UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE UN SISTEMA DE COMUNICACIÓN EN LÍNEA UTILIZANDO EL PROTOCOLO IRC

TESIS

QUE PARA OBTENER EL TÍTULO DE
INGENIERO EN COMPUTACIÓN

PRESENTAN

HONORATO SAAVEDRA HERNÁNDEZ
PEDRO SALINAS BAZÁN

DIRECTOR:

ING. ORLANDO ZALDÍVAR ZAMORATEGUI

MÉXICO, D. F. JUNIO DE 2003

Agradecimientos

A mis padres y hermanos por apoyarme siempre aún cuando estaba muy lejos, a toda la familia Valdovinos por aceptarme y ayudarme durante tanto tiempo, a mis compañeros, profesores y amigos de la Facultad de Ingeniería por todo lo que pasamos juntos y sobre todo a mi esposa Josefina por hacerme feliz y alentarme en el trabajo todos los días.

Gracias.

Honorato Saavedra Hernández

Agradecimientos

A mis padres por darme la vida y apoyarme en todo cuanto les fue posible.
A mis hermanos por su apoyo y ayudarme a no perder de vista mis objetivos.
A mis amigos y compañeros de generación, por todos los momentos que pasamos juntos, que hicieron de mi paso por la Universidad una de las etapas más felices de mi vida.
Al Laboratorio de Multimedia e Internet, a todos los que allí estuvieron, gracias por su ayuda y su amistad.
Al profesor Orlando Zaldívar Zamorategui, por su poyo y la paciencia que tuvo para con nosotros.
Finalmente gracias nato, por ser un buen amigo y compañero de estudios.

Pedro Salinas Bazán

Índice

Capítulo 1

Introducción

El presente documento se centra en el estudio e implementación del protocolo conocido como Internet Relay Chat (IRC) que se traduce al español como "Retransmisión de Conversaciones por Internet", el cual es un estándar de comunicación que se basa en mensajes de texto enviados a través de una red de computadoras. Este protocolo fue creado por Jarkko Oikarinen en Finlandia como una mejora del antiguo programa "talk" y se ha vuelto muy popular debido a que permite conectarse desde cualquier parte de la Internet para participar en discusiones en vivo que pueden incluir a más de dos personas. El programa talk permite copiar líneas de una computadora a otra, esto sirve para comunicar a dos personas solamente, de ahí la necesidad de crear este protocolo, ya que permite más de dos personas en la comunicación.

Para participar en una conversación de IRC, cada participante necesita de un programa "cliente" que se encargará de enviar y recibir mensajes conectándose a otro programa que llamaremos "servidor", que tiene como propósito coordinar la comunicación entre los clientes reenviando los mensajes de cada uno a todos los demás participantes.

Las conversaciones se organizan por medio de "canales" que son grupos de usuarios o clientes que se reúnen para conversar sobre un mismo tema. Un servidor IRC puede manejar varios canales a los que se unen los clientes, según sus gustos o necesidades.

Varios servidores pueden formar una red IRC de manera que los usuarios de un servidor se puedan comunicar con usuarios de otros servidores. El servidor forma la columna vertebral del IRC, dando un punto al cual los clientes se pueden conectar para hablar entre ellos y un punto de conexión para otros servidores para formar una red IRC. La única configuración de red permitida para los servidores IRC es la de un árbol extendido donde cada servidor actúa como nodo central para el resto de la red que puede ver, es decir, no se permiten ciclos en las conexiones.

Este protocolo está diseñado para soportar varias características, como el envío de mensajes privados, la obtención de información de los usuarios, expulsión de usuarios molestos, creación de operadores de canal, creación de administradores de la red IRC, etc.

La implementación de este sistema no abarcará todas las características que define el protocolo IRC, se excluirán aquellas que tienen que ver con la comunicación entre servidores debido a que no se tiene la necesidad de construir una red IRC. Estas funciones o características se refieren a conexiones entre servidores, registro de nuevos servidores, consulta de información de otros servidores y otros detalles que se explicarán más adelante.

Se pretende que nuestro programa sea capaz de establecer comunicación con cualquier programa cliente que se apegue al protocolo IRC. Existen muchos programas "cliente" y uno de los objetivos es que las personas puedan trabajar con nuestro servidor de la manera más sencilla.

Se eligió el protocolo IRC porque tiene un uso muy extendido, una estructura bien definida y sobre todo porque nos permitirá crear un sistema que sea compatible con otros programas que utilicen el mismo protocolo.

Todas las características que debe reunir un servidor IRC están definidas en varios documentos llamados "Request For Comment" (RFC), los cuales contienen información básica para comprender e implementar el protocolo en el programa servidor o en el cliente. Es de estos documentos de donde obtendremos los requerimientos para implementar el protocolo.

Los RFC's son documentos que contienen información sobre la Internet, se les asigna un número para su identificación y tratan sobre algún tema en específico. Nos basaremos en los RFC's 1459, 2810, 2811, 2812 y 2813. El documento original es el 1459 y los otros surgieron como modificaciones o mejoras. En estos documentos se describe el protocolo IRC indicando las características y normas que debe seguir la implementación.

El protocolo IRC ha sido diseñado para usarse en conferencias basadas en texto y ha sido desarrollado en sistemas que usan el protocolo de red TCP/IP[1]. IRC por sí mismo es un sistema de teleconferencia, que a través del modelo cliente/servidor está diseñado para correr en varias máquinas de una manera distribuida. Con esta base podemos confiar en que el sistema será robusto y suficiente para satisfacer las necesidades de comunicación del Laboratorio de Multimedia e Internet y de los estudiantes que lo utilicen, ya sea en la UNAM o en alguna otra parte.

Este proyecto está contemplado dentro de las actividades del Laboratorio de Multimedia e Internet y es parte integral de un conjunto de programas que tienen como fin generar un sistema de comunicación que tenga características de multimedia y que utilice la Internet. No será sólo este proyecto el que logre este fin, pero permitirá que otros se basen en los conocimientos aprendidos durante su desarrollo. Se utilizará para ofrecer asesoría en línea sobre los cursos que se imparten.

El objetivo de este trabajo se puede resumir en el siguiente punto:

- **Desarrollar un programa servidor basado en el protocolo IRC para establecer comunicación escrita en tiempo real a través de la Internet.**

La importancia de este proyecto radica en que resolverá varios problemas de comunicación que existen en el Laboratorio de Multimedia e Internet y por la experiencia y conocimientos que se pueden obtener en su proceso de desarrollo, debido a que por primera vez se está creando un sistema que se apega a un protocolo y que deberá cumplir con muchas características, reglas, convenios y formas en todos los aspectos de su funcionamiento. Se deberá elegir y seguir una metodología que sea adecuada para este tipo de sistema y se entrará en lo que consideramos un nuevo campo de conocimiento que tiene que ver con el manejo de gramáticas y lenguajes.

Con este trabajo se pretende crear un canal de comunicación a través de la Internet para compartir y distribuir información, conocimientos y trabajar en equipo, se decidió crear este sistema apegándonos a IRC porque es capaz de brindar las bases necesarias que satisfagan estos requerimientos cuando participan dos o más personas utilizando la Internet.

[1] Protocolo de Control de Transmisión / Protocolo de Internet

Aunque existen servidores que permiten este tipo de comunicación, tendríamos que depender de terceras personas para establecerla, lo que provoca que los datos viajen por la red hacia lugares que no tienen relación con el Laboratorio creando dependencia hacia un servicio de terceros. Por lo tanto, se decidió instalar un servidor propio, para que la transmisión de los mensajes sea más directa.

Con respecto a la utilización de un programa o paquete ya hecho, se decidió no utilizar ninguno y crear uno propio para tener una base flexible y fácil de modificar a fin de agregar nuevas características muy exclusivas de nuestras necesidades y del sistema operativo en el que estamos trabajando.

El desarrollo de este proyecto exige conocimientos sobre redes de computadoras, TCP/IP, metodologías de desarrollo, programación y manejo de definiciones formales de lenguajes y gramáticas. En este documento se procederá primero con una introducción a los conceptos básicos y después se desarrollará el tema en sí. Está organizado de la siguiente manera:

- El capítulo 1 es una introducción al trabajo presentado y una descripción del contenido del documento.
- El capítulo 2 incluye los conceptos básicos sobre metodologías de desarrollo y en él explicaremos qué metodología se eligió para este sistema.
- El capítulo 3 da una introducción al tema de redes de computadoras y la tecnología que se utilizará en la construcción de nuestro programa. Se hará énfasis en el protocolo TCP/IP debido a su importancia en nuestro desarrollo.
- El capítulo 4 habla sobre el protocolo IRC explicando los detalles mas básicos de su funcionamiento. El análisis a profundidad se hará en un capítulo posterior.
- El capítulo 5 se centra en la representación formal de gramáticas. Este tema es incluido en nuestro marco teórico porque tendremos la necesidad de analizar cadenas de texto que contienen comandos y parámetros con estructuras muy diferentes entre sí.
- El capítulo 6 incluye todos los datos y requerimientos recogidos durante el análisis del sistema.
- El capítulo 7 describe el diseño de la arquitectura, módulos, funciones, etc.
- El capítulo 8 muestra los pasos seguidos durante la construcción de los diferentes módulos y las pruebas realizadas.
- El capítulo 9 incluye las conclusiones a las que hemos llegado, indicando si se cumplió con el objetivo propuesto.

Capítulo 2

Metodologías de desarrollo

Este capítulo tiene por objetivo hacer una definición de los conceptos básicos de ingeniería de software. Se describirán las metodologías empleadas en el desarrollo de software y cuál fue la elegida en este trabajo.

2.1 La industria del software

Cada día son más las actividades en las que intervienen aplicaciones de software. Estas aplicaciones van desde el sistema financiero de un banco hasta el sistema de control de un avión pasando por software educativo, software de ayuda a la construcción de algún artefacto, sistemas de apoyo al trabajo cooperativo, etc. Además, con la popularización de la Internet y, en particular, de la Web, las posibilidades de aplicación han aumentado considerablemente.

Así como los dominios de aplicación de software han aumentado, también ha aumentado la complejidad y el tamaño de los sistemas. Esto es debido, en parte, a que las necesidades de los usuarios son cada vez más exigentes, en términos de interacción, seguridad, retroalimentación, etc., y también, debido a los avances tecnológicos que hacen posible que un sistema esté compuesto de múltiples subsistemas en los que intervienen bases de datos, interfaces gráficas, comunicación entre componentes, recuperación ante fallas, interacción con dispositivos multimedia, sistemas complejos de seguridad, etc.

Los problemas relacionados con el desarrollo y mantenimiento de los sistemas son muchos y muy grandes. Desgraciadamente, no es raro encontrar usuarios de software descontentos debido a que no se satisfacen sus necesidades, el software es ineficiente, no es amigable o no es seguro.

Este problema es importante no sólo por las grandes pérdidas económicas directas de las empresas y organizaciones, sino también por los costos de oportunidad, la seguridad de las personas, el retraso científico y tecnológico, etc.

La construcción de software no se limita a conocer la tecnología, tener la máquina más potente y la última herramienta de desarrollo. Antes que la tecnología, las máquinas y las herramientas están las personas y los procesos que esas personas utilizan para la construcción de los sistemas. La construcción y el mantenimiento de software requiere del esfuerzo y colaboración de equipos de personas. Es por esto y por la complejidad de las tareas que se realizan, que las actividades de desarrollo deben ser planificadas, evaluadas, coordinadas, etc. Éstas son las razones que hacen de la ingeniería de software una disciplina fundamental para los profesionales de la computación.

2.2 ¿Qué es la ingeniería de software?

La ingeniería de software es una disciplina que trata los problemas prácticos del desarrollo de software y tiene por objetivo mejorar la calidad del software produciéndolo dentro de los costos y los plazos previstos. La ingeniería de software ofrece una serie de procesos, metodologías y herramientas para dar soporte a las actividades de desarrollo y

mantenimiento. Esta disciplina está en evolución permanente a causa de los constantes cambios tecnológicos, el afán de estandarización y también, gracias a la experiencia adquirida y documentada de desarrollos, exitosos o no, de software.

En la construcción y desarrollo de proyectos se aplican métodos y técnicas para resolver los problemas.

Los objetivos de la ingeniería de software principalmente son:

- Mejorar la calidad de los productos de software.
- Aumentar la productividad y trabajo de los ingenieros del software.
- Facilitar el control del proceso de desarrollo de software.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.
- Definir una disciplina que garantice la producción y el mantenimiento de los productos de software desarrollados en el plazo fijado y dentro del costo estimado.

2.3 Características de un producto de software

- Muchas y diversas entidades. Un producto de software está constituido por entidades de distinta naturaleza: programas fuente, especificaciones, pruebas, documentos de utilización, instalación, archivos objeto, etc. Estas entidades están fuertemente relacionadas entre sí.

- Larga vida. Un producto de software se construye para que pueda ser utilizado durante muchos años.

- Evolución constante. Un producto de software evoluciona con el tiempo (en desarrollo y después de que ha sido instalado para uso). Esta evolución es consecuencia de mejoras, cambios (lógicos o tecnológicos), correcciones, etc.

- Simultaneidad y variación. Existe un gran número de productos de software que deben funcionar para diferentes usuarios (simultáneamente) con variaciones acerca de las preferencias de éstos sobre las interfaces, la interacción, la comunicación, la plataforma, etc.

2.4 Procesos de software

Las actividades necesarias para la fabricación de un producto de software se conocen con el nombre de procesos de software. Los siguientes son algunos elementos relacionados con estos procesos y que son importantes porque de una u otra forma afectan su realización.

- Contexto exterior cambiante: Los procesos de software se ven afectados por cambios en las necesidades de los usuarios, cambios de políticas, legislaciones, tecnología, etc.

- Naturaleza intelectual de las actividades: Dentro de las muchas actividades necesarias para la fabricación de un software existen algunas que son de naturaleza altamente creativa y que no son automatizables.

- Trabajo en equipo (cooperativo): Para construir un producto de software es necesario distribuir el trabajo en uno o más equipos de personas. Decenas o aún centenas de personas pueden participar en la construcción de un software.

Las características del producto de software y los elementos relacionados con los procesos hacen que la fabricación de software (desarrollo, mantenimiento) sea una actividad compleja que requiere de la aplicación de prácticas de muchas disciplinas y en particular de ingeniería y de administración.

2.5 Procesos de construcción de software

En esta sección se definen, en términos generales, los procesos de construcción y mantenimiento de software. El objetivo es presentar de manera global, todas y cada una de las actividades que deben realizarse al producir software. Este marco global va a permitir ubicar los distintos temas de estudio de la ingeniería de software, sus relaciones e importancia.

2.6 Definición de un proceso de software

Un proceso de software es el conjunto de actividades, métodos, prácticas y transformaciones que realizan las personas, generalmente empleando herramientas, para transformar unas necesidades (expresadas como requerimientos) en un producto de software. La figura 2.1 muestra los agentes involucrados en un proceso de software: personas, actividades y herramientas, y su interrelación.

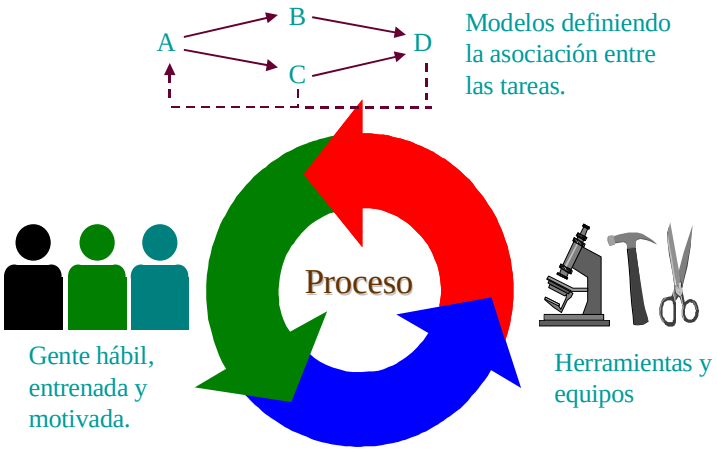

Figura 2.1

La calidad final del producto podrá mejorarse si se asegura la calidad del proceso que lo construye y lo mantiene. Es decir, la definición y realización de un buen proceso de software reduce el riesgo de cometer errores durante el desarrollo y mantenimiento de un producto.

Una organización o un grupo de personas que construyen software siempre utilizan un proceso, el problema es que en la mayoría de los casos, este proceso es a la medida y es improvisado según se va requiriendo.

Con un proceso improvisado no se puede garantizar un buen resultado al final ni tampoco que los recursos se utilizarán de manera eficiente. Si, además, no aprendemos de la experiencia, en el futuro será igualmente difícil realizar un buen trabajo.

Para alcanzar algún objetivo, se necesita poder definir de manera explícita el proceso de software. Esto nos permitirá tener mayor control sobre aspectos administrativos, de coordinación y de colaboración en los proyectos, mayor poder para predecir resultados, la posibilidad de mejorar la calidad de los productos y, sobre todo, aprender de la experiencia en la medida que si el proceso es definido y seguido, éste se podrá evaluar y mejorar.

2.7 Clasificación global de las actividades de los procesos de software

La definición e implantación de un proceso de software no es una tarea fácil. Sin embargo, es posible presentar un marco global sobre el conjunto de procesos que se deben cubrir para construir software.

- Procesos administrativos
- Actividades del ciclo de vida (Análisis de requerimientos, Arquitectura de software, Diseño de software, Programación, Evolución y puesta en marcha).
- Procesos de soporte

2.7.1 Procesos administrativos

Esta clasificación agrupa los mecanismos, procesos y actividades de la administración para patrocinar y asegurar la institucionalización del proceso, es decir, el grado en el cual, el proceso de software, es realmente seguido por los desarrolladores.

2.7.2 Actividades en un ciclo de vida

Estas actividades corresponden a las tareas necesarias para el desarrollo o construcción de software, su mantenimiento y explotación. Podemos mencionar el análisis de requerimientos, el diseño, la programación, la instalación y el mantenimiento. Las personas involucradas en estos procesos son el grupo de desarrollo, el grupo de mantenimiento de las aplicaciones, el grupo de subcontratistas (desarrolladores externos) y el grupo de compradores (clientes) y de usuarios (operadores) quienes realizan el proceso de utilización del software.

2.7.3 Procesos de soporte

Esta clasificación agrupa los mecanismos, procesos y actividades para el aseguramiento de la calidad y seguimiento del proceso de software definido en la organización. Estos procesos dan soporte al grupo de desarrollo en las actividades de administración, planeación y seguimiento al proyecto por lo cual están presentes durante todo el ciclo de vida del software.

Algunas de las actividades que se pueden clasificar en este grupo son la realización de la documentación, la administración de configuraciones o control del cambio, las mediciones y registro del seguimiento del proceso, aseguramiento de calidad, etc.

2.8 Ciclos de vida

La selección del ciclo de vida corresponde a una actividad principal dentro de la planificación del proyecto (proceso administrativo). Es importante resaltar la importancia que tiene la selección de un ciclo de vida dentro del desarrollo de un proyecto de software y mostrar los modelos de ciclo de vida más usados en la práctica, además de justificar la selección de la combinación de los ciclos elegidos.

2.8.1 Codificar y corregir

Uno de los esquemas tradicionales de desarrollo de software consiste en que dada una breve descripción del problema, el programador empieza a codificar el sistema. Mediante esta estrategia el proceso de desarrollo consiste en ir adaptando el software a medida que se descubren los requerimientos reales o se cancela el proyecto. La posibilidad

de éxito con un esquema como éste es bastante baja, y se reduce aun más cada vez que hay cambios en la especificación del problema o aumenta de complejidad.

El ciclo de vida corresponde a las actividades que se realizan desde la descripción de un problema hasta el momento en que el sistema deja de ser utilizado por los usuarios. Es un modelo de las actividades que se llevarán a cabo para la construcción de un software y establece el orden en el cual se realizarán. El esquema de desarrollo anterior corresponde a un ciclo al que llamaremos “Codificar y Corregir”

Salvo que el enunciado de un proyecto sea trivial, codificar y corregir no es realmente el ciclo de vida más apropiado para elegir. El problema más importante es la falta de información, planeación y gestión administrativa. El no saber, en un momento dado cuál es el sistema a construir, cuáles son sus características (complejidad, tamaño, composición, criterios de calidad, etc.). Este desconocimiento hace casi imposible hacer cualquier tipo de planeación sobre el desarrollo del proyecto y traerá consecuencias graves para las actividades de mantenimiento.

El objetivo de utilizar un ciclo de vida en el desarrollo de un proyecto es facilitar la planeación y el seguimiento del mismo, fortalecer la negociación con el cliente y el asegurar la calidad del producto. Los modelos de ciclo de vida tienen el inconveniente de no ser completos en cuanto a los diferentes aspectos del desarrollo, pues tienden a olvidar algunas de las actividades que se realizan a lo largo de todo el desarrollo y en particular, del mantenimiento.

2.8.2 Cascada

En el modelo de ciclo de vida en cascada un proyecto atraviesa por una secuencia de etapas desde el concepto inicial hasta la puesta en marcha del sistema. Aunque es posible regresar en las etapas de desarrollo, en este esquema resulta particularmente costoso arrepentirse; por lo cual es adecuado para proyectos donde los requerimientos son estables, existe una definición clara del producto a realizar y se cuenta con herramientas y métodos definidos de desarrollo. Figura 2.2

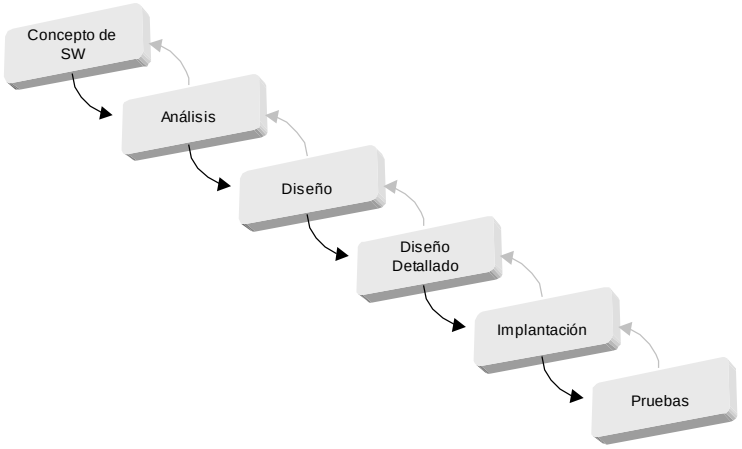

Figura 2.2

El ciclo de vida en cascada es el modelo más conocido y utilizado para desarrollo de software. Se considera como el primer paso a la Ingeniería de Software. Aunque presenta varios inconvenientes en el desarrollo actual, es importante en la medida de ser el padre de los demás ciclos de vida y proporciona una estructura de proyecto clara.

Dentro de las ventajas del ciclo de vida en cascada se encuentra la estructuración y el desarrollo claramente dividido en etapas. Sin embargo, tiene varios inconvenientes. No toma en cuenta labores de soporte y administración, y tiende a dejar las pruebas hasta el final del desarrollo.

2.8.2.1 Investigación preliminar

La solicitud para recibir ayuda de un sistema de información pueden originarse por una persona, cuando se formula la solicitud comienza la primera actividad del sistema. Esta actividad tiene tres partes:

2.8.2.2 Aclaración de la solicitud

Antes de considerar cualquier investigación de sistemas, la solicitud de proyecto debe examinarse para determinar con precisión lo que el solicitante desea; ya que muchas solicitudes no están realizadas de manera clara o bien definida.

2.8.2.3 Estudio de factibilidad

En la investigación preliminar un punto importante es determinar que el sistema solicitado sea factible. Existen tres aspectos relacionados con el estudio de factibilidad, que son realizados por lo general, por analistas capacitados o directivos:

- **Factibilidad técnica.**

Estudia si el trabajo para el proyecto, puede desarrollarse con el software y el personal existente, y si en caso de necesitar nueva tecnología, cuáles son las posibilidades de desarrollarla (no sólo el hardware).

- **Factibilidad económica.**

Investiga si los costos se justifican con los beneficios que se obtienen, y si se ha invertido demasiado, como para no construir el sistema si se cree necesario.

- **Factibilidad operacional:**

Investiga si se utilizará el sistema, qué usuarios lo harán y cómo.

- **Factibilidad de programación**

Investiga si es posible crear el sistema a partir de las herramientas y conocimientos con los que cuenta el equipo de desarrollo.

- **Factibilidad legal**

Indica si el sistema a desarrollar está dentro de la legislación vigente, por ejemplo, si no se está violando alguna patente o prohibición específica de algún país.

2.8.2.4 Aprobación de la solicitud

Algunas organizaciones reciben tantas solicitudes de sus empleados que sólo es posible atender unas cuantas. Sin embargo, aquellos proyectos que son deseables y factibles deben incorporarse en los planes. En algunos casos el desarrollo puede comenzar inmediatamente, aunque lo común es que los miembros del equipo de sistemas estén ocupados en otros proyectos. Cuando esto ocurre, la administración decide qué proyectos son los más importantes y el orden en que se llevarán a cabo.

Después de aprobar la solicitud de un proyecto se estima su costo, el tiempo necesario para terminarlo y las necesidades de personal.

2.8.2.5 Determinación de los requisitos del sistema.

Los analistas, al trabajar con los empleados y administradores, deben estudiar los procesos de una empresa para dar respuesta a ciertas preguntas claves.

Para contestar estas preguntas, el analista conversa con varias personas para reunir detalles relacionados con los procesos de la empresa. Cuando no es posible entrevistar, en forma personal a los miembros de grupos grandes dentro de la organización, se emplean cuestionarios para obtener esta información.

Las investigaciones detalladas requieren el estudio de manuales y reportes, la observación en condiciones reales de las actividades del trabajo y, en algunas ocasiones, muestras de formas y documentos con el fin de comprender el proceso en su totalidad.

Reunidos los detalles, los analistas estudian los datos sobre requerimientos con la finalidad de identificar las características que debe tener el nuevo sistema.

2.8.2.6 Diseño del sistema (diseño lógico)

El diseño de un sistema de información responde a la forma en la que el sistema cumplirá con los requerimientos identificados durante la fase de análisis.

Es común que los diseñadores hagan un esquema del formato o pantalla que esperan que aparezca cuando el sistema está terminado, se realiza en papel o en la pantalla de una

terminal utilizando algunas de las herramientas automatizadas disponibles para el desarrollo de sistemas.

También se indican los datos de entrada, los que serán calculados y los que deben ser almacenados. Los diseñadores seleccionan las estructuras de archivo y los dispositivos de almacenamiento. Los procedimientos que se escriben indican cómo procesar los datos y producir salidas.

Los documentos que contienen las especificaciones de diseño representan a éste mediante diagramas, tablas y símbolos especiales.

La información detallada del diseño se proporciona al equipo de programación para comenzar la fase de desarrollo de software.

Los diseñadores son responsables de dar a los programadores las especificaciones de software completas y claramente delineadas.

2.8.2.7 Desarrollo de software (diseño físico)

Los encargados de desarrollar software pueden instalar software comprado a terceros o escribir programas diseñados a la medida del solicitante. La elección depende del costo de cada alternativa, del tiempo disponible para escribir el software y de la disponibilidad de los programadores.

Los programadores son responsables de la documentación de los programas y de explicar su codificación, esta documentación es esencial para probar el programa y hacer el mantenimiento.

2.8.2.8 Prueba de sistemas

Durante esta fase, el sistema se emplea de manera experimental para asegurarse que el software no tenga fallas, es decir, que funciona de acuerdo con las especificaciones y en la forma en que los usuarios esperan que lo haga. Se alimentan como entradas conjuntos de datos de prueba para su procesamiento y después se examinan los resultados. En ocasiones se permite que varios usuarios utilicen el sistema, para que los analistas observen si tratan de emplearlo en formas no previstas, antes de que la organización implante el sistema y dependa de él.

En muchas organizaciones, las pruebas son conducidas por personas ajenas al grupo que escribió los programas originales; para asegurarse de que las pruebas sean completas e imparciales y, por otra, que el software sea más confiable.

2.8.2.9 Implantación y evaluación

La implantación es el proceso de verificar e instalar nuevo equipo, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para utilizarla.

Cada estrategia de implantación tiene sus méritos de acuerdo con la situación que se considere dentro de la empresa. Sin importar cuál sea la estrategia utilizada, los encargados de desarrollar el sistema procuran que el uso inicial del sistema se encuentre libre de problemas.

Los sistemas de información deben mantenerse siempre al día, la implantación es un proceso de constante evolución.

La evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes. La evaluación ocurre a lo largo de cualquiera de las siguientes dimensiones:

Evaluación operacional

Valoración de la forma en que funciona el sistema, incluyendo su facilidad de uso, tiempo de respuesta, lo adecuado de los formatos de información, confiabilidad global y nivel de utilización.

Impacto organizacional

Identificación y medición de los beneficios para la organización en áreas como finanzas (costos, ingresos y ganancias), eficiencia operacional e impacto competitivo.

Opinión de los administradores

Evaluación de las actitudes de directivos y administradores dentro de la organización, así como de los usuarios finales.

Desempeño del desarrollo

La evaluación del proceso de desarrollo de acuerdo con criterios tales como tiempo y esfuerzo, debe concordar con presupuestos y estándares, y otros criterios de administración del proyecto. Cuando la evaluación de sistema se conduce en forma adecuada proporciona mucha información que puede ayudar a mejorar la efectividad de los esfuerzos de desarrollo de aplicaciones subsecuentes.

2.8.3 Ciclo en V

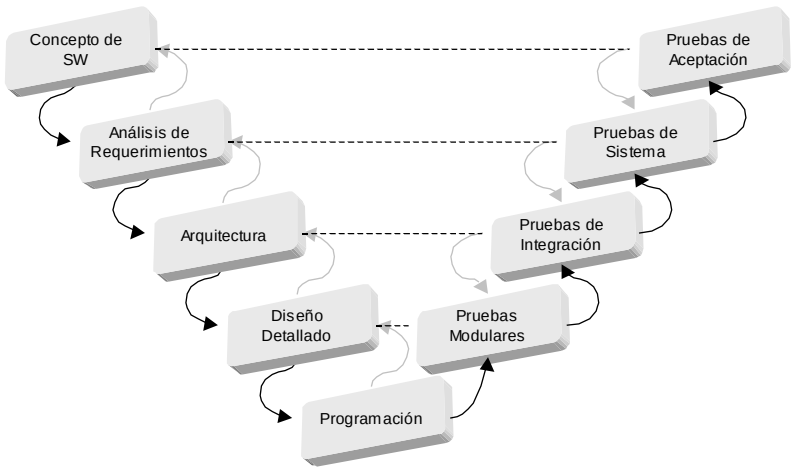

Figura 2.3

Uno de los inconvenientes del modelo en cascada es que las pruebas del software son dejadas al final del desarrollo. El ciclo de vida en V, es una variación del modelo en cascada que trata este problema y es una evolución en el cual se realizan actividades en paralelo y facilita las pruebas del sistema. Se basa en la premisa de que las pruebas de calidad no se deben dejar al final, sino realizarse a lo largo del proceso. El flujo de actividades se muestra en la figura 2.3.

2.8.4 Requisitos incrementales

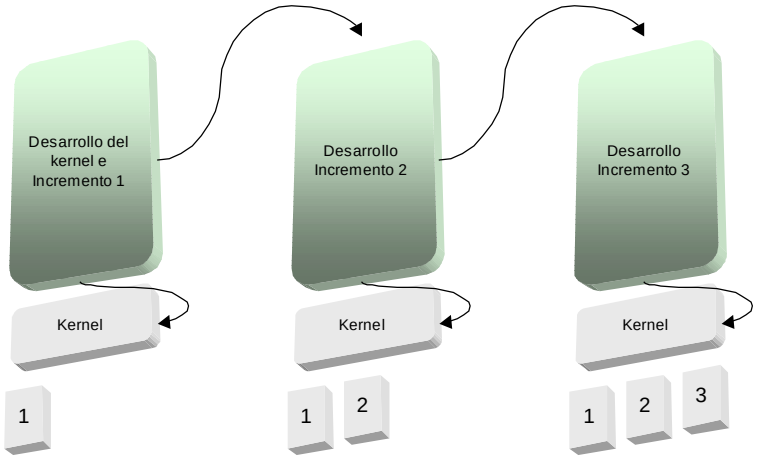

Figura 2.4

El ciclo de vida de requisitos incrementales es un modelo en el que se desarrolla el concepto del sistema a medida que el proyecto avanza. Por lo general, se comienza desarrollando los aspectos básicos del sistema o *kernel* y las características de mayor

prioridad del sistema. Luego se escogen nuevos requisitos por prioridad, el siguiente incremento, y se implementan sobre esta base. El desarrollo de cada uno de los incrementos se puede hacer aplicando otro ciclo de vida como el de cascada o el modelo en V.

2.8.5 Espiral

El modelo en espiral es un ciclo de vida orientado a tener en cuenta los riesgos del proyecto. En este modelo, se divide el proyecto en subproyectos en los que se enfocan uno o más riesgos. Un riesgo dentro de este contexto se refiere a un problema que no tenga requerimientos claros, trabajo con nuevas herramientas o tecnología de punta, una arquitectura indefinida, problemas de eficiencia o control de tiempo real. En cada una de las iteraciones de la espiral se realizan los siguientes pasos:

- Establecer objetivos, alternativas y restricciones.
- Identificar y resolver riesgos.
- Evaluar alternativas.
- Generar las entregas de esta iteración y verificar su corrección.
- Planeación de la siguiente iteración.
- Comprometerse con un enfoque para la siguiente iteración, requerimientos y riesgos (si se decide realizar la siguiente iteración).

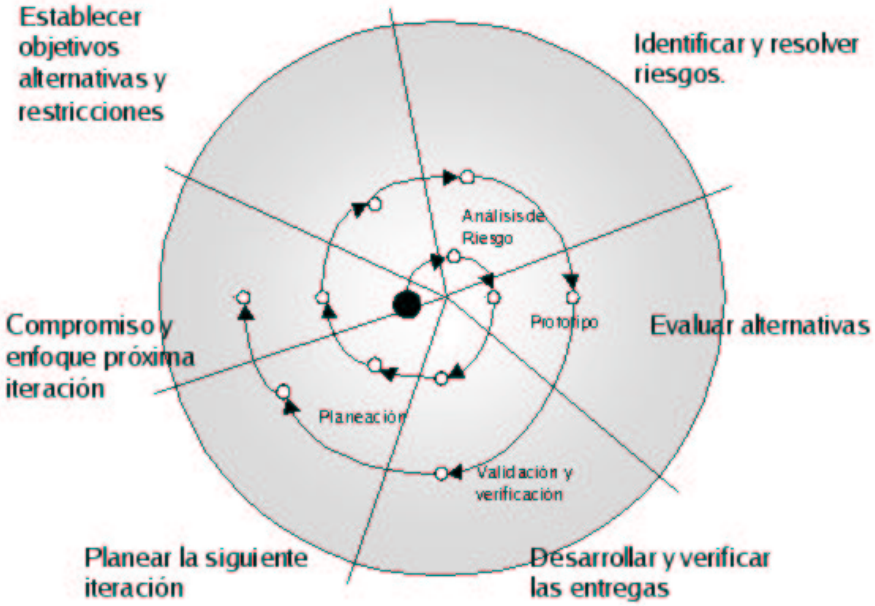

Figura 2.5

Este modelo, al igual que el de requisitos incrementales puede ser combinado con otros ciclos de vida. Su mayor fortaleza radica en que a medida que los costos del proyecto aumentan, los riesgos van disminuyendo. Su mayor debilidad es que es un modelo

complejo y requiere una administración fuerte y altamente capacitada. Este proceso se ilustra en la figura 2.5.

