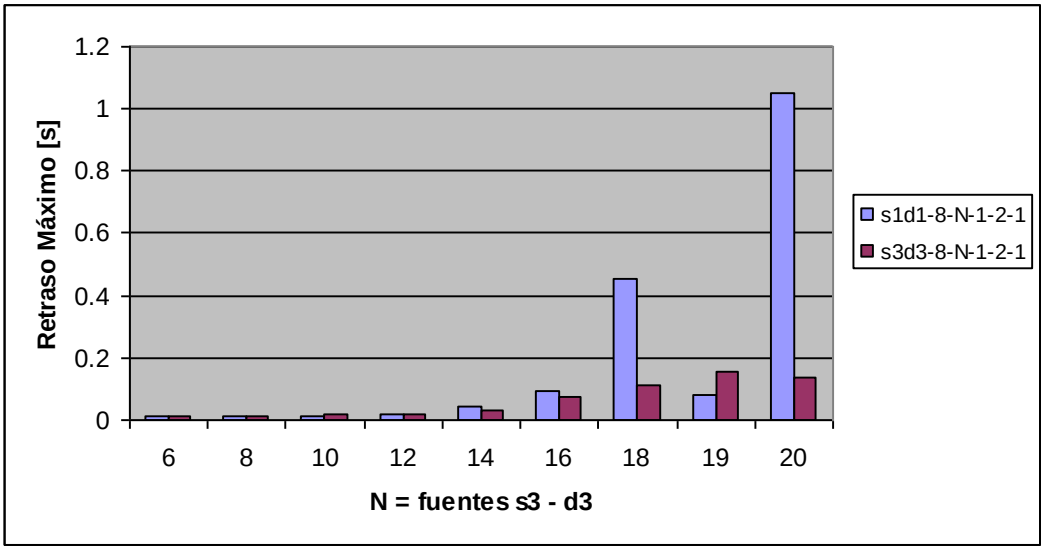
Retrasos máximos en las rutas, según los Envolventes de Servicio Medidos, y según varía el número de fuentes en la ruta s3d3.

Figura 43. Comparativo de los resultados de las dos primeras soluciones. Retraso en las rutas en términos del Envolvente de Servicio Medido, contra el número de fuentes en la ruta s_3-d_3. La etiqueta **s1d1-8-N0-2-0.0** indica el valor del envolvente de la ruta s_1-d_1, con 8 fuentes Pareto *On*/*Off* de 128 *Kb/s*, donde **N0** indica que no hay cambio de pesos, con 2 colas en la interfaz de salida del nodo c_1, y con un valor *m* (pendiente ecuación (61)) de 0.0 (al no haber cambio de pesos la pendiente es 0.0). Las otras etiquetas tienen significados similares.

Se puede observar, en la primera solución (imagen superior de la Figura 43 con una sola cola en la interfaz de salida del nodo c_1), la forma en que el crecimiento del tráfico en la ruta s_3-d_3 afecta el Envolvente de Servicio Mínimo Medido del tráfico de la ruta s_1-d_1. La segunda solución (imagen inferior de la Figura 43 con dos colas de peso constante en la interfaz de salida del nodo c_1) es un caso de reservación de ancho de banda para las rutas combinadas s_1-d_1 y s_2-d_1. Se observa que el tráfico de la ruta s_1-d_1 queda protegido por completo contra el aumento del tráfico de la ruta s_3-d_3, la cual, se ve afectada fuertemente por su mismo aumento de tráfico.

La Figura 44 muestra el valor máximo en cada Envolvente de Servicio Mínimo Medido en las rutas s_1-d_1 y s_3-d_3, cuando se utiliza la Primera Propuesta de Cambio de Pesos, para diversos

valores de *N* (número de fuentes en la ruta s_3-d_3). Se puede observar que no es clara la protección brindada por este método. Al contrario, dado el comportamiento de ráfagas del tráfico en las rutas, con altos valores de *N* inclusive llega a ser contraproducente, para la ruta s_1-d_1, la aplicación de esta Primera Propuesta de Cambio de Pesos.



Retrasos máximos en las rutas, según los Envolventes de Servicio Medidos, y según varía el número de fuentes en la ruta s3d3.

Figura 44. Resultados de la tercera solución, que es la Primera Propuesta de Cambio de Pesos (dos colas en la interfaz de salida del nodo c_1 con pesos variables).
El retraso en las rutas se da en términos del Envolvente de Servicio Medido, contra el número de fuentes en la ruta s_3-d_3. El significado de las etiquetas es similar a los de la Figura 43.

La Figura 45 es una gráfica comparativa de los resultados de las tres soluciones, que muestra el Envolvente de Servicio Mínimo Medido, para la ruta s_1-d_1, donde el valor de *m* se expresa como “la pendiente”.

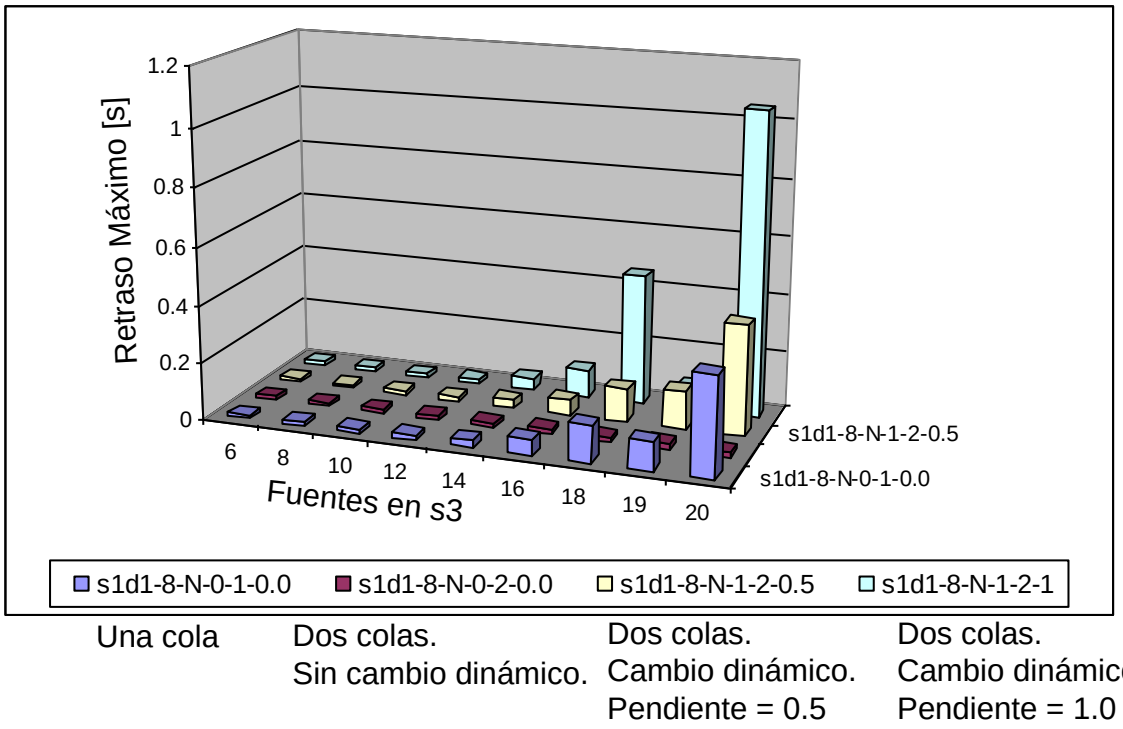


Figura 45. Gráfica comparativa de los resultados de las tres soluciones, mostrando el Envolvente de Servicio Mínimo Medido, para la ruta s_1-d_1, contra N (número de fuentes en la ruta s_3-d_3).

Segunda Propuesta de Cambio de Pesos. Cambio de Peso Suavizado Tendiente a la Longitud Relativa de la Cola

Los resultados de la Primera Propuesta de Cambio de Pesos no indicaban cómo se iban cambiando los pesos, con el tiempo. Asimismo, un inconveniente de esa solución era que cada nuevo peso tenía una expresión que variaba alrededor de un peso inicial, que forzaba a que dicho peso inicial fuese un forma de ancla alrededor del cual variable el peso de operación. No fue clara la ventaja de la aplicación de la Primera Propuesta. Entonces, se buscó una propuesta alternativa, derivando, así, una Segunda Propuesta de Cambio de Pesos.

En la Segunda Propuesta de Cambio de Pesos, los nuevos pesos se calculan como sigue:

$$\phi_i(t+\tau) = (1-\alpha)\phi_i(t) + \alpha \frac{\overline{Q_i(t+\tau)}\,\phi_i'}{\sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'}, \quad i \in \{1, ..., N\}, 0 < \alpha < 1 \tag{62}$$

Donde siempre se debe cumplir que $\sum_{k=1}^{N} \phi_k(t) = 1$. En esta expresión se puede observar que haciendo la sumatoria $\sum_{i=1}^{N} \phi_i(t+\tau)$ el valor de la misma resulta igual a 1.

Esta propuesta cambia con relación a la anterior porque ahora cada nuevo peso depende, en parte, del valor del nuevo peso obtenido anteriormente.

La ecuación (6) se puede despejar para observar que el incremento del peso calculado, $\phi_i(t+\tau) - \phi_i(t)$ depende del valor de α y de la diferencia de los valores $\phi_i(t)$ y $\overline{Q_i(t+\tau)}\,\phi_i' \Big/ \sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'$.

$$\phi_i(t+\tau) - \phi_i(t) = \alpha \left[\frac{\overline{Q_i(t+\tau)}\,\phi_i'}{\sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'} - \phi_i(t) \right] \tag{63}$$

En esta propuesta se siguen calculando las longitudes promedio de las colas en cada evento de llegada o salida de paquete, pero la evaluación de los pesos ahora se hace al final de cada periodo de duración τ, donde τ puede ser igual a 1 *s*. En esta propuesta siempre, al final de cada lapso de tiempo de duración τ, se realiza el cambio de pesos[134]. Se observa que cuando el valor de α es muy cercano a 0, el cambio de pesos es menos sensible al cambio de longitudes promedio de las colas.

Evaluación Experimental de la Segunda Propuesta de Cambio de Pesos.

Las condiciones experimentales para la evaluación de esta propuesta son similares a los presentados en la Primera Propuesta de Cambio de Pesos, usando la misma red, pero los experimentos se hacen solamente para el caso en donde en la salida del nodo c_1 hay dos colas que cambian de pesos, en donde se aplica la ecuación (6) para evaluar los nuevos pesos. Ahora

[134] Aquí, el criterio que se había empleado en la anterior propuesta, para evaluar si se hacía o no un cambio de pesos, se sustituye simplemente por el hecho de que hubiese pasado un tiempo de duración τ. Aun rige la regla de que los pesos propuestos remplazarían a los pesos presentes, cuando fuese apropiado, según la operación del despachador.

solamente las rutas s_1-d_1 y s_3-d_3 tienen tráfico, así que se puede decir que una cola, de la interfaz de salida del nodo c_1 se asocia con la ruta s_1-d_1, y la otra cola con la ruta s_3-d_3.

En la evaluación de esta Segunda Propuesta de Cambio de Pesos se obtuvieron resultados del retraso de los paquetes en las rutas, en términos del Envolvente de Servicio Mínimo Medido. Los resultados de estas pruebas se omiten pues en los mismos tampoco eran evidentes las mejoras obtenidas; sin embargo, los resultados relativos los pesos de las colas, a través del tiempo experimental, en la evaluación de la solución con esta Segunda Propuesta de Cambio de Pesos, sí resultaron interesantes por lo que se muestran en este capítulo.

También se muestran resultados de los tamaños promedio de las colas, a través del tiempo experimental.

Entonces, aquí se presentan diversas pruebas a la Segunda Propuesta de Cambio de Pesos, para observar, exclusivamente, la variación de los pesos y los tamaños de las colas, con el tiempo experimental. Cada prueba consiste de un solo experimento. En cada experimento el número de fuentes en las diversas rutas se mantiene constante. El resultado de cada experimento resulta de una simulación. De una prueba a la otra podía cambiar el número inicial de fuentes. En las cuatro primeras pruebas el valor de α es igual a 0.001, y en la última prueba dicho valor cambia.

Entre las condiciones experimentales, que como se ha dicho son similares a los presentados en la Primera Propuesta de Cambio de Pesos, se usan las siguientes:

- El valor de α es de 0.001 para las 4 primeras pruebas. Este valor suaviza el cambio del valor del nuevo peso.
- Se considera que antes del tiempo experimental las rutas han tenido el mismo número de fuentes, tanto s_1-d_1 como s_3-d_3, por lo que se considera que al inicio de la simulación los pesos de ambas colas deben ser iguales. Ambas colas comienzan con igual peso 1 y 1 (inicialmente los pesos no requieren sumar 1 pero el programa puesto en el simulador hace el escalamiento para que los pesos sumen 1).
- Los paquetes emitidos son de 240 *Byte*.
- El tiempo experimental es de 1800 *s* (30 minutos).

Primera Prueba

En la primera prueba, al inicio del tiempo experimental, el número de fuentes es de 14 para la ruta s_1-d_1 y 16 para la ruta s_3-d_3. El peso de la ruta s_1-d_1 debería de tender a un valor promedio de 14 / (14 + 16) = 0.4666 y el peso de la ruta s_3-d_3 debería de tender un valor

promedio de 16 / (14 + 16) = 0.5333.

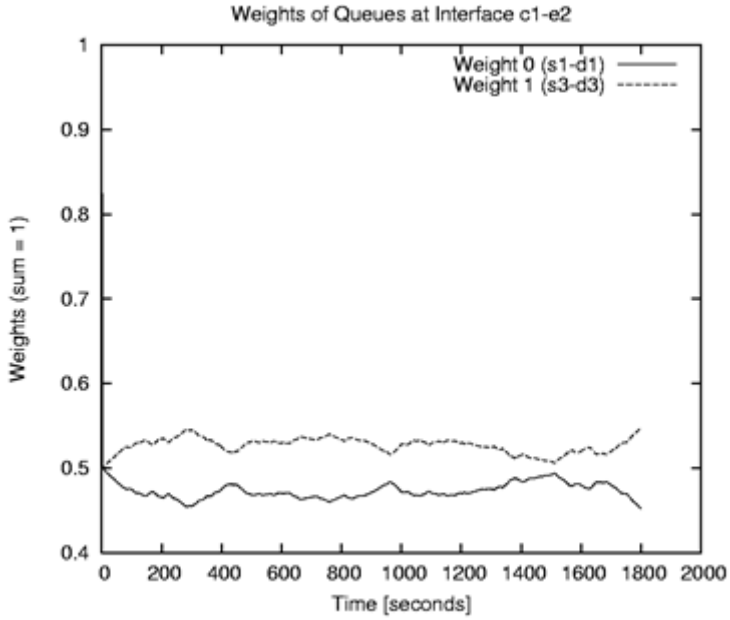


Figura 46. Pesos de las dos colas de la interfaz de salida del nodo c_1, a través del tiempo experimental, en la Primera Prueba de la Segunda Propuesta de Cambio de Pesos.

En la Figura 46 se observan los pesos de las dos colas a través del tiempo experimental (por norma, la suma de los pesos es 1 en todo el tiempo). Se observa que el peso de las rutas se tiende a ajustar a los valores indicados (14/30 y 16/30) pero de forma no asintótica sino que se tienen variaciones todo el tiempo experimental, como si "se rebotara". Esto se debe a que cuando los pesos se alejan se produce un efecto sobre las longitudes de las colas que finalmente produce el efecto contrario de empezar a acercar a los pesos, y cuando los pesos se acercan se produce un efecto sobre las longitudes de las colas, que finalmente produce el efecto contrario de empezar a alejar a los pesos.

Tercera Prueba

Se omite la segunda prueba por ser similar a la primera. En la tercera prueba hay un cambio en el número de fuentes al empezar el experimento, que queda con 8 fuentes en la cola de la ruta s_1-d_1, y 10 fuentes en la cola de la ruta s_3-d_3. Nuevamente, el peso de la cola de la ruta s_1-d_1 debería tender a un valor promedio de 8/(8+10) = 0.4444, y el peso de la cola de la ruta s_3-d_3 debería tender a un valor promedio de 10 / (8 + 10) = 0.5555. En la Figura 47 se observa, del resultado de esta prueba, mostrando el peso que van teniendo las dos colas, a través del tiempo experimental.

Se puede notar que el ajuste varía más suavemente que el ajuste de las pruebas anteriores, porque la diferencia en los tráficos de ambas rutas no es tan grande, y por consecuencia la diferencia de tamaño entre las longitudes medias de las colas tampoco es tan grande. Entonces la rapidez de ajuste de los pesos de las colas es menor.

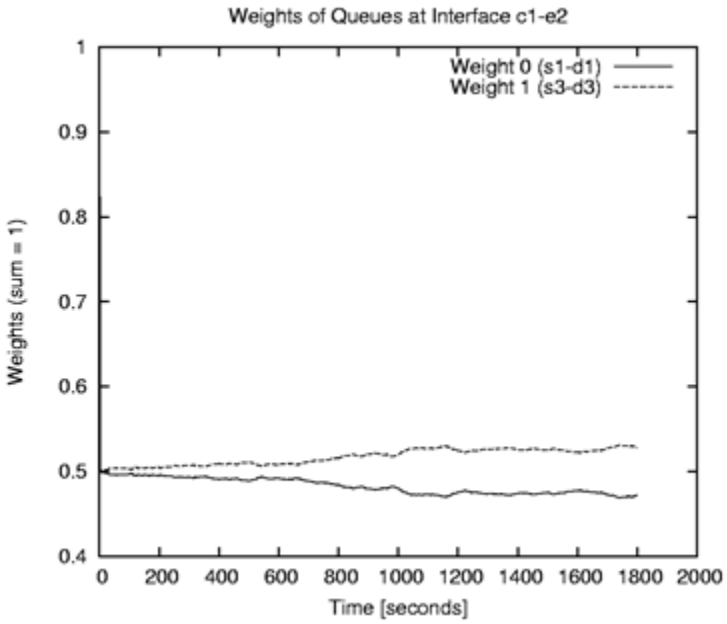


Figura 47**.** Pesos de las dos colas de la interfaz de salida del nodo c_1, a través del tiempo experimental, en la Tercera Prueba, de la Segunda Propuesta de Cambio de Pesos.

Quinta Prueba

Se omite la cuarta prueba por ser similar a la tercera. Para la quinta prueba el número de fuentes en la cola de la ruta s_3-d_3 fue de 16, y el número de fuentes en la cola de la ruta s_1-d_1 fue de 8. Hay más diferencia en tráfico entre estas rutas que en cualquier otra prueba anterior, pero ahora el valor de α fue de 0.00008 (se reduce). En la Figura 48 se observa que un cambio más suave en el peso a través del tiempo experimental.

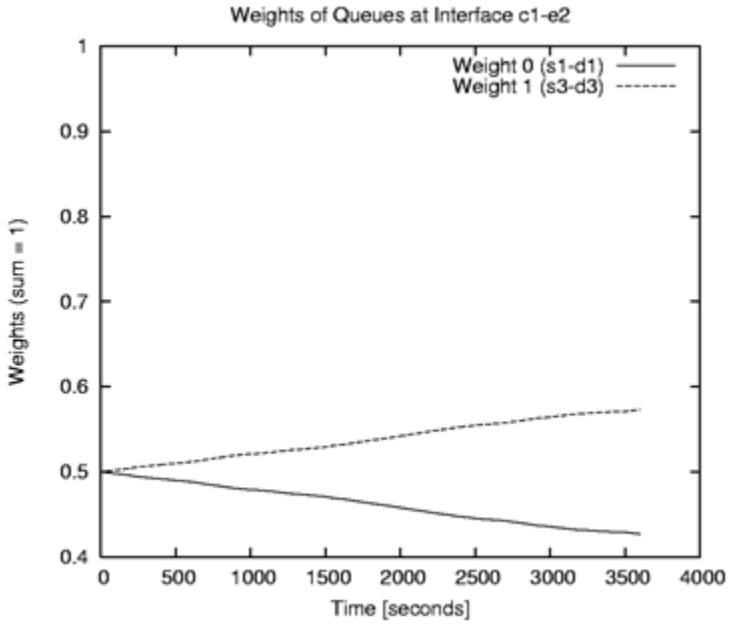


Figura 48**.** Pesos de las dos colas de la interfaz de salida del nodo c_1, a través del tiempo experimental, en la Quinta Prueba, de la Segunda Propuesta de Cambio de Pesos.

Según los resultados de todas las pruebas anteriores, se apreciaba una falta de control sobre la forma en que se daban los cambios de los pesos a través del tiempo. De la ecuación (6) se puede observar que la rapidez de acercamiento de $\phi_i(t+\tau)$ hacia $\overline{Q_i(t+\tau)}\,\phi_i' \Big/ \sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'$ estaba dependiendo de:

1- El valor de α.

2- La cantidad de actualizaciones hechas por segundo.

3- La diferencia misma entre los valores de $\phi_i(t+\tau)$ y $\overline{Q_i(t+\tau)}\,\phi_i' \Big/ \sum_{k=1}^{N} \overline{Q_k(t+\tau)}\,\phi_k'$, que involucraba la diferencia entre la cantidad de tráfico entrante a una y otra colas.

Por estas razones esta propuesta aún tenía problemas y debía replantearse.

Método STP. Evaluación Experimental Inicial

A continuación se presentan los experimentos iniciales hechos con la propuesta de método de atención de colas, llamada en esta Tesis Método STP (Short-Term Protection), por sus siglas en inglés. La explicación del algoritmo de operación de este método se encuentra en el subcapítulo 7.3.

Los experimentos se hicieron con el simulador ns-2 (versiones 2.33). Las condiciones experimentales fueron las siguientes:

Primeros Experimentos

En los primeros experimentos con esta "tercera" propuesta: el Método STP, se empleó una topología pequeña para revisar la operación del método en un solo nodo, el nodo c_1 (ver Figura 49). Estos resultados se muestran en [85] y en el Apéndice A.1).

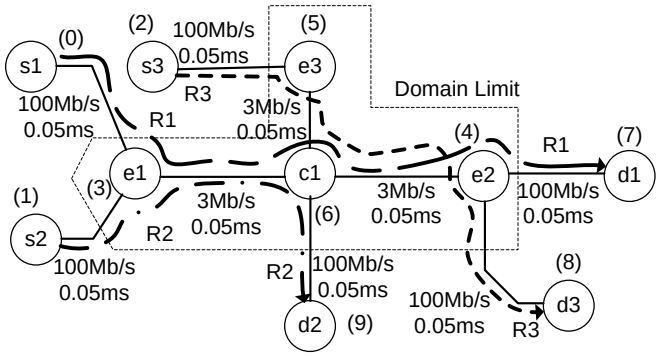


Figura 49**.** Topología de primeros experimentos para la Tercera Propuesta de Cambio de Pesos, el Método STP.
Se tienen dos rutas: la *Ruta* 1 que pasa por los nodos s_1–e_1–c_1–e_2–d_1, y la *Ruta* 3 que pasa por los nodos s_3–e_3–c_1–e_2–d_3. Estas rutas se interfieren en la interfaz de salida del nodo c_1 y se separan en el nodo e_2. Se tiene otra ruta, la *Ruta* 2 que pasa por los nodos s_2–e_1–c_1–d_2.

- Se evalúan tres tipos de soluciones. En el primer tipo hay una cola en la interfaz de salida del nodo c_1 (aquí llamado "caso *q1*"). En el segundo tipo hay dos colas en la interfaz de salida del nodo c_1 (llamado "caso *q2*"), usando el Método STP en dicha interfaz, con una cola para el tráfico que llega de la *Ruta* 1 y otra cola para el tráfico que llega de la *Ruta* 3. Como el tráfico de cada cola corresponde a una ruta en particular a las colas se les llama "Cola de Ruta 1" y "Cola de Ruta 3", respectivamente. Los parámetros del Método STP son: T = 1200 *s*, P = 0.25. El tiempo τ es 1 *s*. El tercer tipo (llamado caso *q2f*) es similar al segundo, pero las colas tienen peso fijo. Todas las demás interfaces de salida de los nodos tienen una sola cola "Droptail". Las longitudes máximas de las colas son tales que la pérdidas que pudiesen darse, por congestionamiento, son menores al 2% en cada experimento.
- La evaluación de las otras soluciones que no son la del Método STP se hace con motivos de comparación de resultados.
- Para la evaluación de cada solución se hacen diversos experimentos. Los resultados de cada experimento se toman de la media de los resultados de 40 simulaciones con idénticas condiciones experimentales. La cantidad de tráfico en cada experimento, a lo largo de tiempo experimental, es constante. Entre un experimento y otro el tráfico de la *Ruta* 3

va aumentando, es decir, el tráfico en cada experimento de la *Ruta* 3 es constante, pero inicia siendo mayor en un experimento, con relación al experimento anterior, para la evaluación de cualquiera de las tres soluciones. Con esto se observa la reacción al aumento de tráfico que tiene cada solución, a lo largo del tiempo experimental.

