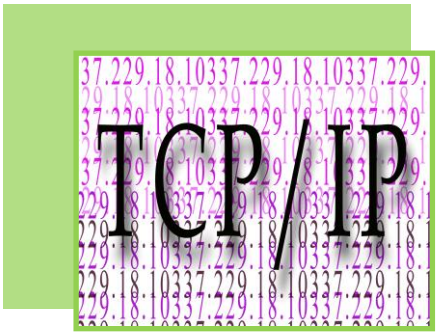
Capítulo 4
Modelo TCP/IP



Como se mencionó en el capítulo 2, la comunicación entre dos dispositivos conectados a una red se divide en capas para facilitar su diseño e implementación. Cada nivel realiza una tarea específica para dar solución a algún problema de la comunicación y requiere los servicios que brindan las capas inferiores para proporcionar un conjunto de servicios a las capas superiores.

Las funciones que realiza cada una de la capas se generalizan en el modelo de referencia OSI, analizado en el capítulo 2. Los diferentes modelos empleados para la implementación de redes pueden ser explicados a partir del modelo OSI, ya que su estructura es similar y cumplen objetivos idénticos.

Los protocolos de redes, que son soluciones prácticas con las que se implementa un modelo, operan en diferentes niveles, es decir, existen protocolos para cada capa del modelo de red empleado para la comunicación. La implementación de dichos protocolos se realiza a través de un conjunto de programas o rutinas de software al que se denomina *pila* o *stack.* En este capítulo se describen los protocolos del modelo TCP/IP.

4.1 Breve historia

Cuando ARPANET (Red de la Agencia de Proyectos de Investigación Avanzada) ya estaba en funcionamiento, su desempeño era aún lento y presentaba fallas y caídas en el sistema de manera frecuente, además del hecho de que siendo una red utilizada por el Departamento de Defensa, era propensa a ataques externos y su integridad estaba constantemente amenazada. Lo anterior motivó a la Agencia de Proyectos de Investigación Avanzada (DARPA) a desarrollar nuevas tecnologías de transmisión de datos. De esta manera se inició el desarrollo de una nueva generación de protocolos, con la finalidad de brindar un servicio más confiable, seguro y robusto. Como resultado de este trabajo nació la familia de protocolos TCP/IP desarrollada por Robert E Kahn y Vint Cerf a mediados de la década de los 70, aunque fue años más tarde que se incorporó a la red ARPANET. Para el año de 1978, los protocolos de la pila se probaron de manera exitosa en una red de Estados Unidos que se conectó a otra red ubicada en Londres. Fue hasta 1982 que se estableció TCP/IP como la pila de protocolos para la red ARPANET y el Departamento de Defensa los declaró estándares para uso militar. Finalmente, en 1983 se estandarizó como pila de protocolos básicos de red.

Poco a poco TCP/IP se ha ido integrando a las redes existentes, hasta llegar a ser el estándar en comunicaciones más usado a nivel mundial. El modelo TCP/IP forma la tecnología base para una red de redes global, su uso general se encuentra en la comunicación de grupos de redes interconectadas.

Actualmente son muchas las industrias que usan TCP/IP como protocolos de comunicación, entre las que se pueden mencionar a las industrias aeroespacial, automotriz, electrónica, hotelera, petrolera, de servicios de impresión, farmacéutica, entre muchas otras, así como escuelas, universidades, laboratorios, centros de investigación y más.

Una ventaja por la cual TCP/IP ha sido preferido es por su capacidad para interconectar diferentes tipos de hardware y equipos con distintos sistemas operativos, ya que funciona sobre muchas tecnologías de redes y posee un alto grado de interoperabilidad. Una característica de la pila de protocolos TCP/IP es que la fiabilidad de la comunicación es responsabilidad del dispositivo, no de la red, con lo que el trabajo de la pila se reduce al mínimo y se hace posible conectar diferentes tipos de redes.

4.2 Estructura del modelo

El diseño del modelo TCP/IP se basa en la idea de que dos dispositivos de una red se pueden comunicar entre ellos sin importar el lugar geográfico en donde residen y que para hacerlo no es necesario que los dispositivos involucrados entiendan la forma de trabajar del sistema subyacente ni la tecnología de red necesaria para llevar a cabo el proceso. Para hacer esto posible dicho sistema se basa en la interconexión de redes independientes.

.

Para cumplir las metas de diseño requeridas el modelo TCP/IP se define en cuatro capas y a cada una le corresponde una o más del modelo OSI, como se muestra en la **¡Error! No se encuentra el origen de la referencia.**. Estas capas son:

1. Capa de aplicación

2. Capa de transporte
3. Capa de red
4. Capa de enlace

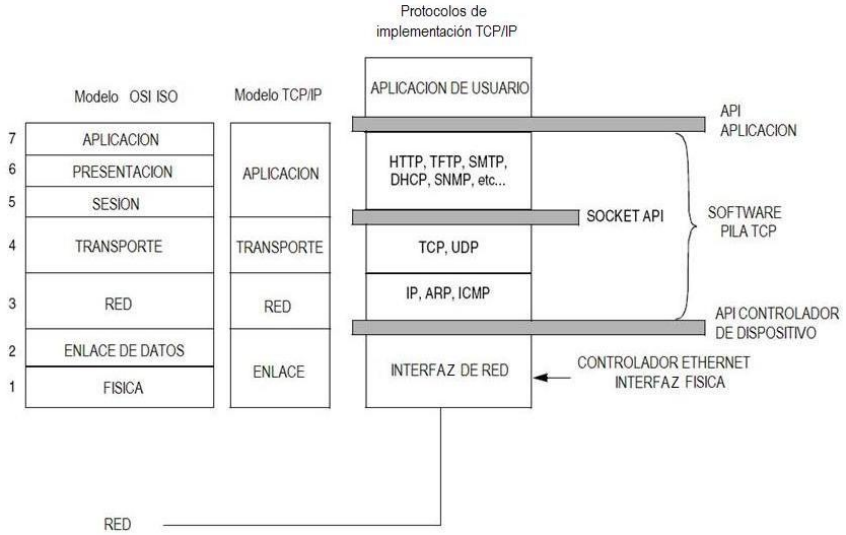


Figura 4.1 Correspondencia entre el modelo OSI y el modelo TCP/IP y ejemplo de algunos protocolos que integran la pila

En la **¡Error! No se encuentra el origen de la referencia.** se muestran las funciones de cada una de las capas del modelo TCP/IP y algunos ejemplos de protocolos de cada capa del modelo.