2.8.6 Técnicas de cuarta generación

Las técnicas de cuarta generación normalmente se refieren a las herramientas que se utilizan para hacer sistemas. Se especifican algunas características del software a construir y luego la herramienta genera automáticamente algunos programas básicos. Cuanto más se especifique el software mejor será la generación del código (más completos los programas). Normalmente a estos programas generados hay que completarlos manualmente y lograr la funcionalidad que se requiere para instalar el producto final. Las herramientas son elementos utilizados en un ciclo de vida de los que hemos visto. Algunas se adaptan mejor a una metodología que a otra. Algunas desventajas que se les atribuyen son:

1. Que no son más fáciles de usar que los lenguajes de programación o herramientas visuales.
2. Que el código generado por las herramientas es ineficiente.
3. El mantenimiento de sistemas grandes es cuestionable.

Son muy utilizados para problemas medianos y pequeños. Los tiempos de desarrollo se reducen notablemente. Existen algunas tendencias que combinan las herramientas 4GL con las técnicas de ensamblaje de componentes y parece ser éste el enfoque dominante hacia el futuro.

2.8.7 Modelo de ensamblaje de componentes.

Este modelo se basa en ir formando con la construcción de cada sistema una biblioteca de componentes (clases /objetos) , cuando se va a construir un nuevo sistema, se hace el proceso de definir los objetos del sistema, buscar en la librería de objetos, programar los que no existen, agregarlos a la biblioteca, ensamblar los objetos, la metodología busca que sea evolutiva pasando por una fase de planificación, análisis de riesgos, ingeniería, construcción y adaptación, evaluación del cliente y repetir estas fases de tal forma que las primeras iteraciones crean los conceptos, al avanzar se desarrollan los nuevos componentes, luego se busca mejorarlos y finalmente se les da mantenimiento.

2.8.8 Modelo de construcción de prototipos

Este modelo arranca con el establecimiento de los requerimientos del sistema, se definen los objetivos y los requisitos conocidos, con base en las áreas de mayor prioridad e importancia para el sistema.

Luego se hace un diseño preliminar, sobre el cual se construye un prototipo o modelo del sistema, compuesto a menudo de ventanas, tablas de la base de datos, formatos de entrada y de salida básicos.

Este prototipo se ajusta lo mejor que se pueda a la solución requerida por el usuario y sobre él se terminan de establecer los demás requerimientos del sistema.

Si el prototipo, es una versión construida sobre un buen conjunto de requerimientos, sólido y real y satisface en buena proporción las necesidades del usuario, en materia de datos y de interface, podría servir como prototipo de trabajo, que es aquél sobre el cual se empieza a construir el sistema definitivo, pero la mayoría de las veces este primer prototipo debe desecharse.

Se encuentran las siguientes críticas a este modelo:

1. Falta de calidad en el prototipo inicial, el usuario se hace a la idea de un gran avance en el desarrollo de su sistema y a veces los analistas se comprometen a arreglar un prototipo que deberían desechar.

2. La técnica es buena si se establecen las reglas de juego desde el comienzo, es de vital importancia que los usuarios, especialmente los gerentes, entiendan y magnifiquen el tamaño del proyecto, sus fases, la metodología y los tiempos.

2.8.9 Modelo de desarrollo rápido de aplicaciones RAD

El Desarrollo Rápido de Aplicaciones (RAD), utiliza un enfoque de desarrollo basado en componentes, donde se deben comprender bien los requisitos y limitar el ámbito del proyecto. Las fases son:

- Modelado de gestión. Consiste en modelar el flujo de información entre las funciones.
- Modelado de datos. Se definen los objetos de datos, sus atributos y las relaciones entre éstos.
- Modelado de proceso. Los objetos se relacionan con las funciones, para lograr el flujo de información. Las transacciones normales son adicionar, modificar, consultar y eliminar un objeto de datos.
- Generación de aplicaciones. Se asume la utilización de herramientas de cuarta generación, tratan de aprovechar al máximo los componentes reutilizables. Usan herramientas generadoras de aplicaciones (CASE).
- Pruebas y entrega. Se debe dividir el problema en módulos y cada módulo se desarrolla por un equipo RAD. Luego se integra el conjunto.

Los problemas que han encontrado a esta metodología son:

1. Se requiere que el problema sea fácilmente modularizable.
2. Se requiere de recursos humanos para cada equipo
3. Cada equipo debe estar altamente comprometido.

RAD no es recomendable cuando los riesgos técnicos del proyecto son altos. Por ejemplo, cuando se introducen nuevas herramientas, nueva tecnología no probada, o cuando se requiere de complicadas interfaces con software ya existente.

2.9 Selección del ciclo de vida

No existe “EL” modelo de ciclo de vida para todos los tipos de proyectos; sin embargo, es posible seleccionar un modelo adecuado dependiendo de las características del proyecto. Por ejemplo, si se tiene un sistema altamente estable con requerimientos bien definidos y claros, el esquema en cascada funciona bastante bien. Si se trabaja con requerimientos poco claros e incompletos el de requisitos incrementales puede ser el más adecuado. Cuando se tiene clientes muy involucrados, se desea un excelente manejo de gestión de riesgos, o se necesita una visibilidad clara del mismo, el ciclo de vida más adecuado sería el de espiral. Si se tiene una fecha fija de entrega no alterable, la combinación de modelo en V y requisitos incrementales es la más adecuada.

Con respecto al desarrollo del servidor de IRC, la mayor parte de los requisitos ya están definidos en los RFC’s, así que podemos argumentar que el sistema no sufrirá muchos cambios en su especificación.

Existen tareas de desarrollo que deben probarse al tiempo de ser realizadas. Por ejemplo, el proceso o programa que verifica que los comandos que lleguen a nuestro servidor IRC estén correctamente escritos, no puede ser probado si antes no hemos logrado la comunicación entre el servidor y los clientes, por lo que existen módulos de nuestro desarrollo que deberán probarse conforme se vaya avanzando, dejando al final sólo las pruebas totales del sistema ya que para ese momento debe estar validado el funcionamiento de todos los módulos y programas auxiliares.

En nuestro caso elegimos el ciclo de vida en V y aunque ningún ciclo de vida se acopla 100% al desarrollo de un sistema, se eligió porque las características del proyecto así lo permitieron. Por ejemplo, los requisitos del sistema se encuentran bien definidos y no se tiene contemplado agregar otros. Las pruebas se realizarán en cada fase y además se dedicará una fase completa a hacer las pruebas unitarias y de integración de los módulos.

Capítulo 3

Internet Relay Chat

El protocolo IRC ha sido diseñado para usarse en conferencias basadas en texto. Se comenzó a desarrollar en 1989 cuando fue originalmente implementado como un medio para que los usuarios de un BBS[2] pudieran conversar entre ellos. Se documentó formalmente a través del RFC 1459 y ha estado evolucionando constantemente.

La "Retransmisión de Conversaciones por Internet" es un estándar de comunicación que se basa en mensajes de texto enviados a través de una red de computadoras. Este protocolo fue creado por Jarkko Oikarinen en Finlandia como una mejora del antiguo programa "talk" que sólo podía comunicar a dos usuarios al mismo tiempo.

Se basa en el modelo cliente/servidor y está diseñado para correr en varias máquinas de una manera distribuida. Una configuración típica incluye un proceso servidor que forma un punto central para que los clientes u otros servidores se conecten, desempeñando el envío y multiplexaje[3] de mensajes requerido.

Podemos describir el proceso de comunicación de una manera muy básica tomando en cuenta que existe un servidor y varios clientes. Los clientes desean participar en una conversación entre ellos, así que se conectan al servidor. El servidor registra a cada uno de los clientes y los mantiene en una lista. Después de conectarse al servidor, cada cliente puede crear un canal, al cual se podrán unir los demás clientes que así lo deseen. Una vez que dos o más clientes se encuentran en un mismo canal, puede empezar la comunicación, ya que cada mensaje que envía un cliente a un canal, es tomado por el servidor y re-enviado a todos los demás clientes que se encuentran en ese mismo canal. También es posible elegir a un solo usuario y enviarle un mensaje privado, es decir, nadie más podrá ver el contenido del mensaje, aunque pertenezcan al mismo canal.

Este modelo distribuido, el cual requiere que cada servidor tenga una copia del estado global de información, es todavía el problema más importante del protocolo, lo cual limita el tamaño máximo que una red puede alcanzar. Si las redes existentes han sido capaces de crecer tanto es porque los fabricantes de hardware nos han dado sistemas más poderosos.

3.1 RFC's 1459, 2810, 2811, 2812 y 2813

Un RFC (Request For Comment) es un documento acerca de la Internet. El protocolo IRC está descrito en varios de estos documentos. El primero fue el número 1459 que contiene una descripción básica del protocolo. Después se escribieron los RFC´s 2810, 2811, 2812 y 2813 que son extensiones del primer documento. La Tabla 3.1 cuadro indica la información que contiene cada uno de ellos.

[2] BBS (Bulletin Board System). Es un servicio en el que los usuarios conectados pueden enviar mensajes y archivos.
[3] Multiplexaje. Técnica que permite que múltiples mensajes o señales compartan un canal de comunicación.

# RFC	Contenido
2810	Arquitectura en general del protocolo
2811	Manejo de los canales
2812	Protocolo del lado del cliente
2813	Protocolo del lado del servidor

Tabla 3.1

La implementación del sistema seguirá la sintaxis de los comandos y mensajes propios del protocolo que se muestran en estos documentos.

3.2 Componentes

Los siguientes puntos definen los componentes básicos del protocolo IRC.

Servidores

El servidor forma la columna vertebral de IRC porque es el único componente del protocolo que es capaz de unir a los otros componentes: provee un punto al cual los clientes se pueden conectar para hablarse unos a otros y un punto para otros servidores para conectarse. El servidor es también responsable de proveer los servicios básicos definidos por el protocolo IRC.

Clientes

Un cliente es cualquier elemento conectado al servidor que no es otro servidor. Hay dos tipos de clientes que sirven para diferentes propósitos. Los clientes se dividen en “clientes usuarios” que son generalmente programas que proveen una interface basada en texto que es usada para comunicarse interactivamente vía IRC. Este tipo particular de clientes es referido a menudo como "usuarios" y en clientes “servicio” que a diferencia de los usuarios, no están hechos para ser usados manualmente ni para hablar. Tienen un acceso más limitado a las funciones de transmisión del protocolo, mientras que opcionalmente, tienen acceso a más datos privados de los servidores.

Los servicios son típicamente autómatas usados para dar algún tipo de servicio (no necesariamente relacionados con IRC) a los usuarios. Un ejemplo es una recolección de estadísticas acerca del origen de los usuarios conectados a la red IRC.

Arquitectura

Una red IRC está definida por un grupo de servidores conectados unos a otros. Un solo servidor forma la red IRC más simple. La única configuración de red permitida es la de un árbol extendido donde cada servidor actúa como nodo central para el resto de la red que puede ver.

3.3 Servicios del protocolo IRC

Esta sección describe los servicios ofrecidos por el protocolo IRC. La combinación de estos servicios permite la conferencia en tiempo real.

Localizador de cliente

Para ser capaz de intercambiar mensajes, dos clientes deben ser capaces de localizarse uno a otro. Después de conectarse a un servidor, un cliente se registra usando una etiqueta, la cual es entonces usada por otros servidores y clientes para saber dónde está localizado el cliente. Los servidores son responsables de llevar un registro de todas las etiquetas que se usan.

Reenvío de mensajes

El protocolo IRC no provee forma para que dos clientes se comuniquen directamente. Toda la comunicación entre clientes es reenviada por el servidor.

Anfitrión de canales y administración

Un canal es un grupo con nombre de uno o más usuarios, los cuales reciben mensajes dirigidos a ese canal. Un canal es caracterizado por su nombre y sus miembros actuales, también tiene un conjunto de propiedades que se pueden manipular por algunos de sus miembros.

Los canales proveen una manera para enviar mensajes a varios clientes. Los servidores mantienen los canales dando el multiplexaje necesario. Los servidores son también responsables de manejar los canales llevando un registro de los miembros de los canales.

3.4 Conceptos

En esta sección se describen los conceptos básicos detrás de la organización del protocolo IRC y cómo las diferentes clases de mensajes son enviadas.

Comunicación uno a uno

La comunicación en una base de uno a uno se desarrolla usualmente por los clientes, debido a que la mayoría del tránsito servidor/servidor no es resultado de servidores hablándose uno al otro. Para proveer un medio a los clientes de hablar a los otros, se requiere que todos los servidores sean capaces de enviar un mensaje en una misma dirección a través del árbol que ellos mismos forman para alcanzar a cualquier cliente. Por lo tanto, la ruta de un mensaje enviado es la ruta más corta entre dos puntos cualesquiera del árbol extendido.

Uno a muchos

La meta principal de IRC es dar un foro que permita las conferencias de manera fácil y eficiente (conversaciones de uno a muchos). IRC ofrece varios medios para lograr esto, cada uno sirviendo a su propio propósito.

A un canal

En IRC el canal tiene un papel equivalente al de un grupo de envío múltiple (un mismo mensaje llega a varios receptores); su existencia es dinámica y la conversación actual de un canal sólo se debe enviar a los servidores que tienen conectados usuarios a ese canal. Más aún, el mensaje debe enviarse solamente a cada conexión local porque cada servidor es responsable de reenviar el mensaje original para asegurar que llegará a todos los recipientes.

A un Host / Máscara de servidor

Para dar algún mecanismo para enviar mensajes a un grupo grande de usuarios relacionados, los mensajes a host y a una máscara[4] de servidor están disponibles. Estos mensajes son enviados a usuarios cuyo host o servidor sea igual al de la máscara.

A una lista

El estilo menos eficiente de una conversación de uno a muchos es a través de una lista de destinos (cliente, canal, máscara). Esto se desarrolla de una manera intuitiva: el cliente da una lista de destinos a los cuales se enviará el mensaje y el servidor envía una copia separada del mensaje a cada destino.

Esto no es tan eficiente como usar un canal, porque la lista de destinos no es revisada para ver si hay duplicados.

Uno a todos

El tipo de mensaje de uno a todos se describe mejor como un mensaje broadcast[5], enviado a todos los clientes o servidores o a ambos. En una red grande de usuarios y servidores, un solo mensaje puede provocar mucho tránsito sobre la red en un esfuerzo para alcanzar los destinos deseados.

Para algunas clases de mensajes, no hay otra opción que el broadcast a todos los servidores de manera que la información de estado mantenida por cada servidor sea consistente entre servidores

[4] Máscara. Patrón con el que pueden corresponder varios nombres, por ejemplo, *.com.mx
[5] broadcast. En transmisiones de red, mandar un mensaje a todos los nodos conectados.

Cliente a cliente

No hay una clase de orden o comando que tenga como resultado que un mensaje vaya directamente de un cliente a otro.

Cliente a servidor

La mayoría de los comandos que provocan cambios en el estado de la información (tales como membresía a un canal, modo del canal, estado de usuario, etc.) deben ser mandados a todos los servidores por default y esta distribución no podrá ser cambiada por un cliente.

Servidor a servidor

Mientras que la mayoría de los mensajes se distribuyen a todos los otros servidores, esto sólo es requerido para cualquier mensaje que afecte a un usuario, canal o servidor. Como éstos son los elementos básicos encontrados en IRC casi todos los mensajes originados en un servidor se envían a todos los otros servidores conectados.

3.5 Problemas actuales

Existen varios problemas relacionados con la arquitectura del protocolo.

Escalabilidad

Se sabe que el protocolo IRC no escala lo suficiente cuando se usa en una red grande. El principal problema viene del requerimiento de que todos los servidores sepan a cerca de todos los otros servidores, clientes y canales y a que esta información debe actualizarse tan pronto como cambie.

Confiabilidad

Como la única red permitida para los servidores IRC es la de un árbol extendido, cada liga entre dos servidores es un punto de falla obvio y serio.

Congestión de la red

Otro problema relacionado con la escalabilidad y confiabilidad, así como con la arquitectura de árbol extendido, es que el protocolo y la arquitectura de IRC son extremadamente vulnerables a congestiones en las redes. Este problema es endémico, y deberá resolverse para la próxima versión del protocolo. Si la congestión y un alto volumen de tránsito causan que una liga entre dos servidores falle, esta falla no sólo genera más tránsito, sino que la reconexión generará aún más.

Privacidad

Además de no escalar bien, el hecho de que los servidores necesitan conocer la información de las entidades hace que el tema de la privacidad sea también un problema. Los mensajes se envían en formato de texto plano que puede ser interceptado e interpretado fácilmente por cualquiera. Un mensaje puede pasar a través de varios servidores antes de llegar a su destino y cualquiera de estas conexiones puede ser víctima de una intercepción de los datos.

Consideraciones de seguridad

Los mensajes de IRC pueden ser interceptados de manera muy sencilla debido a que no están encriptados. El protocolo no considera como prioridad la seguridad en las comunicaciones. El sistema a desarrollar no incluirá características de seguridad porque la información que se desea transmitir por este medio tiene carácter académico y se desea que sea difundido de todas las formas posibles, así que no es importante si alguien se entera de las conversaciones que se llevarán a cabo a través del servidor que implementaremos.

El propósito del sistema no es transmitir conversaciones privadas o datos que puedan resultar confidenciales. Su único propósito es ayudar a transmitir asesoría a los estudiantes que utilicen el sistema y cualquier mensaje o comentario que se haga a través de este medio tiene como fin informar a cualquier persona que esté interesada. Los canales a través de los cuales se transmiten los mensajes son generalmente públicos y cualquiera se puede unir a ellos. La excepción son los canales privados, pero su protección consiste solamente en no dejar que cualquier persona participe en ellos. No existe ningún otro tipo de previsión para ocultar los mensajes por no considerarse necesaria.

Capítulo 4

Redes de computadoras

En este capítulo trataremos sobre las redes de computadoras debido a su importancia central en el sistema que se está construyendo. Es necesario entender cómo funciona la red para lograr comunicar las computadoras utilizando los protocolos estándar. El objetivo principal de esta introducción a la redes de computadoras es entender su estructura para poder utilizarlas eficientemente y conocer los conceptos que se utilizarán durante la construcción del programa.

Una red de computadoras es un grupo de dos o más sistemas computacionales llamados nodos o estaciones, que están conectados o se pueden comunicar unos con otros de alguna manera. En la actualidad las redes se están utilizando cada vez más en todos los campos de la actividad humana debido a que permiten compartir recursos computacionales y facilitan la comunicación, aun cuando las distancias sean muy grandes.

Las computadoras se utilizan en los hogares para aprender y jugar, en las oficinas, bancos e instituciones financieras como procesadores de texto, manejadores de hojas de cálculo y para manejar bases de datos, en las agencias de viajes para hacer y verificar reservaciones, en las escuelas y universidades para el aprendizaje y entrenamiento, análisis científico y simulaciones, en las fábricas para el control de procesos químicos, control de robots, etc. Si combinamos todo lo que podemos hacer con las computadoras por medio de una red, las posibilidades de trabajo en equipo aumentan y la capacidad de compartir los recursos que una computadora en específico puede brindar, facilita el trabajo de muchas personas. Aunque algunas computadoras desempeñan un papel en el que no necesitan de otras, existen aplicaciones que hacen necesaria la comunicación entre dos o más. Por ejemplo, la transferencia de dinero entre dos bancos, la obtención de datos de fuentes públicas, copiar un archivo de una computadora personal a otra, el envío de correo electrónico y otras actividades que cada día son más comunes.

Sea cual sea la aplicación que se le dé a una red de computadoras, debe existir una infraestructura de comunicación que sea independiente de las aplicaciones y nodos que la utilizan, porque no todos los nodos de una red son siempre iguales, no tienen siempre la misma arquitectura, no tienen el mismo sistema operativo o utilizan diferentes implementaciones de una misma aplicación. Se debe separar la estructura de comunicación de manera que los nodos no dependan del tipo de red a la que se conectan. Debe haber alguna manera de aislar la comunicación de usuario a usuario de lo que es la comunicación del nodo a la red, es decir, cualquier nodo debería poder conectarse a cualquier red realizando sólo ajustes pequeños o que por lo menos no impliquen cambios radicales en la configuración del sistema. Si se logra esto, entonces el tipo de red que se utilice se puede elegir de acuerdo a las necesidades de cada aplicación. Por ejemplo, puede utilizarse una simple conexión punto a punto, puede usarse un MODEM[6] para conectarse a la red telefónica pública o se puede instalar una red propia.

Existen muchos tipos de computadoras y de redes y poder unirlas en una red sería imposible si no siguiéramos ciertas reglas comunes. Es por eso que existen subsistemas o módulos de hardware y software que permiten ser intercambiados para lograr casi cualquier

[6] MODEM – Dispositivo de comunicación que convierte las señales binarias en analógicas y analógicas en binarias para su transmisión a través de las líneas telefónicas.

combinación que necesitemos. Podemos pensar que un nodo o una computadora incluye un subsistema de comunicación configurable, el cual se puede ajustar al tipo de red con la que se cuente, así, para conectar una computadora a otro tipo de red, sólo se configura el subsistema de comunicación y el resto de nuestro sistema computacional seguirá comportándose de la misma manera.

El tipo de red depende de la naturaleza de la aplicación, el número de computadoras a conectar y su separación física. Si se tratan de conectar sólo dos computadoras y se encuentran en el mismo lugar, entonces la transmisión puede hacerse con una conexión punto a punto que se lograr con un cable.

Si los nodos se encuentran en diferentes lugares de una ciudad o país, se puede utilizar la red telefónica pública, para lo cual se utiliza un MODEM.

Cuando se trata de conectar varias computadoras, normalmente se utiliza una red para permitir la comunicación de una computadora a otra en cualquier momento. Si las computadoras están en una misma habitación o edificio, es posible instalar una red propia que es conocida como red de área local o LAN por sus siglas en inglés (Local Area Network).

Si las computadoras están localizadas en diferentes sitios, se debe usar alguna red pública. La red resultante es una red de área amplia o WAN (Wide Area Network). Si alguna empresa tiene la necesidad de transmitir grandes volúmenes de datos, puede rentar líneas de transmisión que nadie más utilizará y por lo tanto le darán mayor seguridad y velocidad en sus comunicaciones.

4.1 Estándares

Un estándar es una norma, modelo o conjunto de reglas que se siguen para realizar alguna tarea o actividad. Cuando no se cuenta con estándares, los fabricantes de hardware utilizan cada uno su propia manera de crear y hacer funcionar sus productos y esto pasó con los subsistemas de comunicación. El resultado fue que los subsistemas de comunicación que los fabricantes ofrecían, sólo funcionaban con sus propias computadoras. Estos sistemas son conocidos como sistemas cerrados dado que las computadoras de otros fabricantes no pueden intercambiar información con ellos a menos que adopten el estándar de ese fabricante en particular.

En contraste, varios cuerpos internacionales relacionados con las redes públicas han formulado estándares para conectar dispositivos a estas redes. Estos estándares o recomendaciones a veces se dividen en series. Las recomendaciones han dado como resultado compatibilidad entre equipo de diferentes vendedores permitiendo a un comprador seleccionar equipo útil de un conjunto de fabricantes. Consecuentemente, el equipo de un fabricante que se apegue a estos estándares puede ser usado intercambiablemente con equipo de cualquier otro fabricante que cumpla con los estándares. El equipo resultante es entonces conocido como un sistema abierto o como un ambiente de interconexión de sistemas abiertos u OSIE (Open System Interconnection Enviroment).

A mediados de los 70's muchos tipos de sistemas distribuidos comenzaron a proliferar, las ventajas potenciales de los sistemas abiertos fueron conocidas por la industria computacional. Como resultado, un conjunto de estándares comenzó a ser utilizado. El primero se relacionaba con toda la estructura de un subsistema de comunicación completo dentro de cada computadora. Éste fue producido por la Organización Internacional de Estándares o ISO (International Standards Organization) y es conocido como el modelo de referencia ISO para interconexión de sistemas abiertos u OSI (Open Systems Interconnection).

La idea del modelo de referencia ISO es proveer un marco para la coordinación del desarrollo de estándares y permitir a las actividades existentes y por venir, ser colocadas dentro de un marco común. La idea es permitir a un proceso en cualquier computadora que soporte un conjunto particular de estándares, comunicarse libremente con un proceso en cualquier otra computadora que soporte los mismos estándares, sin importar el fabricante.

La interconexión de sistemas abiertos se relaciona con el intercambio de información entre tales procesos. Se permite así que los procesos cooperen en la realización de un tratamiento de información en particular sin importar las computadoras en las que están funcionando.

4.2 Modelo de referencia de la ISO

Un subsistema de comunicación es una pieza de software y hardware compleja. Los primeros intentos por implementar el software para estos sistemas estaban basados a menudo en un solo programa complejo y sin estructura (normalmente escrito en ensamblador) con muchos componentes interactivos. El software resultante era difícil de probar y a menudo difícil de modificar.

Para superar este problema, la ISO ha adoptado una aproximación por capas para el modelo de referencia. El subsistema de comunicación está dividido en capas, cada una de las cuales desarrolla una función bien definida. Conceptualmente, se puede considerar que estas capas desempeñan una de dos funciones genéricas: funciones dependientes de la red y funciones orientadas a aplicación.

Cada capa desempeña una función definida en el contexto de todo el subsistema de comunicación. Opera de acuerdo a un protocolo definido por mensajes de intercambio que son datos de usuario e información adicional de control con una capa gemela correspondiente en un sistema remoto. Cada capa tiene una interface bien definida entre sí y la capa inmediatamente superior e inferior. Consecuentemente, la implementación de una capa particular de protocolo es independiente de las otras capas.

La estructura lógica del modelo de referencia ISO está hecha de siete capas. Las tres capas más bajas son dependientes de la red y se relacionan con los protocolos asociados con la red de comunicación de datos usada para unir dos computadoras. En contraste, las tres capas superiores están orientadas a la aplicación y se relacionan con los protocolos que

permiten a dos procesos de usuario final interactuar uno con otro, normalmente a través de un conjunto de servicios ofrecidos por el sistema operativo local.

La capa intermedia de transporte enmascara a las aplicaciones superiores de la operación detallada de las capas más bajas. Esencialmente, se construye sobre los servicios que dan las capas bajas para dar a las capas altas un servicio de intercambio de mensajes independiente de la red.

4.3 Las capas

La función de cada capa se especifica formalmente como un protocolo que define el conjunto de reglas y convenciones usadas por la capa para comunicarse con una capa par similar en otro sistema. Cada capa proporciona un conjunto de servicios a la capa inmediatamente superior. También usa los servicios dados por la capa inmediatamente inferior para transportar las unidades de mensaje asociadas con el protocolo a la capa gemela remota. Por ejemplo, la capa de transporte da un servicio de transporte de mensajes independiente de la red a la capa de sesión que está encima y usa los servicios de la capa de red que está debajo para transferir el conjunto de unidades de mensaje a la capa del otro sistema. Conceptualmente, por lo tanto, cada capa se comunica con una capa similar en un sistema remoto de acuerdo a un protocolo definido. En la práctica, las unidades de mensaje resultantes de un protocolo de una capa son pasadas por medio de los servicios de la capa inmediatamente inferior.

4.3.1 Las capas orientadas a aplicación

Son las capas en las que las aplicaciones se comunican con la red utilizando sus servicios de transmisión. Se relacionan con los servicios de alto nivel utilizados por las diferentes aplicaciones, por ejemplo, la identificación, manejo de permisos, sincronización, etc.

4.3.1.1 La capa de aplicación

Proporciona la interface para el usuario hacia un conjunto de servicios de información distribuidos. Esto incluye manejo y administración de transferencia de archivos, así como servicios generales de intercambio de mensajes y documentos generales, tales como correo electrónico. Cierto número de protocolos estándar está disponible o están siendo desarrollados para éstos y otros tipos de servicios. El acceso a los servicios de aplicación se realiza normalmente a través de un conjunto definido de primitivas, cada una con parámetros asociados que son soportados por el sistema operativo local. Las primitivas de acceso son iguales a otras llamadas del sistema operativo (como las usadas para tener acceso a un sistema de archivos local) y resultan en la activación de un procedimiento apropiado del sistema operativo. Estos procedimientos del sistema operativo utilizan el subsistema de comunicación (software y hardware) como si fuera un dispositivo local. El detalle de las operaciones y la implementación del subsistema de comunicación es por tanto transparente al usuario o al proceso de aplicación.

Cuando un proceso de aplicación hace una llamada a estos procesos, uno o varios valores de retorno indican si hubo éxito o falla en la transacción de red que se ejecutó. En adición a la transferencia de información, la capa de aplicación provee servicios como:

- identificación de los socios de comunicación por nombre o por dirección
- determinación de la disponibilidad actual de un socio de comunicación
- establecimiento de autoridad para comunicarse
- conciliación de mecanismos de privacidad (encripción)
- autentificación de un socio de comunicación
- selección de la disciplina de diálogo, incluyendo los procedimientos de iniciación y fin
- conciliación sobre la responsabilidad de recuperación ante errores
- identificación de restricciones en la sintaxis de los datos

4.3.1.2 La capa de presentación

Se relaciona con la representación (sintaxis) de los datos durante la transferencia entre dos procesos de aplicación que se comunican. Para lograr verdadera interconexión de sistemas abiertos, un conjunto de formas comunes de sintaxis de datos abstractos han sido definidos para usarse por procesos de aplicación junto con sintaxis asociadas o concretas. La capa de presentación negocia y selecciona la sintaxis de transferencia apropiada a ser usada durante una transacción de manera que la estructura de la sintaxis del mensaje siendo intercambiado entre dos entidades de aplicación sea mantenida. Entonces, si esta forma de representación es diferente de la forma abstracta interna, la entidad de presentación desempeña la conversión necesaria.

Para ilustrar los servicios ofrecidos por la capa de presentación, podemos considerar una conversación telefónica entre un francés y un español, asumiendo que cada uno utiliza un intérprete y que el único lenguaje que entienden los dos intérpretes es el inglés. Cada intérprete debe traducir desde su lenguaje local al inglés y viceversa. Los dos intérpretes son entonces análogos a dos procesos de aplicación y representan las entidades de la capa de presentación. Francés y español son las dos sintaxis locales y el inglés la sintaxis concreta o de transferencia. Nótese que debe haber un lenguaje entendido universalmente que ha de definirse para permitir que el lenguaje de transferencia acordado sea negociado. También hay que notar que los intérpretes no necesariamente entienden el significado (semántica) de la conversación.

Otra función de la capa de presentación se relaciona con la seguridad de los datos. En algunas aplicaciones, los datos mandados por una aplicación son primero encriptados utilizando una llave, la cual es conocida sólo por la capa de representación recipiente o destino. La última desencripta cualquier dato recibido usando la llave correspondiente antes de pasarlo al recipiente destino.

4.3.1.3 La capa de sesión

Proporciona los medios para habilitar a dos entidades de la capa de aplicación de manera que puedan organizarse y sincronizar su diálogo y manejar su intercambio de datos. Es por lo tanto, responsable de organizar un canal de comunicación entre dos entidades de protocolo de la capa de aplicación durante la transacción de red completa. Además, se proporciona un número de servicios opcionales como pueden ser:

- Manejo de interacción. El intercambio de datos asociado con un diálogo puede ser duplex[7] o half-duplex[8]. En el último caso se proporcionan facilidades para controlar el intercambio de datos de manera sincronizada.

- Sincronización. Para transacciones de red largas, el usuario puede escoger establecer periódicamente puntos asociados con la transferencia. Entonces, si sucede un error durante una transacción, el diálogo puede ser reiniciado en un punto de sincronización acordado.

- Reporte de excepciones. Las excepciones no recuperables ocurridas durante una transacción pueden ser señaladas a la capa de aplicación por la capa de sesión.

4.3.1.4 La capa de transporte

Actúa como la interface entre las capas altas orientadas a aplicación y las capas bajas dependientes de red. Proporciona a la capa de sesión una facilidad para la transferencia de mensajes que es independiente del tipo de red. Dando a la capa de sesión un conjunto definido de facilidades de transferencia de mensajes, la capa de transporte esconde la operación detallada de la red a la capa de sesión.

La capa de transporte ofrece varias clases de servicios como son:

- Clase 0. Proporciona sólo las funciones básicas necesarias para el establecimiento de la conexión y la transferencia de datos.

- Clase 4. Proporciona control de errores completo y procedimientos de control de flujo.

[7] Duplex. Capacidad de transmisión en cualquier sentido al mismo tiempo.
[8] Half-duplex. Capacidad de transmisión en cualquier sentido, pero sólo en uno a la vez.

4.3.2 Las capas dependientes de la red

Como las tres capas más bajas del modelo de referencia ISO son dependientes de la red, su operación detallada varía de un tipo de red a otra. En general, la capa de red es responsable de establecer y limpiar una conexión entre dos entidades de protocolo de capa de transporte. Incluye facilidades como ruteo de red y en algunas instancias control de flujo a través de la interface de computadora a red. En el caso de conexión de red a red provee varias funciones de armonización entre las redes.

La capa de ligado de datos se construye sobre la conexión física que ofrece una red en particular y da a la capa de red facilidades de transferencia de información confiable, es por lo tanto, responsable de funciones como detección de errores y en el evento de errores de transmisión, la retransmisión de mensajes. Normalmente se ofrecen dos tipos de servicios:

- No orientado a conexión, el cual trata a cada marco de información como una unidad auto contenida que es transferida usando la aproximación del "mejor intento"; esto es, si se detectan errores en un marco, entonces ese marco se descarta.

- Orientado a conexión, el cual se esfuerza en dar una facilidad de transferencia libre de errores.

Finalmente, la capa física se relaciona con las interfaces eléctrica y física entre el equipo del usuario y el equipo terminal de red. Da a la capa de ligado de datos, medios para transmitir un flujo serial de bits entre los dos equipos.

4.4 Estándares de sistemas abiertos

El modelo de referencia ISO ha sido formulado simplemente como una base para la estructura de un subsistema de comunicación en el cual las actividades estándares asociadas con cada capa pueden usarse como base. No se trata de que haya un solo protocolo asociado con cada capa, en lugar de esto, un conjunto de estándares está asociado con cada capa, cada uno ofreciendo diferentes niveles de funcionalidad. Entonces, para un ambiente de interconexión de sistemas abiertos específico, un conjunto seleccionado de estándares está definido para usarse por todos los sistemas de ese ambiente.

Los tres mayores cuerpos internacionales que producen estándares para las comunicaciones entre computadoras son la ISO, la IEEE (Institute of Electrical and Electronics Engineers) y la CCITT (Comité Consultatif International Téléphonique et Télégraphique) ahora conocida como ITU (International Telecommunication Union). Esencialmente, ISO e IEEE producen estándares para que los usen los fabricantes de computadoras, mientras que ITU define estándares para conectar equipo a los diferentes tipos de redes públicas nacionales e internacionales. Como el grado de traslape entre las industrias de computadoras y telecomunicaciones es grande, hay un nivel de cooperación que se incrementa entre los estándares producidos por estas organizaciones.