- Como tráfico se emplearon fuentes de tráfico incorporadas al simulador[135]. Las fuentes empleadas fueron de dos tipos: Pareto (PAR) y de tasa constante (CBR). Las fuentes Pareto fueron *On/Off* con 250 *ms* de duración *On* y 250 *ms* de duración *Off*, con 68 *Kb/s* durante los periodos *On*, y paquetes de 95 *Byte* [92][136] con parámetro $\theta = 1.7$ (varianza infinita). Las fuentes CBR tuvieron paquetes de 1500 *Byte* con 256 *Kb/s*[137]. Para el experimento se indicó que las tasas eran pequeñas comparadas con las que podrían usarse en la Internet, pero que servía para propósitos de evaluación del método en un ámbito específico.
- En cada experimento la *Ruta* 1 y la *Ruta* 2 tienen casi el mismo número de fuentes. Las fuentes en la *Ruta* 2 causan retraso en la *Ruta* 1 por el encolamiento en la interfaz de salida del nodo e_1. Aunque en esta interfaz hay un cruce de rutas, como los tráficos de estas rutas son fijos, no se emplea el Método STP en la interfaz de salida del nodo e_1. La *Ruta* 1 tiene 3 fuentes CBR + 17 Fuentes Pareto; y la *Ruta* 2 tiene 3 fuentes CBR + 16 fuentes Pareto. Este tráfico es suficientemente grande para causar un retraso promedio cercano a 3 *ms* en la interfaz de salida del nodo e_1, sin tener pérdidas por congestionamiento. El tráfico de estas rutas es igual para todos los experimentos.
- Se considera que antes de cada experimento ha habido un tráfico medio sin cambios, por un tiempo prolongado, incluyendo el de la *Ruta* 3, con 3 fuentes CBR y 8 fuentes Pareto. Entonces, se considera que la participación inicial de los tráficos medios de la *Ruta1* y la *Ruta3*, en la interfaz de salida del nodo c_1, es de 0.5641 y 0.4359, respectivamente[138]. Estos valores se usan como los pesos iniciales de las colas respectivas para el caso *q*2 (tomando los pesos de las colas con la proporción correspondiente del tráfico en la interfaz). Para el caso *q2f* también se toman estos valores como los pesos fijos de las colas, en dicha interfaz. Desde el inicio de cada experimento la *Ruta* 3 tiene un tráfico aumentado con relación a lo que se consideró que había tenido, por un tiempo prolongado,

[135] En los experimentos de la propuesta posterior (el Método STP) se incorpora en el simulador otro tipo de fuentes, buscando más apego al tráfico real, sensible al retraso.

[136] Considerando que un paquete de voz tiene 67 *Bytes* lo que da un paquete IP de 95 *Byte* añadiendo 8 *Byte* de encabezado UDP y 20 *Byte* de encabezado IP. Con una tasa media de 3.0 *KByte*/*s* en una dirección se obtiene: 34 *Kb/s x* 3 *KByte*/*s* x 8 *bit*/*Byte* x 95 / 67 = 34 *Kb/s*, generando 68 *Kb/s* durante los periodos *On*.

[137] Esta tasa se obtiene de la página Web de Skype [100].

[138] En la
Tabla 21 se dan los valores de tráfico y de participación de tráfico de las rutas para estos experimentos. La *Ruta* 1 tiene el tráfico de 3 CBR + 17 PAR que es (3 x 256) *Kb/s* + (17 x 68 / 2) *Kb/s* = 1346 *Kb/s*. La *Ruta* 3 tiene el tráfico de 3 CBR + 8 PAR que es (3 x 256) *Kb*/s + (8 x 68 / 2) *Kb/s* = 1040 *Kb/s*. La participación del tráfico de la *Ruta* 1 es igual a 1346 / (1346 + 1040) = 0.5641, siendo la participación de la otra ruta igual a 0.4359.

antes del inicio de los experimentos. El aumento de tráfico se denota con +PAR (es decir son fuentes Pareto adicionales).

Observación de los Cambios de Peso

Los primeros resultados de esta evaluación se enfocaron exclusivamente a revisar el comportamiento los pesos en la interfaz de salida del nodo c_1, para el caso $q2$, con el parámetro del Método STP, T = 1200 *s* y diversos valores del parámetro P. Los pesos iniciales de las colas de la *Ruta* 1 y la *Ruta* 3 son 0.5641 y 0.4359 respectivamente. La Figura 50 presenta estos pesos para experimentos con cuatro diferentes valores de P, en el curso de la duración de experimentos en donde el tráfico de la *Ruta* 3 aumenta 30 fuentes Pareto, al inicio de los experimentos[139]. Solamente en este caso el ancho de banda de la interfaz de salida del nodo c_1 se fijó en 4 *Mb/s*. En este caso la cola de la *Ruta* 1 debería tener menor tamaño en casi todos los casos con relación a la cola de la *Ruta* 3.

Cuando $t = T$ = 1200 el peso de la cola para la *Ruta* 1 teóricamente, por la operación del Método STP, no debería llegar a ser menor que lo que se indica en la columna central de la Tabla 20. La tercera columna de esta tabla muestra los datos experimentales, que se observan en la Figura 50, del peso para la cola de la *Ruta* 1, en $t = T$ = 1200.

Valor de P.	Peso mínimo en Ruta 1 en $t = T$.	Peso de Datos Experimentales según la Figura 50.
0.05	0.564 x 0.95 = 0.5358	0.5291
0.15	0.564 * 0.85 = 0.4794	0.4661
P = 0.25	0.564 * 0.75 = 0.4230	0.41

Tabla 20. 4 Método STP. Evaluación Experimental Inicial. Revisión del comportamiento de pesos en la interfaz de salida del nodo c_1.

Se observa que los valores obtenidos (en la tercera columna de la Tabla 20) son cercanos, aunque un poco menores al mínimo esperado. La diferencia con lo esperado se puede deber a que en estos primeros experimentos se aproximó el logaritmo de la ecuación (15) en lugar de obtenerlo directamente[140].

[139] El porcentaje de participación de tráfico que perdería la *Ruta* 1 en este caso es como sigue. Esta ruta tiene 3 CBR y 17 PAR equivalente a 1346 *Kb/s*. La *Ruta* 3 tiene 3 CBR y 38 PAR equivalente a 2060 *Kb/s*. El porcentaje de participación de la *Ruta* 1 es 1346/(2060+1346)=0.3952. Su pérdida de participación de tráfico es (56.41-39.52)/56.41=0.2994 (29.94%).

[140] Para los experimentos siguientes se obtendría el logaritmo con la función "log" de C++.

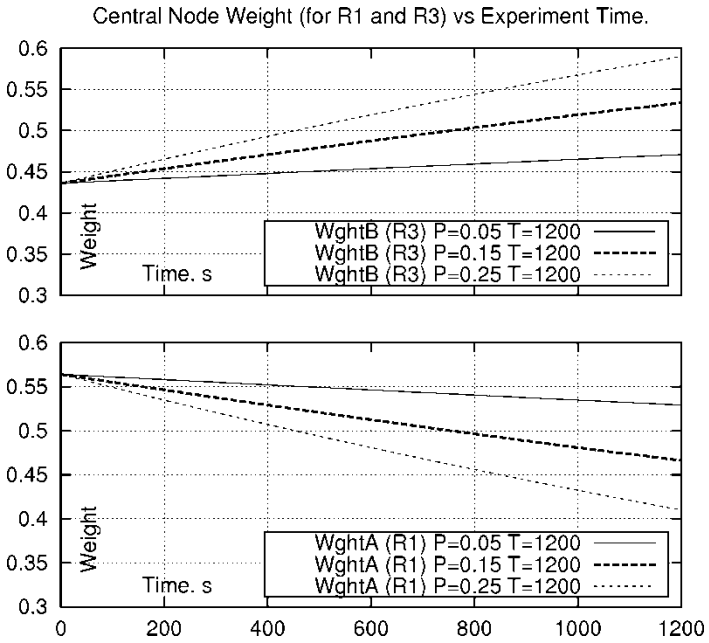


Figura 50. 4 Método STP. Evaluación Experimental Inicial. Resultados experimentales para el peso, en la solución *q*2, con diversos valores del parámetro *P*.
La etiqueta "Wght A (R1) P=0.15 T=1200" indica Peso de la *Ruta* 1 para el caso *q*2 con parámetros *P* = 0.15 y *T* = 1200 *s*.

Observación de Ganancias y Retrasos

Para esta evaluación experimental se utilizó tráfico en las rutas como se ha indicado. Con relación al tráfico de la *Ruta* 3, y su participación con relación al tráfico de la *Ruta* 1, en la interfaz de salida del nodo c_1, se hace uso de la Tabla 21, que muestra las fuentes que tienen la *Ruta* 1 ("R1") y la *Ruta* 3 ("R3"), en los diversos experimentos. En la tabla las fuentes Pareto se designan como "PAR" y las fuentes constantes como "CBR". El encabezado "+PAR" indica el número de Fuentes Pareto incrementadas en la *Ruta* 3, al inicio del cada experimento. Los encabezados "%" indican el porcentaje de ancho de banda que la *Ruta* 1 y la *Ruta* 3 toman de la interfaz de salida del nodo c_1. El encabezado "-%R1" indica el porcentaje de ancho de banda perdido en la *Ruta* 1 con relación a lo que se tenía en la primera fila.

Por ejemplo, en el primer renglón de la Tabla 21, la participación del tráfico de la *Ruta* 1 y *Ruta* 3, en la interfaz de salida del nodo c_1, es de 56.41% y 43.59%, respectivamente (ver cálculos en pie de página 138). En el segundo renglón que contiene datos, la *Ruta* 3 tiene 1 fuente Pareto adicional a lo que se tenía en el primer renglón de la tabla. En este caso la participación de las rutas cambia a 55.62% y 44.38%, respectivamente. En este segundo renglón, bajo el encabezado "-%R1" se tiene el valor "1.41", que se obtiene de la expresión: (56.41 – 55.62) / 56.41 = 0.01405 (1.41%), que quiere decir que el deterioro de porcentaje de

participación del tráfico de la *Ruta* 1 ha sido de 1.41%, con relación a lo que se tenía en el caso del primer renglón.

Con el Método STP el deterioro del peso de la *Ruta* 1, al final del lapso de tiempo de duración *T*, no debe ser mayor al deterioro del porcentaje de pérdida de tráfico de dicha ruta.

CBR R1	CBR R3	PAR R1	PAR R3	+PAR R3	*Kb/s* R1	*Kb/s* R3	*Kb/s* R3+R1	% R3	% R1	–% R1
3	3	17	8	0	1346	1040	2386	43.59	56.41	0.00
3	3	17	9	1	1346	1074	2420	44.38	55.62	1.41
3	3	17	10	2	1346	1108	2454	45.15	54.85	2.77
..	..	..	..							
3	3	17	16	8	1346	1312	2658	49.36	50.64	10.23
3	3	17	17	9	1346	1346	2692	50.00	50.00	11.37
3	3	17	18	10	1346	1380	2726	50.62	49.38	12.47
3	3	17	19	11	1346	1414	2760	51.23	48.77	13.55

Tabla 21**.** 4 Método STP. Evaluación Experimental Inicial. Ganancias y Retrasos. Número de Fuentes en la *Ruta* 1 y en la *Ruta* 3, las tasas en estas rutas, y el porcentaje de ancho de banda en la interfaz de salida del nodo c_1, para estas rutas, en los diversos experimentos.

En esta evaluación se propusieron “Ganancias” y “Retrasos al 2% de percentil” como forma de evaluar las ventajas del método.

Con relación a las ganancias, no se tomó una ganancia total sino ganancias por rutas. Las ganancias otorgaban 1 *punto* por cada paquete que llegaba “a tiempo” en la ruta, y penalizaban con 10 *puntos*, por cada paquete que llegara retrasado, en donde el límite de retraso se fijó en 18 *ms*.

Para el tipo de solución *q*2, los parámetros son *P* = 0.25 y *T* = 1200 *s*, con duración de 1200 *s* por experimento, donde la *Ruta* 3 aumenta su tráfico en *t* = 0 *s* del experimento, sin mayor aumento posterior.

Los valores iniciales de los pesos de las colas de la interfaz de salida of c_1 son: 0.5641 y 0.4359, para la *Ruta* 1 y *Ruta* 3, respectivamente, como lo indica la

Tabla **21**. Los retrasos y las ganancias se evalúan de los archivos de trazas resultantes de los experimentos. Se evalúan ganancias para cada intervalo de 120 *s* del tiempo experimental (por ejemplo el valor de ganancia en el tiempo 1200 *s* corresponde a lo calculado en el intervalo de 1080 a 1200 *s*.

La Figura 51 muestra los resultados obtenidos. La parte izquierda de la figura muestra los retrasos en las rutas para el tipo de solución *q*2, a través del tiempo experimental. Se observa que los retrasos para la *Ruta* 1 aumentan en mayor medida, con el tiempo experimental,

conforme más fuentes se añaden en la *Ruta* 3 (desde 0% cuando no se añaden fuentes hasta un aumento que va de 16 *ms* a 22 *ms* cuando se añaden 11 fuentes). Al contrario, la *Ruta* 3 disminuye su retraso con el tiempo experimental. Esto se debe a que, conforme pasa el tiempo del experimento, la cola de la *Ruta* 1 va disminuyendo de peso, en la interfaz de salida del nodo c_1, y la cola de la *Ruta* 3, en dicha interfaz, va aumentando de peso[141].

La parte derecha de la Figura 51 muestra las ganancias de las rutas para el tipo de solución *q*2, y también para los tipos de solución *q*1 y *q*2*f*, con motivos de comparación.

Cada línea indica las ganancias en una de las rutas (la *Ruta* 1 o la *Ruta* 3) obtenidas, contra el número de fuentes añadidas en la *Ruta* 3. Cada línea, en el tipos de solución *q*2, indica las ganancias que se obtenían a diversos tiempos experimentales, es decir, las tres líneas para el tipo de solución *q*2 dan la ganancia a los 120, 600 y 1200 *s* del tiempo experimental. Se observa cómo la línea para el tipo de solución *q*2 a los 1200 *s* se parece a la línea para el tipo de solución *q*1, y cómo la línea para el tipo de solución *q*2 a los 120 *s* se parece a la línea para el tipo de solución *q*2*f* (ya sea para la *Ruta* 1 o la *Ruta* 3). Es decir, en el tipo de solución *q*2, al principio del experimento (que es poco antes del tiempo 120 *s*) la protección que se da a la *Ruta* 1 es parecida a lo que se tiene en el tipo de solución *q*2*f*. Al final del experimento (en el tiempo 1200 *s*) esa ruta ha perdido su protección, algo más cercano al tipo de solución *q*1. En otras palabras, con estos resultados se podía ver que el Método STP podía proteger a la *Ruta* 1, al principio de los experimentos (como lo hace en todo el tiempo el tipo de solución *q*2*f*), y al final del experimento se perdería la protección (más cercano al tipo de solución *q*1 donde no hay protección alguna).

[141] La falta de suavidad en las curvas obtenidas cuando hay mayor tráfico se debió a que el promedio obtenido en las simulaciones, para cada experimento, se obtuvo con la estadística $\overline{X} = 1/N \sum_{i=1}^{N} X_i$, y que el tráfico en este caso tenía componentes Pareto, de varianza infinita, pero dicha estadística no es buena [93] para distribuciones con extremos "muy pesados", aun cuando se realizaron hasta 40 simulaciones por experimento, y en este caso es mejor emplear la estadística, más robusta, $\overline{X}_{tr}$, en la que primero se truncan, en algún porcentaje, los valores más altos y los valores más bajos de la muestra.

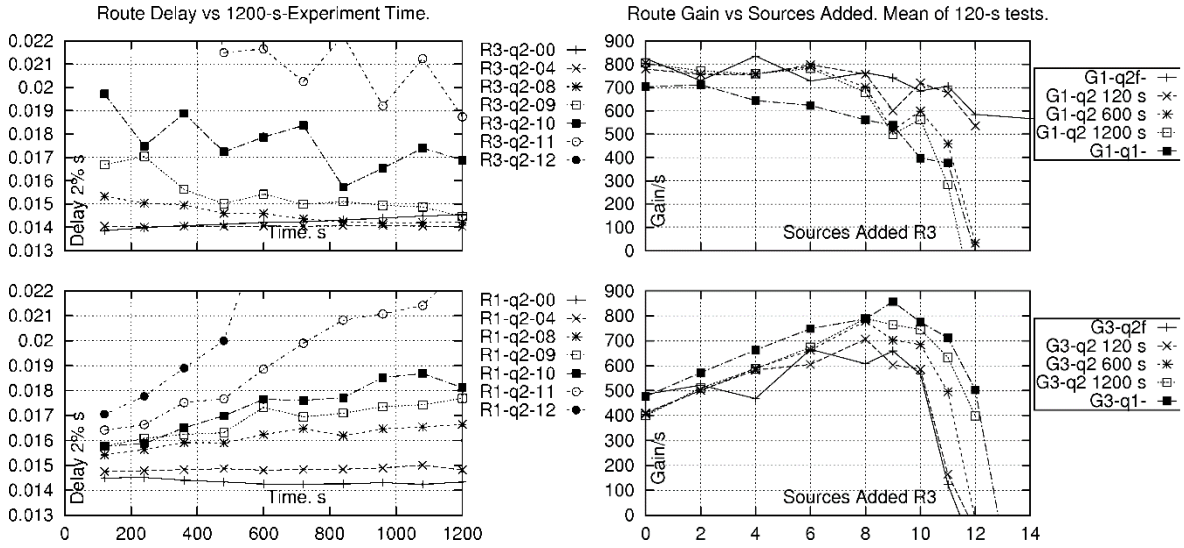


Figura 51. Método STP. Evaluación Experimental Inicial. Ganancias y Retrasos.
La parte izquierda de la figura muestra los retrasos en las rutas para el tipo de solución *q*2. La etiqueta "R1-q2-04" indica "Retraso de la *Ruta* 1 para 4 fuentes añadidas en la *Ruta* 3, para el tipo de solución *q*2. La parte derecha de la figura muestra las ganancias de las rutas para los tipos de solución *q*1, *q2* y *q*2*f*. La etiqueta "G1-q2 120 s" indica "Ganancia en la *Ruta* 1, en el tipo de solución *q*2, a 120 *s* de iniciado el experimento.

En gráficas no mostradas se podía ver que la ganancia total (suma de ganancias de las rutas) era aproximadamente la misma para los tres tipos de solución (*q*1, *q*2 y *q*2*f*) (estos resultados no se reportan aun en algún artículo). La razón de esto es que los despachadores para los tipos de solución *q*2 y *q*2*f* son conservadores de energía, otorgando un servicio tan grande como se tiene en el tipo de solución *q*1. Queda claro que el Método STP no maximiza un indicador así (recordando que este indicador no otorga más importancia al tráfico ya admitido sobre el tráfico nuevo).

Otros resultados, no mostrados, usando el parámetro T = 120 *s*, indican que las ganancias y retrasos se compartan de forma similar a lo mostrado en la Figura 51, pero con cambios 10 veces más rápidos.

Apéndice G. Densidad de Probabilidad Pareto

Las gráficas del tráfico en las redes exhiben características auto-similares en el tiempo entre llegadas de paquetes. Estas características se manifiestan a través de distribuciones de probabilidad que tienen colas de "peso" significativo. La distribución de llegadas de Poisson (con su correspondiente distribución exponencial de tiempo entre llegadas) no tiene esta característica. Para describir los efectos auto-similares se utiliza comúnmente el modelo de densidad de probabilidad de Pareto.

La densidad de probabilidad, *f*(*u*), y la distribución de probabilidad, *F*(*U*), de Pareto, con sus parámetros θ y ξ_a, son:

$$f(u)=\frac{\theta}{\xi_a}\left(\frac{u}{\xi_a}\right)^{-(\theta+1)};\ F(U)=P(u\le U)=1-\left(\frac{\xi_a}{u}\right)^{\theta};\ \theta>0;\ \xi_a>0;\ u>\xi_a. \tag{64}$$

La media μ es:

$$\mu=\begin{cases}\dfrac{\theta}{\theta-1}\xi_a, & \theta>1\\ \infty, & 0<\theta\le 1\end{cases} \tag{65}$$

La varianza σ^2 es:

$$\sigma^2=\begin{cases}\theta\xi_a^2\left[\dfrac{1}{\theta-2}-\dfrac{\theta}{(\theta-1)^2}\right], & \theta>2\\ \infty, & 0<\theta\le 2\end{cases} \tag{66}$$

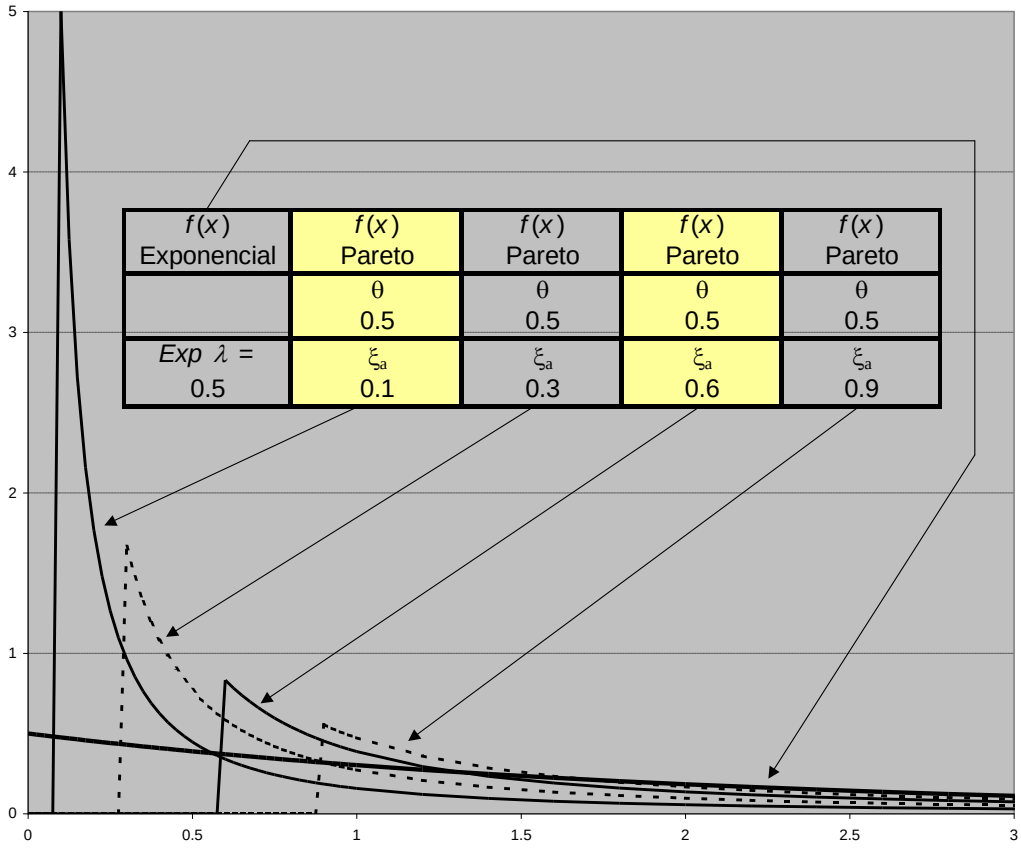


Figura 52**.** Imagen de Densidad de Probabilidad de Pareto con diversos valores de sus argumentos, e Imagen de la Densidad de Probabilidad Exponencial.

Como se observa en la Figura 52, la densidad de probabilidad Exponencial se cruza con la densidad de probabilidad Pareto. En la Figura 53 la densidad de probabilidad de Pareto, graficada para diversos valores del argumento ξ_a , para valores suficientemente grandes de la variable independiente *x*, ya es mayor que la densidad de probabilidad Exponencial.