Tabla 4.1 Funciones de las capas del modelo TCP/IP y ejemplos de protocolos

CAPA	FUNCIÓN	EJEMPLO
Aplicación	Es un conjunto de protocolos que proporcionan servicios específicos a los usuarios de la red, por ejemplo servicio de correo electrónico o de transferencia de archivos.	HTTP → Protocolo de Transferencia de Hipertexto SMTP → Protocolo Simple de Transferencia de Correo POP3 → Protocolo de Oficina Postal Versión 3 Telnet → Red de Telecomunicaciones (acceso a equipos remotos) FTP → Protocolo de Transferencia de Archivos
Transporte	Realiza y mantiene una comunicación entre equipos finales.	TCP → Protocolo de Control de Transmisión UDP → Protocolo de Datagramas de Usuario
Red	Se encarga de la entrega y el encaminamiento de paquetes entre los equipos de diferentes redes interconectadas.	IP → Protocolo de Internet ICMP →Protocolo de Mensajes de Control de Internet ARP → Protocolo de Resolución de Direcciones [12]
Enlace	Implementa la transmisión de información para un medio físico específico.	Ethernet (IEEE 802.3) Token Ring

Como se observa en la **¡Error! No se encuentra el origen de la referencia.** la pila TCP/IP incluye una gran variedad de protocolos diferentes, enfocados a distintos tipos de aplicaciones y servicios. En total abarcan más de cien, pero la base para todos ellos son los dos protocolos que dan nombre a toda la familia: el Protocolo de Control de Transmisión (TCP) y el Protocolo de Internet (IP).

En esta tesis se implementan los protocolos mostrados en la Tabla 4.2. Por ejemplo en la capa de aplicación se implementa el protocolo HTTP, ya que el propósito es habilitar un microcontrolador como servidor web y para ello es necesario hacer uso de las funciones del Protocolo de Transferencia de Hipertexto. Como

[12] El protocolo ARP se encuentra entre la capa de red y la capa de enlace

trabajo futuro se puede dar más funcionalidad al sistema de control por red, añadiendo servicios como el correo electrónico o la transferencia de archivos.

Tabla 4.2 Implementación de protocolos TCP/IP

Capa	Protocolo
Aplicación	HTTP
Transporte	TCP, UDP
Red	IP, ICMP, ARP
Enlace	Ethernet

4.3 Flujo de datos entre las capas del modelo

Al igual que en el modelo OSI, en el modelo TCP/IP los datos pasan de un protocolo a otro (o de una capa a otra) para su tratamiento. La información se genera en la capa de aplicación y se pasa a la capa de transporte, donde se añade el encabezado de la capa de transporte y el paquete resultante se envía al protocolo de la capa de red. En la capa de red se toman los datos provenientes de la aplicación, más el encabezado de la capa de transporte y se añade el encabezado del protocolo de red. Posteriormente los datos se entregan a la capa de enlace, en donde se añade el encabezado correspondiente y los datos son convertidos en señales que finalmente se transfieren por el medio físico.

Una vez que la información llega a su destino ocurre el proceso inverso: la información es recibida por el controlador de red, que implementa las funciones de la capa de enlace, decodifica la señal proveniente del medio, verifica el contenido del encabezado Ethernet, lo quita del paquete y entrega el resto a la capa de red. La capa de red en el nodo receptor encuentra los encabezados de su protocolo y los remueve, realiza sus funciones y si el paquete es válido lo pasa al protocolo de la capa de transporte. Aquí se remueven los encabezados del protocolo de transporte, se realizan las funciones establecidas para los protocolos de esta capa y finalmente se entrega la información al protocolo correspondiente de la capa de aplicación.

En la Figura 4.2 se ilustra el procedimiento de transmisión y recepción de datos entre dos nodos utilizando el modelo TCP/IP.

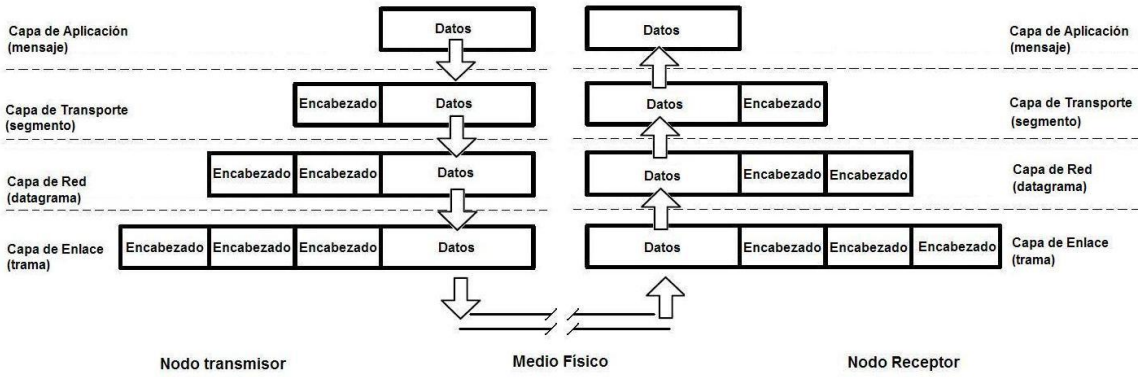


Figura 4.2 Flujo de datos a través de las capas del modelo

4.4 Descripción general

En el nivel más bajo del modelo TCP/IP se encuentra la capa de enlace, donde se implementan las funciones correspondientes a la capa de enlace y la capa física del modelo OSI. Dichas funciones las realiza el controlador de red y para el caso de redes Ethernet (LAN), siguen el estándar IEEE 802.3. El controlador de red posee los medios necesarios para codificar y decodificar señales, así como recibir y transmitir datos. Es necesaria la programación de algunas funciones para manejar el módulo controlador de red. Estas funciones se conocen como *software del controlador* de red y se ocupan del manejo de la información almacenada en los buffers, como recuperarla y enviarla a los siguientes niveles o recibir los datos provenientes de las capas superiores y acomodarlos en el espacio de memoria necesario para que el controlador los transmita por la red. Las funciones que ofrece el *software del controlador de red* son

utilizadas por los protocolos de los niveles más altos y son funciones del tipo: *iniciarTransmisión, recibir, leerBuffer, escribirBuffer, limpiarBuffer* y otras parecidas.