En adición, antes y al mismo tiempo que la actividad de estandarización de ISO, la USDD (United States Defense Deparment) ha hecho investigaciones por varios años sobre comunicaciones y redes a través de la DARPA (Defense Advanced Research Projects Agency). Como parte de esta investigación, las redes de computadoras asociadas con muchas universidades y otros establecimientos de investigación se unieron a DARPA. La red resultante conocida como ARPANET fue extendida para incorporar redes desarrolladas por otras agencias de gobierno. La red internacional combinada es ahora conocida simplemente como la Internet.

El conjunto de protocolos usados con la Internet es conocido como Transmission Control Protocol/Internet Protocol (TCP/IP). Incluye protocolos orientados a red y protocolos de soporte a aplicaciones. Debido a que TCP/IP es muy común, muchos de los protocolos de TCP/IP han sido usados como la base para los estándares ISO. Más aún, como todas las especificaciones de protocolos asociadas con TCP/IP son de dominio público - y por lo tanto no se pagan licencias - han sido utilizadas extensivamente por autoridades públicas y comerciales para crear ambientes de red de sistema abierto. En la práctica, por lo tanto, hay dos sistemas abiertos mayores estándares, el conjunto de protocolos TCP/IP y los que se basan en protocolos ISO.

4.5 Tipos de redes

El tipo de medio de comunicación usado está en función de la naturaleza de la aplicación, el número de computadoras que han de unirse y su separación física.

Si se van a conectar sólo dos computadoras y ambas están en la misma habitación u oficina, entonces el medio de transmisión se puede hacer con una conexión simple de un cable de punto a punto. Sin embargo, si están localizadas en diferentes partes de una ciudad o país, las redes públicas deben ser usadas. Normalmente esto implica la PSTN (Public Switched Telephone Network), la cual requiere un dispositivo conocido como MODEM para transmitir los datos.

Cuando más de dos computadoras están implicadas en la aplicación, una red switcheada de comunicación permite a todas las computadoras comunicarse unas con otras en distintos momentos. Si todas las computadoras están distribuidas en una oficina o edificio, es posible instalar una red propia. Estas redes son conocidas como redes de área local o LANs (Local Area Network). Se cuenta con varios tipos de estas redes y existe mucho equipo disponible.

Cuando las computadoras se localizan en diferentes lugares, las redes públicas deben utilizarse de nuevo. La red resultante es conocida como una red de área amplia o WAN (Wide Area Network). El tipo de WAN usada dependerá de la naturaleza de la aplicación. Por ejemplo, si todas las computadoras pertenecen a la misma empresa y existe un requerimiento de transferir cantidades substanciales de datos entre varios sitios, una forma es simplemente rentar líneas de transmisión de las redes públicas e instalar un sistema privado de switcheo en cada sitio para crear lo que es conocido como una red privada empresarial. Muchas empresas grandes eligen hacer esto, tales redes normalmente incorporan comunicaciones de voz y datos.

Esta solución es sólo viable para empresas grandes, ya que sólo entonces hay suficiente tránsito para justificar el costo de las líneas rentadas y la instalación y puesta en marcha de una red privada. En otros casos, por lo tanto, se deben usar las redes públicas.

Además de proveer un servicio telefónico, la mayoría de los proveedores públicos de telefonía ofrecen un servicio de transmisión de datos. Consecuentemente, para aplicaciones que implican computadoras distribuidas por el país o quizás internacionalmente se usa una red pública switcheada de datos o PSDN (Public Switched Data Network). Alternativamente, muchos proveedores públicos están convirtiendo ahora sus redes telefónicas existentes para permitir transmitir datos sin modems. Las redes resultantes, las cuales operan en un modo totalmente digital son conocidas como redes integradas de servicios digitales o ISDN (Integrated Services Digital Network). Cuando las ISDNs se utilicen mucho, este tipo de facilidad se deberá considerar también.

En algunas aplicaciones el medio de comunicación abarca múltiples redes tales como LAN-WAN-LAN. Por ejemplo, una computadora unida a una LAN en un establecimiento puede requerir comunicarse con una computadora que está unida a una LAN en otro sitio y las dos LANs se conectan por medio de una red pública. Este tipo de comunicación que es conocida como “comunicación entre redes” requiere de elementos adicionales en relación a la red misma y a la interface con las computadoras.

Existen muchos tipos de redes de computadoras y se pueden clasificar con base en las siguientes propiedades o funciones:

- Capacidad de la tasa de transferencia. Indica si la red puede transmitir uno o más mensajes a la vez. Las redes son baseband, carrierband o broadband.

- Cobertura. El rango geográfico sobre el cual los nodos están distribuidos. Las redes se pueden dividir en LAN's, WAN's, MAN's, CAN's, DAN's y GAN's que son local, amplia, metropolitana, de campus, departamental y global respectivamente (local-, wide-, metropolitan-, campus-, departamental- y global- area network).

- Tipos de nodos. Los nodos de una red pueden ser PC's, mainframes, minicomputadoras e inclusive otras redes. Las redes cuyos nodos son otras redes más pequeñas se conocen como redes de acceso o backbones.

- Relación entre nodos. Se pueden categorizar como distribuidas, punto a punto, basadas en servidor y cliente/servidor.

- Topología. La topología puede ser lógica (forma lógica de los nodos en la red) o física (indica el esquema de alambrado con el cual están unidos los nodos). Las principales topologías lógicas son las de bus y de anillo. Las principales topologías físicas son las de bus, de anillo y de estrella.

- Arquitectura. Se define por el cableado utilizado, por el método usado para acceder a la red y por el formato de los paquetes de datos. Algunas arquitecturas comunes son Ethernet, Token Ring, ARCnet y FDDI.

- Posibilidades de acceso. En un extremo están las redes de medio compartido, en las cuales exactamente un nodo puede tener acceso al medio en un momento dado. En contraste las redes switcheadas permiten a múltiples nodos utilizar la red al mismo tiempo. Esto se logra multiplexando.

4.5.1 Capacidad de la tasa de transferencia

En general, las redes broadband[9] soportan una tasa de transferencia más alta, pero en las redes baseband[10] la velocidad de transmisión es muy variable y hay cierto traslape con las broadband.

Si tomamos en cuenta la velocidad, se pueden distinguir varios tipos de redes: Las primeras redes operaban a kilobits por segundo (kbps) hasta unos pocos de cientos de kilobits por segundo. La generación siguiente tenía la velocidad de las redes Ethernet, TokenRing y ARCnet de 1 a 20 Mega bits por segundo (Mbps). Las velocidades tradicionales son de 10 Mbps o menores, la red Token Ring de 16 Mbps y la ARCnet de 20 Mbps son mejoras de los diseños originales.

Más adelante se lograron velocidades de 100 Mbps. Esto incluye FDDI, ATM y Fast Ethernet y de más de 1 gigabit por segundo (Gbps).

4.5.2 Cobertura

Se pueden clasificar las redes por su rango de cobertura. Las categorías más comunes son las redes de área local, las redes de área amplia y las redes de área global que se vuelven cada vez más populares según las compañías internacionales unen sus operaciones en todos los países. Las LAN's incluyen normalmente sólo PC's, las WAN's generalmente incluyen algún tipo de conexión remota.

Tipos de nodos.

Las redes LAN's basadas en PC's son las más comunes y las que más crecimiento han tenido. Se pueden encontrar en escuelas y empresas de todo tipo.

Las redes que incluyen mini computadoras o mainframes se encuentran normalmente en universidades o empresas grandes. Durante varias décadas, estas redes han sido dominadas por las mainframes de IBM. Estas redes consisten principalmente de

[9] Broadband. Se utilizan múltiples canales de comunicación simultáneamente. Los datos se modulan en frecuencia.
[10] Baseband. Transmisión digital de señales sin modulación, opcionalmente utilizando multiplexaje en el tiempo.

terminales o PC's que corren emuladores de terminal (pretenden que no son más que una terminal) para comunicarse con la mainframe.

Las redes basadas en mainframes normalmente cumplen con la arquitectura SNA (Systems Network Architecture) de IBM (International Business MAchines) y si se incluyen PCs que funcionen más que como simples terminales se utiliza SAA (System Application Architecture). SNA y SAA proveen modelos comprensibles para controlar los detalles de la operación de la red y la comunicación a ciertos niveles.

En instalaciones universitarias, las redes distribuidas son muy comunes. En tales redes no hay un control centralizado. En lugar de esto, los nodos ofrecen varios servicios de red. Los ambientes UNIX usan este tipo de redes distribuidas.

Las redes basadas en mini computadoras y mainframes a menudo proveen servicios a LANs. Los nodos en la LAN tienen acceso a la red basada en mainframe a través de compuertas. Las ventajas reales de una arquitectura en capas se ven claramente en las interacciones entre los mundos tan diferentes de las LANs y las redes SNA.

Las redes backbone[11] están diseñadas con pequeñas redes como nodos. Tales redes son capaces de brindar las ventajas de redes muy grandes y heterogéneas y a la vez permiten la sencillez de una LAN. Las redes de acceso pueden operar como redes independientes para la mayoría, pero pueden tener acceso a los recursos de otras redes unidas al backbone siempre y cuando se tengan los derechos necesarios.

4.5.3 Relación entre nodos

Los nodos de una red pueden ser servidores o estaciones de trabajo. Una estación de trabajo hace peticiones y un servidor las atiende. El servidor controla la red dando al usuario a través de la estación de trabajo sólo los servicios que necesita.

Los siguientes términos se usan para describir la relación entre nodos en una red.

- Par a par. Cada nodo puede ser cliente o servidor, esto es, todos los nodos son iguales. Las redes par a par son útiles si es necesario conectar sólo algunas máquinas (generalmente menos de 10) y si nadie va a correr programas que requieran de muchos recursos.

- Distribuidas. Una red sin líder, esto es, una red en la cual cualquier nodo le puede hablar a cualquier otro. Un ejemplo de red distribuida es Usenet, la cual es popular entre la comunidad UNIX. En una red distribuida los servidores son sólo máquinas, dispositivos o programas que dan servicios y no controlan la actividad de la red.

- Basadas en servidor. Una red con un servidor de archivos dedicado. El servidor controla la red concediendo a otros usuarios acceso a los recursos. La mayoría de

[11] Backbone. Una red central a la cual se unen otras redes.

las redes medianas y grandes están basadas en red y los sistemas operativos de redes basadas en PCs más populares (NetWare de Novell, LAN Manager de Microsoft, LAN Server de IBM y Banyan de VINES) asumen una red basada en servidor.

- Cliente/Servidor. Una versión avanzada de una red basada en servidor. Mientras que los nodos de una red basada en servidor tienen acceso a todo tipo de recursos a través del servidor, las estaciones de trabajo deben hacer la mayoría del trabajo. El servidor distribuye los recursos y después deja que la estación de trabajo corra el programa. En la forma más general de la configuración cliente/servidor, la estación de trabajo hace una petición y el servidor procesa la petición y regresa los resultados a la estación de trabajo. De una manera común, un proceso de front-end corriendo en el cliente manda una petición al back-end, éste hace el trabajo pedido y regresa los resultados al cliente.

4.6 Topología

Existen cientos de maneras de conectar computadoras a una red. Afortunadamente, estas posibilidades se reducen a unas pocos tipos fundamentales. Cuando se discute acerca de la forma de las redes o topologías, es útil distinguir entre la forma física y la lógica. La topología lógica especifica el flujo de información y comunicación en la red. La topología física especifica el cableado que une los nodos de la red.

4.6.1 Topologías lógicas

Las dos principales topologías lógicas son las de bus y anillo. En una topología de bus, la información es transmitida a través de un simple cable, llamado el cable principal. Todos los nodos unidos a la red pueden oir la información casi al mismo tiempo. Sólo los nodos a quienes está dirigida la información leen y procesan los paquetes transmitidos. La emisión de información y el acceso simultáneo caracterizan a una topología de bus.

Como todos los nodos oyen una transmisión al mismo tiempo, métodos de contención de acceso a la red tales como CSMA/CD (Carrier Sense Multiple Access/Collision Detection) pueden usarse. En los métodos de contención de acceso al medio, los nodos obtienen derechos de acceso al medio al ser los primeros en pedirlos cuando no hay actividad en la red.

En una topología de anillo, la información es pasada de nodo a nodo en un anillo. Cada nodo obtiene información de exactamente un nodo y la transmite exactamente a un nodo. Los nodos ganan acceso a los mensajes secuencialmente (en una secuencia predeterminada), generalmente basada en las direcciones de red. Como en todas las redes, se espera que los nodos sólo procesen la información que está dirigida a ellos.

Como los nodos no escuchan la transmisión al mismo tiempo, los métodos de contención de acceso no pueden basarse en la contención por derechos de transmisión. En lugar de esto se utilizan métodos determinísticos como token passing.

4.6.2 Topologías físicas

Mientras que la topología lógica controla cómo se mueve la información a través de la red, la topología física o esquema de cableado, controla cómo se mueven las señales eléctricas a través de la red. Esto tiene consecuencias para el estado de la red si un nodo falla.

Por ejemplo, un esquema de cableado en bus requiere poco cable, pero puede hacer la resolución de problemas más difícil que un esquema de cableado en estrella. Si un nodo unido al bus falla, el servidor no tiene manera de saberlo hasta que manda un mensaje al nodo y no obtiene respuesta. En contraste, un cableado en estrella usa mucho cable porque cada nodo puede estar a una distancia considerable del nodo central o hub, pero es fácil determinar cuando un nodo falla, porque el nodo central se puede comunicar directamente con cada nodo.

Aunque existen muchas maneras de etiquetar un esquema de cableado de red, la mayoría de éstos cae en uno de los siguientes grupos:

- Bus. Un cable central forma la columna vertebral de la red y los nodos están unidos a este cable directamente o por medio de una pequeña pieza de cable. Las señales viajan a través del bus y cada nodo investiga en todos los mensajes, leyendo sólo aquellos que están dirigidos a él. Ethernet y ciertas versiones de ARCnet usan una topología de bus. Las variantes de una topología de bus incluyen árbol y árbol extendido.

- Anillo. Los nodos están organizados en más o menos un círculo, cada nodo está conectado al nodo inmediatamente anterior e inmediatamente posterior a él. Los mensajes son pasados a través del anillo en secuencia. De nuevo, un nodo toma el mensaje si es el recipiente, de lo contrario pasa el mensaje.

- Estrella. Todos los nodos están conectados a una máquina central o centro de cableado (como un hub). Los mensajes pueden ser enviados directamente a su destino desde el centro. Algunas versiones de ARCnet usan una topología en estrella. Una red de estrella distribuida es una variante en la cual varios hubs están conectados y cada uno forma una estrella.

- Anillo-Estrella. Todos los nodos están unidos a un centro de cableado en una topología de estrella, pero los nodos son accedidos como si estuvieran en un anillo. Algunas redes Token Ring de IBM usan esta topología.

4.7 Arquitectura

Las arquitecturas de red difieren en el cableado usado (coaxial, par trenzado, fibra óptica), los métodos usados para acceder la red (CSMA/CD, token passing, polling), el formato de los paquetes de datos a través de la red y la topología de la red.

En general, arquitecturas diferentes de red necesitan traductores para hablarse unas a otras. Los ruteadores y los hubs multiarquitectura ayudan a hacer tales conexiones transparentes al usuario.

Las arquitecturas más comunes son Ethernet/IEEE 802.3, ARCnet, Token Ring y FDDI.

4.7.1 Posibilidades de acceso

Las redes pueden funcionar compartiendo el medio de acceso[12] o a través de switches. Comúnmente, en las arquitecturas de medio compartido (tales como Ethernet o Token Ring), sólo un nodo puede transmitir en un momento dado, esto es, el acceso al medio de la red es exclusivo. La forma en que un nodo accede al medio depende del método de acceso usado (por ejemplo, CSMA/CD, token pasing o polling).

Las redes switcheadas, en contraste, establecen conexiones temporales según se necesitan entre dos partes. Tales redes utilizan multiplexación para permitir a múltiples nodos transmitir al mismo tiempo. Las bases usadas para el switcheo distinguen tales redes. Las redes pueden ser de paquete switcheado, circuito switcheado o mensaje switcheado.

4.8 Internet y TCP/IP

Como un término general, Internet es una red internacional, la cual consiste de muchas redes más pequeñas que se pueden comunicar una con otra. Como una referencia específica, la Internet es la red gigantesca internacional creada originalmente para unir varias redes de investigación y defensa. Desde entonces, varias otras redes, grandes y pequeñas, públicas y privadas se han unido a la Internet. Internet es con mucho la red más grande del mundo.

La Internet, sus ancestros y subredes han sido la base para el desarrollo de la mayoría de los protocolos comúnmente usados y de los principios de redes. Por ejemplo, el conjunto de protocolos TCP/IP se desarrolló como parte del proyecto ARPAnet que fue un predecesor de muchas de las subredes y también de la Internet misma.

4.8.1 Estructura de Internet

La Internet tiene una estructura de tres capas:

El backbone es el nivel más alto en la jerarquía de Internet. Es el nivel que mantiene a toda la Internet unida. Consiste de redes como NSFnet y EBONE. El backbone llevará

[12] Medio de acceso. El recurso físico a través del cual los datos o voz se mueven para alcanzar su destino (cable, fibra óptica, aire, etc.)

tráfico y hará el ruteo[13] para los niveles intermedios de red. Debido a este alto volumen de tráfico, las redes backbone tienen un ancho de banda muy grande.

Las redes de nivel medio también conocidas como redes regionales o de tránsito se encuentran bajo el backbone. Éstas llevan datos y hacen el ruteo para las redes de bajo nivel y para sus propios hosts.

Una red de nivel medio debe tener rutas al menos a otras dos redes. Algunos ejemplos de redes de tránsito son: NEARNET, PSINet y SURANET.

Una red de nivel medio a veces es conocida como una red costilla porque está unida al backbone (columna vertebral).

Las redes fragmento son básicamente redes locales o de área metropolitana. Éstas llevan paquetes sólo entre hosts, pero no entre redes. Este es el nivel en el cual la mayoría de los usuarios se comunican.

Cada computadora dentro de Internet tiene asociada una dirección única, formada por una secuencia de 4 números, de 0 a 255 cada uno, separados por puntos, a través de los cuales es posible acceder a él. Sería el equivalente al número de teléfono, utilizando el símil de las comunicaciones telefónicas. Según esto, una dirección IP es un número de 4 bytes, con 2^{32} (4,294,967,296) posibles combinaciones diferentes Además, y para simplificar el sistema de direcciones numéricas, las computadoras pueden tener asignados uno o varios nombres de dominio, que es un nombre descriptivo que permite hacer referencia al equipo y que se forma uniendo el nombre de la computadora con el nombre de la organización en que se encuentra. Los nombres de las organizaciones suelen incluir información sobre el país en el que se encuentran, o bien si se tratan de un organismo educativo (edu), militar (mil), del gobierno (gov o gob) o comercial (com).

4.8.2 TCP/IP

Un número creciente de personas está usando la Internet y muchas por primera vez están utilizando las herramientas que alguna vez estuvieron disponibles a un número limitado de sistemas de computación. Un signo de este crecimiento ha sido el número significativo de libros, artículos y cursos sobre TCP/IP que ahora están disponibles.

TCP/IP es un conjunto de varios protocolos de red desarrollados para usarse sobre Internet. Este conjunto ha probado ser muy popular y es usado por la mayoría de las implementaciones de UNIX y Windows así como otras plataformas. La única competencia real para TCP/IP son los protocolos desarrollados por el modelo de referencia OSI.

[13] Ruteo. Proceso para determinar la ruta entre el emisor y receptor de un paquete de datos

Los principales protocolos del conjunto son:

- SMTP (Simple Mail Transfer Protocol). Provee un servicio de correo electrónico simple. SMTP usa el protocolo TCP para enviar y recibir mensajes.

- FTP (File Transfer Protocol). Permite a los usuarios transferir archivos de una máquina a otra. FTP también usa los servicios del protocolo TCP en la capa de transporte para mover los archivos.

- Telnet provee capacidades de emulación de terminal y permite a los usuarios entrar a una red remota desde sus computadoras.

- SNMP (Simple Network Management Protocol). Es usado para controlar los servicios de administración de red y transferir datos relacionados con la administración.

- TCP (Transmission Control Protocol). Provee servicios de la capa de transporte orientados a conexión y a flujo. TCP usa IP para enviar sus paquetes.

- UDP (User Datagram Protocol). Provee servicios de la capa de transporte orientados a la no-conexión. UDP también usa IP para enviar sus paquetes.

- IP (Internet Protocol). Provee ruteo y servicios de envío sin conexión en la capa de red. IP utiliza switcheo de paquetes y hace el mejor esfuerzo por entregarlos, es decir, no garantiza la entrega.

Aunque no está directamente relacionada con la Internet para propósitos de operación, la asignación de nombres de dominio para Internet es un tema de controversia y actividad. Los hosts de Internet usan una estructura jerárquica de nombres en forma de dominio, subdominio y nombre de host de arriba hacia abajo. El espacio de direcciones IP ha sido manejado por la Internet Assigned Numbers Authority (IANA) y desde abril de 1998 el Internet Network Information Center (InterNIC) tuvo autoridad sobre estos nombres alrededor de todo el mundo. InterNIC también es responsable de la coordinación y manejo del sistema de nombres de dominio (DNS), la base de datos distribuida que reconcilia los nombres de hosts y las direcciones IP en la Internet.

Un cuerpo reciente que maneja los registros es el Internet Corporation for Assigned Names and Numbers (ICANN). Formado en octubre de 1998, ICANN es la organización designada por la National Telecommunications and Information Administration (NTIA) para administrar el DNS

La estructura de los nombres de dominio se entiende mejor si se lee de derecha a izquierda. Los dominios genéricos de más alto nivel incluyen:

.com Organizaciones comerciales
.edu Instituciones Educativas
.net Proveedores de red
.org Organizaciones no lucrativas
.int Organizaciones establecidas por tratos internacionales
.gov Agencias del gobierno federal de los Estados Unidos de América
.mil Milicia de E.U.A.

Una guía para seleccionar los nombres de host es tema del RFC 1178.

Otros dominios utilizan los códigos de país de dos letras definidos en el estándar ISO 3166. Por ejemplo, www.unam.mx es el nombre de un servidor en México.

Otros códigos de dominio son ca (Canadá), de (Alemania), es (España), fr (Francia), il (Israel), ie (Irlanda), jp (Japón), etc. Es importante entender que no existe necesariamente una relación entre el código de un país y su posición geográfica real.

Cada país puede organizar los subdominios en cualquier forma que desee. Muchos países usan un subdominio similar a .com.mx o .edu.mx que son los sufijos para instituciones comerciales y educativas en México.

El esquema de asignación y manejo ha trabajado bien por varios años, pero las presiones del incremento en la actividad comercial, el tamaño de la red y el uso internacional han causado controversia acerca de cómo los nombres pueden asignarse sin violar marcas y reclamos conflictivos de un nombre.

4.8.3 Direcciones IP

Las direcciones IP son números de 32 bits. Se escriben como una secuencia de cuatro números representando el valor decimal de cada uno de los bytes. Los valores están separados por puntos. Un ejemplo de una dirección IP es 192.168.1.6. (Tabla 4.1)

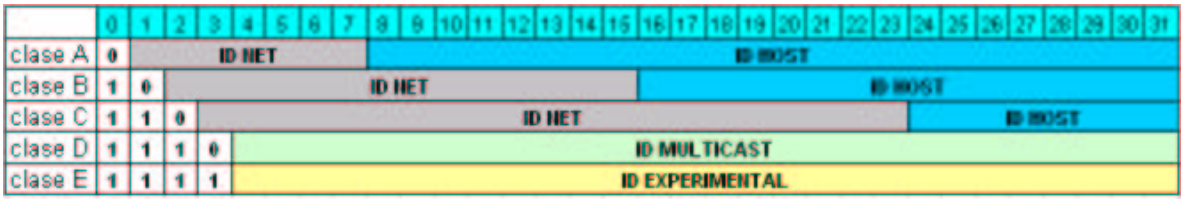

Tabla 4.1

Las direcciones IP son jerárquicas por razones de ruteo y son divididas en dos campos. El identificador de red (ID NET) identifica la subred TCP/IP conectada a la Internet. El ID NET es usado para ruteo de alto nivel entre redes, como se maneja el código

de país, ciudad y área en las llamadas telefónicas. El ID HOST indica el host específico dentro de una subred.

Para permitir redes de diferentes tamaños, IP define varias clases de direcciones. Las clases A, B y C se usan para el direccionamiento de hosts y la única diferencia entre las clases es la longitud del campo ID NET.

Una dirección de clase A tiene un ID NET de 7 bits y un ID HOST de 24 bits. Las direcciones clase A se usan para redes muy grandes y pueden manejar hasta 16,777,216 (2^{24}) hosts por red. El primer dígito de una dirección clase A debe ser un número entre 1 y 126. Relativamente pocas direcciones clase A han sido asignadas, por ejemplo 9.0.0.0 de IBM.

Una dirección de clase B tiene un ID NET de 14 bits y un ID HOST de 16 bits. Las direcciones clase B se usan para redes de tamaño medio y pueden manejar hasta 65,536 (2^{16}) hosts por red. El primer dígito de una dirección clase B será un número entre 128 y 191. El espacio de direcciones de clase B ha sido utilizado por mucho tiempo y es muy difícil obtener una nueva dirección clase B. Por ejemplo 152.163.0.0 (America Online).

Una dirección de clase C tiene un ID NET de 21 bits y un ID HOST de 8 bits. Estas direcciones se usan en redes pequeñas y pueden manejar sólo 254 (2^8-2) hosts por red. El primer dígito de una dirección clase C será un número entre 192 y 223. La mayoría de las direcciones asignadas a redes son clase C (o subclase C), por ejemplo 208.162.102.0.

Las clases D y E se usan sólo para funciones especiales y no son asignadas a hosts individuales. Las direcciones clase D pueden empezar con un valor entre 224 y 239 y son usadas para multicasting IP. Las direcciones clase E comienzan con un valor entre 240 y 255 y están reservadas para uso experimental.

Varios valores de direcciones están reservados y tienen un significado especial. Un ID HOST de 0 es un valor reservado que identifica un contenedor que se refiere a una subred entera. La dirección 208.162.106.0 se refiere a la dirección clase C con un ID NET de 208.162.106. Un ID HOST de puros 1’s es una dirección broadcast y se refiere a todos los hosts en una red. Un ID NET con valor de 127 es usado para realizar pruebas locales y la dirección específica 127.0.0.1 se refiere al localhost.

Varios ID NET han sido reservados en el RFC 1918 para direcciones de red privadas y los paquetes no serán ruteados sobre la Internet a estas redes. Los ID NET reservados son las direcciones clase A 10.0.0.0, las 16 direcciones clase B 172.16.0.0-172.31.0.0 y las 256 direcciones clase C 192.168.0.0-192.168.255.0.

Una herramienta de direccionamiento adicional es la máscara de subred. Las máscaras de subred se usan para indicar la porción de la dirección que identifica la red (o subred) para propósitos de ruteo. La máscara de subred está escrita en notación decimal de puntos y el número de 1’s indica los bits significativos del ID NET (Tabla 4.2).

Clase	Máscara de subred	Número de bits
A	255.0.0.0	8
B	255.255.0.0	16
C	255.255.255.0	24

Tabla 4.2

Dependiendo del contexto y literatura, las máscaras de subred pueden ser escritas en notación decimal con puntos o sólo como un número representando el número de bits significativos para el ID NET. Por lo tanto, 208.162.106.17 255.255.255.0 y 208.162.106.17/24 se refieren a un ID NET de una dirección clase C 208.162.106.

4.8.4 El sistema de nombres de dominio

Mientras las direcciones IP son de 32 bits de longitud, la mayoría de los usuarios no memorizan las direcciones numéricas de los hosts a los cuales se conectan. En lugar de esto, la gente se siente mejor utilizando nombres de host. La mayoría de los hosts tienen una dirección IP numérica y un nombre. Esto es conveniente para las personas, pero el nombre debe ser traducido a una dirección numérica para propósitos de ruteo.

Para manejar el gran número de nombres nuevos sobre la red, el Domain Name System (DNS) fue creado. El DNS es una base de datos distribuida que contiene información del nombre de host y dirección IP de todos los dominios de la Internet. Hay un solo servidor de nombres con autoridad para cada dominio que contiene toda la información relacionada con el DNS. Cada dominio tiene también al menos un servidor de nombres secundario que también contiene una copia de esta información.

Trece servidores raíz alrededor del mundo (la mayoría en EUA) mantienen una lista de todos estos servidores de nombres.

Cuando un host en la Internet necesita obtener la dirección IP de un host basándose en su nombre, una petición DNS se hace al servidor de nombres local. El servidor local puede ser capaz de responder a la petición que fue configurada o guardada en él. Si la información necesaria no está disponible, el servidor de nombres local envía la petición a uno de los servidores raíz.

Capítulo 5

Definición formal de la gramática de un lenguaje

Este capítulo se incluye porque todas las acciones que se generan en el protocolo IRC se manejan por mensajes. Estos mensajes que no son nada más que simples cadenas de texto contienen la información de los comandos que se han de ejecutar en el servidor o en los clientes y es necesario descifrar si estos mensajes están bien construidos. Si son válidos hay que realizar alguna operación en consecuencia. Para reconocer estos comandos es necesario realizar un análisis léxico y sintáctico de estas cadenas, es decir, realizar una búsqueda de elementos como pueden ser palabras clave e identificar los parámetros que vienen contenidos en el mensaje mismo.

Comenzaremos con la teoría básica de manejo de gramáticas y lenguajes y veremos lo referente a scanners y parsers, que son programas que realizan el análisis léxico y sintáctico respectivamente.

5.1 Gramáticas y lenguajes

Un símbolo es cualquier carácter (letras, números, operadores aritméticos: +,-,*,/). Es un objeto que sirve para representar otros entes. Por ejemplo, las letras son símbolos que nos sirven para expresar sonidos; los numerales son símbolos que nos sirven para representar cantidades.

Un conjunto es una colección de símbolos. Las operaciones que pueden realizarse sobre los conjuntos, son las siguientes:

Sean A y B conjuntos:

- Unión:
  $A \cup B = \{x \mid x \in A \text{ o } x \in B\}$

- Intersección
  $A \cap B = \{x \mid x \in A \text{ y } x \in B\}$

- Resta
  $A - B = \{x \mid x \in A \text{ y } x \notin B\}$

- Multiplicación
  $A \times B = \{(a,b) \mid a \in A \text{ y } b \in B\}$

- Complemento:
  Sean U el conjunto de los números naturales y A un subconjunto de U:

  $U=\{0,1,2,3,4,5,6,7,8,9\}$ $A=\{0,2,5\}$

  Entonces, el complemento de A se denota como A' y estaría definido por:

  $A'=\{1,3,4,6,7,8,9\}$

Entonces, el complemento de A se denota como A' y estaría definido por:

A'={1,3,4,6,7,8,9}

- Potencia

$A^n = A \times A \times \ldots n$

Donde siempre:
$A^0 = \varnothing$ (conjunto vacío)

Una cadena o palabra es una secuencia finita de símbolos yuxtapuestos. La longitud de una palabra *p* se denota como $|p|$. La palabra vacía se denota como ε, que es la cadena que no tiene ningún símbolo, de manera que

$$|\varepsilon| = 0.$$

Un prefijo es cualquier número de símbolos iniciales de una cadena. Por ejemplo, los prefijos posibles de la palabra "uno" serían:

ε, u, un y uno.

Un prefijo propio es cualquier prefijo de la cadena, a excepción de la cadena misma.

Un sufijo es cualquier número de símbolos terminales de una cadena, y, al igual que los prefijos propios, un sufijo propio es cualquier sufijo, excepto la cadena misma. La concatenación es la operación de escribir cadenas una después de otra sin espacios intermedios. La cadena vacía es la cadena de identidad para la operación de concatenación. Esto es, sea una cadena *w*:

$$\varepsilon w = w\varepsilon = w$$

La inversa de una cadena w (denotada como w^R) se obtiene al revertir la secuencia de símbolos en la cadena. Por ejemplo, si $w = abc$ entonces $w^R = cba$

Un conjunto no vacío y finito de símbolos se conoce como alfabeto, denotado generalmente por Σ; por ejemplo, el alfabeto del abecedario estaría denotado por :

$$\Sigma = \{a,b,...,z,A,B,...,Z\}$$

Un lenguaje es un conjunto de cadenas de símbolos (palabras) de algún alfabeto. Por convención, el conjunto vacío (denotado por ϕ) cuyo contenido es nulo, y el conjunto cuyo único elemento es la cadena vacía (denotado como {ε}) se consideran lenguajes.

El lenguaje de todas las cadenas de un alfabeto Σ se denota por Σ*. Por ejemplo:

Si $\Sigma = \{a\}$ entonces $\Sigma^* = \{\varepsilon,a,aa,aaa,...\}$

Σ^* denota el conjunto de todas las secuencias de cadenas que están compuestas por cero o más símbolos de Σ.

Σ^+ denota el conjunto de todas las secuencias de cadenas compuestas por uno o más símbolos de Σ. esto es:

$$\Sigma^+ = \Sigma^* - \{\varepsilon\}$$

Por lo tanto, un lenguaje es un subconjunto de Σ^*.

Cualquier cadena que pertenezca a un lenguaje se dice que es una palabra o una oración de dicho lenguaje.

Los lenguajes son conjuntos. De ahí que cualquier operación que se pueda realizar en los conjuntos se pueden realizar sobre los lenguajes.

Si L, L1,L2 son lenguajes, entonces:

- $L1 \cup L2$ es un lenguaje
- $L1 \cap L2$ es un lenguaje
- L1 - L2 es un lenguaje
- -L = Σ^*- L, el complemento de L es un lenguaje.

5.1.1 Gramática

Una gramática es un conjunto finito de reglas que define un lenguaje.

Una gramática G está definida así:

G = {NT, T, S, P}
NT: Elementos no terminales, que son el conjunto finito de símbolos o variables
T: Conjunto finito de símbolos terminales
S: $S \in$ NT. Es un elemento distinguido de NT, llamado símbolo inicial de la gramática
P: Conjunto de reglas de producción

Una regla de producción tiene la forma $X \rightarrow Y$ donde:

$X \in (NT \cup T)^+$ y
$Y \in (NT \cup T)^*$

Esto es, “X” es un miembro del conjunto de cadenas compuestas de cualquier mezcla de variables y símbolos terminales, pero “X” no puede ser la cadena vacía y “Y” es un miembro del conjunto de cadenas compuesta por cualquier mezcla de variables y símbolos terminales y en este caso “Y” sí puede ser la cadena vacía.

Las reglas de producción pueden ser utilizadas para definir las cadenas que pertenecen a un lenguaje.