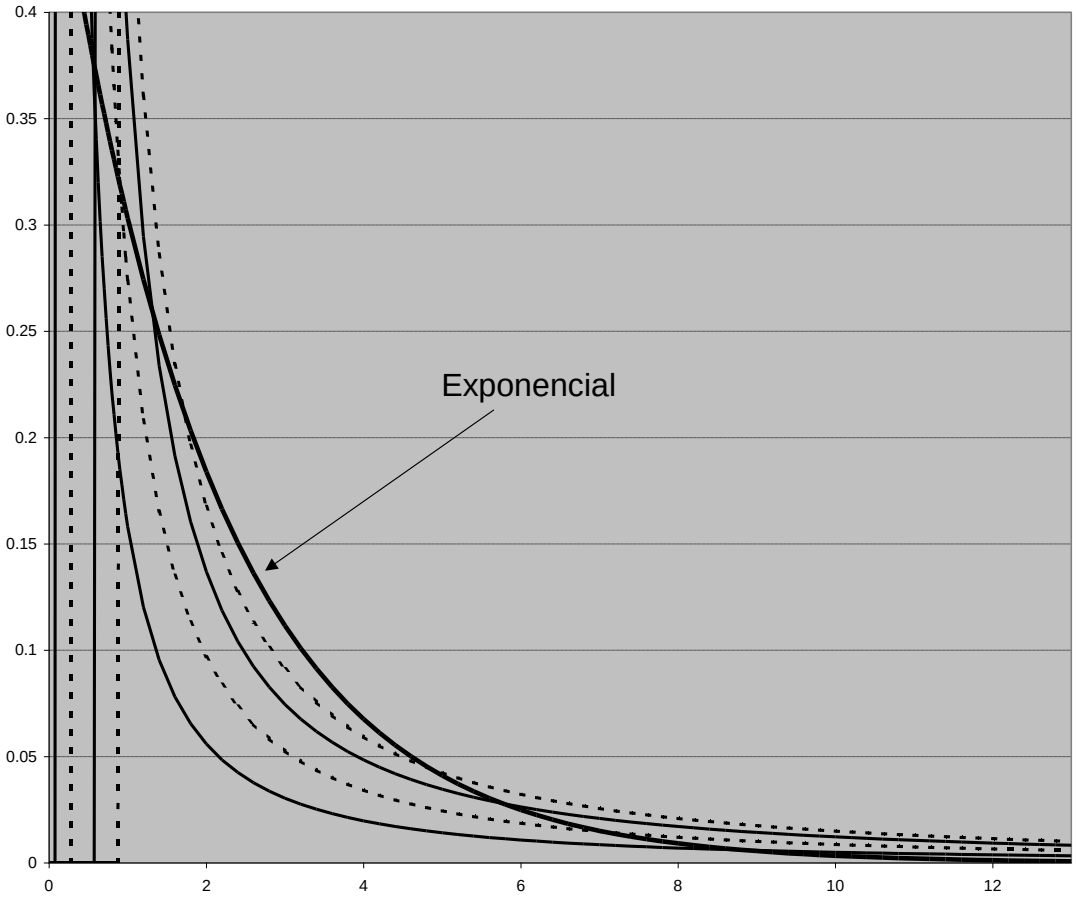


Figura 53**.** Imagen de Densidad de Probabilidad de Pareto con diversos valores de sus argumentos, e Imagen de la Densidad de Probabilidad Exponencial. Todo para valores mayores de la variable *x*.

Las curvas correspondientes a las densidades de Pareto, para un solo valor de θ y diversos valores de ξ_a no se cruzan.

En la Figura 54 se observan varias densidades Pareto para un valor fijo de ξ_a = 0.3, y varios valores de θ. Se puede observar que mientras más pequeño sea el valor de θ más pequeño es el valor de la densidad en valores pequeños del argumento, pero más grande es el valor de la densidad en valores grandes del argumento (aquí sí hay cruce de las curvas de las diferentes densidades Pareto). Por eso, para valores de $\theta \leq 1$, la media es infinita, y para valores de $1 < \theta \leq 2$ la variancia es infinita.

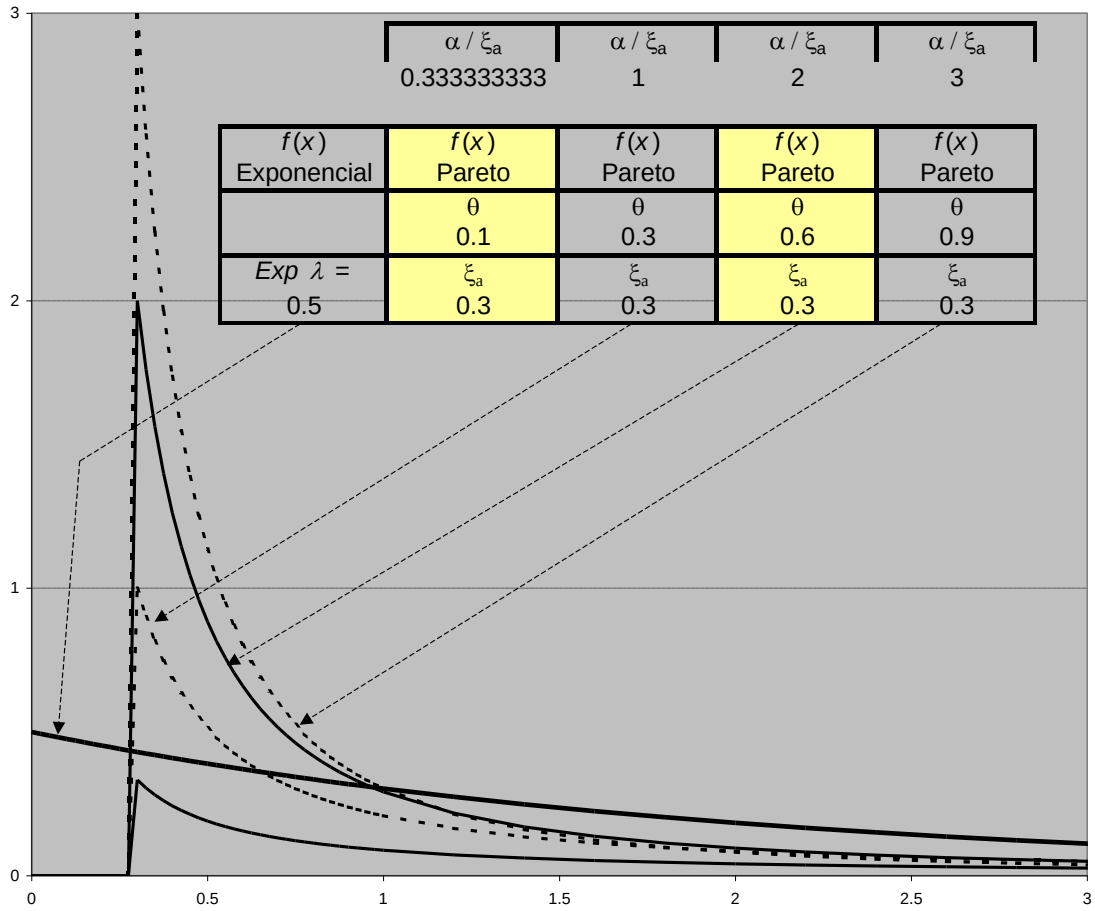


Figura 54. Imagen de Densidad de Probabilidad de Pareto con diversos valores del argumento θ, e Imagen de la Densidad de Probabilidad Exponencial. Todo para valores mayores del argumento *u*.

Apéndice H. Detalles de la Topología Utilizada en la Primera y Segunda Propuestas de Cambio de Pesos

Los detalles de la topología utilizada en la Primera y Segunda Propuestas de Cambio de Pesos incluyen la configuración de los parámetros de DS para emular el comportamiento de que una interfaz de salida tuviese una cola por cada interfaz de entrada. La interfaz de salida con esta característica de la del nodo c_1, yendo hacia el nodo e_2. Una configuración similar se hace para topologías mayores empleadas en la propuesta del Método STP. Las instrucciones para la programación del simulador, utilizando las herramientas de DiffServ se dan en [75].

En la Figura 55 se observan los tipos de colas que se manejan en la topología utilizada para las pruebas con simulación.

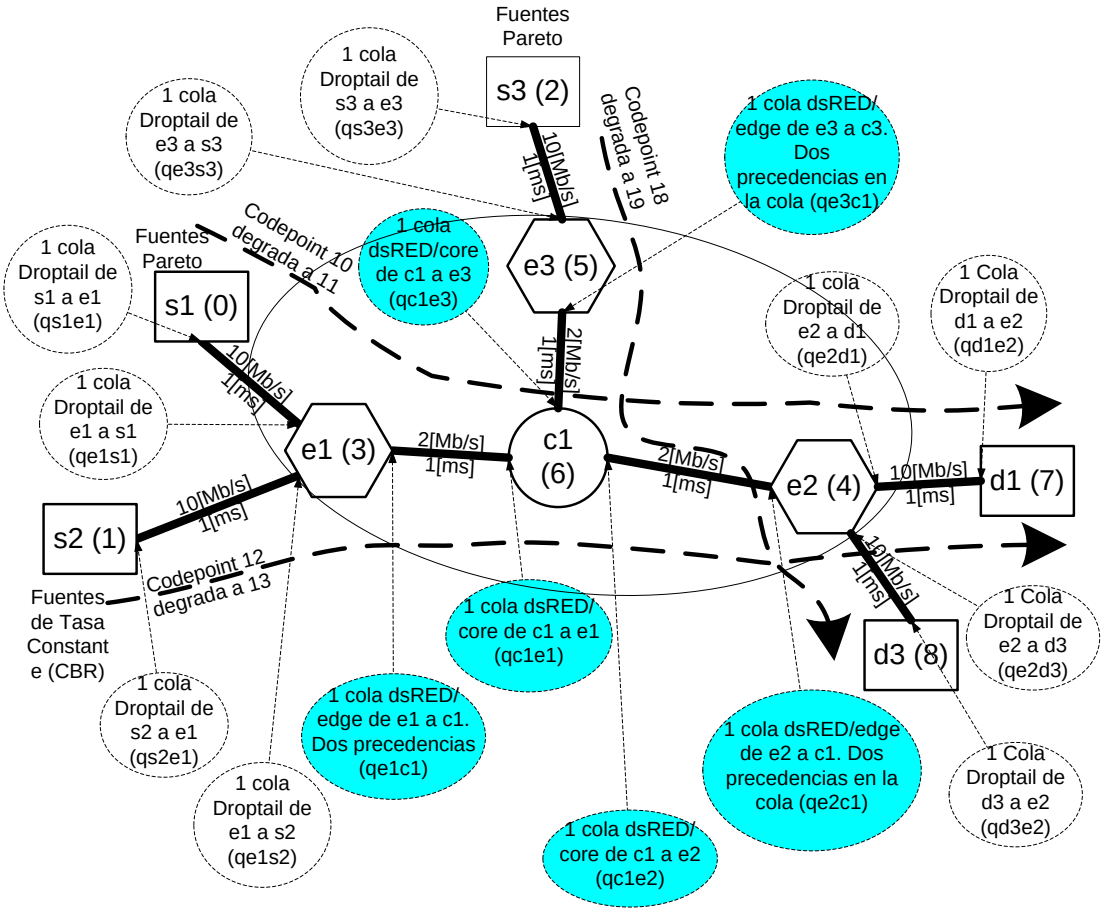


Figura 55. Detalle de la configuración de la topología de red empleada para la evaluación, por simulación, de la primera y segunda propuestas de operación de cambio de pesos.

Un nodo de frontera tiene varias tablas, como se muestra en la Figura 56 (un nodo central solo tiene tablas de PHB). Las tablas de políticas asignan un CodePoint a los paquetes de los flujos, dependiendo de dónde vienen y a dónde van. Es decir que los CodePoints dependen del inicio y destino de los paquetes (no habría paquetes con el mismo origen y destino que pudiesen tener diferente CodePoint ---lo que es una limitante de la operación de esta implantación de DiffServ en ns).

Las tablas de Policers indican que algún paquete de un flujo podría cambiar de CodePoint (a uno con características de tratamiento degradadas en el dominio) si el flujo no está cumpliendo con las características requeridas para el flujo

Tablas de PHBs. Indican en qué colas físicas y virtuales se pondrán los paquetes, según su CodePoint. Esto permitirá dar un tratamiento especial a cada flujo, que está ligado al tratamiento que tienen las colas en los nodos.

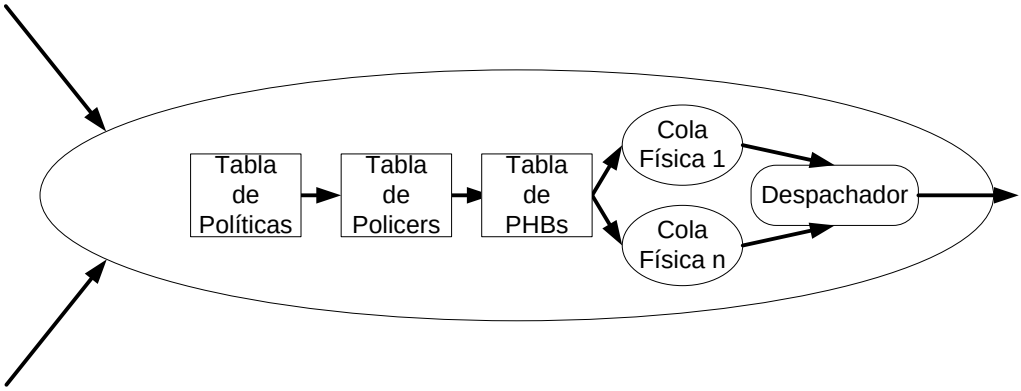


Figura 56. Tablas de DiffServ para operación en los nodos.

Las siguientes tablas indican las políticas, las configuraciones de los Policers y las tablas de PHB utilizadas en los nodos de la red empleada para la evaluación, por simulación, de la primera y segunda propuestas de operación de cambio de pesos. Las variables cir0, cir1, cbs0, cbs1 se indican en las siguientes líneas:

Commited Interface Rate (cir) es 1000000 *bit*/s para todas las fuentes Pareto de s1 (nodo 0 en ns) a d1 (nodo 7 en ns).

Committed Burst Syze (cbs) es 10000 *Byte* para las Fuentes Pareto.

Commited Interface Rate (cir) es 800000 *bit*/s para las fuentes de tasa constante de s2 (nodo 1 en *ns*) a d1 (nodo 7 en *ns*).

Committed Burst Syze (cbs) es 2000 *Byte* para las fuentes de tasa constante.

TABLA DE POLITICAS DEL NODO e1 para ir a c1							
Política #	**Nodo Fuente**	**Nodo Destino**	**Codepoint Inicial**	**Tipo de Policer**	**Tasa de Interfaz**	**Tamaño de Ráfaga**	
1	s1	d1	10	Token Bucket	cir0	cbs0	Fuentes Pareto
2	s2	d1	12	Token Bucket	cir1	cbs1	Fuentes CBR

TABLA DE POLITICAS DEL NODO e2 para ir a c1							
Política #	**Nodo Fuente**	**Nodo Destino**	**Codepoint Inicial**	**Tipo de Policer**	**Tasa de Interfaz**	**Tamaño de Ráfaga**	
1	d1	s1	14	Token Bucket	cir0	cbs0	No se pusieron fuentes
2	d1	s2	16	Token Bucket	cir0	cbs0	No se pusieron fuentes
3	d1	s3	20	Token Bucket	cir0	cbs0	No se pusieron fuentes

TABLA DE POLITICAS DEL NODO e3 para ir a c1							
Política #	**Nodo Fuente**	**Nodo Destino**	**Codepoint Inicial**	**Tipo de Policer**	**Tasa de Interfaz**	**Tamaño de Ráfaga**	
1	S3	D3	18	Token Bucket	cir0	cbs0	Fuentes Pareto

TABLA DE POLICER DEL NODO e1 para ir a c1		
Tipo de Policer	**Codepoint Inicial**	**Codepoint al Degradar**
Token Bucket	10	11
Token Bucket	12	13

TABLA DE POLICER DEL NODO e2 para ir a c1		
Tipo de Policer	**Codepoint Inicial**	**Codepoint al Degradar**
Token Bucket	14	15
Token Bucket	16	17
Token Bucket	20	21

TABLA DE POLICER DEL NODO e3 para ir a c1		
Tipo de Policer	**Codepoint Inicial**	**Codepoint al Degradar**
Token Bucket	18	19

TABLA DE PHB DEL NODO e1 para ir a c1. Relaciona "codepoints" a colas físicas y virtuales en el nodo. Colas RED.		
Code Point	**Cola Física**	**Cola Virtual**
10	0	0
11	0	1
12	0	0
13	0	1

TABLA DE PHB DEL NODO e2 para ir a c1. Relaciona "code-points" a colas físicas y virtuales en el nodo. Colas RED.		
Code Point	**Cola Física**	**Cola Virtual**
14	0	0
15	0	1
16	0	0
17	0	1

TABLA DE PHB DEL NODO e3 para ir a c1. Relaciona "code-points" a colas físicas y virtuales en el nodo. Colas RED.		
Code Point	**Cola Física**	**Cola Virtual**
18	0	0
19	0	1

TABLA DE PHB DEL NODO c1 para ir a e1. Relaciona "code-points" a colas físicas y virtuales en el nodo. Colas RED.		
Code Point	**Cola Física**	**Cola Virtual**
14	0	0
15	0	1
16	0	0
17	0	1

TABLA DE PHB DEL NODO c1 para ir a e2. Relaciona "code-points" a colas físicas y virtuales en el nodo. Colas RED.		
Code Point	**Cola Física**	**Cola Virtual**
10	0	0
11	0	1
12	0	0
13	0	1
18	1	0
19	1	1

TABLA DE PHB DEL NODO c1 para ir a e3. Relaciona "code-points" a colas físicas y virtuales en el nodo. Colas RED.		
Code Point	**Cola Física**	**Cola Virtual**
20	0	0
21	0	1

Tabla 22. Políticas, configuraciones de los Policers y PHB utilizados en los nodos de la red empleada para la evaluación, por simulación, de la primera y segunda propuestas de operación de cambio de pesos.

Apéndice I. Ejemplo de la Obtención del Valor Promedio con el Método del Percentil Bootstrap de Intervalos de Confianza

Apéndice I. Ejemplo de la Obtención del Valor Promedio con el Método del Percentil Bootstrap de Intervalos de Confianza

Con el fin de obtener el intervalo de confianza para indicar la fiabilidad de la estimación de la media, con el promedio obtenido, se aplicó método del percentil Bootstrap de Intervalos de Confianza [93].

Como ejemplo se da la Tabla 23, que muestra 75 de los 1000 renglones de la tabla completa con la que se ejemplifica este método. En el primer renglón de la tabla se muestran 10 números correspondientes al mismo punto en el tiempo de 10 simulaciones. Estos valores se promedian para el ejemplo, siendo este promedio el valor estimado del promedio real del fenómeno. Los valores son: 0.051179; 0.043361; 0.043658; 0.030275; 0.04805; 0.042889; 0.039436; 0.041037; 0.035677; 0.032989; y el valor medio estimado fue 0.0408551.

Para obtener el intervalo de confianza mencionado se hicieron 1000 promedios con los 10 números, tomándolos en combinaciones diferentes tal que siempre los números tomados fuesen de 10 cada vez, donde era válido repetir las veces de tomar uno o más números.

La primera columna de la tabla es un número consecutivo que va del 1 al 1000. Las siguientes 10 columnas indican cómo se tomaron los números para hacer el promedio del renglón (observar que los números tomados siempre son 10 pudiendo repetirse). La siguiente columna indica la media del renglón. La media de los 1000 renglones fue 0.040553469. La siguiente columna muestra la diferencia de la media del renglón menos la media de los 1000 renglones. La última columna es una repetición de la anterior columna pero con los valores ordenados. De estos últimos valores ordenados se obtiene el valor del renglón 50 (como el percentil 5%) y del renglón 950 (como el percentil 95%). Entonces, estos valores de diferencias fueron, respectivamente -0.00285567 y 0.002724331.

Asimismo, se pueden ordenar los valores de las medias, y se obtiene que los valores correspondientes al percentil 5% y al percentil 95% con 0.0376978 y 0.0432778, respectivamente. Estos valores dan lo que se llama el “Intervalo de Confianza Percentil” al 90%, que es: (0.0376978, 0432778), lo que quiere decir que con este método se estima que el valor medio real puede estar en este intervalo, con un 90% de probabilidad.

Entonces, el valor medio real puede estar, con un 90% de probabilidad, 7.72% abajo, o 5.93% arriba del valor obtenido de las 10 muestras, que es el valor 0.0408551. Lo anterior se calcula como:

$$100x\frac{0.0408551-0.0376978}{0.0408551}=7.72\%$$

$$100x\frac{0.0408551-0.0432778}{0.0408551}=-5.93\%$$

Apéndice I. Ejemplo de la Obtención del Valor Promedio con el Método del Percentil Bootstrap de Intervalos de Confianza

	0.051179	0.043361	0.043658	0.030275	0.04805	0.042889	0.039436	0.041037	0.035677	0.032989	MediaEst i	Diferencia	Copia-Ordena
1	0	0	0	0	1	2	2	0	2	3	0.0383021	-0.002251369	-0.00516457
2	0	0	0	0	1	2	2	3	2	0	0.0407165	0.000163031	-0.00502147
3	0	0	0	0	1	3	1	3	1	1	0.040793	0.000239531	-0.00460347
4	0	0	0	0	1	3	2	1	2	1	0.0400969	-0.000456569	-0.00458177
5	0	0	0	0	1	3	2	2	0	2	0.0403641	-0.000189369	-0.00450957
6	0	0	0	0	2	0	1	3	1	3	0.0393291	-0.001224369	-0.00446457
7	0	0	0	0	2	3	1	1	2	1	0.0409583	0.000404831	-0.00440857
8	0	0	0	0	3	0	2	2	2	1	0.0409439	0.000390431	-0.00422727
9	0	0	0	1	0	1	2	3	1	2	0.0376802	-0.002873269	-0.00421827
10	0	0	0	1	0	2	2	2	2	1	0.0381342	-0.002419269	-0.00408317
11	0	0	0	1	1	1	1	2	2	2	0.0380056	-0.002547869	-0.00400937
12	0	0	0	1	1	2	1	2	2	1	0.0389956	-0.001557869	-0.00398937
13	0	0	0	1	1	2	1	3	0	2	0.0392628	-0.001290669	-0.00395877
14	0	0	0	1	1	2	1	3	2	0	0.0398004	-0.000753069	-0.00394887
15	0	0	0	1	1	2	2	1	1	2	0.0385667	-0.001986769	-0.00392787
16	0	0	0	1	2	0	1	1	2	3	0.0377169	-0.002836569	-0.00391277
17	0	0	0	1	2	0	3	0	2	2	0.0382015	-0.002351969	-0.00380417
18	0	0	0	1	2	1	1	1	2	2	0.0387069	-0.001846569	-0.00372177
19	0	0	0	1	2	1	3	0	0	3	0.0386539	-0.001899569	-0.00368737
20	0	0	0	1	2	1	3	0	3	0	0.0394603	-0.001093169	-0.00365647
21	0	0	0	1	2	2	3	1	0	1	0.0404487	-0.000104769	-0.00364407
22	0	0	0	1	2	3	0	2	2	0	0.040847	0.000293531	-0.00360377
23	0	0	0	1	3	1	3	0	1	1	0.0404288	-0.000124669	-0.00354717
24	0	0	0	2	0	1	2	0	3	2	0.035532	-0.005021469	-0.00353187
25	0	0	0	2	0	1	2	2	0	3	0.0363352	-0.004218269	-0.00343887
26	0	0	0	2	1	1	1	2	1	2	0.0374654	-0.003088069	-0.00343747
27	0	0	0	2	1	2	0	2	2	1	0.0380795	-0.002473969	-0.00341857
28	0	0	0	2	1	2	1	0	2	2	0.0371146	-0.003438869	-0.00340867
29	0	0	0	2	2	0	1	0	2	3	0.0366407	-0.003912769	-0.00337807
30	0	0	0	2	2	0	2	0	3	1	0.0375542	-0.002999269	-0.00331477
31	0	0	0	2	2	0	2	2	1	1	0.0386262	-0.001927269	-0.00328717
32	0	0	0	2	2	1	1	0	2	2	0.0376307	-0.002922769	-0.00323997
33	0	0	0	2	2	1	2	1	0	2	0.0385426	-0.002010869	-0.00309537
34	0	0	0	2	2	2	2	0	2	0	0.0392654	-0.001288069	-0.00308807
35	0	0	0	2	2	3	0	1	1	1	0.039502	-0.001051469	-0.00303977
36	0	0	0	2	2	3	2	1	0	0	0.0405226	-3.08691E-05	-0.00299927
37	0	0	0	2	3	1	0	2	1	1	0.0398329	-0.000720569	-0.00296947
38	0	0	0	2	3	1	2	1	1	0	0.0403175	-0.000235969	-0.00294397
39	0	0	0	3	0	2	3	0	0	2	0.0360889	-0.004464569	-0.00293977
40	0	0	0	3	1	0	2	1	1	2	0.0360439	-0.004509569	-0.00293307
41	0	0	0	3	1	2	1	1	2	0	0.037648	-0.002905469	-0.00293047
42	0	0	0	3	2	3	1	1	0	0	0.0396065	-0.000946969	-0.00292277
43	0	0	0	3	2	3	2	0	0	0	0.0394464	-0.001107069	-0.00291127
44	0	0	1	0	0	1	2	1	2	3	0.0376777	-0.002875769	-0.00290887
45	0	0	1	0	1	0	3	2	2	1	0.0396433	-0.000910169	-0.00290547
46	0	0	1	0	1	1	2	2	3	0	0.0402574	-0.000296069	-0.00288807
47	0	0	1	0	1	2	2	1	2	1	0.0401738	-0.000379669	-0.00287577
48	0	0	1	0	2	0	2	2	2	1	0.0405047	-4.87691E-05	-0.00287327
49	0	0	1	0	2	1	0	1	3	2	0.0396693	-0.000884169	-0.00285667
50	0	0	1	0	2	2	0	2	1	2	0.0409265	0.000373031	-0.00285567
51	0	0	1	0	2	2	2	1	1	1	0.0414111	0.000857631	-0.00283657
52	0	0	1	1	0	1	2	2	1	2	0.0379423	-0.002611169	-0.00283627
53	0	0	1	1	0	2	2	2	0	2	0.0386635	-0.001889969	-0.00277417
54	0	0	1	1	0	2	3	2	0	1	0.0393082	-0.001245269	-0.00272797
55	0	0	1	1	1	0	2	1	2	2	0.0379224	-0.002631069	-0.00272247
56	0	0	1	1	1	1	0	1	2	3	0.037623	-0.002930469	-0.00270287
57	0	0	1	1	1	1	2	1	1	2	0.0386436	-0.001909869	-0.00268657
58	0	0	1	1	1	1	2	1	2	1	0.0389124	-0.001641069	-0.00267317
59	0	0	1	1	1	1	2	2	0	2	0.0391796	-0.001373869	-0.00264727
60	0	0	1	1	1	2	2	1	0	2	0.0393648	-0.001188669	-0.00263107
61	0	0	1	1	1	3	0	0	2	2	0.0387982	-0.001755269	-0.00261117
62	0	0	1	1	2	2	1	1	1	1	0.040495	-5.84691E-05	-0.00259367
63	0	0	1	1	2	2	2	1	1	0	0.0411397	0.000586231	-0.00258697
64	0	0	1	1	2	3	0	0	2	1	0.0403043	-0.000249169	-0.00258427
65	0	0	1	1	3	0	0	2	2	1	0.04045	-0.000103469	-0.00257207
66	0	0	1	2	0	0	2	1	2	2	0.0361449	-0.004408569	-0.00256487
67	0	0	1	2	0	0	2	2	2	1	0.0369497	-0.003603769	-0.00255667
68	0	0	1	2	0	1	1	3	1	1	0.037831	-0.002722469	-0.00254787
69	0	0	1	2	0	1	2	1	1	2	0.0368661	-0.003687369	-0.00253727
70	0	0	1	2	0	1	2	1	2	1	0.0371349	-0.003418569	-0.00247507
71	0	0	1	2	0	2	1	2	1	1	0.0380162	-0.002537269	-0.00247397
72	0	0	1	2	1	0	1	0	3	2	0.0364703	-0.004083169	-0.00245217
73	0	0	1	2	1	0	1	1	2	2	0.0370063	-0.003547169	-0.00243197
74	0	0	1	2	1	1	0	1	3	1	0.0376204	-0.002933069	-0.00242267
75	0	0	1	2	1	1	2	1	2	0	0.038641	-0.001912469	-0.00241927

Tabla 23. Se muestran 75 de los 1000 renglones de la tabla completa con la que se ejemplifica el método del percentil Bootstrap de Intervalos de Confianza.