El nivel de enlace de datos puede ser implementado para diferentes tecnologías de redes con la idea de proporcionar un conjunto de funciones similares para cada tipo de red, de tal forma que la interfaz proporcionada al nivel superior sea la misma, sin importar la implementación específica del software, ya que dicha implementación está directamente relacionada con la tecnología del hardware de la red.

Lo importante es contar con las funciones de las que hará uso la capa de red, independientemente de su implementación. Bajo este esquema, si el hardware de la red cambia, sólo tiene que sustituirse el software del controlador de red anterior por el que maneja al nuevo controlador de red, el cual proporciona las mismas funciones al nivel superior y los demás aspectos de la red quedan intactos.

En la capa de red, también llamada capa de internet, se encuentra ubicado el protocolo de Internet (IP). Esta capa se encuentra inmediatamente sobre la capa de enlace y requiere las funciones proporcionadas por el software del controlador de red. El Protocolo de Internet está diseñado para resolver problemas de interconexión de redes, siendo este nivel el que permite que la comunicación se pueda realizar de una red a otra, de manera que los niveles superiores realicen su trabajo como si se tratará de una sola red uniforme.

En la capa de transporte se implementan dos protocolos: TCP y UDP. La diferencia entre estos dos protocolos es que uno maneja un servicio orientado a conexión y el otro no, por lo que la implementación del primero es más compleja que la del segundo.

TCP ofrece un servicio fiable y orientado a la conexión. Cada vez que se requiere enviar información usando el protocolo TCP se debe establecer una conexión entre el emisor y el receptor, cada segmento de datos enviados debe ser confirmado y al término de la comunicación se debe cerrar la conexión. Para manejar el intercambio de información TCP hace uso de temporizadores o *timers*; de tal manera que si un paquete enviado no es confirmado dentro de un periodo de tiempo establecido, TCP lo retransmite antes de enviar un nuevo paquete de datos. Dicho en otras palabras, no se permite el envío de datos hasta que el último paquete enviado haya sido confirmado. De esta manera se asegura la entrega de datos. TCP también se encarga de detectar y corregir errores en los datos, entregarlos en el orden correcto y evitar su duplicidad.

Por otro lado, UDP ofrece un servicio no orientado a conexión en el cual los datos son enviados a un destino sin esperar a que se confirme su recibo, por lo que no garantiza la entrega. Si durante la transmisión un paquete se pierde UDP no tiene manera de saberlo ni toma medidas al respecto. Debido a su naturaleza no orientada a conexión, la implementación de UDP resulta más sencilla que la de TCP, ya que no necesita establecer y cerrar conexiones, manejar temporizadores, esperar confirmaciones, realizar corrección de errores, verificar orden de entrega de paquetes, evitar duplicaciones ni realizar retransmisiones. La ventaja de UDP frente a TCP es que su desempeño es mucho más rápido, simplemente se ocupa en enviar datos tan rápido como le sea posible.

La elección entre un protocolo u otro para la capa de transporte depende de la aplicación. Hay aplicaciones en las que es fundamental que no se presenten pérdidas de datos, aunque se tenga que aumentar el tiempo de transferencia. Un ejemplo de este tipo de aplicaciones sería la transferencia de archivos. Para otras aplicaciones es fundamental que la transmisión se realice lo más rápido posible, como en la transferencia de video en tiempo real, en la cual los datos deben ser enviados tan rápidamente como se generan y no importa mucho si un pequeño pedazo de información no es recibido, ya que esta pérdida no es relevante. Otro ejemplo sería un sistema que sensa el estado de una variable y envía lecturas constantemente a una aplicación por red para realizar la gráfica de la variación de la señal sensada. En este caso la pérdida de un dato no causara inconvenientes, ya que la falta de un punto en la gráfica no es relevante.

Los protocolos de la capa de aplicación pueden estar basados en el protocolo TCP o en el protocolo UDP, según la función a la que estén enfocados. Las aplicaciones HTTP usan TCP. Cuando se genera un mensaje la información se pasa a TCP, el cual abre una conexión y envía los datos. En el extremo opuesto, TCP procesa la información recibida, le avisa a la aplicación que han llegado nuevos datos y envía un segmento de confirmación al nodo emisor. Cuando ya no hay más información que enviar o recibir se cierra la conexión. Hay que hacer notar que para el protocolo TCP el intercambio de información puede ser de datos provenientes de la aplicación, o de datos generados en el nivel de transporte, como es el caso de los paquetes de control que envían una confirmación de recepción o las solicitudes de inicio o término de conexión.

Las aplicaciones que utilizan UDP no necesitan abrir y cerrar conexiones ni enviar segmentos de control como confirmaciones, simplemente generan información y la pasan a UDP para que se encargue de su envío, mientras que en el extremo receptor las aplicaciones reciben información de UDP y la procesan.

Los protocolos descritos anteriormente, junto con el software del controlador de red, cumplen la funcionalidad básica de las capas del modelo TCP/IP, sin embargo se basan en protocolos adicionales para completar sus prestaciones o utilizan protocolos que brindan alguna funcionalidad complementaria. Dos de estos protocolos son el Protocolo de Resolución de Direcciones o ARP y el Protocolo de Mensajes de Control de Internet o ICMP.