Si el lenguaje L está definido por una gramática G = {NT, T, S, P} se puede encontrar una cadena que pertenezca a este lenguaje de la siguiente manera:

1. Empezar con una cadena *w* que consista únicamente del símbolo inicial S
2. Encontrar una subcadena *x* de *w* que sea el lado izquierdo de una producción *p* de P
3. Reemplazar la subcadena *x* de *w* en el lado derecho de la producción *p*
4. Repetir los pasos 2 y 3 hasta que la cadena consista enteramente de símbolos de T (es decir, que no contenga variables)
5. La cadena final pertenece al lenguaje L

Cada aplicación del paso 3 arriba mencionado, se denomina paso de derivación. Supongamos que *w* es una cadena que puede escribirse como "*uxv*" donde:

u y *v* son elementos de $(NT \cup T)^*$
x es un elemento de $(NT \cup T)^+$
existe una producción $x \rightarrow y$

entonces se puede escribir:

$$uxv \Rightarrow uyv$$

y se dice que *uxv* deriva directamente *uyv*

Tipos de gramáticas

De acuerdo a Noam Chomsky, una gramática y el lenguaje correspondiente son independientes del contexto si y sólo si pueden definirse con un conjunto de producciones independientes del contexto. Las gramáticas independientes del contexto son muy importantes en la teoría de los lenguajes de programación ya que los lenguajes que definen, tienen en general una estructura muy sencilla. Las técnicas de análisis sintáctico suelen basarse en gramáticas independientes del contexto.

Una gramática independiente del contexto es no ambigua si y sólo si hay una sola derivación por la derecha (o por la izquierda) y, por ende, un solo árbol de análisis sintáctico (es decir, la secuencia de derivaciones representada como estructura de árbol) para cada frase que pueda derivarse con las producciones de la gramática. En caso contrario se llama ambigua.

Se dice que una gramática G = {NT, T, P, S} es lineal izquierda si cada producción P tiene la forma:

$$A \rightarrow Ba \qquad \text{o} \qquad A \rightarrow a$$

donde A y B $\in$ NT y *a* está en T^*.

La gramática G = {NT, T, P, S} es lineal derecha si cada producción tiene la forma:

$$A \rightarrow aB \qquad o \qquad A \rightarrow a$$

donde A y B $\in$ NT y *a* está en T^*.

Las gramáticas lineales derechas G = {NT, T, P, S} también se conocen como regulares o de estado finito. En general, la sintaxis de un lenguaje de programación no puede expresarse con una gramática regular, pero ciertos elementos del lenguaje, como nombres y números, se describen mejor con gramáticas regulares.
Un símbolo no terminal X $\in$ NT en una gramática independiente del contexto G={NT,T, P, S} es recursivo si:

$$X^* \rightarrow \alpha X \beta$$

para algunas α y β. Se dice que X es recursivo izquierdo si $\alpha = \varepsilon$. De igual modo, se dice que es recursivo derecho si $\beta = \varepsilon$. Una gramática con al menos un X no terminal recursivo izquierdo (o derecho) se conoce como gramática recursiva izquierda (o derecha). Si todos los símbolos no terminales (con la posible excepción del símbolo inicial) son recursivos, decimos que la gramática es recursiva. Algunos de los algoritmos de análisis sintáctico no funcionan con gramáticas recursivas izquierdas, porque estas gramáticas presentan el riesgo de lazos infinitos.

Las gramáticas se clasifican de acuerdo con su complejidad. Esta clasificación, conocida como jerarquía de Chomsky, se establece aumentando las restricciones sobre la forma de las producciones.

La jerarquía de Chomsky es una jerarquía contenedora de clases de gramáticas formales que genera lenguajes formales y fue descrita en 1956. Dicha jerarquía consiste de cuatro niveles, que son los siguientes:

- Gramáticas Tipo 0 (Gramáticas sin restricciones). Incluye todas las gramáticas formales. Las gramáticas sin restricciones son aquellas que no tienen restricciones para el lado izquierdo ni para el lado derecho de las producciones. Una gramática formal es aquella que permite generar cadenas comenzando con un símbolo de inicio especial y la aplicación de reglas que indican cómo ciertas combinaciones de símbolos pueden ser reemplazados por otras combinaciones de símbolos. Una gramática formal consiste de un conjunto finito de símbolos terminales (por ejemplo, las letras de las palabras en el lenguaje normal), un conjunto finito de símbolos no terminales, un conjunto de reglas de producción formados por una parte izquierda y derecha consistente en una palabra de dichos símbolos y un símbolo inicial. Una regla puede ser aplicada a una cadena al reemplazar el lado izquierdo por el lado derecho. Una gramática de este tipo define el lenguaje formal de todas las cadenas que consisten únicamente de símbolos terminales que pueden ser alcanzados por una derivación desde el símbolo inicial. Los elementos no terminales generalmente son representados con letras mayúsculas, los terminales

con letras minúsculas y el símbolo inicial con una "S". La siguiente gramática muestra dicha notación:

NT = {S, A, B}
T = {*a*, *b*}

Reglas de producción:
$S \rightarrow ABS$
$S \rightarrow \varepsilon$
$BA \rightarrow AB$
$BS \rightarrow b$
$Bb \rightarrow bb$
$Ab \rightarrow ab$
$Aa \rightarrow aa$

Esta gramática define el lenguaje de todas las palabras de la forma $a^n b^n$ (esto es, *n* copias de *a* seguidas por *n* copias de *b*).

- Gramáticas Tipo 1 (Gramáticas sensibles al contexto). Una gramática sensible al contexto es una gramática formal de la forma G = {NT, T, P, S} tales que todas las reglas en P son de la forma:

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

donde A es un símbolo no terminal:

$$A \in NT$$

y α,β y γ son cadenas formadas por terminales y no terminales, es decir:

$$\alpha, \beta \in (NT \cup T)^*$$

por lo que las cadenas α y β pueden ser cadenas vacías, pero γ no debe serlo:

$$\gamma \in (NT \cup T)^+$$

Esta gramática también puede tener una producción de la forma:

$$S \rightarrow \varepsilon$$

y si este es el caso, entonces no debe tener S en el lado derecho de una regla de producción.

El adjetivo "sensible al contexto" es explicado por α y β que forman el contexto de A y determinan si A puede ser reemplazada con γ o no. Esto difiere de una gramática libre de contexto donde el contexto de un símbolo no terminal no se toma en consideración.

Un lenguaje que puede ser descrito por una gramática sensible al contexto es llamado lenguaje sensible al contexto.

- Gramáticas Tipo 2 (Gramáticas libres de contexto). Una gramática libre de contexto es una gramática formal en la que cada regla de producción es de la forma:

  $V \rightarrow w$

En la cual V es un símbolo no terminal y *w* es una cadena formada con elementos terminales y/o no terminales. El término "libre de contexto" viene de la característica de la variable V puede ser siempre reemplazada por *w*, sin importar el contexto en el que ocurra. Un lenguaje formal es libre de contexto si existe una gramática libre de contexto que lo genere.

Las gramáticas libres de contexto son importantes debido a que son lo suficientemente poderosas como para describir la sintaxis de los lenguajes de programación; de hecho, casi todos los lenguajes de programación son definidos mediante una gramática libre de contexto. Por otro lado, las gramáticas libres de contexto son lo suficientemente simples para permitir la construcción de eficientes algoritmos de parsers para una cadena dada, determinando si puede y cómo puede ser generada a partir de la gramática. Una gramática que genera expresiones aritméticas sencillas y correctas utilizando como variables *x*, *y*, *z* es la siguiente:

$S \rightarrow T + S$
$S \rightarrow T - S$
$S \rightarrow T$
$T \rightarrow T * T$
$T \rightarrow T / T$
$T \rightarrow (S)$
$T \rightarrow x$
$T \rightarrow y$
$T \rightarrow z$

Esta gramática genera, por ejemplo, la cadena: (*x*+*y*)**x*-*z***y*/(*x*+*x*)

- Gramáticas Tipo 3 (Gramáticas regulares).

Es una gramática formal de la forma G= {NT, T, P, S} tal que todas las reglas de producción en P tengan una de las siguientes formas:

1. $A \rightarrow a$, donde $A \in NT$ y $a \in T$
2. $A \rightarrow aB$, donde $A,B \in NT$ y $a \in T$
3. $A \rightarrow \varepsilon$, donde $A \in NT$

La segunda regla puede ser reemplazada con la regla:

$A \rightarrow Ba$

Ahora un ejemplo de una gramática regular:

Sea NT = {S,A} y T = {*a*,*b*,*c*} y P las siguientes reglas de producción:

$S \rightarrow aS$
$S \rightarrow bA$
$A \rightarrow \varepsilon$
$A \rightarrow cA$

en la cual S en el símbolo inicial. Esta gramática definiría el mismo lenguaje que la expresión regular *a*bc**.

Una expresión regular es una familia de lenguajes compactos y poderosos que describen conjuntos de cadenas. Estos lenguajes son utilizados por muchos editores de texto y utilerías para buscar cuerpos de texto para ciertos patrones y, por ejemplo, reemplazar la cadena encontrada con una cadena distinta. Estas gramáticas restringen sus reglas de producción a sólo un elemento no terminal en su lado izquierdo, posiblemente seguido por un solo elemento terminal. Es permitida la regla $S \rightarrow \in$ sólo si S no aparece en el lado derecho de alguna regla. Las gramáticas regulares describen exactamente todos los lenguajes regulares y en ese sentido son equivalentes a los autómatas de estado finito y las expresiones regulares.

En resumen, todos los lenguajes regulares son libres de contexto, todos los lenguajes libres de contexto son sensibles al contexto y todos los lenguajes sensibles al contexto son recursivamente enumerables. Mientras las gramáticas independientes del contexto definen la sintaxis de las declaraciones, las proposiciones, las expresiones, etc. (es decir, la estructura de un programa), las gramáticas regulares definen la sintaxis de los identificadores, números, cadenas y otros símbolos básicos del lenguaje; por ello, es común encontrar gramáticas independientes de contexto en el análisis sintáctico, a la vez que las gramáticas regulares se emplean como la base del análisis léxico. La Tabla 5.1 muestra el resumen de la jerarquía de las gramáticas según Chomsky.

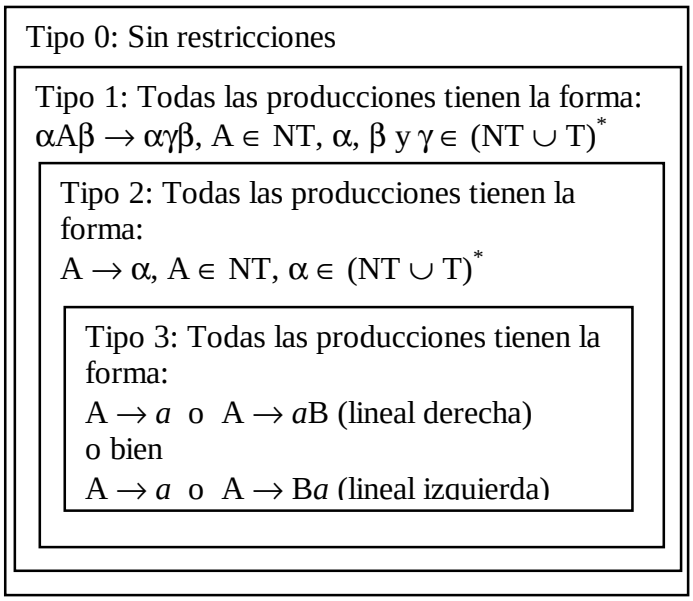

Tabla 5.1

5.2 Análisis léxico

Definición

El proceso de análisis léxico o de reconocimiento puede entenderse como la transformación de un flujo de carácteres en un flujo de símbolos "reducidos". En este caso "reducido" significa que la entrada es filtrada para eliminar aquellos elementos que sólo sirven para hacer legible el programa.

Entre las funciones características de un analizador léxico están:

- Eliminar espacios, comentarios, etc.
- Reconocer identificadores y palabras clave.
- Reconocer constantes y numerales.
- Generar un listado para el compilador.

Eliminación de separadores

Los espacios, tabuladores, comentarios y saltos de línea se denominan separadores y pueden ser muy numerosos en un código fuente, pero no proporcionan información para la traducción, por lo tanto deben ser eliminados del código entregado al analizador sintáctico.

Reconocimiento de operadores y nombres

Es fácil reconocer símbolos de operadores como +, -, *, /, porque constan de un solo carácter. Sin embargo, hay otros operadores como <, <=, < >,:= que comienzan con el mismo carácter y tienen significado diferente. En estos casos, el analizador léxico tiene que

hacer examinar el carácter siguiente por anticipado o pre-análisis, para determinar el operador correcto. Esto también sucede cuando se analiza un número, un identificador o una palabra clave.

El reconocimiento de palabras clave (o reservadas) e identificadores en un flujo de carácteres, se basa en un autómata finito, como por ejemplo el de la Figura 5.1 que reconoce identificadores.

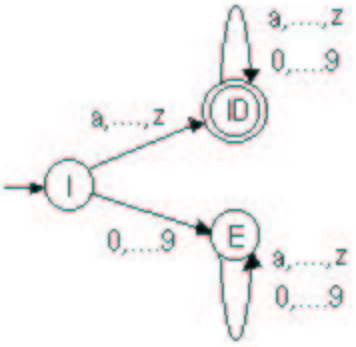

Figura 5.1

La tarea más difícil para el analizador léxico es distinguir entre identificadores y palabras clave, lo cual puede hacerse con tablas de símbolos o con una estructura de datos adicional que contenga todas las palabras reservadas.

Números. El analizador léxico debe transformar un numeral en el símbolo del numeral y su valor.

Tablas de símbolos. Es una estructura de información para manejar los nombres (identificadores y palabras reservadas) del código fuente. Se usa durante la comprobación semántica o dependiente del contexto, además del proceso de generación de código. En general, una tabla de símbolos consta de nombres y atributos.

La información almacenada en una tabla de símbolos varía de un compilador a otro y de un lenguaje a otro; es decir la información que se incluirá en ella depende del lenguaje y del diseñador del compilador. Pero en general, los atributos pueden ser:

- tipo o valor de un nombre (constantes) o ambos;
- la dimensión o el número de parámetros de un procedimiento;
- un puntero al lugar donde se declara y se hace referencia a un nombre;
- algún tipo de dirección, por lo regular un desplazamiento;
- información del alcance en caso de un lenguaje orientado a bloques.

5.3 Análisis sintáctico

Definición:

El análisis sintáctico es el proceso de determinar si una cadena de componentes léxicos puede ser generada por una gramática.

En el estudio de este problema, es útil pensar en construir un árbol de derivación sintáctico, aunque, de hecho, un compilador no lo construya. Sin embargo, un analizador sintáctico sí debe ser capaz de construirlo, pues de otro modo, no se puede garantizar que la traducción sea correcta.

Para cualquier gramática independiente de contexto (GIC) hay un analizador sintáctico que toma como máximo un tiempo de $O(n^3)$, pero esto es demasiado caro, ya que en general, dado un lenguaje de programación, se puede construir una gramática que se pueda analizar con algoritmos lineales. Los analizadores sintácticos de lenguajes de programación suelen hacer un examen simple de izquierda a derecha, viendo un componente léxico de la entrada, a la vez.

La mayoría de los métodos de análisis sintáctico están comprendidos en dos clases, llamadas método descendente y ascendente, dependiendo del orden en que se construyen los nodos del árbol de derivación. En el primero, se construyen comenzando de la raíz, paso a paso, avanzando hacia las hojas y en cada paso se representa una derivación de la palabra por la izquierda. En el segundo, la construcción se inicia en las hojas y avanza hacia la raíz siguiendo el camino contrario en una derivación por la derecha de la palabra.

Ejemplo
S $\rightarrow$T / *ab* / *c*T*c*S
T $\rightarrow$*d* / *e* / *f*

Derivación Descendente
S $\Rightarrow$ *c*T*c*S $\Rightarrow$ *cdc*S $\Rightarrow$ *cdcc*T*c*S $\Rightarrow$ *cdccec*S $\Rightarrow$ *cdccec*T $\Rightarrow$ *cdccecf*

Derivación Ascendente
S $\Rightarrow$ *c*T*c*S $\Rightarrow$ *c*T*cc*T*c*S $\Rightarrow$ *c*T*cc*T*c*T $\Rightarrow$ *c*T*cc*T*cf* $\Rightarrow$ *c*T*ccecf* $\Rightarrow$ *cdccecf*

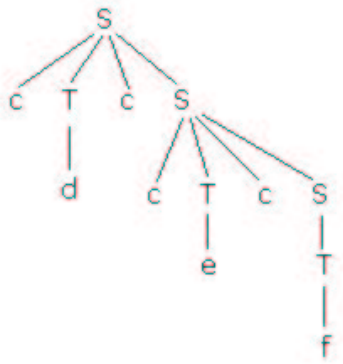

Figura 5.2

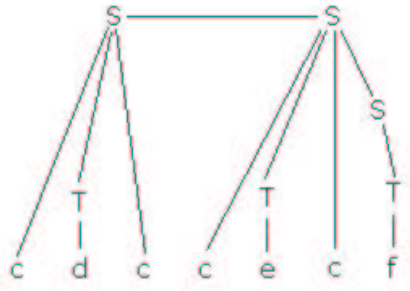

Figura 5.3

Los métodos ascendente y descendente más eficientes trabajan sólo con subclases de gramáticas, pero varias de estas subclases, como las gramáticas LL (left to left: leen en la cadena de izquierda a derecha, derivación por la izquierda) y LR (left to right: leen de izquierda a derecha, derivación por la derecha), son lo suficientemente expresivas para describir la mayoría de las construcciones sintácticas de los lenguajes de programación.

Definición

Se dice que una sentencia w perteneciente a L(G) para alguna gramática libre de contexto, G, se ha analizado sintácticamente cuando se conocen uno (o todos) sus árboles de derivación.

Definición

Sea G = (N, E, P, S) una GLC, con las producciones en P numeradas 1,2,...*p* ; perteneciente a (N U E)*.

Entonces:

i.- Un análisis a la izquierda de *a* es una secuencia de producciones usadas en una derivación más a la izquierda de *a* desde S.

Ejemplo: G= ({S,A,B,C}, {*a*,*b*,*c*}, P, S)

P:

1. S $\rightarrow$ *a*A
2. A $\rightarrow$ *a*B*b*C
3. B $\rightarrow$ *b*
4. C $\rightarrow$ *c*

a = *aabbc*
S $\Rightarrow$ *a*A $\Rightarrow$ *aa*B*b*C $\Rightarrow$ *aabb*C $\Rightarrow$ *aabbc*
1 2 3 4

El orden del análisis a la izquierda de *aabbc* es 1234

ii.- Un análisis a la derecha de *a* es el reverso de la secuencia de producciones usadas en una derivación más a la derecha de *a* desde S.

Ejemplo:
S $\Rightarrow$ *a*A $\Rightarrow$ *aa*B*b*C $\Rightarrow$ *aa*B*bc* $\Rightarrow$ *aabbc*
1 2 3 4

El orden del análisis a la derecha de aabbc es 3421.

5.4 Herramientas de programación de scanners y parsers

El propósito de un analizador léxico (o scanner) es particionar el texto de entrada, entregar una secuencia de comentarios y símbolos básicos. Los comentarios son secuencias de carácteres que deben ser ignorados, mientras los símbolos básicos son secuencias de carácteres que corresponden a símbolos terminales de la gramática que definen la estructura de frase de la entrada.

El análisis léxico es el primer estado en el procesamiento de un lenguaje. El flujo de carácteres que construyen un programa fuente o algún otro tipo de entrada es leído uno a la vez y agrupados en lexemas (conocidos también como "tokens") que tengan características similares, como palabras clave, identificadores, constantes y puntuación. Una vez generados dichos tokens, se pasan a un parser.

Las funciones de un scanner pueden resumirse en las siguientes:

1. Crear la tabla de símbolos.
2. Crear los tokens.
3. Calcular el valor de las constantes.
4. Reconocer comentarios y saltarlos.
5. Ignorar los espacios en blanco(es decir, que una secuencia de dichos espacios las considere como uno solo).
6. Reconocer delimitadores para los componentes léxicos.
7. Detectar el fin del archivo fuente.
8. No debe interrumpirse la ejecución de un programa a consecuencia del programa fuente.

En la tecnología de las computadoras, un parser es un programa (usualmente parte de un compilador) que recibe entradas en forma secuencial de instrucciones de programas fuentes, comandos en línea en forma interactiva o algunas otras de interfaces definidas y los separa en partes (por ejemplo, los sustantivos (objetos), los verbos (métodos) y sus atributos u opciones) que puedan ser manejadas por otro programa (por ejemplo, otros

componentes dentro de un compilador). Un parser puede también verificar que se hayan proporcionado las entradas que se necesitan.

Un parser es un algoritmo o programa para determinar la estructura sintáctica de una sentencia o cadena de símbolos en algún lenguaje. Un parser toma normalmente como entrada una secuencia de tokens que fueron generados previamente por un scanner.

Las funciones que efectúa un parser son:

1. Realizar el análisis sintáctico (agrupar los componentes léxicos del programa fuente en frases gramaticales; comúnmente se representan en un árbol).
2. Añadir y verificar la semántica (revisar los componentes en su significado, así como sus posibles variaciones) de la cadena de entrada (el token).
3. Traducir la cadena de tokens a una cadena de átomos que representa la estructura del programa.
4. Generar el código intermedio.

Capítulo 6

Análisis

6.1 Definición del proyecto

Descripción breve del proyecto

Construir un programa servidor de IRC para crear un canal más de comunicación que permita al Laboratorio de Multimedia e Internet ofrecer otra alternativa para dar asesoría a los alumnos de la Facultad de Ingeniería y que sea utilizado por los miembros del laboratorio mismo como una herramienta más en el desarrollo de los proyectos en los que trabajan.

Objetivo:

- Desarrollar un programa servidor basado en el protocolo Internet Relay Chat (IRC) para establecer comunicación escrita en tiempo real a través de la Internet.

6.2 Descripción del proyecto

Se trata de crear un programa servidor que se apegue al protocolo IRC. El programa se correrá en un servidor propiedad del Laboratorio de Multimedia e Internet y tendrá como propósito ofrecer un servicio que permita que cualquier programa cliente que cumpla con esta especificación se conecte. Esto hará que los clientes válidos puedan utilizar el servicio sin importar el tipo de sistema operativo en el que funcionen o el tipo de computadora. Consideraremos a un cliente como válido cuando se apegue al protocolo.

El sistema deberá funcionar en tiempo real, es decir, deberá generar respuestas en un tiempo razonablemente corto de manera que no impida el flujo continuo de la conversación. Si al estar en una conversación los usuarios deben esperar uno o más minutos entre cada uno de los mensajes, entonces el sistema no servirá, porque no será cómodo ni útil conversar de esta manera debido a que la información fluye muy lentamente.

El sistema deberá permitir al personal del laboratorio darse cuenta de que alguna persona necesita asesoría. Esto se hará por medio de una señal sonora o visual para que no sea necesario que alguien esté atento al programa durante todo el tiempo.

6.3 Descripción del protocolo IRC

Un servidor IRC puede manejar varios canales de comunicación, de los cuales un cliente puede elegir uno o varios para participar, o crear uno propio con un tema específico.

Para participar en una conversación de IRC los clientes se comunican con el servidor y éste se encarga de coordinar los mensajes entre los clientes.

El protocolo IRC ha sido diseñado para usarse en conversaciones basadas en texto y ha sido desarrollado en sistemas que usan el protocolo de red TCP/IP, aunque no es un requerimiento que opere sólo en esta esfera.

El servidor forma la columna vertebral del IRC, dando un punto al cual los clientes se pueden conectar para hablar entre ellos y un punto de conexión para otros servidores para formar una red IRC. La única configuración de red permitida para los servidores es la de un árbol extendido, donde cada servidor actúa como nodo central para el resto de la red que puede ver, es decir, no se permiten ciclos en las conexiones. En nuestro caso, sólo existirá un servidor y, por lo tanto, no se implementarán las funciones que permiten la comunicación con otros servidores.

Un cliente es cualquier elemento conectado al servidor que no es otro servidor. Cada cliente se distingue por un "nickname" o apodo con una longitud máxima de nueve carácteres. Este nickname es el identificador que cada cliente escoge y es el nombre con el cual será conocido por todos los demás. Un cliente puede ser un usuario o un servicio. El servicio es un tipo de cliente que se conecta para dar o pedir información, normalmente un programa autónomo. Interactúa directamente con el servidor. Un usuario es alguien que se conecta para tener una conversación.

El servidor recibe una serie de parámetros en cada mensaje y con base en esto debe generar la respuesta adecuada o un código de error, según sea el caso, además de ejecutar el proceso correspondiente a cada comando que se le envíe.

Existen operadores del sistema y operadores de canal. Los operadores del sistema se encargan de administrar y corregir errores que pudieran presentarse. Los operadores de canal se encargan de moderar los mensajes entre los participantes y de ser necesario pueden expulsar a un participante que no se esté comportando bien.

Las conversaciones se organizan generalmente por medio de canales, los cuales podemos entender como grupos lógicos de usuarios. Cuando se envía un mensaje a un canal, éste es reenviado a todos los clientes que pertenecen a ese canal. Los canales tienen ciertas características que indican cómo será su funcionamiento. Se pueden crear canales a los que nadie pueda entrar a menos que sean invitados por alguien que pertenezca al canal o se pueden crear canales que sean invisibles, es decir, que no aparezcan en la lista de canales que mantiene el servidor. Los canales tienen un nombre y pueden tener un tema o tópico de conversación específico.

6.4 Alcances del proyecto

Todas las características que debe reunir un sistema que cumple con el protocolo IRC están definidas en los documentos RFC's 1459, 2810, 2811, 2812 y 2813, los cuales contienen la información necesaria para comprender e implementar el protocolo, ya sea del lado del servidor o del cliente. Es de estos documentos de donde obtendremos los requerimientos del protocolo IRC ignorando los aspectos relacionados con la comunicación entre servidores, ya que por ahora sólo se implementará una red con un sólo servidor.

El documento más importante es el RFC 2812, ya que contiene la mayor parte de información que se refiere al formato de los comandos, sus parámetros, las respuestas y todo lo que tiene que ver con la comunicación entre cliente y servidor.

6.5 Resultados esperados

Se espera un sistema que actúe como servidor y que se apegue al protocolo IRC para permitir la comunicación entre los integrantes del Laboratorio de Multimedia e Internet de la Facultad de Ingeniería e impartir asesoría en tiempo real.

6.6 Justificación y beneficios

Este sistema se desarrollará para resolver el problema de quienes no pueden acudir a las asesorías y necesitan utilizar la Internet para recibirla, ya que la comunicación por medio de correo electrónico no permite una interacción en tiempo real y en ocasiones es necesario intercambiar puntos de vista o hacer preguntas que surgen en el momento.

Otro problema que se atacará con la construcción de este sistema es el de la imposibilidad de algunos integrantes que se encuentran colaborando en los proyectos del laboratorio de permanecer en el mismo horario que sus compañeros, así que será muy fácil que ellos se comuniquen al laboratorio utilizando un cliente de IRC.

El estudio y manejo de protocolos de comunicación a través de la Internet es un tema de interés para el Laboratorio de Multimedia e Internet y se espera que sirva como una base para introducirnos más en el manejo de la red al permitirnos aprender más sobre este tema. Se eligió el protocolo IRC porque tiene un uso muy extendido, una estructura bien definida y sobre todo porque nos permitirá crear un sistema que sea compatible en cierto grado con otros programas cliente que utilicen el mismo protocolo. Esto es necesario porque la información se va a compartir en grupo y no sólo entre dos personas. Una de las ventajas de apegarnos a este protocolo es que existen muchos programas clientes compatibles para distintas plataformas.

6.7 Identificación de necesidades

El Laboratorio de Multimedia e Internet proporciona asesoría a los alumnos de la Facultad de Ingeniería sobre varios temas de cómputo y desarrolla proyectos que tienen que ver con el manejo de la Internet y medios de comunicación como son el sonido, video, animaciones y por supuesto texto e imágenes.

Debido a problemas de tiempo y de distribución del trabajo, no todas las personas que lo desean pueden estar en el laboratorio cuando quieren que se les ayude por medio de una asesoría, o los integrantes del laboratorio no siempre se encuentran cuando quieren preguntar o avisar algo a sus compañeros de equipo. Es por eso que se pensó en la posibilidad de hacerlo a través de la Internet, por medio de algún programa que permitiera la comunicación.

Se necesita una forma de comunicación que sea rápida y sencilla de usar, que nos permita interactuar en tiempo real, es decir, que permita un flujo de comunicación que no se detenga o se retrase por más de unos cuantos segundos. Debe permitir que la

comunicación se establezca entre dos o más personas al mismo tiempo y debe permitir que se puedan mantener varias comunicaciones separadas, que no tengan nada que ver una con otra, como conversaciones totalmente separadas. Debe permitir además, saber de alguna manera, dentro del laboratorio, que alguien necesita comunicarse o desea asesoría para que pueda ser atendido oportunamente, sin que una persona deba estar frente a una computadora todo el tiempo cuidando que alguien entre al sistema y solicite asesoría.

6.8 Revisión de la situación actual

Actualmente no se tiene manera eficiente de dar las asesorías por medio de la Internet, solamente se logra en el mismo laboratorio. La comunicación de los integrantes del laboratorio que se encuentran en otro sitio se realiza por medio de correo electrónico. Se cuenta con una lista de correo a la cual se envía un mensaje y después éste se distribuye a todos los integrantes de la lista. Cuando se quiere preguntar o comunicar algo se envía el correo y se espera a que los demás contesten, para lo cual puede pasar hasta un día completo o más.

Las personas que no pertenecen al laboratorio pueden escribir a una dirección de correo que revisa constantemente el administrador del sistema de correo. A esta cuenta llegan preguntas sobre el servicio social y los cursos que se imparten. Se tiene una página web en la que se muestran los cursos, horarios, temarios y tutoriales relacionados.

Se cuenta también con un sistema muy limitado que permite enviar mensajes de texto a través de una página web, pero es muy lento y sólo permite el uso de tres canales de comunicación. La experiencia de los integrantes del laboratorio nos dice que no se utiliza debido a su lentitud y a que cuando alguien entra a buscar asesoría, nadie se entera ni es atendido.

6.9 Inventario de recursos

Para el desarrollo del sistema se cuenta con lo siguiente:

- 2 analistas programadores
- 2 computadoras con las siguientes características
  - PC’s Athlon XP a 1.4 GHz
  - 128 MB RAM
  - S. O. Windows 2000 Workstation
- Software de desarrollo
  - Visual Basic 6
  - Visual C++ 6
- Red Ethernet 10 Mbps

Para la implementación del sistema contamos con lo siguiente:

- 1 administrador del servidor.
- 1 servidor

        - PC Pentium III a 800 MHz
        - 128 MB RAM
        - S. O. Windows 2000 Server
    - Conexión directa a la red UNAM y a la Internet.

6.10 Requerimientos de software y hardware

El sistema tiene como una restricción importante que debe funcionar bajo el sistema operativo Windows 2000 Server, ya que el servidor del laboratorio funciona bajo este sistema.

El servidor IRC no dependerá de ningún otro servicio o utilería que funcione en el servidor. Se espera que la capacidad del servidor sea suficiente, ya que no se manejarán grandes volúmenes de datos ni se espera un número muy grande de usuarios. Las asesorías que se brindan no involucran a más de 4 o 5 personas al mismo tiempo y no se espera que se lleven a cabo más de dos asesorías al mismo tiempo.

Un parámetro que se ha impuesto al desarrollo del sistema, es que sea compatible con clientes ya existentes y en especial se ha decidido que sea el programa cliente llamado mIRC con el que se hagan las primeras pruebas de compatibilidad, debido a que es uno de los más difundidos en la Internet y se apega bien al protocolo.

Se requerirán dos computadoras para el desarrollo del sistema y acceso al servidor para realizar las pruebas y la instalación final.

Como software de desarrollo se requiere una herramienta que nos permita generar aplicaciones compatibles con Windows y otras que faciliten el manejo del análisis léxico y sintáctico.

6.11 Factibilidad técnica

El laboratorio cuenta con la infraestructura necesaria para crear un sistema para la Internet, porque cuenta con una conexión directa las 24 horas del día y una intranet que facilita el trabajo del grupo de desarrollo.

El personal de desarrollo tiene experiencia en la programación de sistemas utilizando Visual Basic y posee conocimientos sobre redes de computadoras y metodologías de desarrollo.

El reto más importante será el de manejar el análisis de los mensajes para identificar los comandos y sus parámetros, ya que no se han realizado trabajos que involucren algoritmos de este tipo ni se han manejado las herramientas necesarias.

6.12 Factibilidad operativa

A primera vista la operación del sistema no presenta ninguna dificultad y su éxito dependerá de dos puntos principales:

- El sistema debe generar respuestas rápidas.
- Las peticiones de asesoría deben ser atendidas siempre que sea posible.

Si se logran estas dos metas, la operación del sistema podrá evaluarse como satisfactoria y útil.

6.13 Aceptación del nuevo sistema

Si se logra la meta esperada en la velocidad de respuesta del sistema, se podrá utilizar de manera regular en las asesorías y conversaciones requeridas y seguramente esto animará a los usuarios a trabajar con el sistema de una manera regular.

6.14 Posibles usuarios del sistema

Todos los integrantes del Laboratorio de Multimedia e Internet y los alumnos de la Facultad de Ingeniería que deseen asesoría o simplemente utilizarlo para conversar entre ellos. Los integrantes del Laboratorio de Multimedia e Internet varía entre 10 y 15 personas. El número de alumnos de la Facultad de Ingeniería que podrían interesarse en utilizar el sistema depende de los cursos que imparte el laboratorio y se pueden contar hasta 150 alumnos por semestre.

6.15 Alternativas de desarrollo

Existen varias formas de resolver el problema:

- Se puede construir un sistema que envíe mensajes a través de la Internet por medio de un protocolo propio.
- Se puede utilizar la Web para crear un sistema que pueda usarse desde cualquier navegador por medio de applets o páginas dinámicas.
- Se puede construir un sistema que envíe mensajes a través de la Internet y que se apegue a algún protocolo popular diseñado para ese propósito.

El primer caso parece ser el más simple porque la comunicación puede ser muy sencilla. Sin embargo, tiene el inconveniente de que hay que desarrollar una aplicación para que los usuarios se conecten entre sí y, además, un servidor para atender las conexiones. Cuando el servidor esté funcionando, sólo se podrán conectar quienes tengan el programa cliente que nosotros hayamos generado, lo cual limita a los usuarios que no lo tengan o que cuenten con un sistema operativo que no sea compatible con nuestro programa cliente.

La opción de Web utilizando applets, presenta el inconveniente de que se deben desarrollar tanto el cliente como el servidor. Además, no se cuenta con los recursos

humanos para programar en lenguaje Java y aprender el lenguaje puede llevar bastante tiempo. Los applets son pequeñas aplicaciones escritas en lenguaje Java y que se pueden ejecutar desde un navegador de Web. La solución por medio de páginas dinámicas es la que se encuentra implementada ya. Las páginas se generan cada 5 a 10 segundos con los mensajes y se envían a todos los navegadores que estén participando en la conversación. Esto ha probado ser muy lento y poco eficiente, ya que la información se envía cada 5 segundos a todos los participantes, aunque no se haya escrito ningún mensaje.