Apéndice J. Detalles sobre la Operación de ns-2 con el despachador GPS

J.1 Aspectos Generales

En este apéndice interesa especialmente analizar la operación de ns-2 con DS (Servicios Diferenciados), y usando un despachador WFQ.

El simulador ns-2 es un simulador con una gran cantidad de archivos de código. Independientemente de la revisión de muchas rutinas, en los diversos archivos de código de ns-2, para la implantación del despachador WFQ, y del Método STP, se encontró que era necesario cambiar solamente los archivos dsred.h, dsred.cc y wfq-list.h (este último no estaba incluido en el simulador ns-2 en su versión 2.27 originalmente usada ni en la 2.33 que se utilizó posteriormente). Se requería que el simulador operase con el despachador WFQ, para lo cual se tomó la propuesta de [76], y a la misma se incluyeron las modificaciones relacionadas para que el despachador operase con pesos variables. Esta operación se basaba en la operación de DS integrada en ns-2 [75].

Las clases, y sus dependencias, con las cuales se trabajó, se presentan en el diagrama de la Figura 57. La Figura 58 y la Figura 59 presentan un diagrama de la operación de las rutinas de encolamiento y salida de paquete en ns-2 operando con DS, y con el despachador WFQ. El contenido de estas figuras se va explicando en los siguientes párrafos.

De la operación normal del Scheduler del simulador se van a ejecutar procesos de encolamiento y salida de paquete de manera continuada, llamando a las rutinas enque y deque de la clase Queue. Ambas rutinas en dicha clase son virtuales y son sustituidas (Preempted) por las rutinas enque y deque correspondientes al objeto de la clase derivada de Queue, que define al tipo de colas con los que se trabaja.

Cada cola con la que se trabaja, aquí, es un sistema de colas, es decir, al referirse a una cola, realmente se está refiriendo a un sistema de colas simples. Este sistema está definido por la clase dsREDQueue, que se deriva de la clase Queue, y fue desarrollado en [75] con el objetivo de incluir la operación de DS en ns-2, considerando que cada cola componente contuviese los paquetes de una de diversas clases manejadas en DS.

Lo anterior implica que al estar operando desde un objeto de clase dsREDQueue (objeto que define un conjunto de colas simples –donde cada cola es un objeto), cuando el simulador llama a las rutinas “enque” y “deque” de clase Queue, realmente se estarán llamando a las rutinas “enque” y “deque” de la clase dsREDQueue.

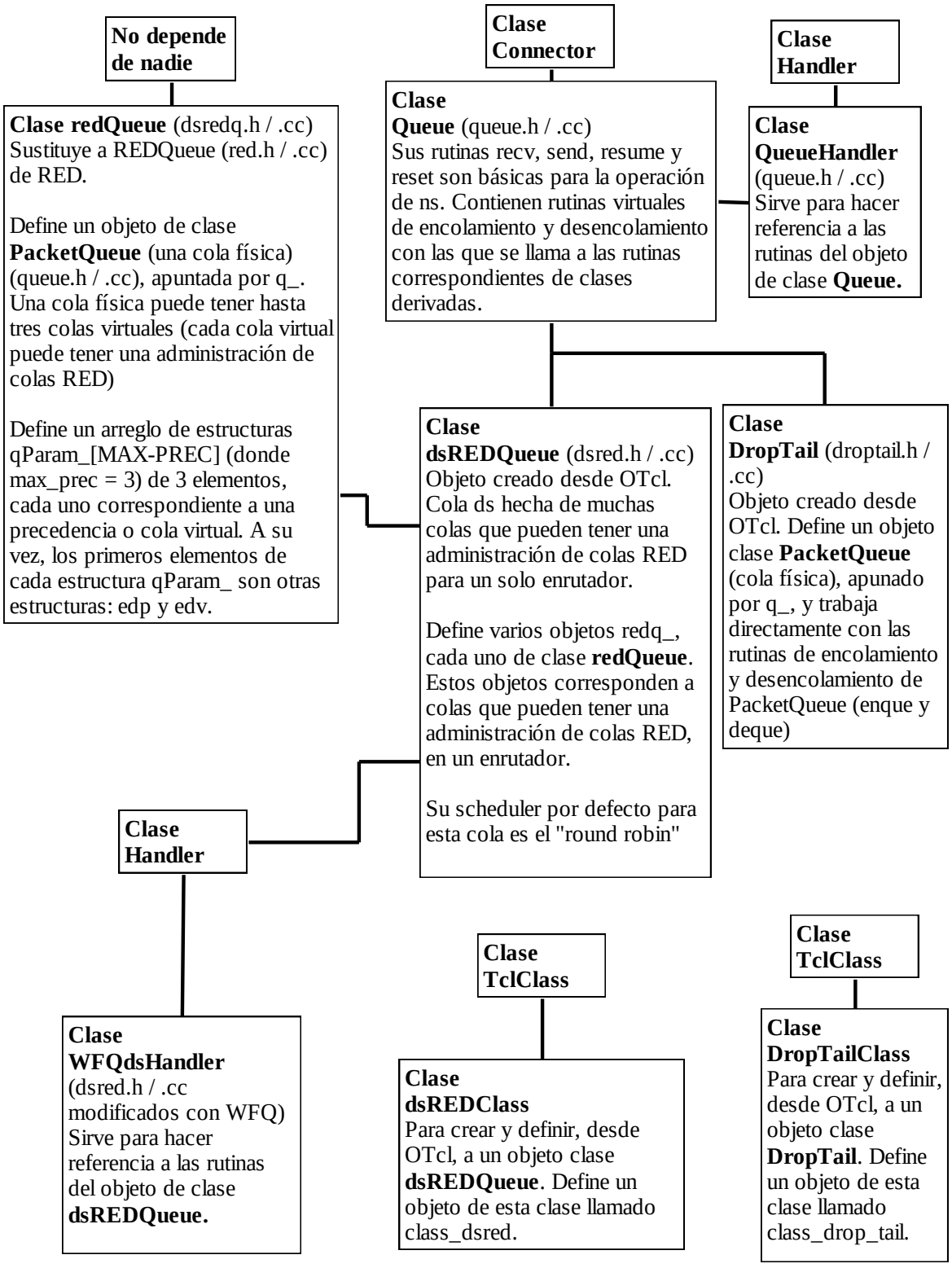


Figura 57. Diagrama de dependencias de clases de ns-2 relacionadas con las clases de interés, en la operación de DS.

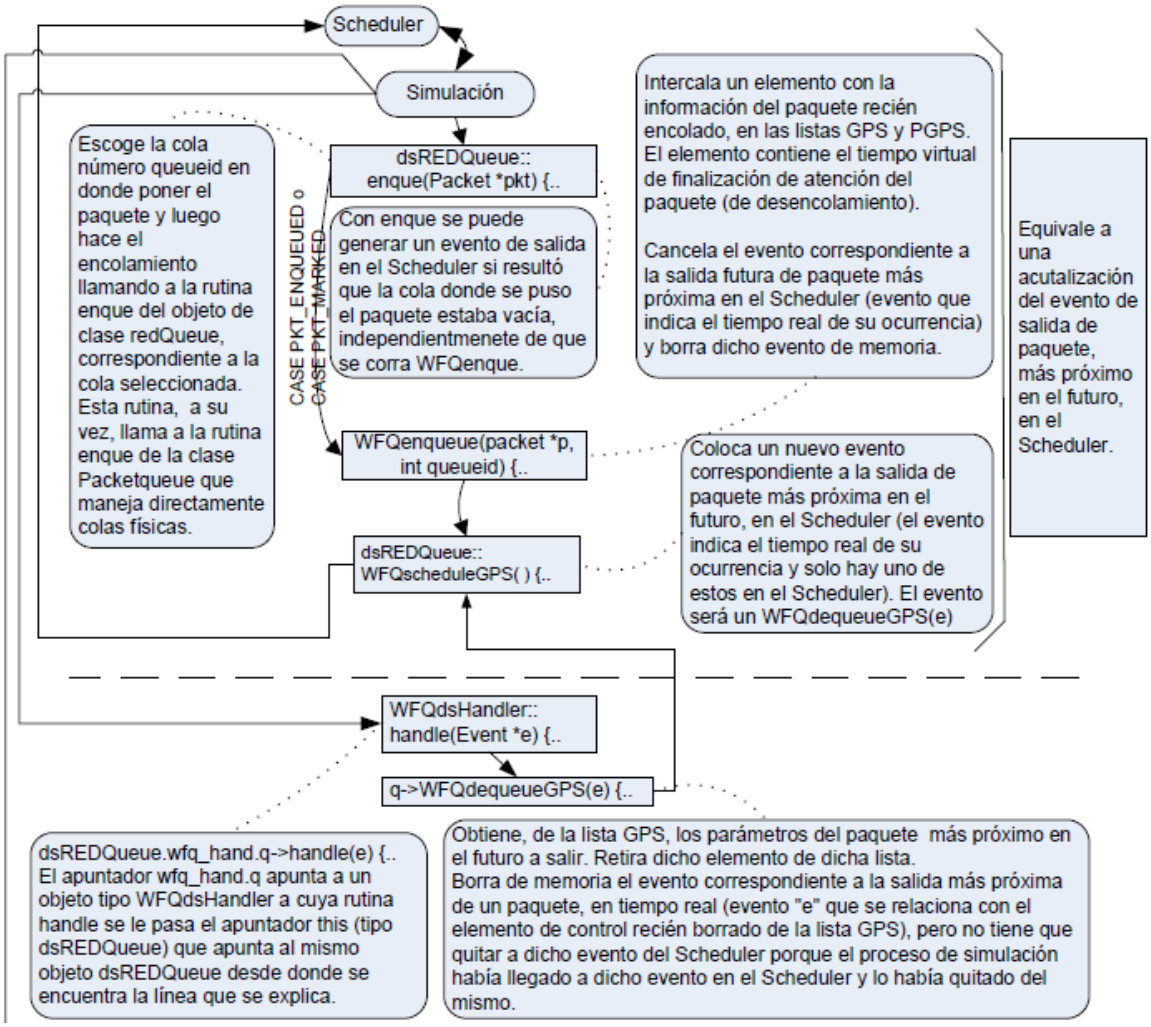


Figura 58. Diagrama de la operación de las rutinas de encolamiento y salida de paquete en ns-2, operando con DS y el despachador GPS / PGPS.

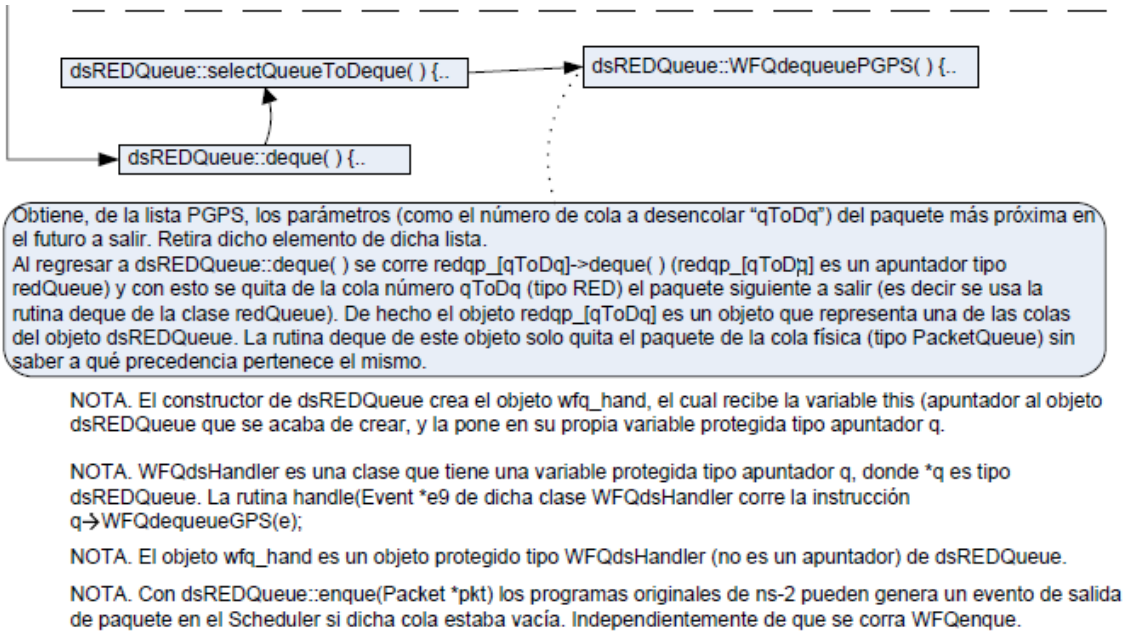


Ilustración 4. Operación de Rutinas de Encolamiento en ns Operando con DS y con Despachador WFQ.

Figura 59. Continuación de diagrama de la operación de las rutinas de encolamiento y salida de paquete en ns-2, operando con DS y el despachador GPS / PGPS.

En este trabajo se aprovecha el tipo de cola múltiple definido por dsREDQueue para implantar una operación en donde una interfaz de salida puede tener una cola por cada interfaz de entrada, en un nodo. En cada interfaz de salida en donde se quiera esta operación se maneja una cola (un objeto) de clase dsREDQueue, de la cual, cada cola componente alberga los paquetes de cada una de las interfaces de entrada del nodo.

Los diseñadores de la clase dsREDQueue, en la codificación C++ del simulador, no incluyeron al despachador WFQ, pero en [76] se integra este despachador en dicha clase. Aprovechando esta integración, para esta Tesis se realizan mayores modificaciones y mejoras a la clase dsREDQueue (trabajo original para esta Tesis) en dos formas: 1- se incluye un algoritmo para que este despachador WFQ opere con pesos variables, 2- se añade la operación de las propuestas de cambio de pesos, como la del Método STP, para que se determinen los cambios que deban tener los pesos, en la operación del despachador.

J.2 Encolamiento de Paquetes

Si al hacerse un enque (de la clase Queue) en una cola resulta que ésta estaba vacía, se debe hacer un deque de la cola y un enque en la siguiente cola "río abajo", considerando, para el deque, el tiempo que se tarde el paquete en acabar de salir (considerando por consiguiente la tasa de transmisión), y para el enque, además, el tiempo que se tarde el paquete en propagarse por el enlace de transmisión. Por esto, los eventos del enque y del deque se deben

colocar en el Scheduler del simulador. Esto se hace con la rutina recv que existe tanto para la clase queue como para la clase linkdelay (en los archivos delay.h y delay.cc).

Cuando el simulador encuentra en el Scheduler un evento de encolamiento, se ejecuta la rutina enque(Packet *pkt) de la clase Queue, lo que causa la ejecución de la rutina enque(Packet *pkt) de la clase dsREDQueue. La rutina va a registrar la llegada de un paquete al sistema de colas, específicamente a una de sus colas componentes (una cola sencilla) de clase redQueue. Así, en la rutina enque(Packet *pkt) de la clase dsREDQueue se ejecuta la rutina enque(pkt, prec, ecn) de la clase redQueue. A esta última rutina se le convoca de la siguiente forma: redqp_[queue]->enque(pkt, prec, ecn)), donde redqp_[queue] es un apuntador a un objeto de la clase redQueue que representa a una cola sencilla, y donde queue es el número de cola, del sistema de colas, en donde se va a poner el paquete. Dicho número de cola se obtuvo en la misma rutina enque de dsREDQueue, mediante una búsqueda a una tabla PHB[142].

> Nota. Para la operación de los algoritmos de las propuestas de cambio de pesos, como la del Método STP, el nodo debería establecer, en la interfaz de salida en donde opere dicho algoritmo, una cola por cada interfaz de entrada, en el nodo. Para emular esta operación, en el simulador, cada ruta de la topología planteada lleva paquetes que, dentro de la operación DS del simulador, van marcados con un valor PHB específico. Este valor indica por qué ruta viene el paquete. Con esto, en la interfaz de salida del nodo se puede saber por qué ruta viene el paquete, y a su vez, por qué interfaz de entrada viene el paquete. Con esto se puede elegir la cola de espera de la interfaz de salida, que se asocia con la correspondiente interfaz de entrada. Debe quedar claro que esta es la manera en que se aprovechan las características de operación de DS para emular la forma de operación del algoritmo, pero esto no implica que el algoritmo requiera el marcado de paquetes, ni que se esté operando con muchas clases DS.

Posteriormente, en la rutina enque de la clase dsREDQueue se ejecuta la rutina WFQenqueue(pkt, queue) de dsREDQueue, en donde se calcula el tiempo virtual de encolamiento del paquete (y también el tiempo virtual de salida del mismo paquete –recordar que en el método GPS se puede calcular este tiempo de salida de paquete desde el momento del encolamiento). Luego, la información referente al paquete se intercala en las listas de eventos

[142] La rutina original de WFQ [76] no manejaba apuntadores pero esto causaba problemas de operación en el simulador ns-2 por lo que el autor de esta Tesis cambia las variables redq_[queue] a apuntadores redqp_[queue].

GPS y PGPS, especialmente creadas para la operación de estos despachadores. Esta información es el número de cola en donde se puso el paquete y su tiempo virtual de salida de paquete. Dichas listas siempre están ordenadas poniendo a la cabeza el menor tiempo virtual.

Al haber llegado un nuevo paquete al sistema de colas de tipo dsREDQueue, el tiempo real del próximo paquete a salir en el sistema GPS debe actualizarse en el Scheduler del simulador, es decir, se cancela el evento que se tenía y se pone el nuevo evento que se tenga relativo a la salida del más próximo paquete a salir del sistema GPS, según el tiempo virtual (recordando que no debe haber simultáneamente más de un evento de este tipo en el Scheduler y que en el mismo el tiempo indicado está dado en tiempo real).

Como resumen, al poner un paquete en cola se calcula el tiempo virtual en que eso ocurre y se obtiene el tiempo virtual de salida de paquete. La información del paquete encolado, sobre su tiempo de salida, y la cola donde se pone, se guarda en dos listas especiales de objetos: la lista GPS y la lista PGPS, objetos que están ordenados en ambas listas según el valor del tiempo virtual de salida de paquete (finalización de atención). Asimismo se actualiza en el Scheduler el evento de salida de paquete (finalización de atención) del paquete más próximo a salir en el sistema GPS.

J.3 Salida de Paquetes en el Sistema PGPS

Cuando el simulador encuentra en el Scheduler un evento de salida para un paquete, lo que equivale a salida de paquete en el sistema PGPS, se ejecuta la rutina deque de la clase Queue, lo que genera se ejecute la rutina deque de la clase dsREDQueue. Esta última selecciona, mediante la rutina selectQueueToDeque de clase dsREDQueue, el número de cola (de objeto de clase redQueue) que se ha de tomar para salida de paquete. Para esto selectQueueToDeque corre la rutina WFQdequeuePGPS( ) con la que obtiene dicho número de cola para sacarle un paquete proveniente del objeto que encabeza la lista PGPS, aquél que contiene el menor tiempo virtual de la lista (tiempo de salida de un paquete), eliminando dicho objeto de la lista. Por facilidad se puede decir que la descarga, tanto de la cola como de la lista PGPS, se hace al mismo tiempo.

Es importante observar que la salida de paquete mencionada no es para el sistema GPS (la lista GPS no se tocó). Esta salida ha requerido del uso de la lista PGPS para conocer cuál es el próximo paquete a salir de entre todas las colas de clase redQueue.

El llevar una lista GPS sirve para poder calcular y actualizar el paso del tiempo virtual. Esto implica poner un evento único de salida de paquete con el menor próximo tiempo virtual en el Scheduler del simulador. En su proceso de revisión del Scheduler, la simulación encuentra este evento único y lo quita del Scheduler, corriendo, a su vez, la rutina WFQdequeueGPS(e) de dsREDQueue, la cual obtiene el objeto que encabeza la lista GPS que debe corresponder

al paquete del evento retirado, borrando dicho objeto de la lista. De este objeto toma el número de cola en donde está el paquete y su tiempo virtual de salida de paquete -lo que sirve para actualizar el tiempo virtual). Asimismo, el simulador corre también la rutina WFQscheduleGPS( ) para poner un nuevo evento en el Scheduler correspondiente a la salida del nuevo paquete con menor tiempo virtual para salir en el sistema GPS. En la lista GPS también queda ahora en la cabeza el objeto correspondiente a dicho paquete.

En suma, se debe notar que para el caso de una salida de paquete el tiempo virtual se actualiza cuando ocurre una salida de paquete en el sistema GPS y no en el sistema PGPS. Para actualizar en el Scheduler el evento del siguiente paquete a sacar en el sistema GPS se debe haber pasado por un proceso de cálculo de tiempos virtuales que inicia con la obtención del tiempo virtual actual. Justamente para eso sirve llevar el control del sistema GPS.

Recapitulación

LAS RUTINAS WFQdequeueGPS, WFQenqueueGPS no quitan ni ponen paquetes en colas. Estas rutinas interactúan con las listas de control GPS y PGPS y con el Scheduler del simulador. Dichas listas de control mantienen elementos que están ligados a los eventos de las salidas de paquetes en el sistema GPS. Cada elemento se relaciona con un paquete que está encolado, y contiene el tiempo virtual en donde el paquete debe salir del sistema, en el sistema GPS.

En el Scheduler hay, cuando mucho, un elemento del evento de la salida futura más próxima de un paquete, indicando el tiempo real de salida en la información del Scheduler. Cada vez que acontece un evento en la Simulación, ya sea de llegada o de salida de un paquete en el Scheduler, se debe actualizar el evento (único en el Scheduler) correspondiente a la próxima salida de un paquete en el sistema GPS.

La rutina dsREDQueue::WFQdequeueGPS no tiene que quitar al evento de próxima salida de paquete del Scheduler porque el proceso de simulación al llegar a dicho evento lo ha quitado, pero sí coloca un nuevo evento correspondiente a la salida de paquete más próxima en el futuro, en el Scheduler (indicando el tiempo real del evento).

La rutina dsREDQueue::WFQenqueue(packet *p, int queueid) intercala un elemento de control nuevo, para el paquete recién encolado, en las listas GPS y PGPS. Dicho elemento indica el tiempo virtual de la finalización de atención del paquete.

En la Figura 60 se pone el diagrama de la operación de estas rutinas.