ARP es un protocolo necesario para determinar la dirección física a la que se debe enviar un paquete a partir de su dirección IP. En el nivel de enlace los paquetes utilizan *direcciones físicas* para su transmisión, mientras que en el nivel de red se hace uso de *direcciones de red*. La capa de enlace se apoya en ARP para enviar paquetes a un nodo del que se conoce su dirección IP pero no su dirección física.

ICMP es un protocolo auxiliar que permite enviar mensajes de error o informativos sobre el funcionamiento de una red. La función de este protocolo no es esencial para la comunicación, pero resulta útil para realizar pruebas y corregir errores.

En los siguientes temas se presenta una descripción detallada de cada uno de los protocolos que se implementan en esta tesis, desde los protocolos de las capas más bajas hasta los de las más altas de la pila TCP/IP.

4.5 ARP – Protocolo de Resolución de Direcciones

Cada dispositivo conectado a una red posee tanto una dirección física (también llamada dirección MAC), como una dirección de red (o dirección IP). Las direcciones de red son usadas por los protocolos de alto nivel para enviar paquetes a través de una red de redes, se deben especificar en el encabezado de los datagramas IP y se tratarán en el protocolo IP. Las direcciones MAC son usadas por los controladores de red para enviar las tramas Ethernet a través de redes físicas y se deben especificar en el encabezado de las tramas.

Cuando un paquete debe ser transmitido pasa de la capa de red a la capa de enlace indicando en el encabezado de red la dirección IP del destino, como se muestra en la Figura 4.3. Para formar una trama Ethernet el emisor necesita conocer la dirección MAC de destino cuya IP se encuentra en el datagrama, para lo cual hace uso de las funciones de ARP. Una vez que la dirección física correspondiente se ha encontrado, ésta se añade al encabezado de la trama Ethernet y el paquete se envía. ARP es el protocolo que hace posible la resolución de una dirección física a partir de una dirección de red y permite que los protocolos de alto nivel trabajen sólo a partir de sus direcciones de red, ocultando los detalles de las direcciones físicas de los dispositivos. ARP fue originalmente diseñado para redes de tipo Ethernet, pero ha sido generalizado para poder ser usado por otros tipos diferentes de interfaces físicas.

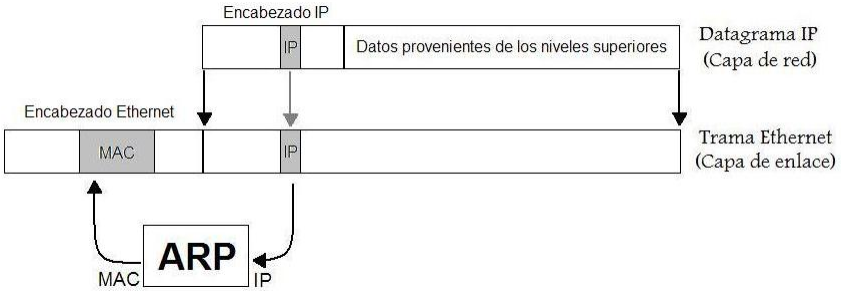
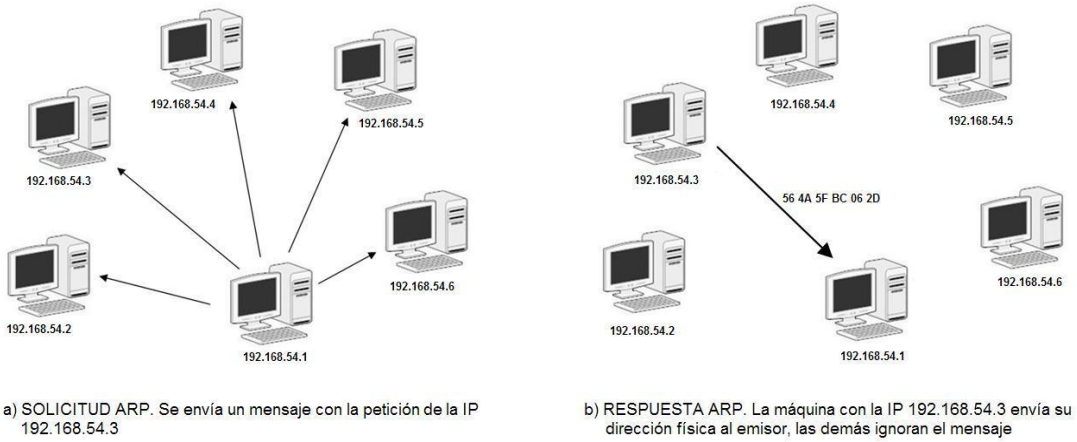


Figura 4.3 Construcción de una trama Ethernet a partir de un datagrama IP

El funcionamiento de ARP se basa en el direccionamiento *broadcast* (o difusión, cuya dirección física corresponde a FF FF FF FF FF FF) para enviar un mensaje de petición (solicitud ARP) a todos los integrantes de una red, informando la dirección IP del dispositivo cuya dirección física se requiere conocer. El dispositivo que reconoce su dirección IP en el mensaje responde enviando su dirección física al emisor,

mientras que los demás dispositivos descartan el mensaje ya que no está dirigido a ellos. La Figura 4.4 ilustra el procedimiento.