La tercer opción tiene la ventaja de que sólo se trabajará en el programa servidor porque se pueden aprovechar los clientes que ya existan para trabajar con algún protocolo. IRC es una muy buena opción y tiene una gran cantidad de clientes para casi cualquier plataforma. Si nos apegamos al protocolo, ganamos compatibilidad y facilidad de uso, aunque el desarrollo pueda ser más complicado. También tenemos la ventaja de que el desarrollo del programa servidor se puede realizar con las herramientas que ya se conocen.

6.16 Análisis de requisitos

6.16.1 Arquitectura

Una red IRC se define como un grupo de servidores conectados entre sí. Un solo servidor forma la red IRC más simple.

La única configuración de red permitida es la de un árbol extendido donde cada servidor actúa como un nodo central para el resto de la red. El protocolo IRC no provee alguna manera para que dos clientes se comuniquen directamente. Toda la comunicación entre los clientes es reenviada por el servidor.

Las siguientes son las especificaciones del protocolo. Cualquier otro comando no funcionará en nuestra implementación, pero eso no significa que no puedan agregarse después.

El protocolo IRC se ha desarrollado ya durante varios años para usarse en conversaciones basadas en texto. El protocolo tiene sus principios en el modelo cliente-servidor. Es un modelo distribuido en el cual pueden participar varios servidores a los cuales se conectan los clientes. Se requiere que cada servidor tenga una copia del estado global de la información, lo cual es el problema más difícil del protocolo.

Cada servidor tiene clientes conectados y maneja diferentes canales. Toda esta información debe ser conocida por todos los demás servidores que formen la red IRC para que un cliente pueda conversar con cualquier otro sin importar a qué servidor estén conectados. Un canal o un cliente son únicos en todos los servidores que forman una red.

Nuestra implementación tomará en cuenta que la red está formada por un solo servidor.

La comunicación que se lleva a cabo a través de IRC se puede catalogar de diferentes formas.

La comunicación uno a uno normalmente la hacen los clientes cuando se envían mensajes privados. Para proveer una manera de hacer esto, se requiere que todos los servidores sean capaces de hablarse entre sí para enviar un mensaje en solamente una dirección a través del árbol extendido para poder alcanzar a cualquier cliente. La ruta de un mensaje debe ser la más corta entre dos puntos del árbol.

La meta principal de IRC es proveer un foro que permita una conversación fácil y eficiente de uno a muchos. En IRC un canal tiene una existencia dinámica en la cual la conversación que se está llevando a cabo debe enviarse sólo a los servidores que soportan usuarios de dicho canal y estos mensajes sólo deben enviarse una sola vez, ya que cada servidor es responsable de que el mensaje original llegue a todos sus usuarios.

6.16.2 Códigos de carácteres

No está especificado ningún conjunto de carácteres. El protocolo se basa en un conjunto de códigos compuestos de 8 bits conocidos como octetos. Cada mensaje puede estar compuesto de cualquier número de octetos, aunque algunos octetos son usados como códigos de control, los cuales actúan como delimitadores de mensajes.

Debido al origen escandinavo de IRC, los carácteres { } | ^ son considerados como los equivalentes minúsculos de los carácteres [ ] \ ^ respectivamente. Esto es importante cuando se determina la equivalencia de dos nicknames o dos nombres de canal.

6.16.3 Servidores

El servidor forma la columna vertebral de la red IRC, ya que es el único componente capaz de unir a todos los demás. Provee un punto al cual los clientes y otros servidores se pueden conectar.

El servidor debe proveer los tres servicios básicos requeridos para conversación o conferencia en tiempo real definidos por la arquitectura IRC:

- Localizador de clientes. Para tener la posibilidad de intercambiar mensajes, dos clientes deben ser capaces de localizarse uno al otro. Al conectarse a un servidor, un cliente se registra usando una etiqueta que entonces es usada por otros servidores y clientes para saber dónde se localiza el cliente. Los servidores son los responsables de llevar el control de las etiquetas utilizadas.

- Reenvío de mensajes. El protocolo IRC no provee ninguna manera para que dos clientes se comuniquen directamente. Toda la comunicación entre los clientes es reenviada por el servidor.

- Manejo de canales. Un canal es un grupo de uno o más usuarios que tiene la propiedad de identificarse por medio de un nombre. Todos estos usuarios reciben los mensajes que se envían al canal. El canal tiene varias propiedades que pueden ser manipuladas por todos o algunos de sus miembros. Los canales proveen una forma de enviar un mensaje a varios clientes. Los servidores mantienen los canales realizando el multiplexaje de mensajes necesario. Los servidores también son responsables de manejar los canales llevando el control de los miembros de cada uno.

Los servidores se identifican de manera única por su nombre, el cual tiene un máximo de 63 carácteres.

6.16.4 Usuarios

Los clientes usuarios son generalmente programas que tienen una interface basada en texto que es usada para comunicarse interactivamente vía IRC. De cada usuario, el servidor debe tener la siguiente información:

- El servidor al cual está conectado.
- El nombre del host en el que el cliente está corriendo.
- El nombre de usuario o login del usuario en ese host.

Cada usuario se distingue de los otros por un nickname único de 9 carácteres como máximo, aunque los clientes deben aceptar cadenas más largas porque podrían usarse en el futuro.

6.16.5 Servicios

A diferencia de los usuarios, los clientes de tipo “servicio” no están hechos para ser utilizados manualmente ni para conversar. Tienen un acceso más limitado a las funciones del protocolo y al mismo tiempo pueden tener acceso a más datos privados de los servidores. Los servicios son generalmente autómatas empleados para dar algún tipo de servicio a los usuarios (no necesariamente relacionado con IRC). Por ejemplo, un servicio que recoge estadísticas acerca del origen de los usuarios conectados a la red IRC. Cada servicio se distingue de los otros por un nombre compuesto de un nickname y un nombre de servidor. Como los usuarios, un servicio tiene un nickname con una longitud máxima de 9 carácteres.

6.16.6 Operadores

Para lograr un orden razonable dentro de la red IRC, se permite que una clase especial de usuarios (operadores) desempeñe funciones generales de mantenimiento sobre la red. Aunque los poderes otorgados a un operador pueden ser considerados como peligrosos, a menudo son necesarios. Un poder más controversial de los operadores es la habilidad de remover un usuario de la red. Los operadores son capaces de cerrar cualquier conexión entre un cliente y el servidor.

6.16.7 Canales

Un canal es un grupo de uno o más usuarios que tiene la propiedad de identificarse por medio de un nombre. Todos estos usuarios reciben los mensajes que se envían al canal. El canal se caracteriza por su nombre, sus propiedades y miembros actuales.

Los nombres de los canales son cadenas que inician con '&', '#', '+' o '!', con una longitud de hasta 50 carácteres. Aparte del requerimiento de que el primer carácter sea '&', '#', '+' o '!', la única restricción sobre el nombre de un canal es que no debe contener espacios, comas o el valor ASCII 7. El espacio es usado como separador de parámetros y la coma es usada como un separador de elementos de una lista. Los nombres de canal no diferencian mayúsculas o minúsculas. Cada prefijo caracteriza un tipo de canal diferente, los canales con el prefijo '&' son locales al servidor en el cual fueron creados.

Una entidad de tipo canal es conocida por uno o más servidores en la red IRC. Un usuario sólo puede ser miembro de un canal conocido por el servidor al cual está conectado directamente. La lista de servidores que saben de la existencia de un canal en particular debe ser manejada como una parte importante de la red IRC para que los mensajes dirigidos al canal lleguen a todos los miembros de éste.

Otros canales son conocidos por uno o más servidores dependiendo de una máscara de canal: si no hay máscara de canal, entonces el canal es conocido por todos los servidores. Si hay una máscara de canal, entonces el canal debe ser conocido sólo por los servidores que tienen un usuario local en el canal y sus vecinos que también corresponden con la máscara. Como otros servidores no tienen ningún conocimiento de la existencia de tal canal, el área formada por los servidores que corresponden con la máscara, deben ser contiguos para que el canal sea conocido por todos estos servidores.

Una máscara es un patrón que se utiliza para hacer corresponder varios nombres de servidor utilizando comodines.

Cada canal tiene propiedades que están definidas por los modos de canal. Los modos de canal pueden ser manipulados por los miembros del canal. Los modos afectan la manera en que los servidores manejan los canales.

Los canales con un prefijo ‘+’ no soportan modos de canal. Esto significa que todos los modos están deshabilitados con excepción de la bandera de modo ‘t’ que explicaremos más adelante.

Para que los miembros de un canal mantengan cierto control, algunos miembros del canal son privilegiados. Sólo estos miembros tienen permitido desarrollar las siguientes acciones sobre el canal:

INVITE	- Invita a un cliente a un canal de sólo invitados (modo +i)
KICK	- Saca a un cliente del canal
MODE	- Cambia el modo del canal, así como los privilegios de los miembros
PRIVMSG	- Envío de mensajes al canal (modo +n, +m, +v)
TOPIC	- Cambia el tópico del canal en un canal con modo +t

Los operadores de un canal dado son considerados como los dueños del canal. La propiedad del canal es compartida por todos los operadores del canal.

Los operadores de canal se identifican por el símbolo ‘@’ al principio de su nickname cuando están asociados a un canal.

Debido a que los canales que empiezan con el carácter ‘+’ no soportan modos de canal, ningún miembro puede tener el estatus de operador de canal.

Un usuario que crea un canal con el carácter ‘!’ como prefijo es identificado como el creador del canal. Después de la creación del canal, a este usuario se le da también el estatus de operador de canal. Como reconocimiento a este estatus, los creadores de canal tienen la habilidad de cambiar ciertos modos del canal que los operadores de canal no pueden.

Con respecto al tiempo de vida de un canal, existen dos grupos de canales: canales estándar cuyo prefijo es ‘&’, ‘#’ o ‘+’, y los canales seguros cuyo prefijo es ‘!’.

Los canales estándar son creados implícitamente cuando el primer usuario se une a él y deja de existir cuando el último usuario lo abandona. Mientras el canal existe, cualquier cliente puede referirse al canal por medio de su nombre.

El usuario que crea un canal, automáticamente se convierte en operador del canal con la notable excepción de los canales cuyos nombres tienen el prefijo ‘+’.

A diferencia de otros canales, los canales seguros no se crean implícitamente. Un usuario que desea crear uno de estos canales, debe pedir su creación enviando un comando JOIN especial al servidor, en el cual el identificador del canal (entonces desconocido) es remplazado por el carácter ‘!’. El proceso de creación para este tipo de canal está controlado estríctamente. El usuario sólo escoge una parte del nombre del canal (conocida como el nombre corto del canal), el servidor automáticamente agrega al final del nombre que da el usuario un identificador de 5 carácteres. El nombre de canal que resulta de la combinación de estos dos elementos es único, haciendo el canal seguro contra abusos

basados en divisiones de la red IRC. Una división de la red sucede cuando dos servidores se desconectan por alguna razón y entonces en cada mitad quedan usuarios que estaban conversando en el mismo canal.

Un servidor no debe permitir la creación de un nuevo canal si otro canal con el mismo nombre corto existe o si otro canal con el mismo nombre corto existió recientemente y cualquiera de sus miembros lo abandonó debido a una división de la red. Los canales dejan de existir después de que el último usuario abandonó y ningún otro miembro del canal lo abandonó recientemente debido a una división de la red.

Los nombres de canal no se vuelven no disponibles; estos canales pueden seguir existiendo después de que el último usuario abandonó. Sólo el usuario que creó el canal se convierte en creador de canal y los usuarios que se unen a un canal vacío existente no se convierten automáticamente en creadores de canal ni en operadores de canal. Para asegurar la unicidad de los nombres de canal, el identificador de canal debe seguir ciertas reglas.

Los modos de canal disponibles aparecen en la Tabla 6.1

O	Otorga el estatus de creador de canal
o	Otorga / toma el privilegio de operador de canal
v	Otorga / toma el privilegio de voz
a	Cambia la bandera de canal anónimo
i	Cambia la bandera de canal de sólo invitación
m	Cambia el modo de canal moderado
n	Cambia a “no mensajes al canal desde clientes de fuera”
q	Cambia la bandera de canal silencioso
p	Cambia la bandera de canal privado
s	Cambia la bandera de canal secreto
r	Cambia la bandera de canal “server reop”
t	Cambia la bandera de canal de tópico modificable por el operador de canal
k	Establece/elimina la clave de canal (password)
l	Establece/elimina el límite de usuario al canal
b	Establece/elimina la máscara de rechazo para mantener usuarios fuera
e	Establece/elimina una máscara de excepción para sobrescribir una máscara de rechazo.
I	Establece/elimina una máscara de invitación para invalidar automáticamente la bandera de sólo invitación

Tabla 6.1

A menos que se mencione más adelante, todos estos modos pueden ser manipulados por los operadores de canal por medio del comando MODE.

Los modos “O,o,v,a,i,m,n” se relacionan con un canal y un usuario que pertenece a ese canal. No decimos que un canal tiene modo “o” sino que un usuario en ese canal tiene ese modo.

El modo ‘O’ es usado sólo junto con canales seguros y no debe ser manipulado por los usuarios. Los servidores lo usan para dar al usuario que ha creado el canal, el estatus de creador de canal.

El modo ‘o’ es usado para cambiar el estatus de operador de canal de un miembro.

El modo ‘v’ es usado para dar y quitar el privilegio de voz a un miembro del canal. Los usuarios con este privilegio pueden hablar en canales moderados.

La bandera de canal ‘a’ define un canal anónimo. Esto significa que cuando un mensaje enviado al canal es enviado por el servidor a los usuarios y el origen es un usuario, entonces debe ser alterado. Para proteger el mensaje el origen es cambiado a “anonymous!anonymous@anonymous”. Debido a esto, los servidores deben prohibir a los usuarios usar el nickname anonymous. Los servidores tampoco deben enviar mensajes QUIT de los usuarios que abandonan esos canales a los otros usuarios, sino generar un mensaje APART en su lugar.

En los canales con el carácter ‘&’ como prefijo, esta bandera puede ser cambiada por los operadores de canal, pero en canales con el carácter ‘!’ como prefijo, esta bandera puede ser establecida (pero no quitada) solamente por el creador del canal. Esta bandera no debe existir en otros tipos de canales.

Las respuestas a los comandos WHOIS, WHO y NAMES no deben revelar la presencia de otros usuarios en canales para los cuales la bandera de anonimato está colocada.

Cuando la bandera de canal ‘i’ está colocada, sólo se aceptan nuevos usuarios si su máscara corresponde con la lista de invitados o si han sido invitados por un operador del canal. Esta bandera también restringe el uso del comando INVITE a los operadores del canal.

La bandera de canal ‘m’ es usada para controlar quién puede hablar en un canal. Cuando está establecida sólo los operadores de canal y los miembros a los que se les ha dado el privilegio de voz, pueden mandar mensajes al canal. Esta bandera sólo afecta a los usuarios.

Cuando la bandera ‘n’ está colocada, sólo los miembros del canal pueden enviar mensajes al canal. Esta bandera sólo afecta a los usuarios.

La bandera de canal ‘q’ es para que la usen sólo los servidores. Cuando está colocada restringe el tipo de datos enviados a los usuarios acerca de las operaciones del canal: la entrada de otros usuarios y los cambios de nickname no se envían. Desde el punto de vista de un usuario, el canal sólo contiene un usuario.

Esto se usa para crear canales locales especiales, en los cuales los servidores envían noticias relacionadas a su operación. Esto es usado como un modo más eficiente y flexible que remplaza el modo de usuario ‘s’ definido en el RFC 1459.

La bandera de canal ‘p’ es usada para marcar un canal como privado y la bandera ‘s’ para marcarlo como secreto. Ambas propiedades son similares y esconden la existencia del canal a otros usuarios.

Esto significa que no hay manera de obtener el nombre de este canal del servidor sin ser un miembro. En otras palabras, estos canales deben ser omitidos en las respuestas a consultas como la del comando WHOIS.

Cuando un canal es secreto, además, el servidor actuará como si el canal no existiera para consultas como TOPIC, LIST y NAMES. Existe una excepción a esta regla: los servidores responderán correctamente al comando MODE. Finalmente, los canales secretos no se cuentan en la respuesta al comando LUSERS cuando el parámetro <mascara> sea especificado.

Las banderas de canal ‘p’ y ‘s’ no deben colocarse al mismo tiempo. Si un mensaje MODE que se origina desde un servidor coloca la bandera ‘p’ y la bandera ‘s’ ya está colocada, el cambio es ignorado silenciosamente.

La bandera de canal ‘r’ sólo está disponible en canales cuyo nombre comienza con el carácter ‘!’ y sólo puede ser cambiada por el creador del canal.

Esta bandera es usada para prevenir que un canal no tenga operador por un período largo de tiempo. Cuando la bandera está establecida, cualquier canal que haya perdido a todos sus operadores de canal por más tiempo que el “reop delay” dispara un mecanismo en los servidores para hacer operador a algunos o a todos los miembros del canal.

La bandera de canal ‘t’ es usada para restringir el uso del comando TOPIC sólo a los operadores de canal.

Un límite de usuarios se puede imponer a los canales usando la bandera de canal ‘l’. Cuando el límite es alcanzado, los servidores deben prohibir a sus usuarios locales unirse al canal. El valor del límite sólo puede hacerse disponible a los miembros del canal en la respuesta a un comando MODE.

Cuando una clave de canal es establecida (usando el modo ‘k’), los servidores deben rechazar las peticiones de sus usuarios locales para unirse al canal, a menos que esta clave sea dada. La clave del canal sólo debe ser visible para los miembros del canal en la respuesta a la consulta MODE.

Para reducir el tamaño total de la base de datos global de modos de acceso de control para los canales, los servidores pueden poner un límite máximo en el número de tales modos para un canal en particular. Si se impone tal restricción, sólo debe afectar a las peticiones de usuario. El límite debe ser homogéneo en toda la red IRC.

Cuando un usuario pide unirse a un canal, su servidor local revisa si la dirección del usuario corresponde a alguna de las máscaras de rechazo impuestas al canal. Si se encuentra una correspondencia, la petición del usuario es rechazada, a menos que la dirección también corresponda con una máscara de excepción del canal.

Los servidores no deben permitir que un miembro de un canal que está relegado del canal (banned), pueda hablar al canal, a menos que este miembro sea un operador o tenga privilegio de voz. Un usuario que tiene prohibido entrar a un canal y que tiene una invitación de un operador del canal puede entrar.

Para los canales que tienen la bandera de sólo invitados, los usuarios que correspondan con una máscara de invitación pueden unirse al canal sin ninguna invitación.

6.16.8 Mensajes

Los servidores y clientes se envían mensajes, los cuales pueden o no generar una respuesta. Si el mensaje contiene un comando válido, el cliente esperará una respuesta, pero no lo hará para siempre. La comunicación es de naturaleza asíncrona.

Cada mensaje de IRC consiste de hasta tres partes principales: el prefijo (opcional), el comando y los parámetros del comando (15 como máximo) y todos los parámetros están separados por un espacio.

La presencia de un prefijo se indica con un carácter ‘:’ el cual debe ser el primer carácter del mensaje mismo. No debe haber espacio entre los dos puntos y el prefijo. El prefijo es usado por los servidores para indicar el verdadero origen del mensaje. Si no existe el prefijo, se asume que se originó en la conexión desde la cual fue recibida. Los clientes no deben usar un prefijo cuando mandan un mensaje. Si usan uno, el único válido es el nickname registrado del cliente.

El comando debe ser un comando IRC válido o un número de tres dígitos representado en texto ASCII.

Los mensajes IRC son siempre líneas de carácteres terminados con un par CR-LF (carriage return – line feed) y no deben exceder 512 carácteres de longitud incluyendo el par CR-LF. Por lo tanto, son 510 carácteres permitidos para el comando y sus parámetros.

Algunos mensajes aceptan comodines como ‘?’ y ‘*’ para sustituir un carácter o cero o más carácteres respectivamente. Las máscaras son cadenas que pueden incluir varios comodines.

6.16.9 Respuestas numéricas

La mayoría de los mensajes enviados al servidor generan una respuesta de algún tipo. La respuesta más común es la numérica, usada para errores y respuestas comunes. La respuesta numérica debe enviarse como un mensaje que consiste del prefijo de quien lo

envía, el número de tres dígitos y el destino de la respuesta. Una respuesta numérica no se puede generar desde un cliente.

6.16.10 Detalles de los mensajes

Debido a la extensión de esta sección, se ha agregado el Apéndice A, en el cual se incluyen las descripciones de cada mensaje reconocido por el protocolo IRC.

6.16.11 Otros requisitos

Un requerimiento especial del sistema es la posibilidad de que las personas que se encuentren en el laboratorio, se den cuenta de que alguien está solicitando asesoría por medio del sistema. El personal del laboratorio no deberá estar forzosamente revisando el sistema, ni tendrá que estar en una computadora para poder enterarse de que se solicita su ayuda. Este servicio sólo se dará dentro de los horarios de asesoría del laboratorio y así se les hará saber a los usuarios por medio de un mensaje al momento de iniciar una conexión con el servidor.

Una vez terminado el análisis de requisitos del sistema, se iniciará el diseño de los módulos del sistema de manera que se cumplan los requerimientos. Se tienen identificados los mensajes que se implementarán, se diseñará el módulo que maneja las comunicaciones y se generará la rutina que nos permita reconocer cada uno de los comandos y parámetros de los mensajes. También se elegirán las estructuras de datos necesarias para llevar el registro de los canales y usarios.

Capítulo 7

Diseño

7.1 Definición de módulos

Del análisis hecho en el capítulo anterior, podemos derivar los siguientes diagramas de flujo de datos.

7.2 Diagrama de flujo de datos de contexto

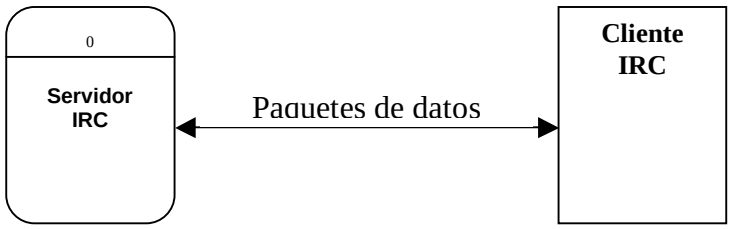

Figura 7.1

De una forma muy general, sólo existe una entidad externa con la que el sistema interactúa. Se trata del cliente y puede ser un usuario o un servicio (Figura 7.1).

7.3 Diagrama de flujo de datos de nivel 0

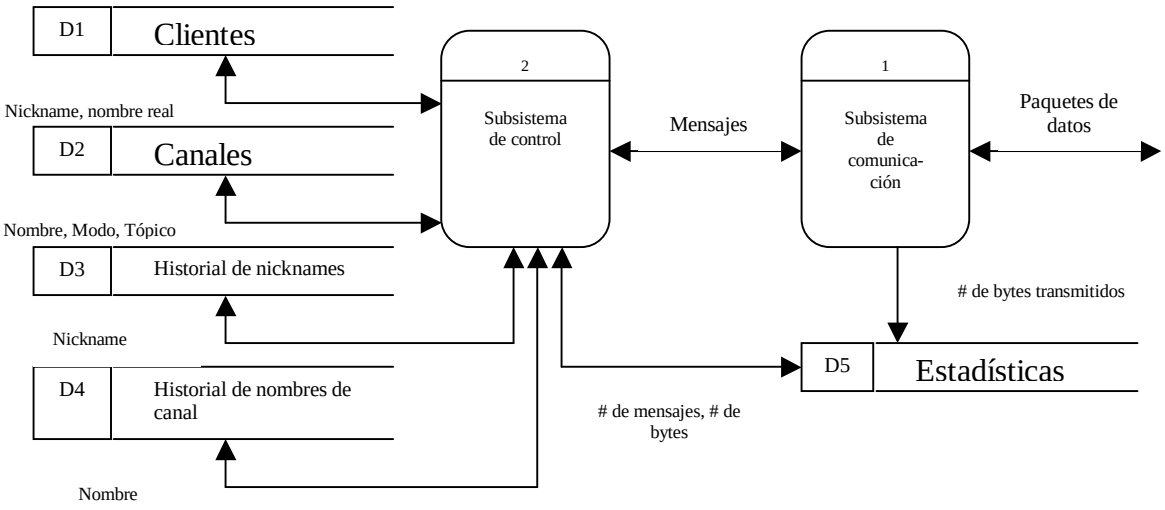

Figura 7.2

Habrán dos módulos principales, el módulo de comunicación y el de control. El módulo de comunicación se encargará de mantener las conexiones con los clientes, llevar estadísticas de los mensajes, ordenar el envío y recepción de los mensajes y enviar y recibir del módulo de control los mensajes que llegan de los clientes y que se van a enviar (Figura 7.2).

El módulo de control se encargará de recibir los mensajes, analizarlos y ejecutar las acciones que correspondan con los comandos recibidos. Después generará los mensajes de respuesta o error necesarios y los pasará al módulo de comunicación.

Existen tres depósitos de datos con información sobre los canales, los clientes y las estadísticas y dos especiales, que contendrán información histórica de nicknames y nombres de canal, para crear un algoritmo que evitará que dos usuarios usen el mismo nickname o que dos canales se llamen igual.

7.4 Diagrama de explosión 1 (proceso subsistema de comunicación)

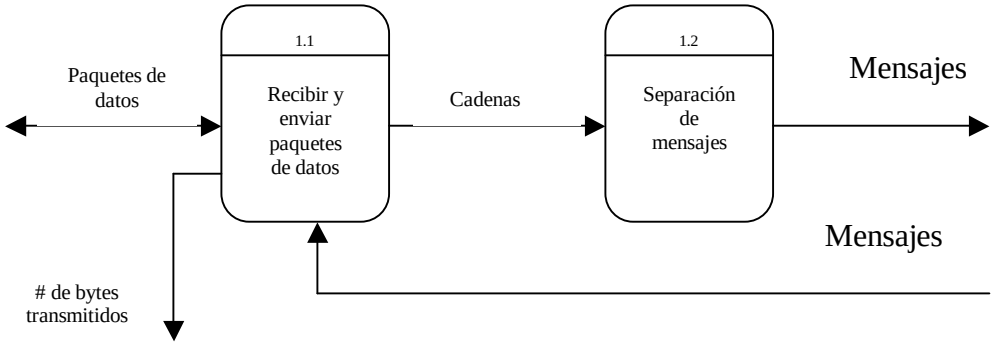

Figura 7.3

El proceso 1.1 (figura 7.3) incluye una conexión por medio de un socket y una pequeña rutina para guardar el número de bytes que se envían y reciben a través de esta conexión. La separación de mensajes (proceso 1.2) tiene como fin buscar los carácteres Carriage Return (CR) o Line Feed (LF), que son los designados en el protocolo para marcar el fin de un mensaje. De la cadena de carácteres que llegue, este proceso deberá generar por separado los mensajes que contenga. Estos dos procesos se repiten por cada conexión que se establezca.

7.5 Diagrama de explosión 2 (proceso subsistema de control)

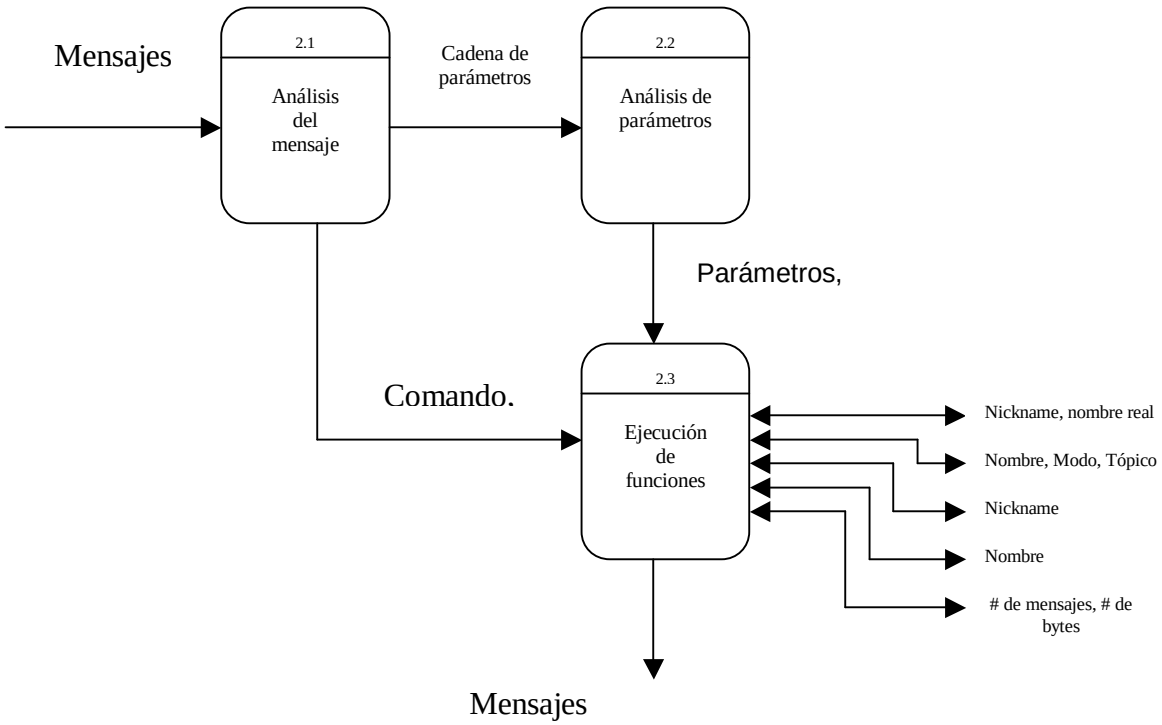

Figura 7.4

El proceso de Análisis del mensaje (2.1) tiene como fin identificar una estructura general del mensaje como correcta o no. Si no es correcta se envía un error al proceso 2.3 para que éste envíe el mensaje correspondiente al módulo de comunicación. Si el mensaje es correcto sintácticamente entonces se analizan los parámetros (proceso 2.2) para verificar que sean correctos. Estos dos análisis son solamente sintácticos y están separados porque el análisis de los parámetros es más complicado que el primero, en el cual sólo hay que identificar tres partes en el mensaje y verificar que la parte que corresponde al comando sea válida. La forma de la cadena de parámetros depende del comando y es muy variada.

El proceso 2.3 recibe las partes del mensaje como unidades separadas y procede a analizar si es posible ejecutar el comando que se ha recibido. Un mensaje dirigido a un canal no existente pasará la verificación sintáctica si está bien construido, pero generará un error cuando se busque el nombre de ese canal y no sea encontrado (Figura 7.4).

Este proceso tiene a su disposición cinco depósitos de datos en los que guardará o de los que obtendrá información según sea necesario. La información que puede manejar es la que se definió en la etapa de análisis al definir las entidades. Como resultado de las acciones que tome se generará un mensaje de error o de respuesta junto con el identificador de a quién va dirigido el mensaje.

7.6 Definición de datos y funciones

Para organizar los datos utilizaremos objetos porque nos ofrecen una forma adecuada de organizar la información. Las entidades que identificamos se convertirán directamente en objetos y dividiremos las funciones o procesos entre éstos. Tenemos entonces una clase que llamaremos usuario, una clase servicio, una clase canal, una clase servidor y una clase mensaje.

Necesitamos una lista de clientes, una de servicios y una de canales. La lista debe poder crecer o decrecer y permitir la búsqueda de un elemento en particular, por medio de un identificador o una o más características. Cada una de estas listas se implementará como un objeto que encerrará una colección. En Visual Basic, una colección es un conjunto ordenado de elementos a los que se puede hacer referencia como una unidad. El objeto "Collection", permite hacer referencia de forma sencilla a un grupo de elementos relacionados como un único objeto. Los elementos o miembros de una colección sólo tienen que estar relacionados por el hecho de existir en la colección y no tienen que compartir el mismo tipo de dato. Esto nos da la facilidad de agregar y borrar elementos a nuestra lista de una manera sencilla.

Cada conexión tiene asociado un socket y un cliente, así que haremos parte del objeto usuario y del objeto servicio a su socket correspondiente por medio de una referencia al socket. A continuación definiremos la estructura de los objetos:

Clase usuario		
Contendrá información sobre un usuario registrado.		
Descripción	**Nombre**	**Tipo**
Servidor local (63 carácteres como máximo)	ServidorLocal	String * 63
Nickname (9 carácteres como máximo)	Nickname	String * 9
Nombre de usuario (cadena)	NombreUsuario	String
Modo a (booleano)	ModoAway	Boolean
Modo i (booleano)	ModoInvisible	Boolean
Modo w (booleano)	ModoWallops	Boolean
Modo r (booleano)	ModoRestringido	Boolean
Modo o (booleano)	ModoOperador	Boolean
Modo O (booleano)	ModoOperadorLocal	Boolean
Modo s (booleano)	ModoNoticiasServidor	Boolean
Unused (reservado)	Unused	Variant
Nombre Real (cadena)	NombreReal	String
Password (cadena)	Password	String

Tabla 7.1

Clase servicio		
Contendrá información sobre un servicio registrado.		
Descripción	**Nombre**	**Tipo**
Nickname	Nickname	String * 9
Servidor local (63 carácteres como máximo)	ServidorLocal	String * 63
Unused (reservado)	Unused	Variant
Distribución (cadena)	Distribucion	String
Tipo (reservado)	Tipo	String
Información (cadena)	Informacion	String
Password (cadena)	Password	String

Tabla 7.2

Clase canal		
Contendrá información sobre un canal registrado.		
Descripción	**Nombre**	**Tipo**
Nombre (50 carácteres como máximo)	Nombre	String * 50
Nombre corto (cadena)	NombreCorto	String
Tipo (caracter)	Tipo	String
Tópico (cadena)	Topico	String
Password (cadena)	Password	String
Máscara (cadena)	Mascara	String
Creador (nickname)	Creador	String
Límite de usuarios (entero)	Limite	Integer
Modo q (booleano)	ModoSilencioso	Boolean
Modo p (booleano)	ModoPrivado	Boolean
Modo s (booleano)	ModoSecreto	Boolean
Modo r (booleano)	ModoReop	Boolean
Modo t (booleano)	ModoTopico	Boolean
Modo k (booleano)	ModoKey	Boolean
Modo l (booleano)	ModoLimite	Boolean
Modo b (booleano)	ModoMascaraBan	Boolean
Modo e (booleano)	ModoMascaraExcepcionBan	Boolean
Modo I (booleano)	ModoMascaraInvitacion	Boolean
Lista de usuarios (array de estructuras)	ListaUsuarios	usuario
Lista de invitados (array de cadenas)	ListaInvitados	usuario
Máscara de usuarios rechazados (cadena)	MascaraBan	String
Máscara de excepcion de usuarios rechazados (cadena)	MascaraExcepcion	String

Tabla 7.3

Clase servidor		
Contendrá información local del servidor.		
Descripción	**Nombre**	**Tipo**
Nombre (63 carácteres como máximo)	Nombre	String
Lista de operadores y sus passwords (array de cadenas)	ListaOperadores	String
Reop delay (entero en segundos)	RetardoReop	Integer
Channel delay (entero en segundos)	RetardoCanal	Integer
User delay (entero en segundos)	RetardoUsuario	Integer

Tabla 7.4

Clase mensaje		
Contendrá información de un mensaje en específico.		
Descripción	**Nombre**	**Tipo**
Contenido (512 carácteres como máximo)	Contenido	String
Hora de recepción (con precisión de segundos)	Hora	Date
Remitente (entero)	Remitente	Integer

Tabla 7.5

Con respecto a las funciones, existirá una por cada mensaje del protocolo que se implemente. Cada una será un módulo separado y hará uso de las listas de usuarios, servicios y canales, así como de los datos locales del servidor. Cada objeto tendrá funciones miembro para obtener los datos que sean necesarios y para modificarlos de manera correcta a través de estas funciones.