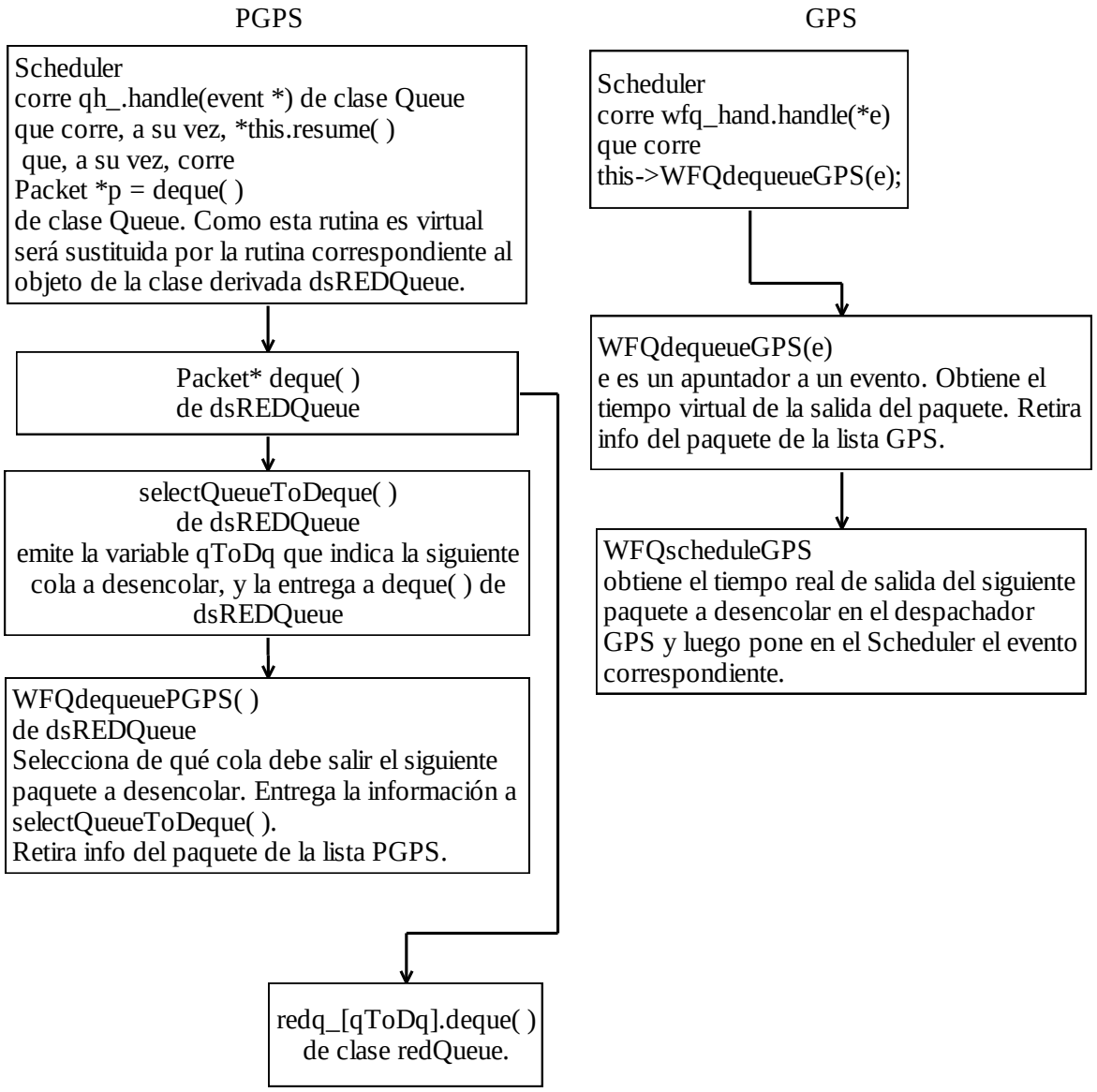


Figura 60. Salida de Paquete con despachador GPS. NOTA. El proceso redq_[qToDq].deque( ) de clase redQueue, simplemente quita el paquete de la cola física.

J.4 Detalles de Rutinas y Variables de Simulador ns-2 Relativas a la Operación de Colas

Estos detalles no incluyen las modificaciones de esta Tesis.

Cuando se escribe dsred.h modificado o dsred.cc modificado quiere decir los archivos dsred.h y dsred.cc modificados por el resultado del trabajo de MrKaic en 2001 [76] para operar originalmente con el algoritmo GPS. Estos archivos fueron originalmente introducidos al simulador ns-2 por el grupo de trabajo de Nortel [75] para que ns-2 operase con DiffServ.

Clase: PacketQueue

Clase Antecesora. TclObject.

Grupo de Archivos. De esta clase dependen todas las colase, iniciando por droptail.

Se Declara y Define en Archivo: queue.h / queue.cc

Clase Básica. Usa objetos tipo Packet (paquetes) que ya estaban definidos y usando espacio en memoria.

Esta es una clase que define una cola física FIFO. De esta clase depende cualquier tipo de cola más sofisticada que se tenga.

En la clase PacketQueue se tienen variables ("head_" y "tail_") que contienen las direcciones, inicial y final, respectivamente, de los paquetes de la cola. Los paquetes de la cola son objetos tipo Packet que están en memoria y que están en tránsito en el sistema. Cada paquete es un objeto que tiene una variable ("next_") la cual, a su vez, contiene la dirección del anterior paquete que está en cola (más próximo a ser atendido)

PacketQueue tiene algunas rutinas muy importantes como "remove" (que quita un paquete específico de la cola al irlo buscando), y las rutinas virtuales "length" (regresa el valor de la variable "len_" que es la longitud de la cola física en paquetes), ByteLength[143] (regresa el valor de la variable "Bytes_" que es la cantidad de Bytes que tiene la cola física), "enque" (pone a un paquete apuntado por un apuntador "p" en la cola física) y "deque" (quita de cola física el paquetes que correspondiente a la dirección indicada por la variable "head_" y regresa dicha dirección (ver queue.h)

Las rutinas enque y deque de PacketQueue aumentan y disminuyen, respectivamente, a la cola virtual FIFO, la contabilidad en paquetes (en la variable len_) y en Bytes (en la variable Bytes_)

Hay otra rutina "remove" de PacketQueue, definida en queue.cc, que quita un paquete específico de la cola (va buscando)

Clase: Queue

Clase Antecesora: Connector

Grupo de Archivos: queue

[143] Para obtener el número de Bytes del paquete se usa la rutina access de la clase hdr_cmn, y de dicha rutina se toma la variable size.

Se Declara y Define en Archivo: queue.h / queue.cc

Define Objetos Tipo: PacketQueue.

Queue es una clase que define un objeto tipo PacketQueue (cola física) apuntado por el apuntador pq_. Las rutinas de Queue recv y send son las primeras rutinas que se usan para poner y quitar paquetes de colas generales. Estas rutinas usan, a su vez, rutinas virtuales queue y deque de la clase Queue que causarán se llame a las rutinas queue y deque de los objetos de clases derivadas desde los cuales se hagan las llamadas a las rutinas.

Clase: DropTail

Clase Antecesora: Queue

Grupo de Archivos: droptail

Se Declara y Define en Archivo: droptail.h / droptail.cc

Define Objetos Tipo: PacketQueue. Usa rutinas de PacketQueue, de Queue.

Clase para definir una cola Droptail. Declara y define dos rutinas “enque” y “deque”, que a su vez usan, las rutinas “enque” y “deque”, consecutivamente, de la clase PacketQueue. Designa un objeto que es una cola física de tipo PacketQueue, apuntada por el apuntador q_.

Clase: dsREDQueue

Clase Antecesora: Queue

Grupo de Archivos: dsred

Se Declara y Define en Archivo: dsred.h / dsred.cc.

Define Objetos Tipo: redQueue.

Clase para definir un grupo de múltiples colas para operación con DS. Esta clase define muchas colas (muchos objetos) redq_[i], i = 1, MAX_QUEUES, de clase redQueue (ver dsred.h). Cada uno de estos objetos corresponde a una cola física. Los objetos (colas físicas) pueden haber sido declarados como colas dropTail, o como rio_c, rio_d, o wred. Esta declaración está inmersa en las rutinas de dsredq.h y dsredq.cc, donde se declara y define la clase redQueue.

En dsredq.h, al principio, se declara:

```
enum mredModeType {rio_c, rio_d, wred, dropTail};
```

Y en dsredq.h, dentro de la definición de la clase redQueue, se declara

```
mredModeType mredMode;
```

En dsredq.cc, se tiene la definición del constructor de la clase redQueue, en donde se tiene la instrucción:

mredMode = rio_c;

todos estos modos tienen diferente forma de calcular su promedio de longitud de cola (ver dsredq.cc—).

En dsred.cc, se tiene la instrucción que inicia la definición de la rutina para cambiar el modo de la cola.

```
void setMREDMode(const char* mode)
```

En dsred.cc se tiene la instrucción para la interacción con Tcl:

```
  if (strcmp(argv[1], "setMREDMode") == 0) {
```

Si mredMode es rio_c, rio_d o wred, entonces la cola física redq_[i] podrá tener varias colas virtuales.

Cada cola física está en espera de ser atendida, en un enrutador. Cada cola virtual de una cola física es una cola RED en varias versiones.

Esta clase tiene una rutina enque que a su vez llama a la rutina enque de la clase redQueue.

Esta rutina tiene la siguienteS líneas de interés:

```
switch(redq_[eq_id].enque(pkt, prec, ecn)) {
  case PKT_ENQUEUED:
    break;
  case PKT_DROPPED:
    stats.drops_CP[codePt]++;
    stats.drops++;
    drop(pkt);
         break;
  case PKT_EDROPPED:
    stats.edrops_CP[codePt]++;
    stats.edrops++;
    edrop(pkt);
    break;
  case PKT_MARKED:
    hf->ce() = 1;   // mark Congestion Experienced bit
```

```
    break;
  default:
    break;
  }
}
```

Sobre las variables ecn y ecn_ (Explicit Congestion Notification)

La operación de ecn con relación a los resultados de la ejecución de la rutina de encolamiento: PKT_ENQUEUED, PKT_DROPPED, PKT_EDROPPED, PKT_MARKED se dan en la página **¡Error! Marcador no definido.** y subsiguientes.

La variable ecn_ se declara en la clase dsREDQueue (ver dsred.h) con la línea:

```
"int ecn_; // used for ECN (Explicit Congestion Notification)"
```

En dsred.cc existe una línea:

```
"bind_bool("ecn_", &ecn_);"
```

La cual liga a la variable ecn_ de OTcl con la variable ecn_ de C++.

También, en la rutina enque de dsREDQueue (Ver dsred.cc) se lleva una variable ecn que se relaciona con ecn_. Se puede ver esto en la línea:

```
"  if (ecn_ && hf->ect()) ecn = 1;"
```

Luego se llama a la rutina enque del objeto redq_ de clase redQueue, pasándole el parámetro ecn, mediante la siguiente instrucción:

```
switch(redq_[eq_id].enque(pkt, prec, ecn)) {    (ver dsred.cc)
```

Sobre el Despachador

En dsred.cc se fija la variable schedMode = schedModeRR (Round Robin) Posiblemente dicho valor para la variable schedMode puede cambiar mediante la interacción con el usuario con Tcl.

En RR (schedMode = SchedModeRR) (Round Robin) se selecciona la cola siguiente de la que se había atendido antes, y si hay una cola vacía se sigue con la siguiente hasta encontrar una cola no vacía (parece que si todas las colas están vacías no se prevé una acción especial de esta rutina al respecto)

En WRR (schedMode = SchedModeWRR) (Weighted Round Robin) se envían tantos paquetes seguidos de cada cola como el peso que tenga la cola (llevado en queueWeight[qToDq] – donde qToDq es la cola que le toca ser atendida por el despachador) (se ve que el peso debe corresponder al número de paquetes seguidos que se quiere que se despachen y debe ser un entero) Si una cola está vacía la atención se va hacia la siguiente cola. La rutina no da prevención si todas las colas están vacías.

Hay otros despachadores como schedMode = schedModeWIRR y schedMode = schedModePRI (Priority Queueing) que no me interesan por el momento. (Ver todo esto en dsred.cc)

En dsred.cc hay una rutina que inicia con la siguiente línea:

```
void dsREDQueue::setSchedularMode(const char* schedtype) {
```

para poner el tipo de modo de despachador. En dsred.cc se tiene la instrucción que inicia el "if" para la relación con Tcl para obtener el valor del tipo de despachador:

```
 if (strcmp(argv[1], "setSchedularMode") == 0) {
```

Hay una rutina dsREDQueue::addQueueWeights (int queueNum, int weight) (en dsred.cc). Se utiliza una ejecución de esta rutina para cada valor de peso (un valor de peso para cada cola) En dsred.cc se ven las instrucciones para la relación de esta rutina con Tcl. Se ve que los pesos entran directamente sin más procesamiento de parte de las rutinas.

Clase: redQueue

Clase Antecesora: No depende de otra clase.

Grupo de Archivos: dsred.

Se Declara y Define en Archivo: dsredq.h / dsredq.cc.

Define Objetos Tipo: PacketQueue.

Esta clase provee especificaciones para una cola física que puede manejar hasta tres colas virtuales tipo RED. Esta clase está hecha especialmente para utilizarse con DS, y hace las veces (sustituye completamente) a la clase REDQueue de RED.

Con un objeto tipo redQueue se declara, define y lleva una cola física (FIFO) que es un objeto tipo PacketQueue que es apuntado por el apuntador q_ (recordar que la clase PacketQueue define una cola física básica FIFO en la cual se basan todas las colas) (Ver dsredq.h).

En la cola física se manejarán tres colas virtuales (cada una asociada a una precedencia) Cada cola virtual es una cola RED. Se lleva el tamaño promedio de cada cola virtual para decidir

sobre su operación RED. Se lleva el tamaño físico de la cola en total. Cuando un paquete se va a encolar, éste se encola siempre al final de la cola física, es decir, su ingreso a la cola es FIFO, y al ingresar el paquete a la cola física se actualiza el tamaño promedio pesado de la cola virtual correspondiente y el tamaño actual no pesado de la cola física (ver dsredq.cc en la rutina enque de redQueue).

La clase redQueue tiene una estructura qParam[MAX_PREC] (donde MAX_PREC = 3) (o sea que la estructura tiene tres elementos donde cada elemento corresponde a una precedencia de la cola) Los primeros elementos de qParam[i], para i = 0, 1 y 2, son: edp, y edv, que son, ambos, a su vez, estructuras, y que se definen en la clase REDQueue (edp son los “early drop parameters” y “edv” son los “early drop variables”) (Ver red.h al principio)

En la clase redQueue se definen las rutinas “enque”, “deque”. Enque utiliza la rutina enque de PacketQueue.

Esta rutina enque es llamada, a su vez, por la rutina enque de dsREDQueue (ver dsred.cc)

Con la rutina enque de redQueue el paquete que llega a la cola se encola si la longitud real de la cola FIFO no ha llegado a su límite, y si además se cumple una de los tres casos siguientes, exclusivos:

1- Se está en modo de administración de congestión de cola “dropTail” y lalongitud real de la cola es menor al parámetro minth (ver la variable correspondiente de dsredq.cc)

2- Se está en cualquier otro modo de administración de congestión y la variable ecn está prendida (posteriormente se podría marcar el paquete como desfavorecido según procedimientos de probabilidad)

3- Se está en cualquier otro modo de administración de congestión, la variable ecn está apagada, pero mediante los métodos RED se ha determinado que el paquete no va a ser marcado o tirado.

Una explicación general del anterior procedimiento es:

De dsredq.cc. Para los casos rioc, riod, wred, para la rutina redQueue::eque(Packet *pkt, int pret, int ect) { ...

Si ecn está encendido un paquete siempre se pone en cola (antes de revisar qué pasará con el tamaño de la cola) de otra forma no se hace.

Cuando un paquete es designado para ser tirado o degradado, por probabilidad (cuando el tamaño promedio de la cola estaba entre minth y maxth), la variable ecn hace la diferencia

entre regresar el mensaje PKT_MARKED (cuando ecn estaba encendido) o PKT_EDROPPED (cuando ecn estaba apagado)

Cuando un paquete es designado para ser tirado o degradado, en el caso de que el tamaño promedio de la cola fuese mayor a maxth, la variable ecn hace la diferencia entre regresar el mensaje PKT_MARKED (cuando ecn estaba encendido) o PKT_DROPPED (cuando ecn estaba apagado)

En todo caso, PKT_MARKED se refiere a un paquete que ya fue puesto en cola pero que después fue marcado (degradado)

Los casos de mensajes PKT_EDROPPED o PKT_DROPPED son casos en donde los paquetes nunca fueron puestos en cola ni se pondrán. Dichos paquetes no se degradan ni se tiran porque nunca fueron puestos en cola.

En caso de que el tamaño de la cola haya sido menor a minth, el paquete se pone en cola cuando ecn estaba apagado (pero no se vuelve a poner en cola si ecn estaba encendido porque ya se había puesto en cola)

Para cualquier paquete que se haya puesto en cola, y que no haya sido designado para tirarse se regresará el mensaje PKT_ENQUEUED.

Entonces:

PKT_MARKED quiere decir que el paquete sí se encoló pero que se seleccionó de forma desfavorable.

PKT_EDROPPED quiere decir que el paquete no se encoló y se seleccionó de forma desfavorable probabilísticamente, por lo que ya no se encolará.

PKT_DROPPED quiere decir casi lo mismo que PCKT_EDROPPED, pero en este caso la selección se hizo de forma no probabilística (el tamaño promedio de la cola virtual sobrepasó thmax)

A continuación se da la operación con más detalle:

Cada cola virtual tiene una longitud real actual llevada por la variable “qParam_[prec].qlen”, donde “prec” es el número de precedencia asociado a dicha cola virtual (de 0, 1 ó 2) La variable qlen es la longitud de la cola real actual, en “paquetes” (no es la longitud pesada) (Ver. dsredq.h).

El tamaño real de la cola física FIFO, en paquetes, que se lleva en q_->length(), donde q_ es un apuntador a la cola física FIFO, que es un objeto PacketQueue.

Si el tamaño real de la cola física FIFO, en paquetes, que se lleva en q_->length(), es mayor o igual al tamaño máximo que se puede tener en paquetes de dicha cola física, lo que se lleva en la variable qlim (ver dsredq.h), entonces el paquete no se encola, no se hace nada, y la rutina termina regresando el valor PKT_DROPPED (PKT_DROPPED y las otras posibilidades de regreso de esta rutina enque de redQueue deben estar asociadas a valores pues la rutina está definida como para regresar enteros) SI ESTÁN ASOCIADAS. En dsred.h se define

```
#define PKT_MARKED 3
#define PKT_EDROPPED 2
#define PKT_ENQUEUED 1
#define PKT_DROPPED 0
```

Si el tamaño real de la cola física en paquetes es menor al máximo tolerado qlim entonces se hace lo siguiente:

Si el modo de administración de congestión de colas que se emplea es “dropTail” (llevado en la variable “mredMode”) entonces se realiza la llamada a la rutina enque de PacketQueue (para poner el paquete en la cola física FIFO) que se realiza con la instrucción “q_->enque(pkt);”, donde debe recordarse que q_ es un apuntador a la cola física FIFO y pkt es un apuntador a un objeto tipo Packet (al paquete) Al final, esta rutina concluye regresando el valor PKT_ENQUEUED.

Para los otros casos de administración de congestión de colas, como: “rio_c”, “rio_d”, o “wred”, a continuación, se calcula la longitud promedio de las colas virtuales (en paquetes) Dicha longitud se ubica en la variable qParam_[prec].edv_.v_ave (dónde prec indica la precedencia o cola virtual que se lleva) Dicho cálculo promedio se realiza según el modo de administración de congestión de colas que se lleve. Para esto se utilizan los valores de límites mínimo y máximo que se han fijado para cada cola virtual, llevados en las variables qParam_[prec].edp_.th_min y qParam_[prec].edp_.th_max.

Una vez obtenida esta longitud promedio, con la metodología de RED se decide si el paquete se ha de tirar o no o marcar o no, de la siguiente forma.

Si ecn está prendida, entonces sí se llama a la rutina enque de Packetqueue (para poner el paquete en la cola física FIFO) con la misma instrucción que aquí se ha comentado, y sin importar el tipo de otro casos de encolamiento mredMode se haya tenido, pero en este caso se aumenta, en uno, el valor real de la cola virtual, en paquetes, en donde se ponga el paquete, que se lleva en la variable qParam_[prec].qlen (longitud en paquetes)

Si el tamaño promedio de la cola, en paquetes, está entre qParam_[prec].edp_.th_min y qParam_[prec].edp_.th_max (parámetros de RED), y si por probabilidad resulta que el paquete

debe ser tirado, entonces, sí ecn estaba prendido la rutina concluye regresando el valor PKT_MARKED, y si ecn estaba apagado la rutina concluye regresando el valor PKT_EDROPPED. Si el tamaño promedio de la cola es mayor a qParam_[prec].edp_.th_max entonces, si ecn estaba prendido la rutina concluye regresando el valor PKT_MARKED, y si ecn estaba apagado la rutina concluye regresando el valor PKT_DROPPED.

Si el tamaño promedio de la cola, en paquetes, es menor a qParam_[prec].edp_th_min, entonces el paquete no se marca para tirar. En este caso, si ecn estaba prendido entonces la rutina concluye regresando el valor PKT_ENQUEUED (porque el paquete ya había sido encolado en la cola física y virtual correspondiente), y si ecn estaba apagado entonces de cualquier forma se llama a la rutina enque de PacketQueue (para encolar el paquete en la cola física FIFO) y también se aumenta en uno el tamaño en paquetes de la cola virtual correspondiente, y regresándose, también, el valor de PKT_ENQUEUED.

Rutina calcAvg(prec, m+1)

La rutina calcAvg(prec, m+1) (ver dsredq.cc) no regresa valor pero pone el promedio de la longitud de la cola virtual, prec, en paquetes, en qParam_[i].edv_.v_ave, donde i puede ser el número de la cola virtual actual prec, que puede estar entre 0 a numPrec – 1 (numPrec es el número máximo de precedencias que se usa), dependiendo del tipo de atención de congestión que se tenga, rio_c, rio_d, o wred. Para rio_d y rio_c se actualiza el tamaño promedio de la cola virtual, en paquetes, usando los tamaños reales, en paquetes, de las colas virtuales qParam_[i].qlen (ver dsredq.h), y para wred se actualizan el tamaño promedio de las colas usando el tamaño total de la cola FÍSICA FIFO, en paquetes, llevado en q_->length() (Ver queue.h en donde la rutina lenght() regresa la variable len_ que es el tamaño de la cola FIFO en paquetes). Las colas que se actualizan son: para el caso de rio_c se hace para las cola i de 0 a prec, para rio_d se hace para la cola prec, y para wred se hace para i de 0 a numPrec – 1.

Para el cálculo de esta longitud promedio, en paquetes, se utiliza la propuesta del artículo de Sally Floyd, donde el peso para actualizar la longitud es “qParam_[prec].edp_.q_w” (Ver dsredq.cc), y la variable *m* corresponde a la variable *m* también del artículo de Sally Floyd (*m* se usa cuando la cola se ha quedado vacía) Dicho valor promedio se queda en el parámetro qParam_[prec].edv.v_ave (Ver red.h)

En dsredq.h se define qParam como una estructura, y en la clase redQueue (ver dsredq.h) se define un grupo de variables qParam_[MAX_PREC], que tienen la estructura de qParam.

qParam_[prec].edv_.v_ave es el tamaño promedio de la cola virtual prec. Esta estructura d qParam tiene, a su vez, las estructuras edv (Early Drop Variables) y edp (Early Drop Parameters) que se pueden consultar en red.h)

En dsredq.cc se asigna mredMode como rio_c inicialmente.

Existen algunas rutinas que dan los valores de longitudes de colas sin tener que acceder a las variables (Ver dsredq.cc), que son.

int redQueue::getRealLenght (void) que regresa el valor de la longitud real física de la cola FIFO, en paquetes.

double int redQueue::getWeigthtedLength() da el valor promedio de toda la cola física. Me parece que esta rutina funciona bien para mredMode = rio_c y mredMode = rio_d, pero tengo mis dudas que la rutina opere bien para mredMode = wred, y desde luego no es para mredMode = dropTail, en paquetes.

double redQueue::getWeightedLenght_v(int prec) que entrega la longitud promedio de la cola virtual prec, en paquetes.

int redQueue::getReal Lenght_v(int prec) que da el valor real de la longitud de la cola virtual, en paquetes.

Existen otras dos rutinas para fijar el “packet time constant” y el “mean packet size” que se debe tener en cuenta.

En dsredq.h, al principio, se declara:

```
enum mredModeType {rio_c, rio_d, wred, dropTail};
```

Y en dsredq.h, dentro de la definición de la clase redQueue, se declara

`mredModeType mredMode;` (ver explicación de dsREDQueue en este documento para más detalles).

Clase: REDQueue.

Clase Antecesora: Queue

Grupo de Archivos: red

Se Declara y Define en Archivo: red.h / red.cc

Para colas RED. Hecha en ns-2 antes de que se incorporase DiffServ en ns.

Esta clase no se analiza aquí porque para DS es sustituida completamente por la clase redQueue. Las estructuras iniciales definidas en red.h son tomadas para definir la clase redQueue en dsredq.h y dsredq.cc.