En una red cada nodo debe mantener una tabla donde almacena las direcciones IP con sus correspondientes direcciones físicas. Cada vez que se resuelve una dirección, el par de direcciones IP–MAC se agrega a la tabla. Esta tabla se llama *tabla de direcciones ARP* y permite el uso constante de una dirección MAC, al reducir la cantidad de peticiones y en consecuencia el retardo en posteriores entregas al mismo nodo. Cuando la capa de enlace requiere enviar una trama, busca dentro de la tabla ARP la correspondiente dirección física que debe añadir en el campo *dirección de destino* de la cabecera Ethernet, si ésta no se encuentra aún, ARP genera un mensaje de solicitud y lo difunde por la red. Al recibir la respuesta la almacena en su tabla y procede a formar la trama Ethernet y a enviar la información a la máquina con la dirección física recibida. De igual forma, cuando una petición llega a cualquier dispositivo de la red, sea o no para él, el receptor puede agregar la dirección IP y dirección física del transmisor, indicadas en la solicitud, a su tabla de direcciones.



a) SOLICITUD ARP. Se envía un mensaje con la petición de la IP 192.168.54.3

b) RESPUESTA ARP. La máquina con la IP 192.168.54.3 envía su dirección física al emisor, las demás ignoran el mensaje

Figura 4.4 Procedimiento de solicitud y respuesta ARP

Los valores de la tabla ARP deben renovarse periódicamente, ya que la dirección física de un dispositivo podría cambiar si se cambia el hardware del controlador de red. En general las entradas de la tabla son valores temporales a los que se les denomina *entradas dinámicas*. Las entradas dinámicas o temporales, tienen un tiempo de vida determinado, si dentro de este tiempo son recibidos nuevos mensajes ARP con los datos de la entrada, se toman las direcciones IP y MAC del emisor y se sobreescriben en la tabla. Si el tiempo de vida de una entrada temporal caduca ésta se borra de la memoria para liberar espacio. Es necesario destacar que también pueden existir valores fijos en las tablas, es decir, se renuevan pero no se borran, a estos valores se les conoce como *entradas estáticas*. Una entrada estática o fija puede ser por ejemplo el par de direcciones IP–MAC del gateway de la red.

Existen dos situaciones posibles al enviar un paquete IP:

1. Que el destino se encuentre dentro de la misma red física que el transmisor.
2. Que el destino se encuentre en otra red, a la cual se accede a través de una o varias compuertas de red (ruteadores, conmutadores o *gateways*).

Para el primer caso, el transmisor forma una trama Ethernet y busca en la tabla ARP la dirección física correspondiente a la IP destino y en caso de no encontrarla genera una solicitud ARP, que al ser respondida permite entregar directamente la trama a su destino.

Para el segundo caso, el transmisor busca en la tabla ARP la dirección física de la compuerta de red y en caso de no encontrarla, genera una petición ARP con la IP de la compuerta, la cual encamina todos los paquetes cuya IP se encuentre fuera de la red. Una vez conocida la dirección MAC de la compuerta, todos los paquetes con IP que no pertenezca a la red se envían a la compuerta con la dirección física de ésta, pero con la IP del dispositivo final. Para determinar si una IP se encuentra dentro o fuera de la red, se utiliza un valor denominado *máscara de subred*, término que se explica en el Protocolo de Internet.

Mientras no se conozca la dirección física de entrega de un paquete, éste debe permanecer almacenado en la memoria. ARP espera un tiempo por la respuesta después de enviar una solicitud, mientras mantiene el paquete en una cola de espera. Si la respuesta no llega y el tiempo de espera se agota, se reenvía la solicitud. Se debe definir un tiempo de espera máximo y un número de retransmisiones permitidas para los mensajes ARP enviados. Cada vez que el tiempo de espera de una respuesta expira, la solicitud se vuelve a enviar, hasta que las retransmisiones se agoten. Una vez que la respuesta llega se procede al envío de la información que está esperando en cola.

4.5.1 Formato de mensajes ARP

El formato de los mensajes ARP se muestra en la Figura 4.5, en cada renglón se representan 4 bytes. El mensaje está formado por 28 bytes, que incluyen los campos que se muestran en la figura y se describen a continuación.

No. de byte	byte + 0 (7 6 5 4 3 2 1 0)	byte + 1 (7 6 5 4 3 2 1 0)	byte + 2 (7 6 5 4 3 2 1 0)	byte + 3 (7 6 5 4 3 2 1 0)
0	Tipo de Hardware		Protocolo	
4	Long. dir. Hardware	Long. Protocolo	Código de operación	
8	Dirección física origen			
12	Dirección física origen		Dirección IP origen	
16	Dirección IP origen		Dirección física destino	
20	Dirección física destino			
24	Dirección IP destino			

Figura 4.5 Formato de un mensaje ARP

Tipo de hardware. Es un campo formado por 2 bytes en el que se especifica el tipo de interfaz física utilizada por la red, ya que ARP se puede usar en diferentes redes físicas. El tipo de hardware para las redes Ethernet es 1. En la Tabla 4.3 se muestran otros tipos de interfaces físicas existentes.

Tabla 4.3 Tipos de interfaz de hardware

Tipo	Interfaz física
1	Ethernet
2	Ethernet experimental
3	Amateur radio X.25
4	Proteon ProNET token ring
5	Chaos
6	IEEE 802 network
7	ARCNET

Protocolo. El campo *protocolo* es de 2 bytes e indica el código de protocolo de red usado. Para el caso de las redes Ethernet el protocolo de red empleado corresponde a IP, cuyo código es 0x0800 (ethertype para IP).

Longitud de la dirección hardware. Este campo consta de 1 byte en el que se especifica el tamaño (en bytes) de la dirección física de la red. Para el caso de redes Ethernet las direcciones MAC están conformadas por 48 bits, es decir 6 bytes.

Longitud del protocolo. Campo formado por 1 byte que indica el tamaño (en bytes) de la dirección de red. Para el caso de las direcciones IP su longitud es de 4 bytes. Este campo y el anterior son necesarios para el protocolo ARP porque, como se mencionó, se usa en diferentes tipos de redes físicas que manejan formatos distintos para sus direcciones de hardware.

Código de operación. Se usan 2 bytes para el código de operación del protocolo. El valor de este campo indica la operación requerida por el mensaje ARP que puede ser una solicitud (código de operación = 1), o una respuesta (código de operación = 2). Cuando un nodo reconoce su dirección IP en un mensaje ARP, verifica el código de operación para saber si se trata de un mensaje de solicitud o de la respuesta a una solicitud previamente enviada.

Dirección física origen. Este campo consta de 6 bytes y especifica la dirección física del dispositivo que genera el mensaje. Cuando una solicitud llega a un nodo la dirección física de quien envía la solicitud (contenida en este campo) se agrega a la tabla de direcciones.