Para el análisis léxico y sintáctico de los mensajes se utilizará una herramienta llamada bumblebee que es un generador de parsers compatible con Bison y que incluye una herramienta compatible con Flex. De esta manera se generará una librería dinámica que podrá utilizarse desde Visual Basic. Se decidió hacerlo de esta manera porque estas dos herramientas nos facilitan la programación de las rutinas para analizar los mensajes, pero como generan código en lenguaje C, tuvimos que compilarlas como librerías de funciones para poder llamarlas desde el código de Visual Basic.

7.7 Diseño de interface gráfica

Aunque no es necesaria una interface para el servidor IRC, porque toda la información se puede consultar a través de un cliente con un password de operador, se generará para mostrar los canales existentes, las estadísticas, los datos del servidor y los

clientes registrados. Esta información se actualizará en la pantalla sólo cuando sea solicitado, porque actualizar la información continuamente puede hacer que el sistema sufra retrasos. También se incluirá una ventana para ver los mensajes que se están enviando y recibiendo, para ayudar a depurar el programa.

7.8 Diseño de la ayuda

Se incluirá un módulo de ayuda que contendrá información sobre los mensajes, su sintaxis y su propósito. Servirá para que los administradores del servidor recuerden la sintaxis de los mensajes.

7.9 Diseño del manejo de errores y mensajes del sistema

Este apartado forma parte del mismo protocolo. Todos los errores y los mensajes que se deben generar están documentados en al capítulo de análisis y se implementarán de esa manera. Existe una gran cantidad de mensajes representados por medio de un número de tres dígitos y para cada uno existe un significado concreto y una explicación de su significado.

Capítulo 8

Desarrollo

Este capítulo nos muestra una idea general de cómo se desarrolló el sistema y las herramientas que se utilizaron. Para tener una idea completa de cómo se programó el sistema iremos recorriendo los módulos desde que se recibe el mensaje hasta que se realiza la acción que corresponde al comando identificado.

En el desarrollo de este sistema utilizamos tres herramientas principalmente, Visual Basic, Visual C++ y el generador de parsers Bumblebee.

8.1 Recibir y enviar paquetes de datos

Este módulo se encarga de la comunicación TCP/IP. Se implementa por medio de un arreglo de controles de tipo Winsock. Este control nos permite crear un socket y nos facilita el manejo del envío y recepción de datos a través de la red. Existe un socket cuya única función es "escuchar" en el puerto 6667 que es un número de puerto estándar para el protocolo IRC, aunque no es obligatorio usarlo. Por "escuchar" entenderemos estar al pendiente de peticiones de conexión de otros nodos de la red. Cada vez que el socket detecta una petición para conectarse al servidor crea un nuevo socket y le delega la tarea de atender esa comunicación; después, el socket vuelve a escuchar esperando nuevas peticiones. Por cada nueva conexión se agrega un elemento al arreglo de controles.

Cuando un cliente se quiere conectar al servidor debe especificar la dirección IP y el puerto. Con estos dos datos se puede identificar a qué programa del servidor se quiere conectar un cliente. En lugar de la dirección IP se puede utilizar el nombre de dominio del servidor, en nuestro caso "mmedia2.fi-b.unam.mx".

Cada socket se encarga de atender a un usuario o a un servicio. Cuando el control detecta que han llegado datos, se dispara un evento que nos permite recuperarlos en una cadena y contar el número de bytes que llegaron, lo cual nos ayudará a actualizar las estadísticas sobre la comunicación que lleva a cabo el servidor. Cuando un cliente se desconecta, el socket es destruido y los recursos que utilizaba se liberan para aprovecharlos de nuevo. Los datos que llegan en realidad pueden recuperarse en una cadena o en variables de tipo entero o de cualquier otro tipo, pero en este caso todos los mensajes se forman exclusivamente de cadenas. Existen respuestas numéricas, pero incluso, éstas se envían como cadenas de dígitos y no como enteros o números reales.

Cada cliente es identificado por medio del índice del elemento en el arreglo de controles, este índice es enviado junto con cada mensaje a las etapas posteriores de procesamiento para no perder de vista quién generó el mensaje.

Cuando el servidor quiere enviar un mensaje, simplemente invoca a la función SendData de uno de los sockets la cual toma como argumento una cadena de texto a ser enviada. La única restricción es el tamaño de la cadena, que no debe ser mayor a 512 carácteres. Si llega un mensaje más grande se truncará y es posible que se convierta en un mensaje no válido.

8.2 Separación de mensajes

Después de recibir los datos, ya tenemos una cadena de texto que puede contener uno o más mensajes, así que comenzamos a separar la cadena en mensajes independientes. Esto se hace buscando un carácter que funciona como delimitador de mensaje. Los carácteres delimitadores son los llamados “Carriage Return” (0xD) y “Line Feed” (0xA). El protocolo especifica que deben ser los dos carácteres juntos o cualquiera de los dos y esto último es lo que pasa precisamente en los sistemas con los que hemos trabajado.

Es muy común que los clientes envíen más de un mensaje de una sola vez, debido a que algunas acciones necesitan más de un mensaje para llevarse a cabo, como por ejemplo el registro de un cliente, que necesita del mensaje USER y del mensaje NICK para completarse.

Cada vez que se identifica uno de estos delimitadores, se separa el mensaje y se envía al módulo de análisis. Esta separación es relativamente sencilla y se hace por medio de varias funciones de tratamiento de cadenas de texto que buscan uno de los carácteres delimitadores y dividen la cadena en mensajes independientes.

Si se llegan a encontrar los dos delimitadores juntos, simplemente se interpretará que han llegado dos mensajes y que el segundo está vacío. Este proceso se realiza con todos los mensajes en todos los sockets.

8.3 Análisis de mensajes

Cada mensaje que llega es analizado en dos etapas. Esta primera etapa comprende la identificación de las partes principales del mensaje. Estas partes son el prefijo, el comando y los parámetros. Se decidió dividir el análisis en dos partes porque cada comando tiene una estructura muy especial en la parte de parámetros, así que preferimos primero identificar el comando y con esta información buscar una estructura específica en los parámetros. Esto simplifica el proceso y genera un módulo de análisis diferente para cada estructura de parámetros de cada comando. A continuación se muestra la definición de la gramática utilizada en esta etapa. La gramática está descrita en formato BNF aumentado.

```
message    =  [ ":" prefix SPACE ] command [ params ] crlf
prefix     =  servername / ( nickname [ [ "!" user ] "@" host ] )
command    =  1*letter / 3digit
params     =  *14( SPACE middle ) [ SPACE ":" trailing ]
           =/ 14( SPACE middle ) [ SPACE [ ":" ] trailing ]

nospcrlfcl =  %x01-09 / %x0B-0C / %x0E-1F / %x21-39 / %x3B-FF
                ; cualquier octeto excepto NUL, CR, LF, " " y ":"
middle     =  nospcrlfcl *( ":" / nospcrlfcl )
trailing   =  *( ":" / " " / nospcrlfcl )

SPACE      =  %x20        ; carácter espacio
crlf       =  %x0D %x0A   ; "carriage return" "linefeed"
```

Aquí, no interesa el significado del mensaje, sino que esté bien construido, es decir, que todos los carácteres sean los aceptados por el protocolo y que la estructura general del mensaje sea correcta. No nos interesa si los parámetros tienen una estructura correcta, sólo si todos los carácteres son válidos.

Se identifican todas las partes del prefijo y el comando. Los parámetros se separan como una sola cadena que será analizada en la segunda etapa.

La programación de este análisis se hizo en una herramienta compatible con Lex y Yacc llamada Bumblebee. En realidad, primero se hicieron pruebas con Flex y Bison y después se utilizó la herramienta, ya que está diseñada para Windows y podíamos utilizar el resultado desde Visual Basic. Esta herramienta se puede utilizar libremente en proyectos de tipo estudiantil, así que decidimos usarla.

El proceso es como sigue: Se genera un archivo que describe la forma de la gramática y otro con una descripción de los tokens que serán reconocidos. Se compilan estos programas y como resultado obtenemos un programa en lenguaje C, que realiza un análisis sobre una cadena o archivo de texto para ver si cumple con la sintaxis. Si una cadena de texto no cumple se genera un error. Cuando la cadena analizada cumple con la especificación es aceptada y se puede realizar cualquier acción con los elementos identificados.

En nuestro desarrollo se generaron los programas lexer.l y parser.y. El programa lexer.l sirve para definir los tokens que identificaremos en nuestros mensajes. Estos son: CRLF, SPACE, USER, LETRAS, TRES_DIGITOS, SERVERNAME, NOSPCRLFCL, HOST y NICKNAME.

El programa parser.y utiliza estos tokens para reconocer la sintaxis del mensaje. La función que generamos se llama mensaje_irc() y recibe varios parámetros por referencia, en los que escribe el resultado del análisis y regresa un entero indicando si el mensaje es válido o no.

La declaración de esta función es la que sigue:

```
int mensaje_irc(char *entrada, char *prefijo_nickname, char *prefijo_servername, char
*prefijo_usuario, char *prefijo_host, char *comando, char *parametros);
```

El parámetro “entrada” es el mensaje a analizar, los demás parámetros son cadenas que sirven para guardar los elementos en que se divide el mensaje. Se hace una división exhaustiva del prefijo, se separa el comando y se separa una cadena con todos los parámetros. Al llamar a esta función, se analiza el número entero que regresa y los valores que entonces tienen las cadenas que se pasaron como parámetros.

Normalmente, tendríamos una función main() para iniciar la rutina de análisis, pero como no se intenta crear un programa independiente, sino generar una librería que ponga a

la función mensaje_irc() disponible desde un programa hecho en Visual Basic, entonces debemos crear una función.

Un programa en lenguaje C no se puede incluir en Visual Basic directamente, así que decidimos crear una Librería de enlace dinámico (DLL) que es un archivo que contiene funciones que se pueden utilizar desde otro lenguaje que soporte su uso. No nos preocupamos de cómo funcionan estas funciones, sólo nos interesa qué parámetros reciben y qué resultado generan. El código en lenguaje C fue compilado en Visual C++ generando la DLL.

Desde Visual Basic sólo llamamos a la función que nos interesa y así sabemos si un mensaje está bien construido o no y, además, obtenemos el prefijo, el comando y los parámetros en cadenas por separado.

Si encontramos que el comando no existe, inmediatamente contestamos con un mensaje de error, indicando que el comando no es conocido. Si el comando es válido, pasamos a la segunda etapa para analizar los parámetros proporcionados.

8.4 Análisis de parámetros

Algunos comandos tienen una estructura muy compleja para los parámetros que reciben y otros ni siquiera tienen parámetros. Debido a estas diferencias es muy difícil crear una sola rutina que pudiera identificar si todo el mensaje está correctamente construido.

Al separar de esta manera el análisis, podemos elegir en esta segunda etapa si hacemos un análisis utilizando nuestra herramienta compatible con Lex y Yacc o si podemos hacer un análisis muy simple con sólo unas cuantas funciones de cadenas. Así, para cada comando se eligió la manera más sencilla de analizar la cadena de parámetros. Si la cadena es muy complicada, es mejor analizarla por medio de un programa hecho en nuestra herramienta por las facilidades que nos otorga.

Si la estructura de los parámetros es correcta, se procede con la ejecución de las acciones que corresponden con la atención que debe darse al comando recibido. Llegar a este punto no significa que el mensaje sea correcto, sólo significa que está sintácticamente bien construido, pero su significado tal vez implique realizar tareas imposibles, como el enviar un mensaje a un canal inexistente o tratar de entrar a un canal sin haber sido invitado, cuando este canal tiene habilitada la bandera de sólo invitados.

La siguiente es la descripción de la gramática de algunos de los parámetros:

```
target     =  nickname / server
msgtarget  =  msgto *( "," msgto )
msgto      =  channel / ( user [ "%" host ] "@" servername )
msgto      =/ ( user "%" host ) / targetmask
msgto      =/ nickname / ( nickname "!" user "@" host )
channel    =  ( "#" / "+" / ( "!" channelid ) / "&" ) chanstring
              [ ":" chanstring ]
servername =  hostname
```

```
host       =  hostname / hostaddr
hostname   =  shortname *( "." shortname )
shortname  =  ( letter / digit ) *( letter / digit / "-" )
              *( letter / digit )
                ; as specified in RFC 1123 [HNAME]
hostaddr   =  ip4addr / ip6addr
ip4addr    =  1*3digit "." 1*3digit "." 1*3digit "." 1*3digit
ip6addr    =  1*hexdigit 7( ":" 1*hexdigit )
ip6addr    =/ "0:0:0:0:0:" ( "0" / "FFFF" ) ":" ip4addr
nickname   =  ( letter / special ) *8( letter / digit / special / "-" )
targetmask =  ( "$" / "#" ) mask
                ; see details on allowed masks in section 3.3.1
chanstring =  %x01-07 / %x08-09 / %x0B-0C / %x0E-1F / %x21-2B
chanstring =/ %x2D-39 / %x3B-FF
                ; any octet except NUL, BELL, CR, LF, " ", "," and ":"
channelid  = 5( %x41-5A / digit )   ; 5( A-Z / 0-9 )
```

El significado de cada parámetro está definido por su posición dentro de la cadena y por el comando al que pertenecen los parámetros.

8.5 Ejecución de funciones

Este módulo realiza todas las acciones sobre las listas de canales, usuarios y servicios y sobre todos los datos con los que cuenta el servidor. Es en esta etapa en la que podremos darnos cuenta de si el mensaje es realmente correcto o no. Si no se puede ejecutar el comando por alguna causa, se deberá generar el correspondiente mensaje de error.

Cada mensaje implica un algoritmo diferente que está sujeto a las especificaciones del protocolo. Es en esta parte en donde se reflejan todas las restricciones y requisitos que deben cumplir los participantes en una conferencia de IRC.

Los mensajes de respuestas y errores son enviados al módulo de envío de paquetes de datos para que alcancen su destino en los clientes correspondientes. Durante el transcurso del proceso completo, siempre se tiene en cuenta qué cliente envió el mensaje para realizar las acciones correctas.

Las acciones que se realizan para cada mensaje están separadas en módulos independientes para permitir su desarrollo en paralelo y para organizar el código. Cada módulo se guarda en un archivo diferente para facilitar los cambios que se realicen.

8.6 Interface gráfica

La figura 8.1 muestra la pantalla principal del sistema y también la única que existe. Muestra los mensajes que recibe y envía el servidor, los clientes conectados, los canales creados y algunas de las estadísticas más comunes. Su propósito es más de depuración del

servidor que de uso en la operación normal del sistema. Toda esta información se puede obtener por medio de un cliente IRC que tenga los privilegios necesarios.

El botón “Actualizar” sirve para mostrar los datos actuales porque la pantalla no se refresca automáticamente, esto es para generar menos carga en los procesos de atención de los mensajes. Sólo la lista de mensajes se actualiza automáticamente.

La lista de mensajes indica desde qué conexión llegó un mensaje y hacia qué conexión se envían las respuestas. Para cada conexión se indica la dirección IP.

La pantalla muestra el nombre del servidor, una caja de texto con los mensajes que se están enviando a todos los clientes y una etiqueta con el número de clientes conectados actualmente al servidor y que además están registrados. Se incluyen también, por razones de administración, dos contadores del número de bytes enviados y recibidos por el servidor.

Existen dos listas en las cuales se pueden ver los nicknames de los usuarios registrados actualmente y de los canales que han sido creados por estos usuarios.

El botón “Salir” termina la ejecución del servidor IRC.

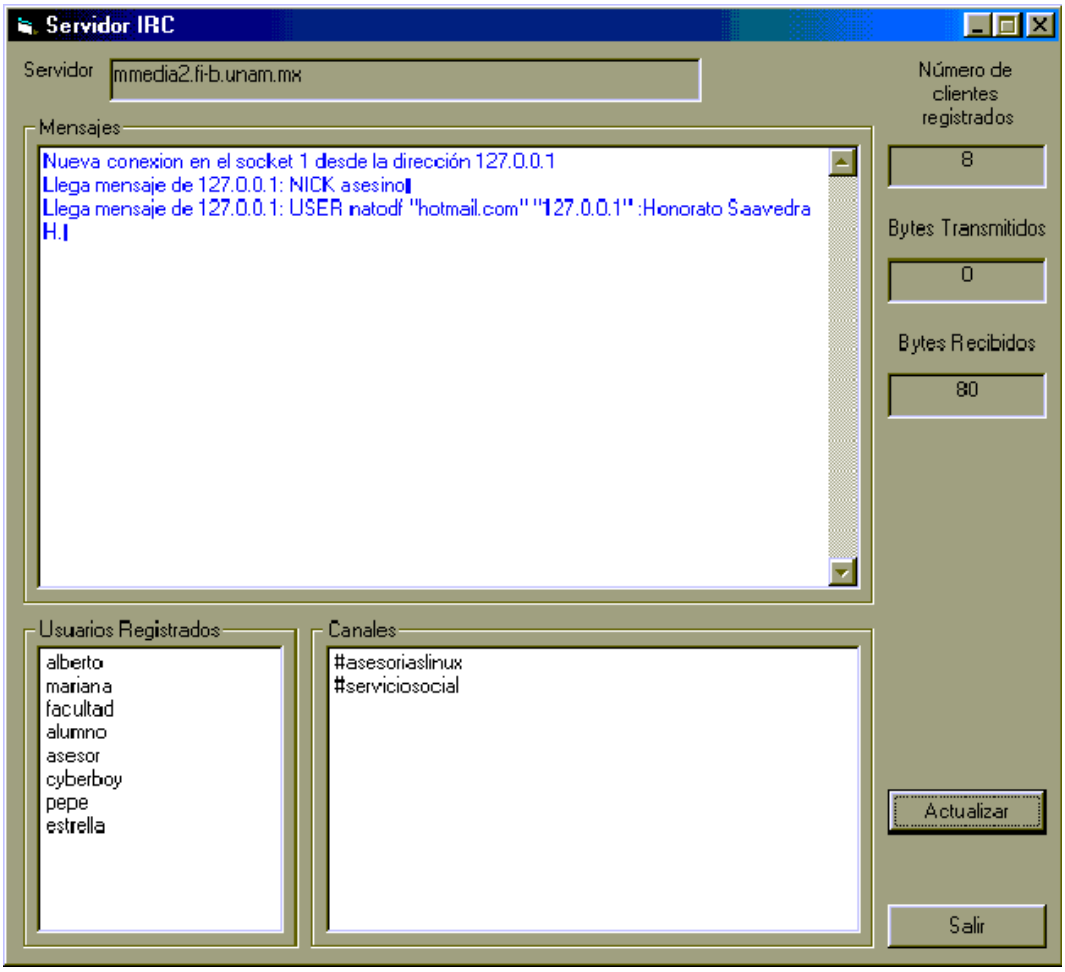

Figura 8.1

8.7 Pruebas

Se sometió el sistema a las siguientes pruebas:

8.7.1 Compatibilidad con varios clientes IRC

Se probó el funcionamiento con los clientes mIRC, Xchat, mozilla IRC. Todos los clientes funcionaron correctamente con el servidor. Los clientes corrían en plataforma Windows y Linux. En estas pruebas se detectó que el mensaje PING era diferente en cada uno de ellos, pero la estructura general era parecida y el servidor es capaz de contestar a las diferentes variaciones.

8.7.2 Comparación de respuestas generadas contra otros servidores

Cada mensaje se comparó en estructura con los emitidos por varios servidores IRC que están funcionando continuamente y que mantienen cientos de conexiones simultáneas. Se comprobó que se respeta el protocolo y que nuestro servidor genera los mismos mensajes con la misma estructura. La Tabla 8.1 muestra un ejemplo.

servidor	mmedia2.fi-b.unam.mx
mensaje	PRIVMSG #micanal :hola asesor

servidor	jailhouse.ma.us.acestar.org
mensaje	:AceRelay!outsider@relay.key2peace.org PRIVMSG #micanal :hola asesor

Tabla 8.1

Los dos mensajes envían el comando PRIVMSG. El prefijo es opcional así que los dos mensajes realizan la misma operación. El canal es #micanal y el mensaje es hola asesor.

8.7.3 Conexión con 20 clientes simultáneos

Se realizó una prueba en la que se conectaron 20 clientes IRC simultáneamente. Cada cliente tenía la tarea de crear canales, borrarlos y enviar mensajes de manera que se pudiera simular un volumen de información en tránsito similar a la que podrían generar 20 clientes IRC de manera común. No se detectaron fallas ni respuestas retrasadas o lentas. Todos los clientes pudieron comunicarse sin ningún problema.

8.7.4 Resultados

Como resultado de estas pruebas se comprobó el cumplimiento del protocolo IRC por parte del servidor y la posibilidad de utilizar los clientes más comunes.

Los requerimientos del sistema son la base fundamental para medir la calidad del programa. La inconsistencia con los requerimientos provoca que la calidad del producto sea deficiente. El programa construido cumple con el estándar IRC en lo que corresponde a comunicación con los clientes. La comunicación con otros servidores está fuera del alcance de este proyecto.

Los mensajes que se pueden utlizar en el servidor son los siguientes y constituyen la interface que cumple con el protocolo IRC.

número	Comando
1	NICK
2	USER
3	OPER
4	MODE
5	SERVICE
6	QUIT
7	JOIN
8	PART
9	TOPIC
10	NAMES
11	LIST
12	AWAY
13	KICK
14	PRIVMSG
15	NOTICE
16	MOTD
17	LUSERS
18	INVITE
19	WHO
20	WHOIS
21	USERHOST
22	KILL
23	PING
24	PONG
25	ERROR

Tabla 8.2

En total son 25 mensajes que permiten realizar las funciones básicas de un servidor IRC y son suficientes para los propósitos de este proyecto en particular. La integración del sistema se realizó de manera descendente, así que se construyeron los dos módulos principales y se agregó un módulo aparte para cada mensaje según se fueron programando.

Capítulo 9

Conclusiones

Como resultado del desarrollo de este sistema podemos concluir lo siguiente:

La elección de la metodología fue acertada con respecto a la suposición de que los requisitos del sistema no cambiarían, ya que se siguió exactamente la especificación del protocolo de los RFC's. No se decidió agregar ningún otro mensaje ni modificar los que ya estaban definidos para lograr la mayor compatibilidad posible con los clientes existentes.

La división del trabajo en etapas como el análisis, diseño e implementación permitió conservar el orden necesario para crear un producto de calidad que cumple con los requerimientos necesarios y con las características que se identificaron como más importantes, además de permitir realizar cambios o mejoras de una manera más ordenada. Durante la implementación se constataron las ventajas de la utilización de módulos al lograrse la eliminación de subsistemas muy complejos y difíciles de programar.

Los conocimientos sobre herramientas de programación de scanners, parsers y redes de computadoras fueron fundamentales en la programación del sistema. Las partes más complicadas del desarrollo fueron las que tienen que ver con el manejo de las conexiones de red y con la identificación de los comandos y parámetros dentro de un mensaje.

El análisis realizado en el capítulo 6 muestra que el protocolo IRC se trata de una especificación extensa, debido al número de mensajes y respuestas existentes dentro del protocolo y que la programación debería separarse en módulos para repartir el trabajo entre los integrantes del equipo de desarrollo. Esta división se pudo lograr de manera eficiente ya que el diseño permitió trabajar con funciones bien definidas e independientes unas de otras.

Con respecto a lo aprendido por el equipo de desarrollo, se manejaron conceptos de programación en redes TCP/IP, herramientas de programación de scanners y parsers como Flex y Bison y, lo más importante, fue trabajar con una especificación perfectamente definida, a la cual se debía apegar el desarrollo, aún cuando la implementación era totalmente libre. Un punto a favor de este sistema es que a pesar de los cambios que se hagan al protocolo IRC, pasará mucho tiempo antes de que la mayoría de los clientes IRC ya no sean capaces de comunicarse a través de este servidor, porque el protocolo ya no cambia drásticamente.

Con respecto a los objetivos, se logró cumplir con la compatibilidad que era el principal obstáculo a vencer y se logró la construcción de un sistema suficientemente rápido como para llevar a cabo una conversación fluida en tiempo real a través de la Internet. Que el objetivo de crear un nuevo canal de comunicación para el Laboratorio de Multimedia e Internet se logre, se tendrá que verificar a través del tiempo al adoptarse o no como un recurso ordinario de comunicación.

La implantación del sistema no tuvo ninguna dificultad y se espera que ahora que el programa ha sido probado, sea utilizado de manera regular por los integrantes del Laboratorio de Multimedia e Internet.

La solución a la que se llegó para lograr la atención oportuna por parte de los integrantes del Laboratorio de Multimedia e Internet a las asesorías sin tener que estar

siempre atentos en una computadora, fue generar un sonido de aviso suficientemente fuerte para que se escuche en todo el laboratorio. Al percibir este sonido, un integrante del laboratorio que esté destinado a dar asesoría toma una computadora y se conecta al servidor de IRC. El aviso se genera cuando el cliente que desea usar el servicio crea un canal llamado #asesoria. Al conectarse el cliente al servidor se le envía un mensaje en el cual se le indica el horario en el cual puede utilizar el servicio de asesoría y cómo puede avisar que ya está conectado y listo para hacer sus preguntas.

La implementación del sistema propuesto permite ahora realizar la tarea de asesorar a otros estudiantes sobre los distintos temas relacionados con la ingeniería de una forma muy sencilla que complementa el esquema tradicional que se ha practicado hasta ahora. Con esta herramienta de comunicación aumenta la infraestructura existente que está basada en cursos, información por medio de la Web y correo electrónico. El uso del sistema se publicitará por medio de la página Web del laboratorio.

A pesar de que el método de brindar asesoría por medio de un servidor de IRC puede resultar difícil de usar debido a la necesidad de atención continua por parte de los asesores, en el Laboratorio de Multimedia e Internet se cuenta con las condiciones necesarias para hacerlo porque la mayor parte del día se encuentran varias personas con la capacidad suficiente para ayudar a los estudiantes que así lo requieren o en los casos difíciles orientarlos hacia la referencia o persona más adecuada para resolver un problema específico. El sistema ya está siendo utilizado por los integrantes del laboratorio de manera regular y se espera que cada vez se integre más a las herramientas de las que disponemos.

Se probó que el sistema es compatible con IRC, así que se aprovechan los programas cliente existentes en las distintas plataformas de cómputo y se cuenta con una forma de comunicación escrita de tiempo real gracias a que la información que se transmite es muy pequeña y los retardos generados por la internet son aceptables en la mayoría de los casos.

Apéndice A

Detalle de los mensajes

Este apéndice incluye las descripciones de cada mensaje reconocido por el protocolo IRC.

Cuando se da la respuesta ERR_NOSUCHSERVER, significa que el destino del mensaje no se puede encontrar. El servidor no debe mandar ninguna otra respuesta después de este error para cualquier comando.

El servidor al cual se conecta el cliente debe revisar el mensaje completo y regresar cualquier error apropiado.

Si se encuentran múltiples parámetros, entonces cada uno debe ser revisado y las respuestas apropiadas se deben enviar al cliente. En el caso de mensajes incorrectos que usan una lista de parámetros separada por comas, una respuesta debe enviarse por cada elemento.

La información referente a los mensajes fue obtenida de los RFC's y se incluye para aclarar varias inconsistencias del protocolo detectadas en los servidores y clientes de IRC de más uso en la actualidad.

Registro de la conexión

Los comandos descritos en esta sección son usados para registrar una conexión con un servidor IRC y para desconectarse correctamente.

Un comando PASS no es requerido para la conexión de un cliente, pero debe preceder a la combinación NICK/USER para una conexión de usuario o al comando SERVICE para una conexión de servicio. El orden recomendado es el siguiente:

1 Mensaje PASS	1 Mensaje PASS
2 Mensaje NICK	2 Mensaje SERVICE
3 Mensaje USER	

Al tener éxito, el cliente recibirá un mensaje RPL_WELCOME (para los usuarios) o RPL_YOUSERVICE (para los servicios) indicando que la conexión está ahora registrada. El mensaje de respuesta debe contener el identificador del cliente completo con el cual se registró.

Comando	**PASS**
Parámetros	<password>
Descripción	El comando PASS es usado para establecer un password de conexión. El password opcional puede y debe ser establecido antes de cualquier intento de registrar la conexión. Actualmente esto requiere que un usuario envíe un comando PASS antes de mandar la combinación NICK/USER. **Este mensaje NO se implementará en nuestra versión.**
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_ALREADYREGISTRED
Ejemplos	PASS passwordsecreto

Comando	**NICK**
Parámetros	<nickname>
Descripción	El comando NICK es usado para dar a un usuario un nickname o para cambiar el existente
Respuestas Numéricas	ERR_NONICKNAMEGIVEN ERR_ERRONEUSNICKNAME ERR_NICKNAMEINUSE ERR_NICKCOLLISION ERR_UNAVAILRESOURCE ERR_RESTRICTED
Ejemplos	NICK Wiz ; Introducción de un nuevo nickname "Wiz" si la sesión aún no se registra, o un usuario cambiando su nickname a "Wiz" :WiZ!mmedia2@fi-b.unam.mx NICK Kilroy ; El servidor avisando que WiZ cambió su nickname a Kilroy.

Comando	**USER**
Parámetros	<usuario> <modo> <no usado> <nombre real>
Descripción	El comando USER se utiliza al inicio de la conexión para especificar el nombre de usuario, nombre del host y el nombre real de un nuevo usuario. El parámetro <modo> debe ser numérico y puede usarse para ajustar modos de usuario cuando se registran en el servidor. Este parámetro es una máscara de bits con dos bits que tienen significado: si el bit 2 está encendido el modo de usuario 'w' se establecerá y si el bit 3 está encendido se establecerá el modo 'i'. Se cuenta desde el bit 0. El parámetro <nombre real> puede contener espacios.
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_ALREADYREGISTRED
Ejemplos	USER guest 0 * :Juan Perez ; Un usuario registrándose a sí mismo con el nombre de usuario "guest" y el nombre real "Juan Perez". USER guest 8 * :Juan Perez ; Un usuario registrándose a sí mismo con el nombre de usuario "guest" y el nombre real "Juan Perez" y pidiendo ser un usuario invisible.

Modo de usuario

Comando	OPER
Parámetros	<nombre> <password>
Descripción	Un usuario normal utiliza el comando OPER para obtener privilegios de operador. La combinación de <nombre> y <password> es requerida para ganar los privilegios. Si tiene éxito, el usuario recibirá un mensaje MODE indicando los nuevos modos de usuario.
Respuestas Numéricas	ERR_NEEDMOREPARAMS RPL_YOUREOPER ERR_NOOPERHOST ERR_PASSWDMISMATCH
Ejemplos	OPER yomero secretin ; Intento de registrarse como operador utilizando un nombre de usuario "yomero" y "secretin" como password.

Comando	**MODE**
Parámetros	<nickname> *(("+" / "-") *("i" / "w" / "o" / "O" / "r"))
Descripción	Los modos de usuario típicamente son cambios que afectan cómo es visto el cliente por los otros o qué mensajes "extra" se le enviarán. Un comando MODE de usuario sólo debe aceptarse si el que envía el mensaje y el nickname dado como parámetro son el mismo. Si no se da otro parámetro, entonces el servidor regresa la configuración actual del nickname. Los modos disponibles son los siguientes:

	a – el usuario está marcado como retirado i – marca a un usuario como invisible w – el usuario recibe wallops r – conexión de usuario restringida o – bandera de operador O – bandera de operador local s – marca a un usuario como receptor de noticias del servidor Otros modos adicionales pueden estar disponibles más adelante. La bandera 'a' no debe ser alterada por el usuario usando el comando MODE, en su lugar se requiere usar el comando AWAY. Si un usuario intenta hacerse a sí mismo operador usando la bandera "+o" o "+O", el intento debe ser ignorado para que los usuarios no burlen los mecanismos de autenticación del comando OPER. No hay restricción en abandonar los privilegios de operador usando "-o" o "-O". Si un usuario trata de hacerse a sí mismo "no restringido" usando la bandera "-r", el intento debe ser ignorado. No hay restricción en usar "+r". Esta bandera es establecida por el servidor después de la conexión por razones administrativas. Mientras las restricciones impuestas se dejan a la implementación, es típico que un usuario restringido no pueda cambiar nicknames ni hacer uso del estatus de operador de canal. La bandera 's' es obsoleta pero puede ser usada aún.
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_USERSDONTMATCH ERR_UMODEUNKNOWNFLAG RPL_UMODEIS
Ejemplos	MODE WiZ –w ; Comando de WiZ para deshabilitar la recepción de mensajes WALLOPS. MODE Juan +i ; Comando de Juan para hacerse invisible. MODE WiZ –o ; WiZ removiendo su estatus de operador.

Comando	SERVICE
Parámetros	<nickname> <reservado> <distribucion> <tipo> <reservado> <informacion>
Descripción	El comando SERVICE se usa para registrar un nuevo servicio. Los parámetros del comando especifican el nickname del servicio, la distribución, tipo e información de un nuevo servicio. El parámetro <distribucion> es usado para especificar la visibilidad de un servicio. El servicio puede ser conocido sólo por servidores que tienen un nombre que corresponde con la máscara de distribución. Para que un servidor que corresponde tenga conocimiento del servicio, la ruta de red entre ese servidor y el servidor en el que está conectado el servicio debe estar compuesta de servidores que corresponden con la máscara de distribución. El parámetro <tipo> está reservado para uso futuro.
Respuestas Numéricas	ERR_ALREADYREGISTRED ERR_NEEDMOREPARAMS ERR_ERRONEUSNICKNAME RPL_YOURESERVICE RPL_YOURHOST RPL_MYINFO
Ejemplos	SERVICE dict * *.fr 0 0 :Diccionario de frances ; registro de un servicio con nombre "dict". Este servicio sólo estará disponible en servidores cuyo nombre corresponda con "*.fr".

Comando	QUIT
Parámetros	[<Mensaje de salida>]
Descripción	Una sesión de cliente es terminada con un mensaje de salida. El servidor acepta esto mandando un mensaje ERROR al cliente.
Respuestas Numéricas	Ninguna.
Ejemplos	QUIT :Me fui a comer ; Formato de mensaje preferido. :syrk!kalt@millennium.stealth.net QUIT :Me fui a comer ; El usuario syrk salió de IRC y se fue a comer.