Clase: dsREDClass

Clase Antecesora: TclClass.

Grupo de Archivos: dsred

Se Declara y Define en Archivo: dsred.cc

Liga la nueva clase dsREDQueue con una clase del mismo nombre de Tcl. Contiene una rutina llamada create que crea en memoria espacio para un objeto llamado dsREDQueue en Tcl, y regresa, en C++, un apuntador a dicho objeto (el ancestro de dicho objeto es TclObject), por eso la llamada a dicha rutina es:

```
TclObject* create(int, const char*const*) {
  return (new dsREDQueue);
}
```

Y crea el objeto class_dsred tipo dsREDClass, en C++.

Clase: QueueHandler

Clase Antecesora: Handler.

Grupo de Archivos: queue.

Se Declara y Define en Archivo: queue.h / queue.cc

Cuando se crea un objeto tipo Queue se reserva en memoria espacio para sus variables (no para sus rutinas que solo están en memoria una vez) como parte de la estructura de la clase Queue. Una de las variables del objeto es el objeto protegido qh_ que es de tipo QueueHandler (ver queue.h). A continuación se ejecuta el constructor de Queue para dicho objeto (ver queue.cc) en donde qh_ recibe el valor *this (this es el apuntador al objeto Queue) por lo que qh_ recibe al mismo objeto tipo Queue. El constructor del objeto tipo Queue hace otras cosas como varios binds. Al crearse el objeto qh_ (tipo QueueHandler), a su vez se reserva espacio en memoria para sus variables (similar a como se hizo con el objeto tipo Queue referido) y se ejecuta el constructor de QueueHandler para qh_, y finalmente la variable privada queue_, tipo Queue, del constructor de qh_, acaba referenciada al objeto tipo Queue creado inicialmente.

Entonces, cuando se corre la rutina pública handle(Event*) de QueueHandler se corre la rutina pública resume de queue_ (ver queue.cc) con la instrucción única:

```
queue_.resume();
```

Dicha rutina tiene un argumento de entrada Event* pero no lo utiliza para nada.

Clase: WFQdsHandler

Clase Antecesora: Handler.

Grupo de Archivos: dsREDQueue modificado.

Se Declara y Define en Archivo: dsred.h modificado / dsred.cc modificado.

Cuando se crea un objeto tipo dsREDQueue se reserva en memoria espacio para sus variables (no para sus rutinas que solo están en memoria una vez) como parte de la estructura de la clase dsREDQueue. Una de las variables del objeto es el objeto protegido wfq_hand que es de tipo WFQdsHandler (ver dsred.h modificado). A continuación se ejecuta el constructor de dsREDQueue para dicho objeto (ver dsred.cc modificado) en donde el objeto wfq_hand recibe el valor this (this es el apuntador al objeto dsREDQueue mismo por lo que el constructor de wfq_hand recibe un apuntador al objeto dsREDQueue creado). El constructor de dsREDQueue hace otras cosas como varios binds y algunas inicializaciones. Al crearse el objeto wfq_hand como tipo WFQdsHandler, a su vez, se reserva espacio en memoria para sus variables (similar a como se hizo con el objeto tipo dsREDQueue referido) y se ejecuta el constructor de WFQdsHandler para wfq_hand. En dicho constructor, el apuntador protegido q de WFQdsHandler, queda apuntando al objeto dsREDQueue creado.

La rutina pública handle(Event *e) de WFQdsHandler corre, a su vez, a la rutina pública WFQdequeueGPS de dsREDQueue (ver dsred.cc modificado) con la instrucción única:

```
q->WFQdequeueGPS (e);
```

Esto se puede hacer gracias a que q apunta al objeto dsREDQueue.

Así, el objeto dsREDQueue contiene un objeto wfq_hand (tipo WFQdsHandler) que tiene un apuntador q que apunta al mismo objeto dsREDQueue.

Clase: dsREDQueue

Clase Antecesora: Queue

Grupo de Archivos: dsREDQueue modificado.

Se Declara y Define en Archivo: dsred.h modificado / dsred.cc modificado.

DsREDQueue :: WFQenqueue (Packet *p, int queuid) { (ver dsred.cc modificado).

Esta rutina considera que ha llegado un paquete, apuntado por el apuntador *p*, a la cola *queueid*. Se calcula el tiempo virtual actual que corresponde al tiempo virtual del evento de llegada del paquete, y se pone en la variable *virt_time*. Se calcula el tiempo real correspondiente y se pone en la variable *last_vt_update*. Se calcula el tiempo virtual de finalización de atención del paquete y se pone en la variable *finish_t[queueid].* Se intercala, en las listas GPS y PGPS, el elemento de control correspondiente al paquete que acaba de llegar, que contiene el tiempo virtual de finalización de atención del paquete, *finish_t[queueid*, y el número de cola en donde está el paquete, *queueid*.

Se actualiza adecuadamente $\sum_{i \in B_j} \phi_i$ (VARIABLE *sum*) y el número de paquetes en espera en la cola de interés *queueid*.

El evento correspondiente a la partida futura más cercana de un paquete del sistema GPS, es el objeto tipo event apuntado por wfq_event (objeto que ya estaba dado al inicio de esta rutina) si dicho evento no es nulo, entonces se cancela del Scheduler porque se va a actualizar.

Ahora se corre la subrutina dsREDQueue :: WFQscheduleGPS() en donde se crea un nuevo objeto (un evento) tipo event, apuntado por wfq_event, y se calcula el valor de la variable *Next(t)* (que indica el tiempo real correspondiente al tiempo virtual futuro más próximo de finalización de atención a un paquete que está guardado en la lista GPS) con *Next(t)* se asigna en el Scheduler este evento (manejado por la variable *tmp*).

Requerimientos

Se debe saber el tiempo virtual del último evento, en la variable *virt_time* (que se representa aquí como $V(t)$) y cuyo tiempo real correspondiente llega en la variable *last_vt_update* (que se representa aquí como t) se debe tener el tiempo virtual del evento de finalización de atención del anterior paquete en la cola, que llega en la variable *finish_t[queueid]* (que se representa aquí como F_i^{k-1} donde el número de paquete anterior es el #*k* – 1 y el número de la cola es #*i*) también se requiere la variebla tipo evento wfq_EVENT.

En la rutina se ejecutan las siguientes acciones

El tiempo actual real se pone en la variable *now* (se representa aquí en las ecuaciones como $t + \tau$), y el tiempo real del último evento llega en la variable *last_vt_update.*

Se actualiza el tiempo virtual del evento de llegada del paquete a partir del tiempo virtual del evento anterior. Dicho tiempo se encuentra en la variable *virt_time*. En términos de ecuaciones se calcula $V(t+\tau)=V(t)+\frac{\tau}{\sum_{j\in B}\phi_j}$, donde *virt_time* = $V(t)$, variable que se va a actualizar para acabar siendo igual a $V(t+\tau)$ La variable del programa *sum* = $\sum_{j\in B}\phi_j$. Es correcto que *sum* aún no se actualice pues depende de las llegadas de paquetes hasta justo antes de la llegada de este paquete que acaba de llegar. El tiempo que ha pasado entre este evento de llegada y el último evento es τ = *now – last_vt_update*. Todas estas variables son variables protegidas de dsREDQueue.

Se calcula el tiempo virtual finalización de atención del paquete, $F_i^k = S_i^k + L_i^k / R\phi_i$ (ver fórmula 10 de [40]), donde $S_i^k = \max\left\{F_i^{k-1}, V\left(a_i^k\right)\right\}$ (en esta fórmula se considera que el paquete que ha llegado es el paquete número *i* que ha llegado a la cola *k* desde que la cola dejó de estar inactiva). La variable protegida de dsREDQueue, *finish_t[queueid]*, equivale a F_i^{k-1}, y, con esta fórmula, se va a actualizar para acabar siendo igual a F_i^k.

Se tiene una nota en donde se preguntan si la variable protegida *bandwidth* es realmente una variable que representa bits.

Ahora sí, se actualiza la sumatoria *sum* = $\sum_{j\in B}\phi_j$ (aquí la variable protegida *queue-Weight[queueid]* equivale a $\phi_{queueid}$), lo que significa que se le sumará el valor $\phi_{queueid}$, siempre y cuando la variable protegida *B[queueid]* hubiese sido igual a 0, es decir, que la cola *queueid* hubiese estado inactiva. *B[queueid]* es el número de paquetes en espera en la cola *queueid*. Posteriormente se suma 1 a *B[queueid]*.

Se inserta el elemento de control correspondiente al paquete apuntado por *p*, que contiene la información de las variables *queueid* y *finish_t[queueid]*, tanto en la lista PGPS como en la lista GPS, con las rutinas PGPS_list.insert_order, y GPS_list.insert_order, definidas en wfq-list.h. Dichas listas están formadas de elementos, o entradas, tipo elem (de estructura elem) de la forma dada en la Figura 61.

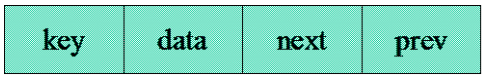


Figura 61. Formato de un elemento de la lista GPS o de la lista PGPS.

El campo *key* recibe el valor finish_t[queueid] (variable tipo double que representa el tiempo virtual, que está representado con la variable *k* en la rutina insert_order).

El campo *data* recibe a la variable *queueid* (variable tipo int que representa al número de cola física y se representada con la variable el en la rutina insert_order)

El campo *next* es un apuntador que contiene la dirección del elemento siguiente hacia atrás (hacia head) (head es el elemento de menor tiempo virtual y se considera el último de la lista – es elemento que se va a tomar en el próximo dequeue –ver WFQdequeuePGPS).

El elemento *prev* es un apuntador que contiene la dirección de elemento siguiente hacia delante (hacia tail) (tail es el elemento de mayor tiempo virtual y se considera el primero de la lista).

Cada lista (class List) contiene las variables *head* y *tail* que son apuntadores a los elementos de la lista head y tail. El elemento *tail* es aquél que tiene el tiempo virtual (el) más grande de la lista y se considera que es el primer elemento.

Si el apuntador protegido *wfq_event* a un objeto tipo evento, es nulo o cero entonces el evento relacionado se cancela y se borra de memoria.

Se ejecuta la rutina WFQscheduleGPS.

El evento correspondiente a la finalización futura más cercana de un paquete está apuntado por wfq_event (ya estaba desde el inicio de la rutina). Si dicho evento no era nulo, entonces se cancela el evento del Scheduler porque se va a actualizar.

Ahora se crea un nuevo objeto (un evento) tipo Event apuntado por wfq_event. Se calcula el valor de la variable *Next(t)* (que está en la variable *tmp* y que indica el tiempo real correspondiente al tiempo virtual futuro más próximo de finalización de atención a un paquete que está guardado en la lista GPS). Se asigna en el Scheduler dicho evento.

Nota. No hay una cuenta de longitud de colas.

No hay una relación entre los elementos de la lista y los paquetes. No es necesaria esta relación pues los paquetes, en cada cola, están en orden de llegada y se atienden de forma FIFO. Cuando se va a hacer un dequeuePGPS lo que se hace es seleccionar la próxima cola que va a ser atendida (dequeued) (con la rutina WFQdequeuePGPS) (esta rutina toma el elemento

head de la lista y la quita de ella usando las rutinas PTPS_list.get_data_min y PGPS_list. extract). Con la metodología seguida, de forma automática, se mantiene la relación entre cada paquete en las colas y los elementos correspondientes a sus tiempos virtual y real de salida, en las listas.

Clase: dsREDQueue

Clase Antecesora: Queue

Grupo de Archivos: dsREDQueue modificado.

Se Declara y Define en Archivo: dsred.h modificado / dsred.cc modificado.

`dsREDQueue::WFQdequeueGPS (Event *e) {` …

La rutina recibe el apuntador de un evento en el Scheduler, el apuntador *e*, que es el evento correspondiente a la finalización futura más cercana de un paquete, dado en tiempo real.

Para que esta rutina suceda se ha quitado un paquete de cola. Dicha partida debe corresponder al evento apuntado por *e*. La simulación debe haber llegado a dicho evento en el Scheduler y deben haberse disparado las acciones correspondientes, y dicho evento se debe haber removido del Scheduler (pero parece que no se borró de memoria porque esta rutina, WFQdequeueGPS, lo borra de memoria).

WFQdequeueGPS extrae el elemento de control correspondiente al paquete proximo a salir, de la lista GPS, y borra dicho elemento de dicha lista.

Se actualizan adecuadamente $\sum_{i \in B_j} \phi_i$ (VARIABLE *sum*) y el número de paquetes B[queueid] en espera, en la cola de interés *queueid*.

Finalmente si el sistema gps no esta oscioso, se corre la subrutina dsREDQueue :: WFQscheduleGPS() en donde se crea un nuevo objeto (un evento) tipo event, apuntado por wfq_event, en donde se va a poner en el Scheduler la partida más próxima futura, en tiempo real, del sistema GPS. Para esto, se calcula el valor de la variable *Next(t)* (que indica el tiempo real correspondiente al tiempo virtual futuro más próximo de finalización de atención a un paquete que está guardado en la lista GPS). Con *Next(t)* se asigna en el Scheduler este evento (manejado por la variable *tmp*).

dsred.h modificado / dsred.cc modificado

Todos los cambios que se hacen en la modificación de WFQ a dsred.h y dsred.cc, en donde se debe incluir el archivo wfq-list.h (para llevar la lista de elementos de control) no modifican en

lo absoluto la actualización de las longitudes de las colas cuando hay encolamientos o salidas de paquetes.

Operación del Manejador Handler

wfq_hand.handle(wfq_event)

wfq_hand es el nombre del manejador "handler" del objeto (cola) tipo dsREDQueue.

wfq_event es un apuntador a un objeto tipo evento.

La rutina handle solo tiene la línea

q->WFQdequeueGPS(e) donde e es un apuntador a una variable tipo Event.

q es un apuntador igualado al apuntador this (que apunta a la tipo dsREDQueue en operación)

La instrucción wfq_hand.handle(*e) corre (ver scheduler.cc) this->WFQdequeueGPS(e); (ver llamada a Scheduler en WFQscheduleGPS en dsred.cc).

Variable que representa el Número de Colas en una Cola DiffServ (clase dsREDQueue)

La variable numQueues_ es el número de colas con las que se trabaja. Se debe dar un valor a esta variable entre 1 y el número máximo NUM_QUEUES que es igual a 8 y que está definida así en dsred.h y dsred.h modificado.

J.5 Modificaciones al Simulador ns-2 para la Operación de GPS con Pesos que tienen Cambio Dinámico

En esta Tesis se modifica el simulador ns-2 en los siguientes términos:

La versión del simulador ns-2 que se modifica es la 2.33 (originalmente la 2.27). Estos cambios sirven para modificar la operación de ns, a fin de que el despachador GPS/PGPS opere de manera dinámica con respecto a sus pesos. Estos cambios dependen de las longitudes de las colas.

Los cambios se hacen exclusivamente en los archivos: *dsred.h*, *dsred.cc* y *wfq-list.h* (en C++). A estos archivos se les añadieron rutinas nuevas y modificaron rutinas existentes. Todas las adiciones y modificaciones están claramente documentadas en los mencionados archivos.

La forma de hacer el cambio fue tomar los archivos *dsred.h*, *dsred.cc* y *wfq-list.h* tal como fueron generados por Mrkaic [76], tal como estaban en su publicación, y cambiarlos por los archivos *dsred.h* y *dsred.cc* que estaban en la versión del simulador ns-2 que se tenía para

esta Tesis (el archivo *wfq-list.h* no existía en la versión que se tenía para los trabajos de esta Tesis). Hay que hacer notar que las modificaciones de Mrkaic no forman parte oficial de las versiones del simulador ns-2 y solo se pueden encontrar en las ligas ofrecidas en [76].

Ya que la versión del simulador ns-2 que tenía Mrkaic era anterior a aquella con la que se trabajó en esta Tesis, entonces se hizo una revisión cuidadosa para identificar diferencias en los archivos debido a cambios en las versiones, que tuviesen que corregirse para dejar los archivos compatibles con las nuevas versiones 2.27. Efectivamente se encontraron diferencias por versiones, las que se indican más adelante.

Dado el método anterior, en esta Tesis ya no se tuvo que copiar las adiciones de Mrkaic a *ns*-2.

Rutinas Modificadas de *dsred.cc* (*dsred.h*) para Implantar la Operación Dinámica en los Despachadores GPS y WRR

dsREDQueue::dsREDQueue( )

*void dsREDQueue::WFQenqueue(Packet *p, int queueid) {*

*void dsREDQueue:: WFQdequeueGPS(Event *e) {*

void dsREDQueue::reset( ) {

void dsREDQueue::selectQueueToDeque( ) {

void dsREDQueue::addQueueWeights(int queueNum, int weight) {

*int command(int argc, const char*const* argv)*

Las variables y rutinas añadidas se enlistan a continuación. Las explicaciones de las mismas se pusieron en inglés, en el trabajo de esta Tesis, en el cuerpo de los archivos mencionados.

Variables Añadidas a *dsred.cc* (*dsred.h*) para Implantar la Operación Dinámica en los Despachadores GPS y WRR

double Fmaxtmp; // Maximum virtual time of a weight change process block.

double Fmax; // Maximum virtual time.

double queueWeightpr[MAX_QUEUES]; // Present queue weight. One per queue.

double queueWeightnw[MAX_QUEUES]; // New queue weight. One per queue.

int wchpr; // Weight Change Process. If wchpr = 1 the process is on.

double start_t[MAX_QUEUES]; // Part of formula 11 of Gallager and Pareck

// article. Used in the WFQenqueue routine of the dsREDQueue class.

int set_chw_op; // If this variable = 1 the option for Weight-change Process

// is operational. This option has a 0 default value and is changed via OTcl

// instructions which call the setChgWghtOpt routine. Each dsREDQueue object

// has its own the set_chw_op variable.

Es muy importante observar la variable *set_chw_op* con la cual se puede tener, desde OTcl, comunicación con la parte compilada del simulador, sirve para indicar si se va a utilizar la forma de operación en donde los pesos cambian de forma dinámica, o no.

Rutinas Añadidas a *dsred.cc* (y *dsred.h*) para Implantar la Operación Dinámica en los Despachadores GPS y WRR

*double dsREDQueue::getWeightedLength_q(redQueue *q); // Get weighted length of*

// one physical queue.
*int dsREDQueue::getRealLength_q(redQueue *q); // Get real length of one physical*

// queue.
*double dsREDQueue::getWeightedLength_q_prec(redQueue *q, int); // Get weighted*

// length of one virtual queue of one physical queue.
*int dsREDQueue::getRealLength_q_prec(redQueue *q, int); // Get real length*

// of one virtual queue of one physical queue.
double dsREDQueue::get_sum_WeightedLengths(); // Get the sum of weighted

// lenghts of entire physical queues.
void dsREDQueue::set_new_Weights(); // The weights queueWeightnw[ ] get

// new weight values.

int dsREDQueue::change_weights(); // If result = 1 the weights must be changed.

void dsREDQueue::setChgWghtOpt(int); // Routine run from the interpreter, for

// setting the option for the Weight-Change process to operate. It modifies

// the value of the variable set_chw_op.

// The instruction from Otcl is: "$dsredq setChgWghtOpt 1", to set the

// Weight-Change Option on (assuming that dsredq is the name of the OTcl variable

// that refers to the dsREDQueue object)

Rutinas Añadidas a *wfq-list.h* (y *dsred.h*) para Implantar la Operación Dinámica en los Despachadores GPS y WRR

double get_key_max( )

Comentarios Importantes sobre algunas Modificaciones Realizadas para Corregir Diferencias en la Versión de Mrkaik con relación a la Versión 2.27 con la que se trabajó en esta Tesis

En *dsred.cc* se dejó puesta la rutina *void applyTSWMeter(Packet *pkt)* con la versión utilizada por Mrkaic [76] (antigua con relación a la versión del simulador ns-2 utilizada en esta Tesis), porque no había influencia en la operación para los usos de esta Tesis. El cambio de esta rutina a la forma de la versión usada para esta Tesis hubiese sido bastante fácil y podría hacerse en cualquier momento. La versión utilizada para esta Tesis hubiese contenido una llamada a dicha rutina con diferentes parámetros:

void dsREDQueue::applyTSWMeter(int q_id, int pkt_size). Esta rutina se llama desde *Packet* dsREDQueue::deque( )*.

La versión de Mrkaic tenía, en la rutina *int command(int argc, const char*const* argv)*, en su primera línea if: *if (strcmp(argv[1], "configQ") == 0)*, se hace una llamada a la rutina *config* definida en *dsredq.cc* (Pág. 1), que era anticuada por su parámetros, con relación a la versión del simulador ns-2 utilizada en esta Tesis, así que esta forma de llamada fue actualizada para esta Tesis, en el archivo *dsred.cc*, en donde también se escriben los comentarios al respecto, con la explicación de esta situación.

La siguiente modificación es en extremo importante, y viene de la versión 2.27, del simulador ns-2 (con la que se trabajó en esta Tesis)

En las versiones 2.33 y 2.27 de ns-2, con las que se trabajó, en el archivo *dsred.h*, en la definición de la clase dsREDQueue (cola tipo dsRED que incluye varias colas RED) se tenía la línea siguiente (donde MAX_QUEUES era una variable de sistema definida para ser igual a 8):

redQueue redq_[MAX_QUEUES]

Cuando se ejecutaban las línea de *dsred.h*, con esta instrucción no solo se estaba declarando que habría una arreglo redq_[8] (redq_[0] a redq_[8]) de objetos de clase redQueue (objeto que define las colas RED), como parte de la clase dsREDQueue, sino que además, por la forma

de la línea, se estaba reservando espacio de memoria para estos objetos. Este espacio quedaba reservado para la clase dsREDQueue pero aún no había objetos dsREDQueue que se hubiesen creado. Esto crearía un problema.

Lo anterior no creaba error de compilación, pero sí error en tiempo de ejecución por un "error de acceso a memoria" (Segmentation Fault). Era curioso que el simulador ns-2 antes de ser modificado para esta Tesis, no daba problemas, aun existiendo esta situación, pero después de la modificación, estos problemas de tiempo de ejecución de presentaron.

La corrección a este problema se hizo cambiando la instrucción indicada, a la forma:

*redQueue *redqp_[MAX_QUEUES];*

Es decir, ahora en la definición de la clase dsREDQueue se incluía solamente un arreglo de apuntadores de tipo redQueue que aquí se denotan *redqp_* (refiriéndonos con la última *p* a "Pointer"), y se quitaron las variables *redq_* .

También, en el constructor de la clase dsREDQueue, definido en el archivo *dsred.cc*, casi al final de dicha definición, antes de la última instrucción "*reset( )*", se incluyen las líneas en donde se crea espacio de memoria para las ocho variables apuntadoras *redqp_* que forman ya parte de la clase dsREDQueue. Esto se hizo con las siguientes líneas añadidas:

```
// CHANGE BY ALFREDO MATEOS -- SEE CHANGE IN the declaration of
//  dsredQueue in dsred.h.
// Now a number of MAX_QUEUES objects are created, with pointers redqp_[ ]
  for (i = 0; i < MAX_QUEUES; i++) {
   redqp_[i] = new redQueue;
 }
// END OF CHANGE BY ALFREDO MATEOS.
```

Por último, cuando dentro de las operaciones correspondientes a alguna cola dsREDQueue se hiciese una llamada a una rutina o variable de una de sus colas RED (apuntadas por redqp_[0], …, redqp_[7]), dichas llamadas ya no se harían refiriéndose al nombre de dicha cola RED, sino que se haría ahora con llamadas refiriéndose al apuntador de dicha cola RED. Por ejemplo, la línea que antes era:

```
for (i = 0; i < MAX_QUEUES; i++) redq_[i].qlim = limit();
```

Ahora ha cambiado a

```
for (i = 0; i < MAX_QUEUES; i++) redqp_[i]->qlim = limit();
```

J.6 Otros Cambios Hechos a los Archivos dsred.cc y dsred.h de ns, en su Versión 2.33 (antes 2.27), para la Implantación del Despachador GPS

La Versión del simulador ns-2 utilizada por Mrkaic [76], para incorporar la funcionalidad del despachador GPS en ns, con relación a las versiones utilizada por A. Mateos de ns, es más antigua. Por esta razón, se hicieron los siguientes cambios.

Se dejó la rutina applyTSWMeter de la versión utilizada por Mrkaic [76]. Dicha rutina es diferente en los archivos dsred.h y dsred.cc.

En la versión del simulador ns-2 utilizada por Mrkaic, en el archivo dsred.cc, se tenían las siguientes líneas, en la función dsREDQueue::command.

```
if (strcmp(argv[1], "configQ") == 0) {
  redq_[atoi(argv[2])].config(atoi(argv[3]), argv);
  return(TCL_OK);
}
```

Estas líneas fueron sustituidas por las siguientes, para que estuviesen acordes con las versiones de ns-2, 2.27 y 2.33.

```
if (strcmp(argv[1], "configQ") == 0) {
  // modification to set the parameter q_w by Thilo
  redq_[atoi(argv[2])].config(atoi(argv[3]), argc, argv);
  return(TCL_OK);
}
```

En las anteriores líneas, de debe notar que la llamada a la rutina config del objeto redq_[atoi(argv[2])] está cambiada en la versión de Mrkaic.