Dirección IP origen. Es un campo de 4 bytes en donde se especifica la dirección de red del nodo emisor. Este valor también se agrega a la tabla ARP junto con la dirección MAC de origen especificada en el campo anterior.

Dirección física destino. Este campo consta de 6 bytes para la dirección física del dispositivo al que se requiere enviar un mensaje ARP. En el caso de las solicitudes, como aun no se conoce la dirección física del destinatario (puesto que el objetivo del mensaje es precisamente conocerla), en este campo se escribe la dirección de difusión, es decir el valor FF FF FF FF FF FF, que permite que el mensaje sea recibido por todos los miembros de la red. En el caso de la respuesta, el valor de este campo se llena con el valor del campo *dirección física de origen* contenido en la solicitud (que ahora es el destino para enviar la respuesta), y se especifica la dirección física solicitada (es decir la propia) en el campo *dirección física de origen* del mensaje de respuesta.

Dirección IP destino. Este campo consta de 4 bytes y especifica la IP del nodo remoto. En el caso de las solicitudes el valor de la dirección IP destino indica el dispositivo del que se desea conocer su dirección física. Para las respuesta, la dirección IP destino corresponde a la dirección del dispositivo que envió la solicitud.

4.6 IP – Protocolo de Internet

El Protocolo de Internet, o protocolo IP (*Internet Protocol*), se encarga de la entrega y el encaminamiento de paquetes a través de una red formada por varias redes interconectadas entre ellas. Este nivel recibe datos de la capa de transporte, los ensambla en datagramas y selecciona la mejor ruta para entregarlos. A la unidad de datos manejada por IP se le denomina *datagrama* o *paquete*.

IP utiliza un sistema de entrega de paquetes sin conexión, es un protocolo no confiable, ya que la entrega de los datos no está garantizada. En su recorrido del origen al destino un paquete probablemente tendrá que atravesar varias redes y podría perderse, retrasarse, llegar duplicado, alterado o desordenado. IP no toma medidas al respecto, toda la fiabilidad en la entrega de datos, en caso de ser necesaria, se deja al protocolo de la capa de transporte. La única seguridad que ofrece IP es un mecanismo de *checksum* (suma de verificación) aplicado sobre la cabecera y que no incluye los datos.

Debido a que IP es un protocolo sin conexión, a diferencia de los servicios orientados a conexión que establecen un camino único (circuito virtual) para todos los datos enviados a través de una conexión, los datagramas en IP se envían de manera independiente y pueden tomar caminos distintos, por lo tanto cada datagrama enviado debe contener la dirección de red del destino en el encabezado IP.

Como protocolo de nivel de red, IP proporciona los servicios de encaminamiento y entrega de paquetes entre redes. Si el destino se encuentra dentro de la misma red física que el transmisor, los algoritmos de encaminamiento no tienen mucho sentido, la información se entrega de manera directa. De no cumplirse la condición anterior, el datagrama se entrega al ruteador de la red, que debe contener información sobre la localización de otras redes y decide hacia dónde encaminar el paquete para que alcance su destino. Si este destino se encuentra en alguna red alcanzable por el ruteador, se pasa el paquete a dicha red o, en caso contrario se envía hacia otro ruteador que lo acerque más a la red a la que desea llegar. En cada ruteador se evalúa la dirección de red del paquete para implementar el algoritmo de encaminamiento. El último ruteador está conectado a la red que contiene el dispositivo final y el paquete se pasa a dicha red para ser entregado finalmente a su destino.

El tamaño máximo de un datagrama IP está limitado a 65 536 bytes. Sin embargo, las restricciones del hardware ponen un límite a la capacidad de transmisión a través de una red, cantidad a la que se denomina *Unidad Máxima de Transmisión* o MTU (Maximum Transmission Unit). El MTU es característico de cada red y define la cantidad máxima de bytes que una red física puede transportar. Esta cantidad está relacionada con la tecnología de la red, por ejemplo, el tamaño máximo de transferencia para redes

Ethernet es de 1518 bytes y para redes PPP[13] es de 575 bytes. El MTU se refiere al tamaño de las tramas, ya que es la unidad de información que se utiliza a nivel físico.

Ya que un datagrama IP viaja dentro de una trama, puede suceder que su tamaño no siempre se ajuste de manera adecuada para poder ser transportado por una red física. Cuando el tamaño de un paquete IP excede la capacidad de transferencia de una red, es necesario establecer un mecanismo que resuelva el problema. Aun cuando se resuelva para una red específica, ajustando el tamaño del datagrama de acuerdo a las limitaciones de la red donde se genera, al viajar a su destino el datagrama puede atravesar muchas otras redes, en donde quizá la consideración hecha para su tamaño no satisface las necesidades de las otras redes.

El mecanismo establecido por IP para resolver el inconveniente se denomina *fragmentación*, y consiste en descomponer un datagrama en fragmentos cuando es demasiado grande para viajar por una red física. El tamaño de cada fragmento es tal que pueda ser transportado a través de un hardware específico y cada fragmento se envía en una trama como un datagrama independiente.

Al llegar a su destino, el paquete fragmentado se reensambla, para lo cual IP debe informar al receptor si un paquete está fragmentado y en caso de ser así debe proporcionar un número de secuencia que permita al dispositivo destino reensamblar el paquete en el orden correcto antes de pasarlo a la siguiente capa.

4.6.1 Formato de datagramas IP

Los datagramas IP están formados por un encabezado que contiene información sobre el protocolo, el datagrama y los datos que provienen de los niveles superiores. El formato de los paquetes IP se muestra en la Figura 4.6, donde se representan 4 bytes en cada renglón. El encabezado está formado por los primeros 20 bytes y algunos campos opcionales de longitud variable.