Comando	SQUIT
Parámetros	<servidor> <comentario>
Descripción	El comando SQUIT está disponible sólo para los operadores. Se usa para desconectar ligas entre servidores. Los servidores también pueden generar mensajes SQUIT en condiciones de error. Un mensaje SQUIT se puede enviar también a una conexión de servidor remoto. En este caso, el mensaje SQUIT simplemente será enviado al servidor remoto sin afectar a los servidores que se encuentran entre el operador y el servidor remoto. El parámetro <comentario> debe darlo cualquier operador que ejecute SQUIT para un servidor remoto. El servidor al cual se le ordenó desconectarse genera mensajes WALLOPS con <comentario> incluido de manera que otros usuarios sepan de la razón de esta acción. **Este mensaje NO se implementará en nuestra versión.**
Respuestas Numéricas	ERR_NOPRIVILEGES ERR_NOSUCHSERVER ERR_NEEDMOREPARAMS
Ejemplos	SQUIT tolsun.oulu.fi :Conexion lenta ; Comando para desconectar al servidor tolson.oulu.fi con el comentario “Conexion lenta”. :Trillian SQUIT cm22.eng.umd.edu :Servidor fuera de control ; Comando generado por Trillian para desconectar a "cm22.eng.umd.edu" de la red con el comentario “Servidor fuera de control”.

Operaciones de canales

Este grupo de mensajes se relaciona con la manipulación de canales, sus propiedades (modos de canal) y sus contenidos (usuarios típicos). Por esta razón, estos mensajes no deben hacerse disponibles a los servicios.

Todos estos mensajes son peticiones que pueden ser otorgadas o no por el servidor. El servidor debe enviar una respuesta informando al usuario si su petición fue aceptada, rechazada o generó un error. Cuando el servidor acepta la petición, el mensaje es regresado al usuario (eventualmente reformateado) usando como prefijo el nickname del mismo usuario. Las reglas que gobiernan cómo se manejan los canales son aplicadas por los servidores.

Comando	JOIN
Parámetros	(<canal> *("," <canal>) [<clave> *("," <clave>)]) / "0"
Descripción	El comando JOIN lo utiliza un usuario para pedir empezar a escuchar en el canal especificado. Los servidores deben ser capaces de analizar los argumentos en la forma de lista, pero no deben usar listas cuando se envíen mensajes JOIN a los clientes. Una vez que el usuario se ha unido a un canal, recibe información acerca de todos los comandos que su servidor recibe y que afectan al canal. Esto incluye JOIN, MODE, KICK, PART, QUIT y por supuesto PRIVMSG/NOTICE. Esto permite a los miembros del canal llevar el rastro de los otros miembros del canal, así como de los modos del canal. Si un JOIN es exitoso, el usuario recibe un mensaje JOIN como confirmación y entonces se le envía el tópico del canal (usando RPL_TOPIC) y la lista de usuarios que están en el canal (usando RPL_NAMREPLY), la cual debe incluir al usuario que se unió. Nótese que este mensaje acepta un argumento especial "0" que es una petición especial para abandonar todos los canales de los cuales el usuario es miembro actualmente. El servidor procesará este mensaje como si el usuario hubiera mandado un comando PART para cada canal del cual es miembro.
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_BANNEDFROMCHAN ERR_INVITEONLYCHAN ERR_BADCHANNELKEY ERR_CHANNELISFULL ERR_BADCHANMASK ERR_NOSUCHCHANNEL ERR_TOOMANYCHANNELS ERR_TOOMANYTARGETS ERR_UNAVAILRESOURCE RPL_TOPIC
Ejemplos	JOIN #asesorias ; Comando para unirse al canal #asesorias. JOIN &asesorias xxxx ; Comando para unirse al canal #asesorias usando la clave "xxxx". JOIN #asesorias,&bar abc ; Comando para unirse al canal #asesorias usando la clave "abc" y al canal &bar sin utilizar clave. JOIN #asesorias,#bar abc,def ; Comando para unirse al canal #asesorias usando la clave "abc" y al canal &bar utilizando la clave "def". JOIN #asesorias,#bar ; Comando para unirse a los canales #asesorias y #bar. JOIN 0 ; Abandona todos los canales en los que se encuentra actualmente. :WiZ!jto@tolsun.oulu.fi JOIN #Dimension_desconocida ; mensaje JOIN de WiZ en el canal #Dimension_desconocida

Comando	PART
Parámetros	<canal> *("," < canal >) [<Mensaje de salida>]
Descripción	El comando PART causa que el usuario que envía el mensaje sea removido de la lista de miembros activos de los canales dados en la lista de parámetros. Si se da un <Mensaje de salida>, este se enviará en lugar del mensaje por default (el nickname). Esta petición siempre es aceptada por el servidor. Los servidores deben ser capaces de analizar los argumentos en la forma de lista, pero no deben usar listas cuando envíen mensajes PART a los clientes.
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL ERR_NOTONCHANNEL
Ejemplos	PART #Dimension_desconocida

	; Comando para abandonar el canal "#Dimension_desconocida" PART #operadores,&grupo5 ; Comando para abandonar los canales "#operadores" y "&grupo5". :WiZ!jto@tolsun.oulu.fi PART #playzone :He perdido ; Usuario WiZ abandonando el canal "#playzone" con el mensaje "He Perdido".

Modo de canal

Comando	**MODE**
Parámetros	<canal> *(("-" / "+") *<modos> *<parametros de modo>)
Descripción	El comando MODE permite que los usuarios puedan consultar y cambiar las características de un canal. Note que hay un máximo de tres cambios por comando para los modos que toman un parámetro (porque el número total de parámetros permitidos son 15).
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_KEYSET ERR_NOCHANMODES ERR_CHANOPRIVSNEEDED ERR_USERNOTINCHANNEL ERR_UNKNOWNMODE RPL_CHANNELMODEIS RPL_BANLIST RPL_ENDOFBANLIST RPL_EXCEPTLIST RPL_ENDOFEXCEPTLIST RPL_INVITELIST RPL_ENDOFINVITELIST RPL_UNIQOPIS
Ejemplos	MODE #Finnish +imI *!*@*.fi ; Comando para hacer al canal #Finnish moderado y de sólo invitación e invitando automáticamente a los usuarios que tenga un nombre de host que corresponda con *.fi. MODE #Finnish +o Kilroy ; Comando para dar privilegios de operador de canal a Kilroy en el canal #Finnish. MODE #Finnish +v Wiz ; Comando para permitir a WiZ hablar en #Finnish. MODE #Fins -s ; Comando para remover la bandera de 'secreto' del canal #Fins. MODE #42 +k oulu ; Comando para establecer la clave del canal “#42” a "oulu". MODE #42 -k oulu ; Comando para remover la clave del canal “#42”. MODE #operadores +l 10 ; Comando para establecer el límite para el número de usuarios en el canal "#operadores" a 10. :WiZ!jto@tolsun.oulu.fi MODE #operadores -l ; El usuario "WiZ" removiendo el límite para el número de usuarios en el canal "#operadores". MODE &oulu +b ; Comando para listar las máscaras de rechazo impuestas al canal "&oulu".

	MODE &oulu +b *!*@* ; Comando para prevenir que cualquier usuario se una al canal. MODE &oulu +b *!*@*.edu +e *!*@*.bu.edu ; Comando para prevenir que cualquier usuario de un host *.edu pueda unirse, excepto si corresponde con *.bu.edu. MODE #bu +be *!*@*.edu *!*@*.bu.edu ; Comando para que cualquier usuario de un host *.edu pueda unirse, excepto si corresponde con *.bu.edu MODE #meditacion e ; Comando para listar máscaras de excepción para el canal "#meditacion". MODE #meditacion I ; Comando para listar las máscaras de invitación para el canal "#meditacion". MODE !12345ircd O ; Comando para preguntar quién es el creador del canal "!12345ircd".

Comando	TOPIC
Parámetros	<canal> [<topico>]
Descripción	El comando TOPIC se usa para cambiar o ver el tópico de un canal. El tópico del canal <canal> es regresado si no se da el parámetro <topico>. Si el parámetro <topico> está presente, el tópico para ese canal será cambiado si esta acción está permitida para el usuario que lo pide. Si el parámetro <topico> es una cadena vacía, el tópico para ese canal será removido.
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_NOTONCHANNEL RPL_NOTOPIC RPL_TOPIC ERR_CHANOPRIVSNEEDED ERR_NOCHANMODES
Ejemplos	:WiZ!jto@tolsun.oulu.fi TOPIC #test :Nuevo tópico ;Usuario Wiz estableciendo el tópico. TOPIC #test :otro topico ; Comando que establece el tópico en #test a "otro topico". TOPIC #test : ; Comando para eliminar el tópico en #test. TOPIC #test ; Comando para revisar el tópico de #test.

Comando	NAMES
Parámetros	[<canal> *("," <canal>) [<destino>]]
Descripción	Al usar el comando NAMES, un usuario puede listar todos los nicknames que son visibles para él. El parámetro <canal> especifica de qué canales regresar información. No hay respuesta de error para los nombres de canal erróneos. Si no se da el parámetro <canal>, una lista de todos los canales y sus ocupantes es regresada. Al final de esta lista, se da otra lista de usuarios visibles que no están en ningún canal o que están en un canal no visible y se listan como pertenecientes al canal "*". Si el parámetro <destino> se especifica, la petición es reenviada a este servidor para que genere la respuesta. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	ERR_TOOMANYMATCHES ERR_NOSUCHSERVER RPL_NAMREPLY RPL_ENDOFNAMES
Ejemplos	NAMES #Dimension_desconocida,#42 ; Comando para listar los usuarios visibles en #Dimension_desconocida y #42 NAMES ; Comando para listar los usuarios

	y canales visibles.

Comando	LIST
Parámetros	[<canal> *("," <canal>) [<destino>]]
Descripción	El comando LIST es usado para listar los canales y sus tópicos. Si el parámetro <canal> se usa, sólo el estado de ese canal es desplegado. Si el parámetro <destino> se especifica, la petición es reenviada a ese servidor el cual generará la respuesta. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	ERR_TOOMANYMATCHES ERR_NOSUCHSERVER RPL_LIST RPL_LISTEND
Ejemplos	LIST ; Comando para listar todos los canales. LIST #Dimension_desconocida,#42 ; Comando para listar los canales #Dimension_desconocida y #42

Comando	INVITE
Parámetros	<nickname> <canal>
Descripción	El comando INVITE es usado para invitar a un usuario a un canal. El parámetro <nickname> es el nickname de la persona a ser invitada al canal <canal>. No se requiere que el canal al cual es invitado el usuario exista o sea un canal válido. Como sea, si el canal existe, sólo los miembros del canal pueden invitar a otros usuarios. Cuando el canal tiene la bandera de sólo invitación, sólo los operadores de canal pueden ejecutar un comando INVITE. Sólo el usuario que invita y el usuario invitado recibirán notificación de la invitación. Otros miembros del canal no son notificados. (Esto es diferente a los cambios MODE y es ocasionalmente fuente de problemas entre los usuarios.)
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_NOSUCHNICK ERR_NOTONCHANNEL ERR_USERONCHANNEL ERR_CHANOPRIVSNEEDED RPL_INVITING RPL_AWAY
Ejemplos	:Angel!wings@irc.org INVITE Wiz #Dust ; Mensaje a WiZ cuando ha sido invitado por el usuario Angel al canal #Dust INVITE Wiz #Dimension_desconocida ; Comando para invitar a WiZ a #Dimension_desconocida

Comando	KICK
Parámetros	<canal> *("," <canal>) <usuario> *("," <usuario>) [<comentario>]
Descripción	El comando KICK puede ser usado para pedir remover forzosamente a un usuario de un canal. Causa que el usuario <usuario> salga del canal <canal> a la fuerza. Para que el mensaje sea sintácticamente correcto, debe haber un parámetro canal y múltiples parámetros usuario o tantos parámetros canal como parámetros usuario. Si un <comentario> es dado, éste será enviado en lugar del mensaje por default, que es el nickname del usuario ejecutando KICK. El servidor no debe mandar mensajes KICK con múltiples canales o usuarios a los clientes, Esto es necesario para mantener compatibilidad con clientes antiguos.
Respuestas Numéricas	ERR_NEEDMOREPARAMS ERR_NOSUCHCHANNEL ERR_BADCHANMASK ERR_CHANOPRIVSNEEDED ERR_USERNOTINCHANNEL ERR_NOTONCHANNEL
Ejemplos	KICK &Melbourne Matias ; Comando para sacar a Matias de &Melbourne.

	KICK #Finnish Juan :Speaking English ; Comando para sacar a Juan de #Finnish usando "Speaking English" como la razón (comentario). :WiZ!jto@tolsun.oulu.fi KICK #Finnish Juan ; mensaje KICK de WiZ en el canal #Finnish para remover a Juan del canal.

Enviando mensajes

El propósito principal del protocolo IRC es dar una base a los clientes para comunicarse entre ellos. PRIVMSG, NOTICE y SQUERY son los únicos mensajes disponibles que realmente desarrollan envío de mensajes de texto de un cliente a otro. El resto sólo lo hace posible y trata de que suceda de una manera confiable y estructurada.

Comando	PRIVMSG
Parámetros	<destino de mensaje> <texto a ser enviado>
Descripción	PRIVMSG es usado para enviar mensajes privados entre usuarios, así como para mandar mensajes a los canales. <destino de mensaje> es usualmente el nickname del recipiente del mensaje o el nombre de un canal. El parámetro <destino de mensaje> también puede ser una máscara de host (#<mascara>) o una máscara de servidor ($<mascara>). En ambos casos el servidor sólo enviará el PRIVMSG a aquellos que tengan un servidor o host que corresponda con la máscara. La máscara debe tener al menos un punto '.' en ella y sin comodines al final del último punto '.'. Esto es para prevenir que las personas envíen mensajes a "#*" o a "$*", lo cual enviará un mensaje a todos los usuarios. Los comodines son los carácteres '?' y '*'. Esta extensión al comando PRIVMSG sólo está disponible para los operadores.
Respuestas Numéricas	ERR_NORECIPIENT ERR_NOTEXTTOSEND ERR_CANNOTSENDTOCHAN ERR_NOTOPLEVEL ERR_WILDTOPLEVEL ERR_TOOMANYTARGETS ERR_NOSUCHNICK RPL_AWAY
Ejemplos	:Angel!wings@irc.org PRIVMSG Wiz :¿Estás recibiendo este mensaje? ; Mensaje de Angel a Wiz. PRIVMSG Angel :si, lo estoy recibiendo ; Comando para mandar un mensaje a Angel. PRIVMSG gato@tolsun.oulu.fi :Hola ; Comando para enviar un mensaje a un usuario en el servidor tolsun.oulu.fi con nombre de usuario "gato". PRIVMSG carlos%almoloya.stealth.net@irc.stealth.net :Me ves y me oyes? ; Mensaje a un usuario en el servidor irc.stealth.net con nombre de usuario "carlos", y conectado desde el host almoloya.stealth.net. PRIVMSG carlos%almoloya.stealth.net :Necesitas dinero? ; Mensaje a un usuario en el servidor local con nombre de usuario "carlos", y conectado desde el host almoloya.stealth.net.

	PRIVMSG Wiz!gato@tolsun.oulu.fi :Hola ! ; Mensaje al usuario con nickname Wiz que está conectado desde el host tolsun.oulu.fi y tiene el nombre de usuario "gato". PRIVMSG $*.fi :El servidor tolsun.oulu.fi se va a apagar. ; Mensaje a todos en un servidor que tiene un nombre que corresponde con *.fi. PRIVMSG #*.edu :La red está en mantenimiento ; Mensaje a todos los usuarios que vienen de un host que corresponde con *.edu.

Comando	NOTICE
Parámetros	<destino de mensaje> <texto>
Descripción	El comando NOTICE es usado igual que PRIVMSG. La diferencia entre NOTICE y PRIVMSG es que las respuestas automáticas nunca se deben enviar en respuesta a un mensaje NOTICE. Estas reglas también aplican a los servidores, no deben enviar ninguna respuesta de error al cliente al recibir un mensaje NOTICE. El objetivo de esta regla es evitar ciclos entre clientes que envían “algo” automáticamente en respuesta a “algo” recibido. Este comando está disponible para los servicios y usuarios. Típicamente es usado por los servicios y autómatas (clientes con inteligencia artificial u otro programa de control interactivo). Vea PRIVMSG para más detalles sobre respuestas y ejemplos.
Respuestas Numéricas	
Ejemplos	

Consultas al servidor

El grupo de comandos de consultas al servidor ha sido diseñado para regresar información acerca de cualquier servidor que está conectado a la red.

En estas consultas, cuando un parámetro <destino> aparece, las máscaras usualmente son válidas. Para cada parámetro, sólo una consulta y conjunto de respuestas serán generados. En la mayoría de los casos, si un nickname es dado, indica el servidor al cual el usuario está conectado.

Estos mensajes normalmente tienen poco valor para los servicios, es por lo tanto recomendado no permitir a los servicios que los usen.

Comando	MOTD
Parámetros	[<destino>]
Descripción	El comando MOTD es usado para obtener el mensaje del día del servidor dado o del servidor actual si <destino> se omite.
Respuestas Numéricas	RPL_MOTDSTART RPL_MOTD RPL_ENDOFMOTD ERR_NOMOTD
Ejemplos	

Comando	LUSERS
Parámetros	[<mascara> [<destino>]]
Descripción	El comando LUSERS es usado para obtener estadísticas acerca del tamaño de la red IRC. Si no se dan parámetros, la respuesta será acerca de toda la red. Si se especifica una máscara, la respuesta será acerca de la parte de la red formada por los servidores que corresponden con la máscara. Finalmente, si el parámetro <destino> se especifica, la respuesta es reenviada a ese servidor, el cual generará la respuesta. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	RPL_LUSERCLIENT RPL_LUSEROP RPL_LUSERUNKOWN RPL_LUSERCHANNELS RPL_LUSERME ERR_NOSUCHSERVER
Ejemplos	

Comando	VERSION
Parámetros	[<destino>]
Descripción	El comando VERSION es usado para consultar la versión del programa servidor. Un parámetro opcional <destino> es usado para consultar la versión del programa servidor al cual un cliente no está directamente conectado. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_VERSION
Ejemplos	VERSION tolsun.oulu.fi ; Comando para revisar la versión del servidor "tolsun.oulu.fi".

Comando	STATS
Parámetros	[<consulta> [<destino>]]
Descripción	El comando STATS es usado para consultar estadísticas de cierto servidor. Si el parámetro <consulta> se omite, sólo el final de la respuesta de estadísticas es enviado de regreso. Una consulta puede darse para cualquier letra la cual es revisada por el servidor destino y es pasada por los servidores intermedios ignorada e inalterada. Se permiten comodines en el parámetro <destino>. Excepto para los siguientes, la lista de consultas válidas es dependiente de la implementación. Las consultas estándar deben ser soportadas por el servidor: l – regresa una lista de las conexiones del servidor mostrando que tanto tiempo han estado establecidas y el tránsito sobre esas conexiones en Kbytes y número de mensajes para cada dirección; m – regresa el contador de uso para cada uno de los comandos soportados por el servidor; los comandos para los cuales el contador de uso es cero se pueden omitir; o – regresa una lista de usuarios privilegiados configurados, operadores; u – regresa una cadena mostrando cuanto tiempo ha estado funcionando el servidor. Se recomienda también que la configuración de acceso de clientes y servidores se publique de esta manera.
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_STATSLINKINFO RPL_STATSUPTIME RPL_STATSCOMANDOS RPL_STATSOLINE RPL_ENDOFSTATS
Ejemplos	STATS m ; Comando para revisar el uso de comandos en el servidor en el que estás conectado.

Comando	LINKS
Parámetros	[[<servidor remoto>] <mascara de servidor>]
Descripción	Con LINKS un usuario puede listar todos los nombres de servidor que son conocidos por el servidor que responde a la consulta. La lista regresada de servidores debe corresponder con la máscara, o si no se da una máscara, se regresa la línea completa. Si <servidor remoto> es proporcionado además de <mascara de servidor>, el comando links es reenviado al primer servidor encontrado que corresponda con este nombre (si existe) y se requiere que el servidor responda a la consulta. **Este mensaje NO se implementará en nuestra versión.**
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_LINKS RPL_ENDOFLINKS
Ejemplos	LINKS *.au ;Comando para listar todos los servidores que corresponden con *.au; LINKS *.edu *.bu.edu ;Comando para listar los servidores que corresponden con *.bu.edu como los ve el primer servidor que corresponda con *.edu.

Comando	TIME
Parámetros	[<destino>]
Descripción	El comando TIME es usado para consultar la hora local del servidor especificado. Si el parámetro <destino> no es proporcionado, el servidor que reciba el comando debe responder a la consulta. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_TIME
Ejemplos	TIME tolsun.oulu.fi ;revisa la hora en el servidor "tolson.oulu.fi"

Comando	CONNECT
Parámetros	<servidor destino> <puerto> [<servidor remoto>]
Descripción	El comando CONNECT puede usarse para pedir a un servidor que intente establecer una nueva conexión a otro servidor inmediatamente. CONNECT es un comando privilegiado y debe estar disponible sólo para los operadores de la red IRC. Si <servidor remoto> es proporcionado y su máscara no corresponde con el servidor que analiza el mensaje, el intento de conexión es enviado a la primer correspondencia de servidor remoto. En cualquier otro caso, el intento de conexión es hecho por el servidor que procesa la respuesta. El servidor que reciba un comando CONNECT remoto debe generar un mensaje WALLOPS describiendo la fuente y el destino de la petición. **Este mensaje NO se implementará en nuestra versión.**
Respuestas Numéricas	ERR_NOSUCHSERVER ERR_NOPRIVILEGES ERR_NEEDMOREPARAMS
Ejemplos	CONNECT tolsun.oulu.fi 6667 ;Comando para intentar conectarse el servidor local a tolsun.oulu.fi en el puerto 6667

Comando	TRACE
Parámetros	[<destino>]
Descripción	El comando TRACE es usado para encontrar la ruta a un servidor específico y la información acerca de sus vecinos. Cada servidor que procesa este comando debe reportar al que lo envía acerca de él. Las respuestas forman una cadena que muestra la ruta hacia el destino. Después de mandar esta respuesta de regreso, debe enviar el mensaje TRACE al siguiente servidor hasta que el servidor dado sea alcanzado. Si se omite el parámetro <destino>, se recomienda que el comando TRACE envíe un mensaje al que lo envía diciendo a que servidores el servidor local tiene conexión directa. Si <destino> es un servidor actual, se requiere que el servidor de destino reporte todos los servidores, servicios y operadores que están conectados a él. Si el comando fue ejecutado por un operador, el servidor debe reportar también todos los usuarios que están conectados a él. Si <destino> es un nickname, entonces sólo se da una respuesta para este nickname. Si el parámetro <destino> se omite, se recomienda que el comando TRACE sea analizado como dirigido al servidor que está procesando el mensaje. Se permiten comodines en el parámetro <destino>. **Este mensaje NO se implementará en nuestra versión.**
Respuestas Numéricas	ERR_NOSUCHSERVER Si el mensaje TRACE está destinado a otro servidor, todos los servidores intermedios deben regresar una respuesta RPL_TRACELINK para indicar que el mensaje TRACE pasó a través de ellos y a donde se envió. RPL_TRACELINK Una respuesta TRACE puede componerse de cualquiera de las siguientes respuestas numéricas. RPL_TRACECONNECTING RPL_TRACEHANDSHAKE RPL_TRACEUNKNOWN RPL_TRACEOPERATOR RPL_TRACEUSER RPL_TRACESERVER RPL_TRACESERVICE RPL_TRACENEWTYPE RPL_TRACECLASS RPL_TRACELOG RPL_TRACEEND
Ejemplos	TRACE *.oulu.fi ; TRACE a un servidor que corresponde con *.oulu.fi

Comando	ADMIN.
Parámetros	[<destino>]
Descripción	El comando ADMIN se usa para encontrar información acerca del administrador del servidor dado o del servidor actual si el parámetro <destino> se omite. Cada servidor debe tener la habilidad de reenviar mensajes ADMIN a otros servidores. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_ADMINME RPL_ADMINLOC1 RPL_ADMINLOC2 RPL_ADMINEMAIL
Ejemplos	ADMIN tolsun.oulu.fi ; pide una respuesta ADMIN de tolsun.oulu.fi ADMIN syrk ; petición ADMIN para el servidor al que está conectado el usuario syrk

Comando	INFO
Parámetros	[<destino>]
Descripción	El comando INFO tiene que regresar información describiendo el servidor; su versión, cuándo fue compilado, el nivel de parches, cuándo fue iniciado y cualquier otra información miscelánea que se considere relevante. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_INFO RPL_ENDOFINFO
Ejemplos	INFO csd.bu.edu ; pide una respuesta INFO desde csd.bu.edu INFO Angel ; pide información del servidor al que está conectado Angel.

Consultas de servicios

El grupo de comandos de consultas a servicios ha sido diseñado para regresar información acerca de cualquier servicio que esté conectado a la red.

Comando	SERVLIST
Parámetros	[<mascara> [<tipo>]]
Descripción	El comando SERVLIST se usa para listar los servicios actualmente conectados a la red y visibles al usuario que ejecuta el comando. Los parámetros opcionales pueden usarse para restringir el resultado de la consulta.
Respuestas Numéricas	RPL_SERVLIST RPL_SERVLISTEND
Ejemplos	

Comando	SQUERY
Parámetros	<nombre de servicio> <texto>
Descripción	El comando SQUERY es usado al igual que PRIVMSG. La única diferencia es que un servicio debe recibir el mensaje. Esta es la única forma de que un mensaje de texto sea enviado a un servicio. Vea PRIVMSG para obtener más detalles y ejemplos.
Respuestas Numéricas	
Ejemplos	SQUERY irchelp :HELP privmsg ; Mensaje al servicio con nickname irchelp. SQUERY dict@irc.fr :fr2en blaireau ; Mensaje al servicio con nombre dict@irc.fr.

Consultas sobre los usuarios

Las consultas de usuario son un grupo de comandos que se relacionan con la búsqueda de detalles sobre un usuario o grupo de usuarios en particular. Cuando se usan comodines con cualquiera de estos comandos, si corresponden, sólo regresan información sobre usuarios que son visibles por quien hace la consulta. La visibilidad de un usuario se determina por la combinación de modo de usuario y el conjunto de canales comunes a ambos.

Aunque los servicios no deben usar esta clase de mensajes, lo tienen permitido.

Comando	WHO	
Parámetros	[<mascara> ["o"]]	
Descripción	El comando WHO es usado por un cliente para generar una consulta que regresa una lista de información que corresponde con el parámetro <mascara> dado por el cliente. En la ausencia del parámetro <mascara>, todos los usuarios visibles (usuarios que no son invisibles (modo +i) y los que no tienen un canal común con el cliente que hace la petición) son listados. El mismo resultado se puede lograr usando una <mascara> de “0” o cualquier comodín que corresponda con cada usuario visible. La <mascara> pasada a WHO es revisada contra el host del usuario, el servidor, el nombre real y el nickname si el canal <mascara> no se encuentra.	
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_WHOREPLY	RPL_ENDOFWHO
Ejemplos	WHO *.fi WHO jto* o	; Comando para listar todos los usuarios que corresponden con "*.fi". ; Comando para listar todos los usuarios que corresponden con "jto*" si son operadores.

Comando	WHOIS	
Parámetros	[<destino>] <mascara> *("," <mascara>)	
Descripción	Este comando es usado para consultar información acerca de un usuario en particular. El servidor responderá este comando con varios mensajes numéricos que indican diferentes estados de cada usuario que corresponda con la máscara (si puedes verlos). Si no está presente un comodín en la máscara, cualquier información acerca de este nickname que tengas permitida ver se presentará. Si el parámetro <destino> se especifica, manda la consulta al servidor especificado. Es útil si se desea saber cuánto tiempo el usuario en cuestión ha estado sin hacer nada ya que sólo el servidor local conoce esa información, mientras que todo lo demás se conoce globalmente. Se permiten comodines en el parámetro <destino>.	
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_WHOISUSER RPL_WHOISCHANNELS RPL_AWAY RPL_WHOISIDLE RPL_ENDOFWHOIS	ERR_NONICKNAMEGIVEN RPL_WHOISCHANNELS RPL_WHOISSERVER RPL_WHOISOPERATOR ERR_NOSUCHNICK
Ejemplos	WHOIS wiz WHOIS eff.org trillian	; regresa la información disponible del usuario con nickname WiZ ; pregunta al servidor eff.org información sobre el usuario trillian

Comando	WHOWAS
Parámetros	<nickname> *("," <nickname>) [<contador> [<destino>]]
Descripción	WHOWAS pide información sobre un nickname que ya no existe. Esto puede deberse a un cambio de nickname o a un usuario que abandona IRC. En respuesta a esta consulta, el servidor busca en su historial de nicknames. La historia es revisada hacia atrás regresando la primer entrada más reciente. Si hay múltiples entradas, hasta <contador> respuestas serán regresadas (o todas si no se especifica <contador>). Si se pasa un número no positivo como <contador>, entonces se hace una búsqueda completa. Se permiten comodines en el parámetro <destino>.
Respuestas Numéricas	ERR_NONICKNAMEGIVEN ERR_WASNOSUCHNICK RPL_WHOWASUSER RPL_WHOISSERVER RPL_ENDOFWHOWAS
Ejemplos	WHOWAS Wiz ; regresa toda la información del historial de nicknames acerca de "WiZ"; WHOWAS Mermaid 9 ; regresa a lo máximo, las 9 entradas más recientes en el historial de nicknames para "Mermaid"; WHOWAS Trillian 1 *.edu ; regresa la entrada más reciente para "Trillian" del primer servidor que corresponda con "*.edu".

Mensajes misceláneos

Los mensajes de esta categoría no caben en ninguna otra, pero de todos modos son parte del protocolo y son requeridos.

Comando	KILL
Parámetros	<nickname> <comentario>
Descripción	El comando KILL es usado para causar que una conexión cliente-servidor sea cerrada por el servidor que tiene la conexión actual. Los servidores generan mensajes KILL en las colisiones de nicknames. Puede estar también disponible a usuarios que tienen el estatus de operadores. Los clientes que tienen algoritmos de reconexión hacen este comando inútil ya que la desconexión es muy corta. Como sea, rompe el flujo de datos y puede ser usado para detener grandes cantidades de flujo de usuarios abusivos o debidos a accidentes. Los usuarios abusivos usualmente no se preocupan porque se reconectan pronto y vuelven a su mismo comportamiento. Para prevenir el abuso de este comando, cualquier usuario puede elegir recibir los mensajes KILL generados por otros para estar alerta. En un escenario donde los nicknames deben ser globalmente únicos en todo momento, los mensajes KILL son enviados cuando se detectan duplicados esperando que los dos desaparezcan y después sólo uno reaparezca. Cuando un cliente es removido como resultado de un mensaje KILL, el servidor debe agregar el nickname a la lista de nicknames no disponibles en un intento de evitar que los clientes vuelvan a usar este nombre inmediatamente, el cual es usualmente el patrón de comportamiento abusivo. El comentario dado debe reflejar la razón actual para el KILL. Para KILL's generados por el servidor, normalmente se trata de detalles sobre el origen de los dos nicknames en conflicto. Para los usuarios es dejado a su criterio proveer una razón adecuada que satisfaga a otros que la ven. Para prevenir que los mensajes KILL sean generados escondiendo la identidad de quien los envía, el comentario también muestra una ruta que es actualizada por cada servidor por donde pasa el mensaje, cada uno agregando su nombre a la ruta. NOTA: Se recomienda que sólo los operadores puedan desconectar a otros usuarios con el comando KILL. Este comando ha sido sujeto de muchas controversias.
Respuestas Numéricas	ERR_NOPRIVILEGES ERR_NEEDMOREPARAMS ERR_NOSUCHNICK ERR_CANTKILLSERVER
Ejemplos	

Comando	PING
Parámetros	<servidor1> [<servidor2>]
Descripción	El comando PING se usa para probar la presencia de un cliente activo o servidor al otro lado de la conexión. Los servidores envían un mensaje PING a intervalos regulares si no se detecta actividad en una conexión. Si una conexión falla al responder a un mensaje PING dentro de un intervalo de tiempo, la conexión es cerrada. Un mensaje PING puede enviarse aún si la conexión está activa. Cuando se recibe un mensaje PING, el mensaje PONG apropiado debe enviarse a <servidor1> (el servidor que envió el mensaje) tan pronto como sea posible. Si el parámetro <servidor2> es especificado, representa el destino del PING y el mensaje se envía a él.
Respuestas Numéricas	ERR_NOORIGIN ERR_NOSUCHSERVER
Ejemplos	PING tolsun.oulu.fi ; Comando para mandar un mensaje PING al servidor PING WiZ tolsun.oulu.fi ; Comando de WiZ para enviar un mensaje PING al servidor "tolsun.oulu.fi" PING :irc.funet.fi ; mensaje Ping enviado por el servidor "irc.funet.fi"

Comando	PONG
Parámetros	<servidor> [<servidor2>]
Descripción	El mensaje PONG es una respuesta al mensaje PING. Si el parámetro <servidor2> está dado, este mensaje debe reenviarse al destino dado. El parámetro <servidor> es el nombre de la entidad que ha respondido al mensaje PING y generado este mensaje.
Respuestas Numéricas	ERR_NOORIGIN ERR_NOSUCHSERVER
Ejemplos	PONG csd.bu.edu tolsun.oulu.fi ; mensaje PONG desde csd.bu.edu a tolsun.oulu.fi

Comando	ERROR
Parámetros	<mensaje de error>
Descripción	El comando ERROR es para que lo usen los servidores cuando reportan un error serio o fatal a los servidores a los que están conectados directamente. Puede ser enviado de un servidor a otro pero no debe ser aceptado desde ningún cliente normal desconocido. Sólo se debe usar un mensaje ERROR para reportar errores que ocurren en una liga de servidor a servidor. Un mensaje de error se envía al servidor del otro lado y a los usuarios locales y archivos de registro apropiados. No es para pasarse a ningún otro servidor si se recibe desde un servidor. El mensaje ERROR también se usa antes de terminar una conexión con un cliente. Cuando un servidor envía un mensaje ERROR a sus operadores, el mensaje debe ser encapsulado dentro de un mensaje NOTICE indicando que el cliente no fue responsable del error.
Respuestas Numéricas	Ninguna.
Ejemplos	ERROR :El servidor *.fi ya existe ; mensaje ERROR al otro servidor que causó este error. NOTICE WiZ :ERROR desde csd.bu.edu – El servidor *.fi ya existe ; el mismo mensaje ERROR anterior pero enviado al usuario WiZ en el otro servidor.

Características opcionales

Esta sección describe los mensajes opcionales. No son requeridos en una implementación de un servidor del protocolo descrito aquí. En ausencia de alguna característica debe generarse una respuesta de error o un error de comando desconocido. Si el mensaje está destinado a otro servidor entonces debe ser pasado (se requiere un análisis elemental). Las respuestas numéricas reservadas para esto se incluyen aquí también.

De esta sección, sólo los mensajes USERHOST e ISON están disponibles para los servicios.

Comando	AWAY
Parámetros	[<texto>]
Descripción	Con el comando AWAY los clientes pueden establecer una cadena de respuesta automática para cualesquiera comandos PRIVMSG dirigidos a ellos (no al canal en el que están). El servidor envía una respuesta automática al cliente que envía el comando PRIVMSG. El único servidor que responde es aquel al que el cliente está conectado. El comando AWAY es usado también con un parámetro para establecer un mensaje o sin parámetros para remover el mensaje AWAY. El mensaje AWAY sólo debe usarse para comunicación cliente-servidor. Un servidor puede elegir ignorar el mensaje silenciosamente recibido de otros servidores. Para actualizar el estado de un cliente en todos los servidores el modo de usuario 'a' debe usarse.
Respuestas Numéricas	RPL_UNAWAY RPL_NOWAWAY
Ejemplos	AWAY :Fui a comer ; Comando para establecer la cadena del mensaje AWAY a "Fui a comer".