Casi al final del archivo desred.cc de la versión utilizada por Mrkaic, faltaban rutinas como "getAverageV, y "getCurrentV", las cuales se pusieron.

Modificación al archivo wfq-list.h el 24 de febrero de 2011

En la clase List , en su rutina insert_order(X el, double k) se hace la siguiente modificación.

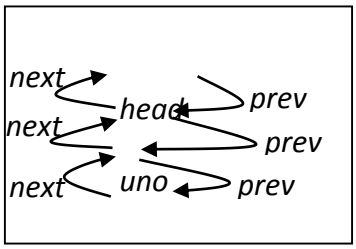
En lugar de:

```
if(p->next!=) {
 p->next->prev = tmp;
 tmp->next = p->next;
} else
```

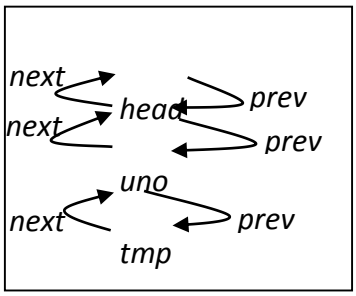
Se pone

```
if(p->next!=) {
 (p->next)->prev = tmp;
  tmp->next = p->next;
  p->next = tmp;
  tmp->prev = p;
} else
```

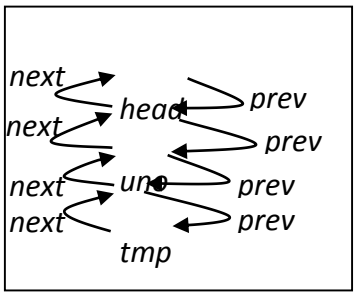
Notar que se arreglan unos paréntesis. Asimismo, lo anterior es importante cuando se va a acomodar un nuevo elemento en una lista (apuntado con el apuntador *tmp*) mediante la adición de una línea de código que se coloca antes de la línea `p->next = tmp, dado que` se carecía de ligas entre los elementos, se debía hacer un arreglo al código. Para ejemplificar, considérese que el elemento que está arriba de aquél apuntado por el apuntador *p* está apuntado por el apuntador *uno*. El primer elemento de la lista es el apuntado por el apuntador *head* y el último es el apuntado por el apuntador *tail*, y considérse que ni *uno* ni *p* coinciden con *head* y *tail* respectivamente. Así que el siguiente cuadro ejemplifica esquemáticamente las posiciones:



Considérse que va a entrar un nuevo elemento que sería apuntado por *tmp*, elemento que se supone que quedará arriba del elemento apuntado por *p*. Según lo que se tenía en el programa quedaba lo siguiente:



Entonces las líneas adicionales de corrección agregan los faltantes para que quede lo siguiente (el elemento apuntado por *uno* quedaba desligado).


next
head
prev
next
prev
next
uno
prev
next
prev
tmp

Apéndice K. Inclusión del Código para Generación de Tráfico MPEG4 en el Simulador ns-2

K.1 Generalidades

La contribución de Ashraf Matrawy y Loannis Lambadaris [70], añade la generación de tráfico MPEG4 en el simulador ns-2. Estos autores indican que la generación tiene casi las mismas estadísticas de primero y segundo orden que la traza original de Cuadros ("Frames") que se obtiene con un codificador MPEG4.

MPEG4 maneja 3 tipos de cuadros: cuadros *I*, *P*, *B*. Cada cuadro tipo *I* (Intra-Coded Frame) contiene información de imagen fija, y es el tipo que tiene la menor de las compresiones. Su decodificación no depende de otros cuadros. Cada cuadro tipo *P* (Predictively Coded Frames) tiene compresión media, y su decodificación depende del cuadro *I* anterior, o del cuadro *P* anterior. Cada cuadro tipo *B* (Bidirectionally Predicively Coded Frames) es el tipo que tiene la mayor compresión, y su decodificación depende del cuadro *I* o *P* anterior, y del cuadro *I* o *P* posterior.

Para la codificación (en el lado transmisor), y el despliegue (en el lado receptor) de los cuadros se usa el siguiente orden, según el tipo de cuadro [70]:

```
I B B P B B P B B P B B I
```

Dado que la codificación (y decodificación) de cada cuadro tipo *B* depende del cuadro tipo *I* o tipo *P* que le antecede, y del cuadro tipo *I* o tipo *P* que le sigue, el orden de transmisión de los cuadros debe cambiar, para facilitar la decodificación en el lado receptor. Este orden de transmisión es [70]:

```
I P B B P B B P B B I B B
```

De tal forma que los cuadros tipo *B* puedan decodificarse al ser recibidos.

En el escenario de trabajo de esta Tesis se utilizó tráfico con flujos MPEG4, obtenido de un generador de tráfico con casi las mismas estadísticas de primer orden (tamaño de los cuadros) y de segundo orden (correlación de los tamaños de los cuadros), con relación a las estadísticas de las trazas originales MPEG4.

K.2 Generación

Para la generación de tráfico se utiliza la metodología TES (Transform Expand Sample). La generación tiene dos fases:

1- Generación de una serie de tiempo de variables (llamadas secuencias "Background").
    a. Con distribución probabilidad marginal uniforme

b. Con autocorrelación que se varía (cambiando una función parámetro llamada densidad de probabilidad de innovación f_v "que generalmente puede ser arbitraria").

2- Generación de series de tiempo sintéticas (llamadas secuencias "Foreground") que se parecen a las muestras reales (empíricas).

a. Se usa la técnica de inversión, que permite la transformación de cualquier variable aleatoria uniforme a una con distribución arbitraria.

i. Desde series de tiempo uniformes.

ii. A partir de la inversión (aplicando una fórmula de inversión) al histograma construido a partir de la secuencia empírica de video semejante al de la serie de tiempo de entrada.

b. Se obtiene una serie con una distribución probabilidad marginal "H" y una autocorrelación que depende de f_v. La distribución de probabilidad de la serie generada se obtiene con la inversión, pero la correlación se obtiene haciendo de manera visual, interactivamente con realimentación visual, para obtener el mejor empate de f_v con la densidad de probabilidad de la autocorrelación de la serie empírica.

Para las pruebas de [70] se utiliza la serie de tiempo de entrada se obtiene codificando, a 30 *cuadros*/*s*, una emisión de la emisora BBC.

Cada variable de la serie de tiempo obtenida (serie de entrada) resulta de los bits de cada cuadro.

Se obtiene una serie de tiempo para cada tipo de cuadro, *I*, *P*, *B*.

Se obtiene el histograma y la correlación para cada secuencia, *I*, *P*, *B* (ver las figuras 3 a 5 respectivamente). Las subfiguras muestran tanto las gráficas obtenidas de los datos empíricos (en verde) como de los datos generados (en azúl). Las subfiguras de cada figura muestran:

w1 (arriba izquierda) las secuencias de tiempo.

w2 (abajo izquierda) la función de correlación.

w3 (arriba derecha) el histograma.

w4 (abajo derecha) la densidad de probabilidad de innovación.

Las series de tiempo generadas se intercalan para generar la serie final. No se dice pero me imagino que de las series de tiempo se generan tantos paquetes como sean necesarios, enviados en concordancia temporal con los envíos de datos de dichas series.

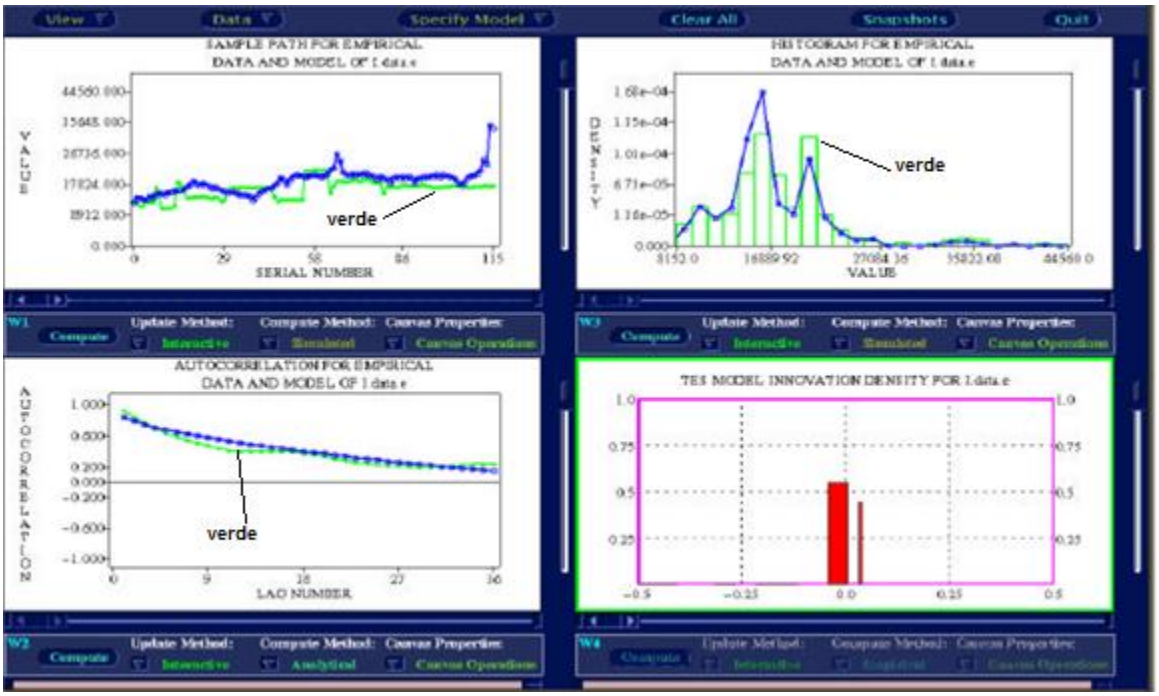


Fig. 3. Histograma y la correlación para cada secuencia de cuadros tipo *I*. Tomado de [70].

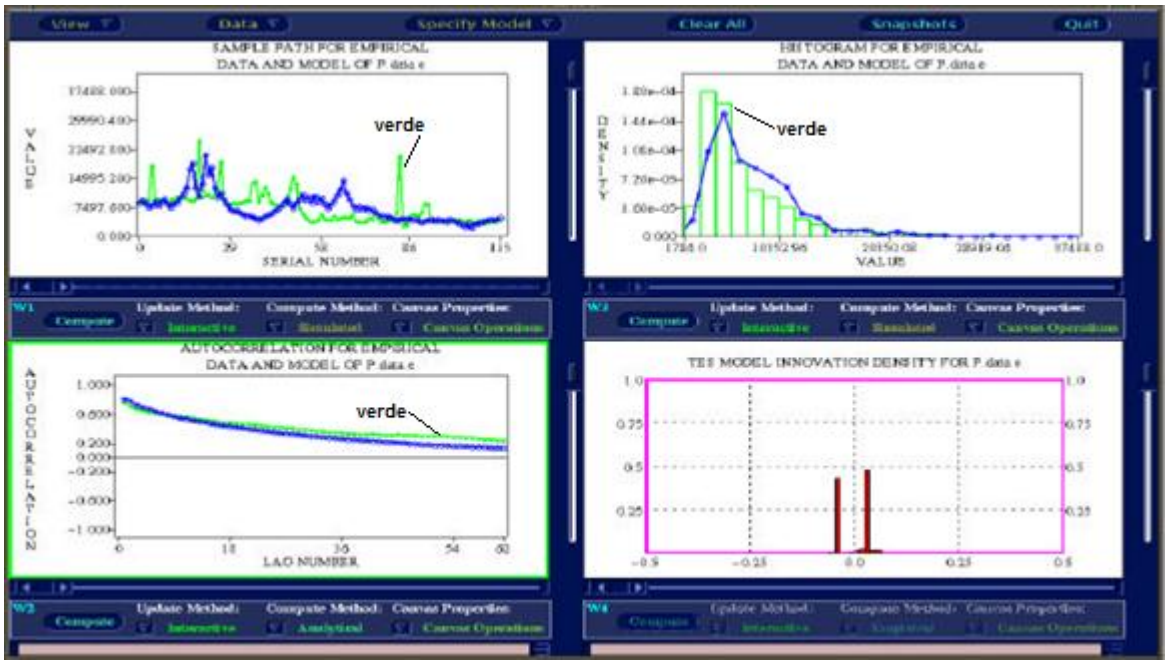


Fig. 4. Histograma y la correlación para cada secuencia de cuadros tipo *P*. Tomado de [70].

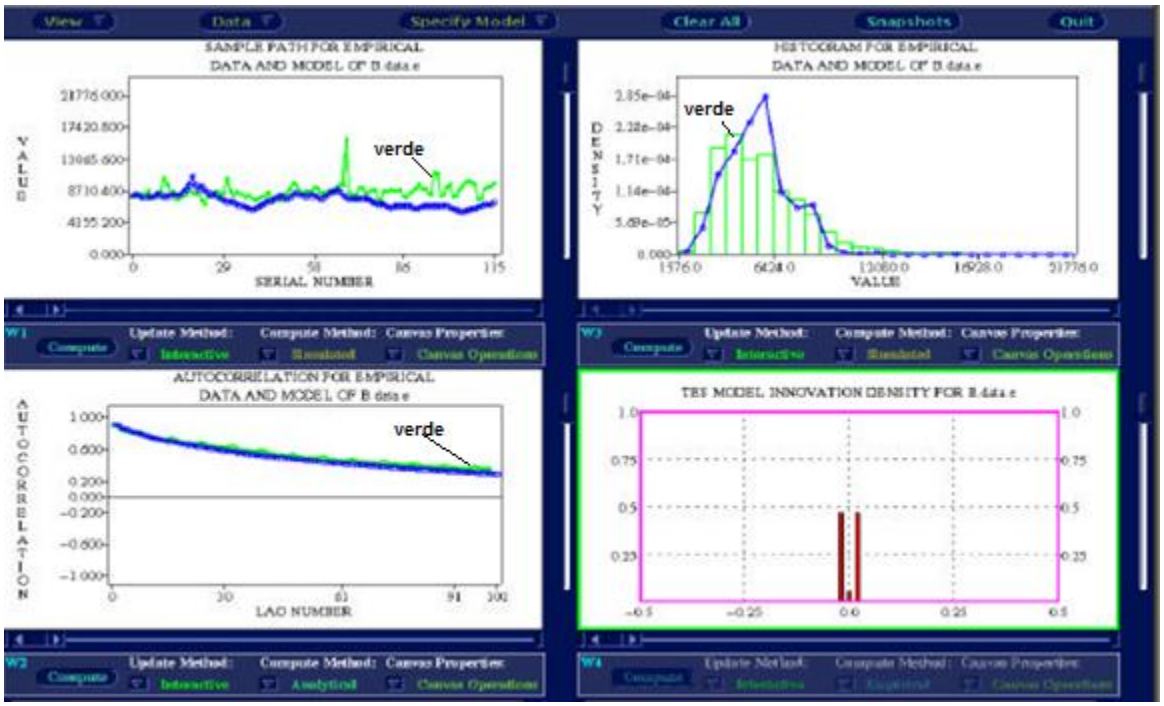


Fig 5. Histograma y la correlación para cada secuencia de cuadros tipo *B*. Tomado de [70].

K.3 Implementación en ns-2

Para encontrar los archivos y datos de esta contribución, se debe buscar en la página Web del Código Contribuido de ns-2 [94]. En dicha página, bajo el título de “Topology and Traffic Generation”, se busca “Video traffic generator based on TES (Transform Expand Sample) model of MPEG4 trace files”. Aquí hay una liga a la documentación para incluir este generador en ns-2, que es: http://www.sce.carleton.ca/~amatrawy/mpeg4/.

En los documentos encontrados en la liga, se tiene el documento mpeg4_traffic_README.txt en donde se indica que los autores saben que el código se ha puesto a operar con las versiones 2.27 de ns-2, pero que no comentarían sobre problemas de compilación en esta versión.

Sobre lo anterior, para esta Tesis se instaló ese código en la versión 2.33 de ns-2.

Se indica que el código genera cuadros (Frames) cada 1/30 *s*, pero que esto se puede cambiar en el código, y recompilar.

Se requiere poner el archivo mpeg4_traffic.cc en donde se tienen los archivos .cc para ns, e incluir el nombre de este archivo en el archivo make de ns-2, y recompilar.

En el archivo tcl/lib/ns-default.tcl se deben incluir las líneas:

```
Application/Traffic/MPEG4 set rateFactor_ 1
Application/Traffic/MPEG4 set initialSeed_ 0.5
```

Las variables rateFactor e initialSeed indicant:

rateFactor indica cuánto se quiere escalar hacia arriba o abajo la entrada de video.

intialSeed se usa para el inicio de la generación del primer cuadro (“Frame”) en el modelo.

NOTA: Sep. 2011. Después de lo anterior, es importante volver a compilar porque el archivo ns-default.tcl debe quedar integrado en ns. Esto lo dice en el archivo tclcl/tcl-object.tcl en su línea 196 en adelante.

Se requieren 6 archivos de estadísticas para uso del modelo empleado (TES -Transform Expand Sample), los cuales se deben poner en el directorio "video_model" que se inserta en donde se tienen los Scripts TCL. Los seis archivos indicados contienen información de los Cuadros (Frames) tipo I, P y B. Se tienen dos archivos por Cuadro. Los archivos, por pares, para los Cuadros I, P y B, son:

```
"./video_model/Imodel_hist_1";
"./video_model/Imodel_inv_1";

"./video_model/Pmodel_hist_1";
"./video_model/Pmodel_inv_1";
```

```
"./video_model/Bmodel_hist_1";
"./video_model/Bmodel_inv_1";
```

*hist_* es el histograma para el tipo de Cuadro.

*inv_* es la función de distribución acumulada de probabilidad para la función de innovación para este tipo de Cuadro.

Se dan referencias para estudiar el modelo TES:

B. Melamed, "An Overview of TES Processes and Modeling Methodology", in Performance Evaluation of Computer and Communication Systems, 1993

B. Melamed et al., "TES-Based Modeling for Performance Evaluation of Integrated Networks", in Proc. of INFOCOM 1992

D. Reininger et al., "Variable Bit Rate MPEG video: Characterstics, Modeling and Multiplexing", in Proc. of ITC 1994

A. Matrawy, I. Lambadaris and C. Huang, "MPEG4 Traffic Modeling using The Transform Expand Sample Methodology", in Proc. of 4th IEEE International Workshop on Networked Appliances, January 2002

K.4 Memoria de Pasos Específicos para Incluir el Generador MPEG4 en ns-2

Los siguientes pasos hacen referencia a los directorios específicos empleados para esta Tesis.

En la terminal se escribe: cd /usr/local/ns-allinone-2.33/ns-2.33

Se corre make. No debe haber compilación porque aún no se ha actualizado archivo alguno .cc y el archivo mpeg4_traffic.cc no ha sido incluido en makefile. Efectivamente no se hizo compilación alguna.

K.4.1 Procedimiento para Incluir en ns-2 la generación de tráfico MPEG4

Se guardan los archivos originales[144] de código C++, y también una copia de los nuevos archivos de generación a incluirse[145].

Ahora se pone el archivo mpeg4_traffic.cc en el directorio "tools" (justo en donde está el generador "Pareto", entre otros). Ahora, con el fin de que posteriormente se haga la compilación, se debe abrir este archivo mpeg4_traffic.cc y guardarlo para ponerle una nueva fecha.

Ahora, siguiendo en el directorio "/usr/local/ns-allinone-2.33/ns-2.33", se toma "makefile" y ahí se pone, en la penúltima línea, en donde se define la variable "OBJ_CC", la línea "tools/mpeg4_traffic.o" (línea 326).

Se compila con "make".

Se obtiene una buena compilación.

El código genera Cuadros (Frames) cada 1/30 *s*, y esta tasa se puede cambiar en el código.

Del directorio "/usr/local/ns-allinone-2.33/ns-2.33/tcl/lib" se toma el archivo "ns-default.tcl" y después de las líneas que se refieren a "Application/Traffic/", justo abajo de la línea 493, se añaden las líneas (justo como dice el archivo de instrucciones "mpeg4_traffic_README.txt"):

```
Application/Traffic/MPEG4 set rateFactor_ 1
Application/Traffic/MPEG4 set initialSeed_ 0.5
```

NOTA. Después de lo anterior, es importante volver a compilar porque este archivo ns-default.tcl deberá quedar integrado en ns. Esto lo dice en el archivo tclcl/tcl-object.tcl en su línea 196 en adelante.

En el archivo "mpeg4_traffic.cc", hay una expresión que dice:

```
frame_in_Bytes = rateFactor_*size_/8 ;
```

O sea que rateFactor_ cambia el tamaño de los cuadros (Frames).

---

[144] Protección de Archivos DiffServ. Los archivos de diffserv creados de la versión 1.5 se guardan en "/media/OS/RESULTADOS-NO-BORRAR/RESULTADOS_V1.6/Diffserv-V1.5", para protección.

[145] En el directorio "/media/OS/RESULTADOS-NO-BORRAR/RESULTADOS_V1.5/BITACORAS-CONFERENCIAS/POST-AMCS-Ago-2011/MPEG4 -TES" se tienen los archivos informativos y de código del generador de tráfico MPEG4. En el directorio "/media/OS/RESULTADOS-NO-BORRAR/RESULTADOS_V1.5/BITACORAS-CONFERENCIAS/POST-AMCS-Ago-2011/MPEG4 - TES/CODIGO" se tienen los archivos de código, entre ellos el archivo "mpeg4_traffic.cc". También ahí se tienen el archivo "mpeg4_traffic_README.txt" de instrucciones para instalación.

Las variables rateFactor_ y intialSeed_ están contenidas en líneas de código (en el archivo "mpeg4_traffic.cc") que están ligadas a variables del SCRIPT con la función "bind", o sea que su valor se puede cambiar desde el Script.

Por último, en el archivo "mpeg4_traffic.cc" hay una la línea en donde se maneja el número de *Cuadros*/*s* (en inglés *Frames*/*s*) que está por default puesta en el valor 1/30. Esta variable sí se tiene que cambiar en el mismo código del archivo:

```
inter_frame_interval_ = 1.0/30.0;   // 30 frame/sec
```

Para la inclusión de los Archivos de Estadística, en el directorio "/media/OS/RESULTADOS-NO-BORRAR/RESULTADOS_V1.5/BITACORAS-CONFERENCIAS/POST-AMCS-Ago-2011/MPEG4 - TES/CODIGO/video_model" se tienen los archivos estadísticos. Este subdirectorio "video_model" se debe copiar completo en donde estén los Scripts del programa manejado.

K.5 Revisión de Ejemplos de Uso

En el directorio "/media/OS/RESULTADOS-NO-BORRAR/RESULTADOS_V1.5/BITACORAS-CONFERENCIAS/POST-AMCS-Ago-2011/MPEG4 - TES/CODIGO" está el archivo "part-of-example.txt" que tiene un ejemplo del uso de esta generación. El archivo contiene exactamente lo siguiente:

```
This is the part of the code that uses MPEG4

set source [$ns node]
set udp0 [new Agent/UDP]
$ns attach-agent $source $udp0
set vdo [new Application/Traffic/MPEG4]
$vdo set initialSeed_ 0.4
$vdo set rateFactor_ 5
$vdo attach-agent $udp0
$ns at 1.8 "$vdo start"
```

Una parte del Script utilizado tiene las siguientes líneas, parecidas a las del ejemplo dado, anterior.

```
set udpMPEG4 [new Agent/UDP]
set nulMPEG4 [new Agent/Null]
$ns attach-agent $s1 $udpMPEG4
$ns attach-agent $d1 $nulMPEG4

set srcMPEG4 [new Application/Traffic/MPEG4]
$srcMPEG4 set initialSeed_ 0.4
$srcMPEG4 set rateFactor_ 5

$srcMPEG4 attach-agent $udpMPEG4

$ns connect $udpMPEG4 $nulMPEG4
```

```
$ns at 1.8 "$srcMPEG4 start"
$ns at $testTime "$srcMPEG4 stop"
```

Para Correr ns Ahora con MPEG4.

Para correr el Script utilizado de ns-2 (con sus argumentos), con el interés de revisar la característica del generador de tráfico MPEG4, se escribe la siguiente línea (específica de la forma de llamar al Script en el trabajo de esta Tesis):

ns topologia7-1.ns PARAM_GEN 360 95 1000 PARAM_qm 3 1 3 1 3 1 PARAM_GEN_qm 1 120 0.25 PESOS 1 1 1 1 1 1 1 1 1 N_FT_CBR 0 0 0 0 0 0 0 N_FT_PAR 0 0 0 0 0 0 0 N_FT_MPEG4 1 0 0 0 0 0 0 IMPRESN 1 0 0 ARCHIVS salida.trc

Para Obtener la Tasa de MPEG4 en ns-2, se corre:

LC_ALL=DA gawk -f tasa-mejorado-1.awk lapso=10 salida="../Arch/tasa1.txt" salida.trc

Se observa que, con un rateFactor_ de 5 la tasa es aproximadamente de 1 *Mb/s* por fuente, y con un rateFactor_ de 10 la tasa es aproximadamente de 2 *Mb/s* por fuente.