<table>
<tr><td>No. de byte</td><td colspan="8">byte + 0
7 6 5 4 3 2 1 0</td><td colspan="8">byte + 1
7 6 5 4 3 2 1 0</td><td colspan="8">byte + 2
7 6 5 4 3 2 1 0</td><td colspan="8">byte + 3
7 6 5 4 3 2 1 0</td></tr>
<tr><td>0</td><td colspan="4">Versión</td><td colspan="4">Hlen</td><td colspan="8">TOS</td><td colspan="16">Longitud total</td></tr>
<tr><td>4</td><td colspan="16">Identificación</td><td colspan="4">Flags</td><td colspan="12">Desplazamiento del fragmento</td></tr>
<tr><td>8</td><td colspan="8">TTL</td><td colspan="8">Protocolo</td><td colspan="16">Checksum</td></tr>
<tr><td>12</td><td colspan="32">Dirección IP origen</td></tr>
<tr><td>16</td><td colspan="32">Dirección IP destino</td></tr>
<tr><td>20–60</td><td colspan="24">Opciones IP (opcional)</td><td colspan="8">Relleno</td></tr>
<tr><td>--</td><td colspan="32">Datos</td></tr>
</table>

Figura 4.6 Formato de un datagrama IP

A continuación se describen los campos del encabezado IP.

Versión. Campo de 4 bits que indica la versión del protocolo con la que fue creado el datagrama. Cuando un dispositivo recibe un paquete IP, debe evaluar el número de versión para saber el formato de los datos que incluye. La versión que actualmente se encuentra en uso es la número 4, pero la versión 6 ya está desarrollada. Si un dispositivo que sólo implementa la versión 4 recibiera un datagrama con el formato de la versión 6, al revisar el campo *versión* descartaría el paquete, ya que no sabría cómo interpretarlo. El formato que se utiliza en esta tesis tema corresponde a la versión 4 (IPv4).

Longitud de la cabecera (Header Length). Debido a que la cabecera no tiene un tamaño fijo, en este campo de 4 bits se indica su longitud medida en palabras de 4 bytes, para que el receptor sepa dónde termina el encabezado y comienzan los datos. El último campo de la cabecera tiene una longitud variable y es opcional. Si este campo no existe, la cabecera tiene un tamaño de 20 bytes (el campo *longitud de la cabecera* contiene el valor 5). El campo variable puede tener una longitud máxima de 40 bytes (10 palabras), por lo tanto, el valor mínimo del campo *longitud de la cabecera* es 5 y el máximo 15 (los paquetes con un campo opcional de tamaño máximo contienen 60 bytes en la cabecera).

[13] *Point to Point Protocol*. Es un protocolo usado para conectar 2 dispositivos directamente

Longitud total. Este es un campo de 16 bits que especifica la longitud total en bytes del paquete IP, incluyendo el encabezado y los datos. Al sólo contar con 16 bits para expresar la longitud de un paquete, el tamaño máximo que puede tener es de 65 535 bytes. Para conocer el tamaño de los datos se puede restar la longitud del encabezado a la longitud total.

Tipo de servicio. Se le conoce como *TOS* (*Type Of Service*). Es un campo de ocho bits que se divide en cinco subcampos para expresar el tipo de servicio que requiere un dispositivo que envía paquetes IP.

7	6	5	4	3	2	1	0
—	—	R	T	D	Prioridad		

Figura 4.7 Contenido del campo Tipo de Servicio en la cabecera de un datagrama IP

La prioridad indica la importancia que se le debe dar al datagrama, comprende valores desde el 0 (prioridad normal) hasta el 7 (usado para control). Los valores D, T y R especifican el tipo de transporte solicitado por el emisor, cuando contienen un valor de uno:

D: solicita procesamiento con retardos cortos.
T: solicita alto desempeño
R: solicita alta confiabilidad

Los valores de este campo indican qué tipo de servicio requiere la aplicación y ayudan a los ruteadores a seleccionar las rutas que mejor cumplen con el criterio indicado, por ejemplo, seleccionar una red física más rápida que las otras. Los nodos finales no ocupan estos valores, sólo los ruteadores.

Identificación. Este campo está compuesto por 2 bytes. Cada datagrama IP debe estar marcado con un número que lo identifique. Si un datagrama se fragmenta, el número de identificación permite saber a qué datagrama pertenece cada fragmento y le permite al destino reensamblarlo a partir de los fragmentos, ya que cada uno tiene el mismo número de identificación que tenía el datagrama original. El número de identificación puede ir desde el 0 hasta el 65 535.

Flags. El campo *Flags* incluye los bits 7, 6 y 5 de un byte del encabezado (consultar Figura 4.6). Los bits 5 y 6 informan el estado de la fragmentación de un paquete. Al bit 6 se le conoce como el bit de *no fragmentación*. Cuando un datagrama incluye esta bandera con el valor lógico 1, se le informa a los ruteadores que el paquete no debe ser fragmentado, ya que sólo es útil si se transporta completo. Si algún ruteador en el camino está conectado a una red con MTU menor al tamaño del datagrama y el datagrama incluye la bandera de no fragmentación, el ruteador descarta el paquete por ser imposible su transmisión a través de la red física a la que está conectado.

El bit 5 se denomina bit de *más fragmentos* y sirve para indicarle al nodo final si un fragmento es el último de un datagrama fragmentado (bit 5 = 0), o si a continuación vienen más fragmentos (bit 5 = 1).

Desplazamiento del fragmento. Este campo está formado por 13 bits. En un datagrama fragmentado, cada fragmento contiene una cantidad determinada de bytes. En el campo *desplazamiento* se especifica un valor para cada fragmento que indica la cantidad de bytes que dicho fragmento se tiene que desplazar, desde el inicio, para formar el datagrama original. Por ejemplo, supongamos que un datagrama de 1400 bytes se fragmenta en tres partes para poder ser transportado por una red con tamaño de MTU = 620. El tamaño de los fragmentos se fija en 600, 600 y 200 bytes respectivamente. El primer fragmento tiene un valor de desplazamiento de 0, ya que es el inicio del datagrama; el segundo fragmento tiene un desplazamiento de 600 bytes desde el inicio del datagrama y en el tercer fragmento su desplazamiento es de 1200 bytes. El valor del campo *desplazamiento* se especifica en unidades de 8 bytes, es decir, que para los 3 datagramas anteriores sería: 0, 75 y 150. El campo longitud total de cada fragmento de datagrama indica el tamaño del fragmento, no el tamaño del datagrama sin ser fragmentado. Cuando el datagrama se reensambla se utilizan los valores de *longitud total* de cada fragmento y sus valores de *desplazamiento* para calcular el tamaño total del datagrama.