Comando	REHASH
Parámetros	Ninguno.
Descripción	El comando REHASH es un comando administrativo que puede usarse por un operador para forzar que el servidor vuelva a leer y procesar su archivo de configuración.
Respuestas Numéricas	RPL_REHASHING ERR_NOPRIVILEGES
Ejemplos	REHASH ; mensaje de un usuario con estatus de operador pidiendo al servidor que reprocese su archivo de configuración.

Comando	DIE
Parámetros	Ninguno.
Descripción	Un operador puede usar el comando DIE para apagar el servidor. Este mensaje es opcional ya que puede ser visto como un riesgo que permitiría a cualquier persona que se conecte como operador ejecutar este comando. El comando DIE debe siempre ser analizado sólo por el servidor al cual está conectado el cliente que lo manda y no debe ser pasado a otros servidores.
Respuestas Numéricas	ERR_NOPRIVILEGES
Ejemplos	DIE ; no se requieren parámetros.

Comando	RESTART
Parámetros	Ninguno.
Descripción	Un operador puede usar el comando RESTART para forzar al servidor a reiniciarse. Este mensaje es opcional ya que puede ser visto como un riesgo que permitiría a cualquier persona conectarse al servidor como operador y ejecutar este comando causando al menos una interrupción en el servicio. El comando RESTART debe siempre ser totalmente analizado por el servidor al cual está conectado el cliente que lo manda y no debe ser pasado a otros servidores.
Respuestas Numéricas	ERR_NOPRIVILEGES
Ejemplos	RESTART ; no se requieren parámetros.

Comando	SUMMON
Parámetros	<usuario> [<destino> [<canal>]]
Descripción	El comando SUMMON puede usarse para dar a los usuarios que están en un host corriendo el servidor IRC un mensaje pidiéndoles que se unan a la red IRC. Este mensaje sólo es enviado si el servidor destino tiene SUMMON habilitado, el usuario tiene una sesión abierta y el proceso servidor puede escribir a las terminales de los usuarios o a algo equivalente. Si no se da el parámetro <servidor> se envía el mensaje al usuario <usuario>. Si SUMMON no está habilitado en el servidor, debe regresar ERR_SUMMONDISABLED.
Respuestas Numéricas	ERR_NORECIPIENT ERR_FILEERROR

	ERR_NOLOGIN ERR_SUMMONDISABLED	ERR_NOSUCHSERVER RPL_SUMMONING
Ejemplos	SUMMON jto SUMMON jto tolsun.oulu.fi	; invita al usuario jto del servidor ; invita al usuario jto en el host donde corre un servidor llamado “tolsun.oulu.fi".

Comando	USERS	
Parámetros	[<destino>]	
Descripción	El comando USERS regresa una lista de usuarios que tienen una sesión en el servidor en un formato similar a los comandos UNIX who, rusers y finger. Si está deshabilitado, el mensaje correcto debe ser enviado para indicar esto. Debido a las implicaciones de seguridad de tal comando, debe estar deshabilitado por default en las implementaciones de servidor. Habilitarlo debe requerir recompilar el servidor o hacer algún cambio equivalente en lugar de sólo cambiar alguna opción y reiniciar el servidor. El procedimiento para habilitar este comando debe incluir también los comentarios suficientes. **Este mensaje NO se implementará en nuestra versión.**	
Respuestas Numéricas	ERR_NOSUCHSERVER RPL_USERSSTART RPL_NOUSERS ERR_USERSDISABLED	ERR_FILEERROR RPL_USERS RPL_ENDOFUSERS
Ejemplos	USERS eff.org	; pide la lista de los usuarios con una sesión en el servidor eff.org

Comando	WALLOPS
Parámetros	<Texto a ser enviado>
Descripción	El comando WALLOPS es usado para enviar un mensaje a todos los usuarios actualmente conectados que tienen el modo de usuario ’w’. Después de implementar WALLOPS como un comando de usuario se vio que era a menudo usado para mandar mensajes a muchas personas. Debido a esto, se recomienda que la implementación de WALLOPS permita sólo a los servidores enviar este mensaje.
Respuestas Numéricas	ERR_NEEDMOREPARAMS
Ejemplos	:csd.bu.edu WALLOPS :CONNECT '*.uiuc.edu 6667' de Alberto ; mensaje WALLOPS de csd.bu.edu anunciando un mensaje CONNECT recibido de Alberto.

Comando	USERHOST	
Parámetros	<nickname> *(ESPACIO <nickname>)	
Descripción	El comando USERHOST toma una lista de hasta 5 nicknames, cada uno separado por un espacio y regresa una lista de información acerca de cada nickname que encontró. La lista obtenida tiene cada respuesta separada por un espacio.	
Respuestas Numéricas	RPL_USERHOST	ERR_NEEDMOREPARAMS
Ejemplos	USERHOST Wiz Michael syrk :ircd.stealth.net 302 yournick :syrk=+syrk@millennium.stealth.net ; Respuesta para el usuario syrk	; petición USERHOST de informacion de los nicknames "Wiz", "Michael", y "syrk"

Comando	ISON
Parámetros	<nickname> *(ESPACIO <nickname>)
Descripción	El comando ISON fue implementado para dar un medio rápido y eficiente para obtener una respuesta acerca de si un nickname dado está actualmente en IRC. ISON sólo toma un tipo de parámetro: una lista de nicknames separada por espacios. Cada nickname que está presente en la lista, es agregado por el servidor a su cadena de respuesta. Por lo tanto, la cadena de respuesta puede regresar vacía (ninguno de los nicknames está presente), puede ser una copia exacta de la cadena parámetro (todos están presentes) o puede ser cualquier subconjunto de los nicknames dados en el parámetro. El único límite en el número de nicknames es que la combinación no debe ser demasiado larga como para causar que el servidor la corte para que quepa en 512 carácteres. ISON sólo es procesado por el servidor local del cliente que manda el comando y no debe

	pasarse a otros servidores.
Respuestas Numéricas	RPL_ISON ERR_NEEDMOREPARAMS
Ejemplos	ISON telefono trillian WiZ jarlek Avalon Angel Juan ; Ejemplo de ISON preguntando por 7 nicknames.

Respuestas

La siguiente es una lista de respuestas numéricas que son generadas en respuesta a los comandos dados. Cada número se da con su nombre y cadena de respuesta.

Las constantes en el rango de 001 a 099 se usan para conexiones cliente-servidor solamente y nunca deben viajar entre servidores. Las respuestas generadas a los comandos se encuentran en el rango de 200 a 399.

#	Constante	Mensaje
001	RPL_WELCOME	"Bienvenido a la red IRC <nickname>!<usuario>@<host>"
002	RPL_YOURHOST	"Tu host es <nombre de servidor>, corriendo la version <ver>"
003	RPL_CREATED	"Este servidor fue creado el <fecha>"
004	INFO	"<nombre de servidor> <version> <modos de usuario disponibles> <modos de canal disponibles>"

El servidor envía las respuestas 001 a 004 a un usuario después de un registro exitoso de la conexión.

#	Constante	Mensaje
005	RPL_BOUNCE	"Intenta el servidor <nombre de servidor>, puerto <numero de puerto>"
302	RPL_USERHOST	Enviado por el servidor a un usuario para sugerir un servidor alternativo. Esto se usa a menudo cuando la conexión es rechazada debido a que el servidor está lleno. ":*1<respuesta> *(" " <respuesta>)" Formato de respuesta usada por USERHOST para listar respuestas a la lista de consulta. La cadena de respuesta se compone como sigue: respuesta = nickname ["*"] "=" ("+" / "-") hostname El '*' indica si el cliente está registrado como un operador. El '-' o '+' representan si el cliente ha enviado o no un mensaje AWAY respectivamente.
303	RPL_ISON	Formato de respuesta usado por ISON para la lista de respuestas. ":*1<nickname> *(" " <nickname>)"
301	RPL_AWAY	"<nickname> :<mensaje away>"
305	RPL_UNAWAY	":Tu ya no estas marcado como ausente"
306	RPL_NOWAWAY	":Haz sido marcado como ausente"

Estas respuestas son usadas con el comando AWAY (si está permitido). RPL_AWAY se envía a cualquier cliente que envía un PRIVMSG a un cliente que está ausente. RPL_AWAY sólo es enviado por el servidor al cual el cliente está conectado. Las respuestas RPL_UNAWAY y RPL_NOAWAY se envían cuando el cliente remueve y establece un mensaje AWAY.

#	Constante	Mensaje
311	RPL_WHOISUSER	"<nickname> <usuario> <host> * :<nombre real>"
312	RPL_WHOISSERVER	"<nickname> <servidor> :<informacion del servidor>"
313	RPL_WHOISOPERATOR	"<nickname> :es un operador IRC"
317	RPL_WHOISIDLE	"<nickname> <integer> :segundos sin movimientos"
318	RPL_ENDOFWHOIS	"<nickname> :Fin de la lista WHOIS"
319	RPL_WHOISCHANNELS	"<nickname> :*(("@" / "+") <canal> " ")"

Las respuestas 311 - 313, 317 - 319 son todas las respuestas generadas a un mensaje WHOIS. Si hay suficientes parámetros presentes, el servidor que responde debe formular una respuesta (si el nickname es encontrado) o regresar un mensaje de error. El '*' en RPL_WHOISUSER está como un carácter literal y no como un comodín. Para cada conjunto de respuestas, sólo RPL_WHOISCHANNELS puede aparecer más de una vez (para listados largos de nombres de canales). La '@' y el '+' junto al nombre de canal indican si un cliente es un operador de canal o se le ha dado permiso

para hablar en un canal moderado. La respuesta RPL_ENDOFWHOIS es usada para marcar el fin del procesamiento del mensaje WHOIS.

#	Constante	Mensaje
314	RPL_WHOWASUSER	"<nickname> <usuario> <host> * :<nombre real>"
369	RPL_ENDOFWHOWAS	"<nickname> :Fin de WHOWAS" Cuando se contesta a un mensaje WHOWAS, un servidor debe usar las respuestas RPL_WHOWASUSER, RPL_WHOISSERVER o ERR_WASNOSUCHNICK para cada nickname en la lista presentada. Al final de todas las respuestas debe haber un RPL_ENDOFWHOWAS (aún si sólo hubo una respuesta y fue un error).
321	RPL_LISTSTART	Obsoleto. No usado
322	RPL_LIST	"<canal> <# visible> :<topico>"
323	RPL_LISTEND	":Fin de LIST" Las respuestas RPL_LIST, RPL_LISTEND marcan las respuestas reales con datos y fin de la respuesta del servidor a un comando LIST. Si no hay canales disponibles que regresar, sólo la última respuesta debe enviarse.
325	RPL_UNIQOPIS	"<canal> <nickname>"
324	RPL_CHANNELMODEIS	"<canal> <modo> <parametros de modo>"
331	RPL_NOTOPIC	"<canal> :No se ha establecido topico"
332	RPL_TOPIC	"<canal> :<topico>" Cuando se envía un mensaje TOPIC para determinar el tópico del canal, se envía una de dos respuestas. Si el tópico está establecido, RPL_TOPIC se envía. De lo contrario se envía RPL_NOTOPIC.
341	RPL_INVITING	"<canal> <nickname>" Regresado por el servidor para indicar que el mensaje INVITE fue exitoso y que se está pasando al cliente final.
342	RPL_SUMMONING	"<usuario> :Convocando al usuario a IRC" Regresado por el servidor preguntando un mensaje SUMMON para indicar que está convocando a un usuario.
346	RPL_INVITELIST	"<canal> <mascara de invitacion>"
347	RPL_ENDOFINVITELIST	"<canal> :Fin de la lista de invitados del canal" Cuando se listan las máscaras de invitación para cierto canal, se requiere que un servidor envíe la lista de regreso usando los mensajes RPL_INVITELIST y RPL_ENDOFINVITELIST. Un mensaje RPL_INVITELIST separado es enviado por cada máscara activa. Después de que las máscaras han sido listadas (o si no hay ninguna) un RPL_ENDOFINVITELIST debe ser enviado.
348	RPL_EXCEPTLIST	"<canal> <mascara de excepcion>"
349	RPL_ENDOFEXCEPTLIST	"<canal> :Fin de la lista de excepciones de canal" Cuando se listan las máscaras de excepción para cierto canal, se requiere que un servidor envíe la lista usando los mensajes RPL_EXCEPTLIST y RPL_ENDOFEXCEPTLIST. Un mensaje RPL_EXCEPTLIST separado es enviado por cada máscara activa. Después de que las máscaras han sido listadas (o si no hay ninguna) un RPL_ ENDOFEXCEPTLIST debe ser enviado.
351	RPL_VERSION	"<version>.<nivel de depuracion> <servidor> :<comentarios>" Respuesta del servidor mostrando los detalles de su versión. La <version> es la versión del software que se está usando (incluyendo cualquier revisión) y el <nivel de depuracion> es usado para indicar si el servidor está corriendo en modo de depuración. El campo <comentarios> puede contener cualquier comentario acerca de la versión.
352	RPL_WHOREPLY	"<canal> <usuario> <host> <servidor> <nickname> ("H" / "G" > ["*"] [("@" / "+")] :<contador hop> <nombre real>"
315	RPL_ENDOFWHO	"<nombre> :Fin de la lista WHO" El par RPL_WHOREPLY y RPL_ENDOFWHO se usa para responder un mensaje WHO. El RPL_WHOREPLY sólo es enviado si hay una correspondencia apropiada a la consulta WHO. Si se da una lista de parámetros con el mensaje WHO, un RPL_ENDOFWHO MUST debe enviarse después de procesar cada elemento de la lista.

353	RPL_NAMREPLY	"("=" / "*" / "@") <canal> :["@" / "+"] <nickname> *(" " ["@" / "+"] <nickname>) "@" se usa para canales secretos, "*" para privados y "=" para los otros (canales públicos).
366	RPL_ENDOFNAMES	"<canal> :Fin de la lista NAMES" Para responder a un mensaje NAMES, un par de respuesta que consiste de RPL_NAMREPLY y RPL_ENDOFNAMES se envía del servidor al cliente. Si no se encontró el canal de la consulta, entonces sólo se regresa RPL_ENDOFNAMES. La excepción a esto es cuando un mensaje NAMES se envía sin parámetros y todos los canales visibles y su contenido se envían en una serie de mensajes RPL_NAMEREPLY con un RPL_ENDOFNAMES para marcar el fin.
364	RPL_LINKS	"<mascara> <servidor> :<contador hop> <informacion de servidor>"
365	RPL_ENDOFLINKS	"<mascara> :Fin de la lista LINKS" En respuesta al mensaje LINKS, un servidor debe mandar respuestas usando RPL_LINKS y marcar el fin de la lista usando una respuesta RPL_ENDOFLINKS.
367	RPL_BANLIST	"<canal> <mascara de rechazo>"
368	RPL_ENDOFBANLIST	"<canal> :Fin de la lista de rechazos de canal" Cuando se listan los rechazos activos para cierto canal, se requiere que un servidor envíe la lista usando los mensajes RPL_BANLIST y RPL_ENDOFBANLIST. Un mensaje separado RPL_BANLIST se envía por cada máscara de rechazo activa. Después de que las máscaras de rechazo han sido listadas (o si ninguna está presente), un mensaje RPL_ENDOFBANLIST debe enviarse.
371	RPL_INFO	":<cadena>"
374	RPL_ENDOFINFO	":Fin de la lista INFO" Un servidor respondiendo a un mensaje INFO tiene que mandar todos sus “info” en una serie de mensajes RPL_INFO con un RPL_ENDOFINFO para indicar el fin de las respuestas.
375	RPL_MOTDSTART	":- <servidor> Mensaje del dia - "
372	RPL_MOTD	":- <texto>"
376	RPL_ENDOFMOTD	":Fin del comando MOTD” Cuando se responde al comando MOTD y se encuentra el archivo MOTD, el archivo se despliega línea por línea, con un límite en cada línea de 80 carácteres usando formatos de respuesta RPL_MOTD. Estos deben estar entre un RPL_MOTDSTART (antes de los RPL_MOTDs) y un RPL_ENDOFMOTD (después).
381	RPL_YOUREOPER	":Ahora eres un operador IRC" RPL_YOUREOPER se envía a un cliente que ha corrido exitosamente un mensaje OPER y ha ganado el estatus de operador.
382	RPL_REHASHING	"<archivo de configuracion> :Rehashing" Si la opción rehash es usada y un operador envía un mensaje REHASH, un RPL_REHASHING se envía de regreso al operador.
383	RPL_YORESERVICE	"Eres el servicio <nombre de servicio>" Enviado por el servidor a un servicio después de un registro exitoso.
391	RPL_TIME	"<servidor> :<cadena mostrando la hora local del servidor>" Cuando se contesta al mensaje TIME, un servidor debe enviar la respuesta usando el formato RPL_TIME. La cadena que muestra la hora sólo necesita contener el día correcto y la hora. No hay más requerimientos para esta cadena.
392	RPL_USERSSTART	":UserID Terminal Host"
393	RPL_USERS	":<nombre de usuario> <ttyline> <nombre de host>"
394	RPL_ENDOFUSERS	":Fin de USERS"
395	RPL_NOUSERS	":Nadie tiene una sesión" Si el mensaje USERS es manejado por un servidor, las respuestas RPL_USERSTART, RPL_USERS, RPL_ENDOFUSERS y RPL_NOUSERS son usadas. RPL_USERSSTART debe enviarse primero, seguido por una secuencia de RPL_USERS o un solo RPL_NOUSER. Después de esto va un RPL_ENDOFUSERS.
200	RPL_TRACELINK	"Liga <version y nivel de depuracion> <destino>

		<siguiente servidor> V<version de protocolo> <tiempo de conexion en segundos> <backstream sendq> <upstream sendq>"
201	RPL_TRACECONNECTING	"Intenta. <clase> <servidor>"
202	RPL_TRACEHANDSHAKE	"H.S. <clase> <servidor>"
203	RPL_TRACEUNKNOWN	"???? <clase> [<direccion IP del cliente en formato de puntos>]"
204	RPL_TRACEOPERATOR	"Oper <clase> <nickname>"
205	RPL_TRACEUSER	"User <clase> <nickname>"
206	RPL_TRACESERVER	"Serv <clase> <int>S <int>C <servidor> <nickname!usuario|*!*>@<host|servidor> <version de protocolo>"
207	RPL_TRACESERVICE	"Servicio <clase> <nombre> <tipo> <tipo activo>"
208	RPL_TRACENEWTYPE	"<tipo nuevo> 0 <nombre del cliente>"
209	RPL_TRACECLASS	"Class <clase> <contador>"
210	RPL_TRACERECONNECT	No usado.
261	RPL_TRACELOG	"Archivo <archivo log> <nivel de depuracion>"
262	RPL_TRACEEND	"<nombre del servidor> <version y nivel de depuracion > :Fin de TRACE" Los RPL_TRACE* son regresados por el servidor en respuesta al mensaje TRACE. Cuantos regresa depende del comando TRACE y de si fue enviado por un operador o no.. No hay orden predefinido para que alguno ocurra primero. Las respuestas RPL_TRACEUNKNOWN, RPL_TRACECONNECTING y RPL_TRACEHANDSHAKE se usan para conexiones que no han sido totalmente establecidas y son desconocidas, están intentando conectarse o están en el proceso de completar el saludo de servidores. RPL_TRACELINK es enviado por cualquier servidor que maneja un mensaje TRACE y tiene que pasarlo a otro servidor. La lista de RPL_TRACELINKs enviados en respuesta a un comando TRACE que atraviesa la red IRC debe reflejar las conexiones actuales entre los servidores a través de una ruta. RPL_TRACENEWTYPE se usa para cualquier conexión que no cabe dentro de las otras categorías pero que es desplegada de todos modos. RPL_TRACEEND se envía para indicar el fin de la lista.
211	RPL_STATSLINKINFO	"<nombre de la liga> <sendq> <mensajes enviados> <Kbytes enviados> <mensajes recibidos> <Kbytes recibidos> <tiempo de conexion>" Reporta estadísticas sobre una conexión. <nombre de la liga> identifica la conexión en particular, <sendq> es la cantidad de datos que está en cola y esperando a ser enviada. <tiempo de conexión> indica cuanto tiempo conexión en segundos ha estado abierta la conexión.
212	RPL_STATSCOMMANDS	"<comando> <contador> <contador de bytes> <contador remoto>" - Reporta estadísticas sobre uso de comandos.
219	RPL_ENDOFSTATS	"<letra de estadísticas> :Fin del reporte STATS"
242	RPL_STATSUPTIME	":Servidor funcionando por %d dias %d:%02d:%02d" - Reporta el tiempo que ha estado funcionando el servidor.
243	RPL_STATSOLINE	"O <mascara de host> * <nombre>" Reporta los hosts permitidos desde los cuales los usuarios se pueden volver operadores IRC.
221	RPL_UMODEIS	"<cadena de modo de usuario>" Para contestar una consulta acerca del modo de un cliente, se envía RPL_UMODEIS.
234	RPL_SERVLIST	"<nombre> <servidor> <mascara> <tipo> <contador hop> <informacion>"
235	RPL_SERVLISTEND	"<mascara> <tipo> :Fin del listado de servicios" Cuando se listan los servicios en respuesta al mensaje SERVLIST, se requiere que un servidor envíe la lista usando los mensajes RPL_SERVLIST y RPL_SERVLISTEND. Un mensaje RPL_SERVLIST separado es enviado por cada servicio. Después de que los servicios han sido listados (o si ninguno está presente) un mensaje RPL_SERVLISTEND debe ser enviado.
251	RPL_LUSERCLIENT	":Hay <integer> usuarios y <integer> servicios en <integer> servidores"
252	RPL_LUSEROP	"<integer> :operador(es) en linea"
253	RPL_LUSERUNKNOWN	"<integer> :conexión(es) desconocida(s)"
254	RPL_LUSERCHANNELS	"<integer> :canales formados"
255	RPL_LUSERME	":Tengo <integer> clientes y <integer> servidores"

Al procesar el mensaje LUSERS, el servidor envía un conjunto de respuestas de entre RPL_LUSERCLIENT, RPL_LUSEROP, RPL_USERUNKNOWN, RPL_LUSERCHANNELS y RPL_LUSERME. Cuando conteste, un servidor debe enviar RPL_LUSERCLIENT y RPL_LUSERME. Las otras respuestas se envían sólo si se encuentra un contador diferente de cero para ellos.

#	Constante	Mensaje
256	RPL_ADMINME	"<servidor> :Informacion administrativa"
257	RPL_ADMINLOC1	":<informacion administrativa>"
258	RPL_ADMINLOC2	":<informacion administrativa>"
259	RPL_ADMINEMAIL	":<informacion administrativa>"

Cuando se contesta a un mensaje ADMIN, un servidor debe usar las respuestas desde RPL_ADMINME hasta RPL_ADMINEMAIL y dar un mensaje con cada una. Para RPL_ADMINLOC1 se espera una descripción de en qué ciudad, estado y país está el servidor, seguida de detalles de la institución (RPL_ADMINLOC2) y finalmente el contacto administrativo para el servidor (se requiere una dirección de correo) en RPL_ADMINEMAIL.

#	Constante	Mensaje
263	RPL_TRYAGAIN	"<comando> :Por favor espere un momento e intente de nuevo." - Cuando un servidor ignora un comando sin procesarlo debe usar la respuesta RPL_TRYAGAIN para informar al cliente que lo originó.

Respuestas de error

Las respuestas de error se encuentran en el rango de 400 a 599.

#	Constante	Mensaje	Descripción
401	ERR_NOSUCHNICK	"<nickname> :No existe tal nickname/canal"	Usado para indicar que el parámetro nickname dado a un comando no se está usando actualmente.
402	ERR_NOSUCHSERVER	"<nombre del servidor> :No existe tal servidor"	Usado para indicar que el nombre de servidor dado no existe actualmente.
403	ERR_NOSUCHCHANNEL	"<nombre de canal> :No existe tal canal"	Usado para indicar que el canal dado es inválido.
404	ERR_CANNOTSENDTOCHAN	"<nombre de canal> :No se puede enviar al canal"	Enviado a un usuario que no está en un canal con modo +n o no es operador de canal (o modo +v) en un canal que tiene el modo +m o donde el usuario es rechazado y está tratando de enviar un mensaje PRIVMSG a ese canal.
405	ERR_TOOMANYCHANNELS	"<nombre de canal> :Te has unido a demasiados canales"	Enviado a un usuario cuando ha alcanzado el número máximo de canales permitido e intenta unirse a otro.
406	ERR_WASNOSUCHNICK	"<nickname> :No existia tal nickname"	Regresado por WHOWAS para indicar que no hay información para ese nickname en el historial.
407	ERR_TOOMANYTARGETS	"<destino> :<codigo de error> recipientes. <mensaje de cancelacion>"	Regresado a un cliente que intenta enviar un mensaje PRIVMSG/NOTICE usando el formato de destino usuario@host y para cuando un usuario@host tiene varias ocurrencias. Regresado a un cliente que trata de enviar un mensaje PRIVMSG/NOTICE a demasiados recipientes. Regresado a un cliente que trata de unirse a un canal seguro usando su nombre corto cuando hay más de un canal con ese nombre.
408	ERR_NOSUCHSERVICE	"<nombre de servicio> :No existe tal servicio"	Regresado a un cliente que trata de enviar un mensaje SQUERY a un servicio que

			no existe.
409	ERR_NOORIGIN	":No se especifico origen"	Mensaje PING o PONG sin parámetro origen.
411	ERR_NORECIPIENT	":No se dio recipiente (<comando>)"	No hay mensaje
412	ERR_NOTEXTTOSEND	":No hay texto para enviar"	No hay mensaje
413	ERR_NOTOPLEVEL	"<mascara> :No se especifico dominio de nivel superior"	No hay mensaje
414	ERR_WILDTOPLEVEL	"<mascara> :comodín en dominio de nivel superior"	No hay mensaje
415	ERR_BADMASK	"<mascara> :Máscara de Servidor/host erronea"	

412 - 415 son regresados por PRIVMSG para indicar que el mensaje no fue enviado por alguna razón. ERR_NOTOPLEVEL y ERR_WILDTOPLEVEL son errores que se envían cuando hay un uso inválido de "PRIVMSG $<servidor>" o "PRIVMSG #<host>".

#	Constante	Mensaje	Descripción
421	ERR_UNKNOWNCOMMAND	"<comando> :Comando desconocido"	Regresado a un cliente registrado para indicar que el comando enviado es desconocido por el servidor.
422	ERR_NOMOTD	":El archivo MOTD no existe"	El archivo MOTD del servidor no puede ser abierto por el servidor.
423	ERR_NOADMININFO	"<servidor> :No hay información administrativa disponible"	Regresado por un servidor en respuesta a un mensaje ADMIN cuando hay un error al buscar la información apropiada.
424	ERR_FILEERROR	":Error de archivo al hacer <operacion de archivo> en <archivo>"	Error genérico usado para reportar una operación de archivo que falló durante el procesamiento de un mensaje.
431	ERR_NONICKNAMEGIVEN	":No se dio el nickname"	Regresado cuando el parámetro nickname que se esperaba por un comando no se encuentra.
432	ERR_ERRONEUSNICKNAME	"<nickname> :Nickname erroneo"	Regresado después de recibir un mensaje NICK que contiene carácteres que no caen dentro del conjunto definido.
433	ERR_NICKNAMEINUSE	"<nickname> :El nickname está siendo usado"	Regresado cuando un mensaje NICK es procesado y resulta un nickname existente.
436	ERR_NICKCOLLISION	"<nickname> :KILL de colisión de nickname desde <usuario>@<host>"	Regresado por el servidor a un cliente cuando detecta una colisión de nicknames (registro de un nickname que ya existe en otro servidor).
437	ERR_UNAVAILRESOURCE	"<nickname/canal> :El nickname/canal no está disponible temporalmente"	Regresado por un servidor a un usuario que trata de unirse a un canal actualmente bloqueado por el mecanismo de retardo de canal. Regresado por un servidor que trata de cambiar un nickname que está bloqueado por el mecanismo de retardo de nickname.
441	ERR_USERNOTINCHANNEL	"<nickname> <canal> :No está en ese canal"	Regresado por el servidor para indicar que el usuario destino del comando no está en ese canal.
442	ERR_NOTONCHANNEL	"<canal> :No estás en ese canal"	Regresado por el servidor siempre que un cliente trata de ejecutar un comando que afecta a un canal del cual el cliente no es miembro.
443	ERR_USERONCHANNEL	"<usuario> <canal> :ya está en el canal"	Regresado cuando un cliente trata de invitar a un usuario a un canal que ya está en el canal.
444	ERR_NOLOGIN	"<usuario> :El usuario no tiene una sesión abierta"	Regresado por el servidor después de un comando SUMMON para un usuario que no se pudo realizar porque no tiene una

			sesión abierta.
445	ERR_SUMMONDISABLED	":SUMMON ha sido deshabilitado"	Regresado como respuesta al comando SUMMON. Debe ser regresado por cualquier servidor que no lo implementa.
446	ERR_USERSDISABLED	":USERS ha sido deshabilitado"	Regresado como respuesta al comando USERS. Debe ser regresado por cualquier servidor que no lo implementa.
451	ERR_NOTREGISTERED	":No te has registrado"	Regresado por el servidor para indicar que el cliente debe registrarse antes de que el servidor permita analizar sus mensajes en detalle.
461	ERR_NEEDMOREPARAMS	"<comando> :No hay suficientes parámetros"	Regresada por el servidor a numerosos comandos para indicar al cliente que no dio suficientes parámetros.
462	ERR_ALREADYREGISTRED	":Comando no autorizado (ya esta registrado)"	Regresado por el servidor a cualquier liga que trate de cambiar parte de los detalles registrados (como el password o detalles del usuario en un segundo mensaje USER).
463	ERR_NOPERMFORHOST	":Tu host no está entre los privilegiados"	Regresado a un cliente que intenta registrarse con un servidor que no está configurado para permitir conexiones desde el host en el que se intentó la conexión.
464	ERR_PASSWDMISMATCH	":Password incorrecto"	Regresado para indicar un intento fallido de registrar una conexión para la que se requiere un password y éste no fue dado o es incorrecto.
465	ERR_YOUREBANNEDCREEP	":Estás excluido de este servidor"	Regresado después de un intento de conectarse y registrarse con un servidor que está configurado para negar explícitamente las conexiones.
466	ERR_YOUWILLBEBANNED		Enviado por el servidor a un usuario para informar que el acceso al servidor pronto le será negado.
467	ERR_KEYSET	"<canal> :Clave de canal ya establecida"	No hay mensaje
471	ERR_CHANNELISFULL	"<canal> :No puedes unirte al canal (+l)"	No hay mensaje
472	ERR_UNKNOWNMODE	"<char> :es un carácter de modo desconocido por mi para <canal>"	No hay mensaje
473	ERR_INVITEONLYCHAN	"<canal> : No puedes unirte al canal (+i)"	No hay mensaje
474	ERR_BANNEDFROMCHAN	"<canal> : No puedes unirte al canal (+b)"	No hay mensaje
475	ERR_BADCHANNELKEY	"<canal> : No puedes unirte al canal (+k)"	No hay mensaje
476	ERR_BADCHANMASK	"<canal> :Máscara de canal incorrecta"	No hay mensaje
477	ERR_NOCHANMODES	"<canal> :El canal no soporta modos"	No hay mensaje
478	ERR_BANLISTFULL	"<canal> <char> :La lista de canales está llena"	No hay mensaje
481	ERR_NOPRIVILEGES	":Permiso denegado- No eres un operador IRC"	Cualquier comando que requiera de privilegios de operador debe regresar este error para indicar que el intento fracasó.
482	ERR_CHANOPRIVSNEEDED	"<canal> :No eres operador de canal"	Cualquier comando que requiera de privilegios de operador de canal (tales como mensajes MODE) debe regresar este error si el cliente que hace este intento no es un operador de canal en el canal especificado.

483	ERR_CANTKILLSERVER	":¡No puedes matar a un servidor!"	Cualquier intento de usar el comando KILL en un servidor será rechazado y este error será regresado inmediatamente.
484	ERR_RESTRICTED	":¡Tu conexión está restringida!"	Enviado por el servidor a un usuario después de una conexión para indicar la naturaleza restringida de la conexión (modo de usuario "+r").
485	ERR_UNIQOPPRIVSNEEDED	":Tú no eres el operador de canal original"	Cualquier mensaje MODE que requiera de privilegios de creador de canal debe regresar este error si el cliente que hace el intento no es un operador de canal en el canal especificado.
491	ERR_NOOPERHOST	":No hay lineas-O para tu host"	Si un cliente envía un mensaje OPER y el servidor no ha sido configurado para permitir conexiones desde el host del cliente como operador, este error debe regresarse.
501	ERR_UMODEUNKNOWNFLAG	":Bandera de modo (MODE) desconocida "	Regresado por el servidor para indicar que un mensaje MODE fue enviado con un parámetro nickname y que la bandera de modo no fue reconocida.
502	ERR_USERSDONTMATCH	":No puedes cambiar el modo de otros usuarios"	Error enviado a cualquier usuario que trata de ver o cambiar el modo de otro usuario.

Valores reservados

Estas constantes no se describen antes porque caen dentro de una de las siguientes categorías:

1. Ya no se usan;
2. Están reservadas para uso futuro;
3. Se utilizan pero son parte de una característica no genérica del servidor IRC.

#	Constante	#	Constante
231	RPL_SERVICEINFO	232	RPL_ENDOFSERVICES
233	RPL_SERVICE		
300	RPL_NONE	316	RPL_WHOISCHANOP
361	RPL_KILLDONE	362	RPL_CLOSING
363	RPL_CLOSEEND	373	RPL_INFOSTART
384	RPL_MYPORTIS		
213	RPL_STATSCLINE	214	RPL_STATSNLINE
215	RPL_STATSILINE	216	RPL_STATSKLINE
217	RPL_STATSQLINE	218	RPL_STATSYLINE
240	RPL_STATSVLINE	241	RPL_STATSLLINE
244	RPL_STATSHLINE	244	RPL_STATSSLINE
246	RPL_STATSPING	247	RPL_STATSBLINE
250	RPL_STATSDLINE		
492	ERR_NOSERVICEHOST		

Se deberá tener cuidado de no utilizar estos valores en alguna otra constante que se defina para utilizarse en los mensajes.

Bibliografía

1. Pressman, Roger S. *Ingeniería del Software: Un Enfoque práctico*. Cuarta edición McGraw-Hill, México, 1998.

2. Jackson, M.A. *System Development.* Editorial Prentice Hall, 1983.

3. Hopcroft, A. V. J. *Data structures and Algorithms.* Editorial Addison-Wesley, 1983.

4. Liu, Cricket. Peek, Jerry. Jones, Russ Jones. Buus, Bryan. & Nye, Adrian, *Administración de Servicios de Información en Internet*, Editorial Mc Graw Hill, 1997.

5. Hopcroft, John E. y Ullman, Jeffrey D. *Introduction to Automata Theory, languages and Computation*. Editorial Addison-Wesley, México, 1998.