El promedio de tasa total que se obtuvo de 10 fuentes MPEG4 de las usadas fue de 6209240.44292612, es decir que por fuente se tiene un promedio de 620,924[*b*/*s*]. Se calcularon 619 *Kb/s* de promedio, para un rateFactor_ de 3. Se usó ns2.34.

Revisando los valores obtenidos, despejando el número ce X *byte*/*frame*: 30 *frame*/s *X Byte*/*frame* 8 *bit*/*Byte* = 1,000,000 *bit*/s, lo que implica que el *cuadro (*o *frame)* tiene un valor indicativo de su tamaño promedio de 4,166 *Byte*/*frame* (los tamaños de los *cuadros* varían según el histograma).

Con el uso de fuentes MPEG4 con 619 *Kb/s* (rateFactor_ de 3), pasando el tráfico de 44 fuentes por el nodo c_0 se obtiene un retraso promedio de 15 *ms* para el curso de un paquete por ese nodo, y aumentando a 46 de esas fuentes el retraso aumenta a 33 *ms*. Por esto, se propone que inicialmente por cada nodo c_0, c_1 y c_2, pasen 39 fuentes, de la siguiente forma:

Entre los nodos c_0 a c_1, y c_1 a c_2 pasaría el tráfico de 13 fuentes en el trayecto entre los nodos s_1 a d_1.

Entre los nodos c_0 a c_1 pasaría el tráfico de 13 fuentes en el trayecto entre los nodos s_2 a d_2.

Entre los nodos c_0 a c_1 pasaría el tráfico de 13 fuentes en el trayecto entre los nodos s_3 a d_3.

Entre los nodos c_1 a c_2 pasaría el tráfico de 13 fuentes en el trayecto entre los nodos de s_4 a d_4.

Entre los nodos c_1 a c_2 pasaría el tráfico de 13 fuentes en el trayecto entre los nodos de s_5 a d_5.

Entre los nodos c_2 a e_8 pasaría el tráfico de 13 fuentes en el trayecto entre los nodos de s_6 a d_6.

Entre los nodos c_2 a e_8 pasaría el tráfico de 13 fuentes en el trayecto entre los nodos de s_7 a d_7.

Desde 0[s] hasta 480[s] de una simulación se tendrían estas fuentes. En 480[s] aumentarían las fuentes, y desde 480[s] hasta 1800[s] se conservarían los números de fuentes.

Apéndice L. Aspectos de Tiempos entre Cuadros MPEG4

L.1 Generalidades

Con los parámetros seleccionados del generador de tráfico MPEG4 [70], la tasa promedio obtenida[146] por cada flujo MPEG4 fue de 0.621 *Mb*/*s*. Se calcula el número promedio de Bytes por cada cuadro como: (0.621 *Mb*/*s*) / (30 *cuadros*/*s*) / (8 *bits*/*Byte*) = 2,587 *Bytes*/*cuadro*.

Sobre los histogramas obtenidos de la serie empírica (ver Apéndice K), un cuadro tipo *P* puede llegar a tener hasta 60,000 *Bytes* (solo el 2% de los cuadros tendrían un tamaño igual o mayor). El tiempo de transmisión de un cuadro tipo *P* por un enlace de 30 *Mb*/*s* (como los usados en el escenario de trabajo de esta Tesis) podría ser tan bajo como 0.7 *ms*, para un cuadro de 2,587 *Bytes*, o de 16 *ms*, para un cuadro de 60,000 *Bytes*. Si dicho enlace se compartiese entre 36 flujos iguales, el tiempo de transmisión sería de 24.83 *ms*, para el cuadro de 2,587 *Bytes*, o de 512 *ms* para el cuadro de 60,000 *Bytes*.

El tiempo de captura de un cuadro, para su codificación, puede ser 3 *ms* [24]; y el tiempo de formación de una imagen, en la pantalla receptora, considerando una frecuencia del monitor de 60 a 100 Hz, podría tomar hasta de 17 *ms* [24].

En la Figura 62 se representa una forma simplificada de los tiempos de generación, transmisión y despliegue de los cuadros, cuando se transmiten 30 cuadros/s (el tiempo de inicio entre cuadros generados es de 33.33 *ms*), sin considerar el tiempo de decodificación de los cuadros ni el tiempo de formación de la imagen en la pantalla receptora.

[146] Al argumento rateFactor_ se le dio un valor de 3. Este argumento indica cuánto se quiere escalar hacia arriba o abajo la entrada de video.

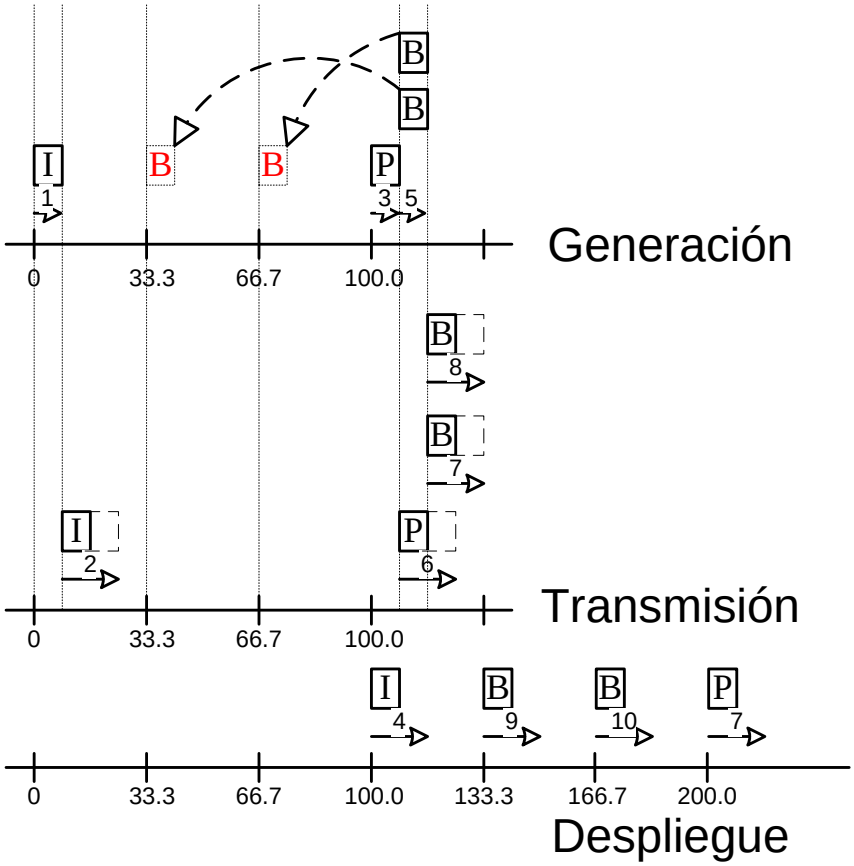


Figura 62. Tiempos de generación, transmisión y despliegue de cuadros, en *ms*. Los números indican la acción: 1- Generación de cuadro tipo *I*. 2- Transmisión del cuadro tipo *I*. 3- Generación de cuadro tipo *P*. 4- Despliegue de cuadro tipo *I*, etc.

En la Figura 62 se observa que, por la dependencia de codificación de los cuadros, los órdenes de secuencia, en la transmisión y despliegue de los mismos es diferente, por lo que el despliegue de cada cuadro (en el lado receptor) no inicia sino 100 *ms* después de que fue el inicio de la generación del mismo (en el lado transmisor), y siempre que el tiempo de generación y transmisión de un cuadro tipo *P* se mantenga dentro de 33.3 ms. El tener 100 *ms* de retraso, de extremo a extremo en una transmisión, no causa incomodidad en los usuarios de una videoconferencia [24] [25].

Apéndice M. Inclusión del Código para la operación del Servicio de Despacho de Multisalto Coordinado (CMS) en ns-2

Apéndice M. Inclusión del Código para la operación del Servicio de Despacho de Multisalto Coordinado (CMS) en ns-2

En total se modificaron los siguientes archivos:

```
packet.h    (en ns-2.33/common)
dsred.h     (en ns-2.33/diffserv)
dsred.cc    (en ns-2.33/diffserv)
dsredq.h    (en ns-2.33/diffserv)
dsredq.cc   (en ns-2.33/diffserv)
ns-default.tcl   (en ns-2.33/tcl/lib/)
```

Agregar en `../ns-2.33/tcl/lib/ns-default.tcl` las líneas que se indican a continuación.

```
# A. Mateos Oct. 2011. For Priroty Index Scheduler. To indicate if link originates from an ingress node.

Queue/dsRED set IngrRoute_ 0
```

Modificación directamente de hdr_cmn del paquete. Esto para que el paquete contenga la indicación de su Priority Index. Se incluye directamente en el hdr_cmn. Se debe hallar packet.h que se encuentra en el directorio `/usr/local/ns-allinone-2.33/ns-2.33/common`, y ahí, en struct hdr_cmn buscar las dos líneas que contienen a la variable txtime_ para añadir, después de esas líneas, nuevas tres líneas, para que quede como sigue:

```
// tx time for this packet in sec

double txtime_;

inline double& txtime() { return(txtime_); }

// Priority index (prindex_) for this packet (in sec). Alfredo Mateos --Oct 2011.

double prindex_;

inline double& prindex() { return(prindex_); }

// End of insertion --Oct 2011.
```

En el Script de ns-2 agregar lo siguiente para dar el valor de 1 a la variable `IngrRoute_` a alguna cola (el objeto que representa a la cola), con lo que se indica que el nodo asociado a la misma es de ingreso).

```
# Agregado por A Matos. Oct. 2011. Para Scheduler Priority Index. Si IngrRoute_ es 1 el nodo es de ingreso (en este caso como ejemplo el nodo e3).

$qe3c1 set IngrRoute_ 1; # Indica que el nodo e3 es de ingreso.
```

PARA dsred.h

```
// Insertion by Alfredo Mateos for the Scheduler using Priority Index. Oct. 2011.

int IngrRoute_;  // Indication if the interface is in a link which orig-
inates from an ingress node (1=Yes).

// End addition Oct. 2011.
```

PARA dsred.cc

```
// Inserción de Alfredo Mateos, Oct. 2011, para el Scheduler Priority
Index.

bind("IngrRoute_", &IngrRoute_); // Indica si la interfaz está en un
enlace que emana de un nodo que es de ingreso (1 = si).

// printf("En Constructor. IngrRoute_ = %d\n", IngrRoute_);

// FIN DE AGREGADO Oct. 2011.
```

PARA dsred.h se hace el cambio en la estructura siguiente.

```
// Estructura cambiada en Oct. 2011 por A Mateos para incluir parámetros
para Priority Index.

/*---------------------------------------------------------------------
struct phbParam
This struct is used to maintain entries for the PHB parameter table, used
to map a code point to a physical queue-virtual queue pair.
---------------------------------------------------------------------*/

struct phbParam {

 int codePt_;

 int queue_;      // physical queue

 int prec_; // virtual queue (drop precedence)

 double PrInlp_;  // Priority Index for last packet received of flow PHB, in
sec. A Mateos. Oct. 2011.

 double DelPr_;   // Increment to Priority Index for flow PHB, in sec. A
Mateos. Oct. 2011.

 double GamFl_;   // Guaranteed Rate for flow PHB, in bits/sec. A Mateos.
Oct. 2011.
 int iR_;   // Counter.

};
```

De tal forma que en el Script de ns-2 se tienen las entradas para la tabla PHB, como la siguiente:

```
$qe1c0 addPHBEntry 10 0 0 0 0.05 125000
```

En dsred.h se modifica la línea que declaraba la rutina lookupPHBtable por lo siguiente.

```
// Modificaciones Oct. 2011. Alfredo Mateos. Para incluir datos para Priority
Index.

void lookupPHBTable(int codePt, int* queue, int* prec, double* PrInlp, dou-
ble* DelPr, double* GamFl, int *iR); // looks up queue and prec corresp to
a code point.

void addPHBEntry(int codePt, int queue, int prec, double PrInlp, double
DelPr, double GamFl); // edits phb entry in the table.

// Concluye modificación Oct. 2011.
```

En dsred.cc se cambia la rutina lookupPHBTable por lo siguiente.

```
// Rutina de lookupPHBTable modificada en sus argumentos por A Mateos
Oct. 2011 para incluir datos Priority Index.

void dsREDQueue::lookupPHBTable(int codePt, int* queue, int* prec,
double* PrInlp, double* DelPr, double* GamFl, int* iR) {

 for (int i = 0; i < phbEntries; i++) {
   if (phb_[i].codePt_ == codePt) {
     *queue = phb_[i].queue_;
     *prec = phb_[i].prec_;
     *PrInlp = phb_[i].PrInlp_;
     *DelPr = phb_[i].DelPr_;
     *GamFl = phb_[i].GamFl_;
     *iR = phb_[i].iR_;
     return;
   }
 }
 printf("ERROR: No match found for code point %d in PHB Table.\n",
codePt);

}
```

En dsred.cc la primera parte de la rutina deque() se cambia como sigue: (realmente no sería necesario inicializar ni llamar a lookupPHBTable con tantos argumentos como se hace aquí porque lo único que importa es la variable queue, y anuqe esto se podría mejorar por ahora se deja así.

```
/*---------------------------------------------------------------------
```

```
Packet* deque()
    This method implements the dequing mechanism for a Diffserv router.
------------------------------------------------------------------------*/
Packet* dsREDQueue::deque() {
       Packet *p;
       int queue, prec, iR;
       double PrInlp, DelPr, GamFl;
       hdr_ip* iph;
       int fid;
       // Select queue to deque:
       selectQueueToDeque();
       // Dequeue a packet from the underlying queue:
       p = redqp_[qToDq]->deque();
       if (p != 0) {
        iph= hdr_ip::access(p);
        fid = iph->flowid()/32;
        pktcount[qToDq]+=1;

        if (schedMode == schedModePRI && queueMaxRate[qToDq]) applyTSWMe-
ter(p);

         /* There was a packet to be dequed. Find the precedence level
          (or virtual queue id) to which this packet was attached. */

         // Búsqueda a lookupPHBTable modificada en sus argumentos por A
Mateos Oct. 2011 para incluir datos Priority Index.

         lookupPHBTable(getCodePt(p),  &queue,  &prec,  &PrInlp,  &DelPr,
&GamFl, &iR);

         // Update virtual queue reinserted for this version of dsred.cc by
Alfredo Mateos.

         // decrement virtual queue length
         // Previously in updateREDStateVar, moved by xuanc (12/03/01)
         redqp_[qToDq]->updateVREDLen(prec);

         // update state variables for that "virtual" queue
         redqp_[qToDq]->updateREDStateVar(prec);

       } // End if.
       // Return the dequed packet:
       return(p);
} // End deque.
```

En dsred.cc la primera parte de la rutina enque() se cambia como sigue:

```
/*-----------------------------------------------------------------------
void enque(Packet* pkt)
The following method outlines the enquing mechanism for a Diffserv
router. This method is not used by the inheriting classes; it only serves
as an outline.
------------------------------------------------------------------------*/
void dsREDQueue::enque(Packet* pkt) {
   int codePt, queue, prec, iR;
```

```
   double PrInlp, DelPr, GamFl;
   hdr_ip* iph = hdr_ip::access(pkt);

   codePt = iph->prio(); // Extracting the marking done by the edge
router.

   int ecn = 0;

   // Looking up queue and prec numbers for that codept.
   // Búsqueda a lookupPHBTable modificada en sus argumentos por A Mateos
Oct. 2011 para incluir datos Priority Index.

   lookupPHBTable(codePt, &queue, &prec, &PrInlp, &DelPr, &GamFl, &iR);
```

En dsred.h se añade lo siguiente

```
// *** schedModeWFQ added ***
enum schedModeType {schedModeRR, schedModeWRR, schedModeWIRR, schedModePRI,
schedModeWFQ, schedModePR_IN};
```

En dsred.cc se cambia la siguiente rutina

```
// Rutina cambiada por A. Mateos. Oct. 2011, para incluir parámetros de
Priority Index.
/*-----------------------------------------------------------------------
void addPHBEntry(int codePt, int queue, int prec)
 Add a PHB table entry.   (Each entry maps a code point to a queue-
precedence pair.)
-----------------------------------------------------------------------*/
void dsREDQueue::addPHBEntry(int codePt, int queue, int prec, double
PrInlp, double DelPr, double GamFl) {
      if (phbEntries == MAX_CP) {
      printf("ERROR: PHB Table size limit exceeded.\n");
      } else {
            phb_[phbEntries].codePt_ = codePt;
            phb_[phbEntries].queue_ = queue;
            phb_[phbEntries].prec_ = prec;
            phb_[phbEntries].PrInlp_ = PrInlp;
            phb_[phbEntries].DelPr_ = DelPr;
            phb_[phbEntries].GamFl_ = GamFl;
            phb_[phbEntries].iR_ = phbEntries;
            stats.valid_CP[codePt] = 1;
            phbEntries++;
      }
}
```

En dsred.cc se cambia la sigiente rutina.

```
// La siguiente rutina se modifica por A. Mateos. Oct. 2011, para incluir
variables de Priority Index.
```

```
if (strcmp(argv[1], "addPHBEntry") == 0) {

  addPHBEntry(atoi(argv[2]),        atoi(argv[3]),        atoi(argv[4]),
atof(argv[5]), atof(argv[6]), atof(argv[7]));

  return (TCL_OK);

}
```

Arreglar enque de la siguiente forma (agregando dos veces lo siguiente -se puede ven en la rutina.

```
// **Priority Index **ADDITION START*
      if(schedMode == schedModePR_IN) {
//      printf("Entrando a PrIenque\n");
        PrIenque(pkt, PrInlp, DelPr, GamFl, iR);
      }
```

Poner la rutina siguiente en dsred.cc (después de la rutina enque(...)

```
// ****New for Priority Index. Llamada desde enque()

void dsREDQueue::PrIenque(Packet* pkt, double PrInlp, double DelPr, dou-
ble GamFl, int iR) {

// Rutina de Priority Index para poner datos en paquete a encolar.
// Se lama a la rutina como PrIenque(pkt, PrInlp, DelPr, GamFl, iR)
// pkt es el apuntador del paquete a encolar.

  double PrIn_loc; // Variable local para Priority Index.
  hdr_cmn *hdr = hdr_cmn::access(pkt);

// Se pregunta si se está en un nodo de ingreso.
// IngrRoute_ es un variable general en la interfaz dsREDQueue.
//  printf("En PrIenque. IngrRoute_ = %d; ", IngrRoute_);

  if(IngrRoute_ == 1) {
    // Sí, se está en un nodo de ingreso.

    PrIn_loc = PrInlp + ( (double) hdr->size() * 8.0 / GamFl ); // size
está en Bytes.

    phb_[iR].PrInlp_ = PrIn_loc;
    hdr->prindex_ = PrIn_loc;

//    printf("Revisando paso de GamFl = %f\n", GamFl);

//    printf("Nodo ingreso = %p. Packet act = %p. PrIndx Last Packet =
%f. Tam paq actual = %d. Tasa Asegur = %f\n", this, pkt, PrInlp, hdr-
>size(), GamFl);
```

```
  } else {

    // No se está en un nodo de ingreso.
    PrIn_loc = hdr->prindex() + DelPr;
    hdr->prindex_ = PrIn_loc;

  // printf("Nodo central = %p. Packet act = %p. PrIndx Packet Actual =
%f. Delta_PrIndex = %f\n", this, pkt, hdr->prindex(), DelPr);

  } // Se acaba if-else.

//  printf("DelPr = %f. Nuevo PrIndx del PHB %d = %f (CdPt num. %d en
estruct) ", DelPr, phb_[iR].codePt_, phb_[iR].PrInlp_, iR);

// phb_ es una variable general en la interfaz dsREDQueue.

//  printf("El nuevo PrIndx en Packet Actual %p = %f\n\n", pkt, hdr-
>prindex());

}; // Concluye PrIenque

// ***Concluye rutina para Priority Index.
```

Se agrega para dsred.h lo siguiente.

```
// Nuevo para Priority Index.
     void PrIenque(Packet* pkt, double PrInlp, double DelPr, double
GamFl, int iR);
```

Se modifica en dsred.cc el selector de Schedular mode.

```
/*-----------------------------------------------------------------
void setSchedularMode(int schedtype)
   sets up the schedular mode.
-----------------------------------------------------------------*/
void dsREDQueue::setSchedularMode(const char* schedtype) {
     if (strcmp(schedtype, "RR") == 0)
     schedMode = schedModeRR;
     else if (strcmp(schedtype, "WRR") == 0)
          schedMode = schedModeWRR;
     else if (strcmp(schedtype, "WIRR") == 0)
          schedMode = schedModeWIRR;
     else if (strcmp(schedtype, "PRI") == 0)
          schedMode = schedModePRI;
     // **WFQ**ADDITION**START*****************
     else if (strcmp(schedtype, "WFQ") == 0)
          schedMode = schedModeWFQ;
     // **WFQ**ADDITION**END****************

     // **Priority Index**ADDITION**START****************
```

```
      else if (strcmp(schedtype, "PR_IN") == 0)
            schedMode = schedModePR_IN;
      // **Priority Index**ADDITION**END*****************
        else
           printf("Error:   Scheduler   type   %s   does   NOT   ex-
ist\n",schedtype);
} // End setSchedularMode
```

Hay que agregar a la rutina selectQueueToDeque() de dsred.cc lo siguiente:

```
// **Priority Index**ADDITION**START*****************

      else if (schedMode == schedModePR_IN) {

       PrIdequeue(); // End else if.

      // Esta rutina debe modificar una variable general entera que se
llama qToDq

      } // End if schedMode == schedModePR_IN

// **Priority Index**ADDITION**END*****************
```

En dsredq.h añadir la declaración de la rutina lookup después de deque(void).

```
  // Adición de A Mateos Oct. 2011.
  Packet* lookup(int);            // looks up for packets
  // Acaba adición de A. Mateos Oct. 2011.
```

En dsredq.cc añadir la definición de la rutina lookup después de la rutina deque().

```
// **ADDED by Alfredo Mateos. Oct. 2011.
//  Packet* lookup(int i)
//    Looks up a packet from the physical queue.
//   With parameter 0 looks up for the header (first) packet.
Packet* redQueue::lookup(int i) {
  return(q_->lookup(i));
}
// Ends addition of Alfredo Mateos Oct. 2011.
```

AÑADIR LA SIGUIENTE FUNCION en dsred.cc.

```
// FUNCION añadida por Alfredo Mateos para Priority Index . Oct. 2011.

void dsREDQueue::PrIdequeue() {

  printf("EN PrIdeque. Link = %p. numQueues_ = %d\n", this, numQueues_);
```

```
  // Se debe entregar una variable que se llama qToDq
  Packet *p;
  hdr_cmn *hdr;
  int i, j;
  i = j = 0;
  double PrIn_loc, PrIn_min; // Variable local para Priority Index.
  PrIn_loc = PrIn_min = 0.0;
  // Los objetos redqp_ son objetos (colas) tipo redQueue.
  // A Mateos ha añadido en dsredq.{h,cc} a la clase redQueue la rutina
lookup(int i) que busca un paquete y entrega su apuntador (sin quitarlo de
la cola seleccionada por el argumento qToDq de la clase (la cola múltiple)
dsREDQueue.

  //  Cuando el argumento i es 0 se indica que se busca el paquete de
cabezera.

  // A su vez dicha rutina utiliza la rutina lookup(int i) que está en la
clase PacketQueue (que es una cola de objetos tipo packet) que hace esta
búsqueda, y que a su vez está contanida en el archivo ../ns-
2.33/queue.{h,cc}.

  while ((i < numQueues_)) {

    printf("** i(cola) = %d. Long_cola= %d; ", i, redqp_[i]->getReal-
Length());

            if(redqp_[i]->getRealLength() > 0) {
                  p = redqp_[i]->lookup(0);
                  hdr = hdr_cmn::access(p);
                  PrIn_loc = hdr->prindex();

                  if( j == 0 ) { PrIn_min = PrIn_loc; j++; qToDq = i; } else
if(PrIn_loc < PrIn_min) {qToDq = i; PrIn_min = PrIn_loc;}

                  printf("Pck=%p, Pri_Pck=%f; ", p, PrIn_loc);
            }

            i++;

      } // End while.

      if(j == 0) { qToDq = 0; }

      printf("Saliendo de PrIdequeue. qToDq = %d\n\n", qToDq);

} // End PrIdequeue

// Acaba FUNCION añadida por A. Mateos. Oct. 2011.
```

En el archivo dsred.h en la declaración de la clase dsREDQueue se añade:

```
// ADICION Alfredo Mateos Oct. 2011. Priority Index
      void PrIdequeue();
// Acaba adición Oct. 2011.
```

Apéndice M. Inclusión del Código para la operación del Servicio de Despacho de Multisalto Coordinado (CMS) en ns-2