TTL (*Time To Live*). Es un campo de 1 byte que especifica el tiempo (en segundos) que un datagrama puede existir dentro de una red de redes. Cuando un ruteador recibe un datagrama IP decrementa el número de este campo en 1. Si el ruteador está muy congestionado, registra el tiempo que transcurre desde que el paquete llega hasta que es enviado y, antes de hacerlo, decrementa el tiempo de vida del paquete la cantidad de segundos que hayan transcurrido. Cuando el tiempo de vida de un datagrama llega a cero, se

descarta. El tiempo de vida puede llegar hasta los 255 segundos, el valor típico de un datagrama IP es de 64 segundos.

Protocolo. Campo formado por 1 byte que le indica al receptor qué tipo de información acarrea el paquete IP. Es un valor entero que identifica el protocolo de los datos contenidos en el paquete y puede ser TCP (6) o UDP (17). También el protocolo ICMP (1) usa IP para enviar mensajes (aunque no es un protocolo de nivel superior a IP). Cualquier otro protocolo que haga uso de los servicios de IP para enviar información debe agregar su código de protocolo en el campo *protocolo*.

Checksum. La suma de verificación es un valor de 2 bytes que se usa para asegurar la integridad del encabezado IP, sin garantizar nada sobre los datos contenidos en el datagrama. Se calcula sumando el contenido de toda la cabecera mediante el complemento aritmético a uno, tomando los datos como enteros de 16 bits contiguos. Después de realizar la suma se obtiene el complemento a uno. Para realizar el cálculo al campo de checksum se le da el valor de cero.

Dirección IP origen. Campo de 4 bytes. Dirección que identifica la red en donde se encuentra el dispositivo emisor.

Dirección IP destino. Campo de 4 bytes. Dirección que identifica la red en donde se encuentra el dispositivo al cual está dirigido el paquete.

Opciones IP. Este es un campo de longitud variable que puede no aparecer en un datagrama. Se utiliza para agregar opciones y la longitud depende de la opción que se agregue. Algunas opciones sólo ocupan un byte mientras que otras incluyen parámetros. El valor máximo del campo de opciones es de 40 bytes (o 10 palabras de 4 bytes).

Relleno. Este campo se utiliza para ajustar la longitud de las opciones a un múltiplo de 4 bytes en caso de ser necesario, ya que la longitud del encabezado se mide en palabras formadas por 4 bytes cada una. Por ejemplo si el datagrama sólo incluye una opción de un byte, el relleno se compone de 3 bytes todos iguales a cero y la longitud total de la cabecera será de 6 palabras (o 24 bytes).

Datos. El área de datos en el datagrama IP incluye la información proveniente del nivel superior.

4.6.2 Direcciones IP

Las direcciones IP son los números que identifican a cualquier nodo dentro de una red de redes. Están formadas por 32 bits, lo cual equivale a direcciones entre 0 y 4294967296. Como esta cifra es muy grande e implicaría la memorización de cantidades comprendidas dentro de ese intervalo, para un manejo más fácil, las direcciones IP se representan mediante cuatro números enteros de ocho bits, separados entre ellos por un punto, es decir, para representar la dirección 4294967296 se usa el número 255.255.255.255, (todos los bits de la dirección puestos a uno, separados en grupos de 8 bits) el cual es más fácil de manejar comparado a la cifra completa de 32 bits.

Otra ventaja de este tipo de representación es que permite separar las direcciones IP en dos partes, la primera identifica la dirección de una red (llamada *identificador de red*), y la segunda identifica la dirección de los dispositivos conectados a ella (llamada *identificador de host*). Por ejemplo, si se usan 24 de los 32 bits para expresar la dirección de la red, los otros 8 bits se utilizan para especificar el dispositivo conectado a esa red. Con lo anterior se tiene la posibilidad de direccionar 2^{24} (16777216) redes diferentes, con 2^8 (256) dispositivos cada una. Bajo esta consideración la dirección 194.163.25.11 identifica al nodo número 11 de la red 194.163.25.

ID de red																				ID de host							
1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	1	1
194								163								25								11							

Figura 4.8 Ejemplo de formato de dirección IP

Cada red reserva dos valores del identificador de host: el primero y el último, es decir, todos los bits iguales a cero y todos los bits iguales a uno. El valor de cero se reserva para referenciar a la red completa (*dirección de red*). En el ejemplo anterior la dirección de red se especifica mediante 194.163.25.0. Cuando todos los bits del identificador de host valen uno, la dirección resultante corresponde a la *dirección de difusión* de la

red (o dirección broadcast), para este ejemplo 194.163.25.255. Las combinaciones restantes pueden ser empleadas para asignar direcciones a los dispositivos conectados a la red 194.163.25.0 y van desde la dirección 194.163.25.1 hasta la dirección 194.163.25.254, es decir, en realidad sólo puede haber 254 dispositivos conectados a una red de estas y no 256.

Las direcciones IP se organizan en cinco diferentes clases de acuerdo a la cantidad de bits que se designen para el identificador de red y para el identificador de host. Los bits más significativos de la dirección se utilizan para especificar el tipo de red del que se trata. En la Figura 4.9 se muestra la distribución de bits para cada clase de direcciones. El formato de la dirección que se utilizó como ejemplo corresponde a la clase C.

Figura 4.9 Clases de direcciones IP

La clase A se designa cuando el bit más significativo de la dirección es igual a cero. Para la identificación de la red se usan 7 bits y 24 son usados para identificar al host, de manera que es posible tener 128 redes con 16777214 nodos cada una. El rango de direcciones clase A abarca desde 1.0.0.0 hasta 127.0.0.0.

Para identificar a una dirección clase B, los 2 bits más significativos deben ser 1 0. Estos 2 bits fijos, junto con los siguientes 14 se usan para el identificador de red y los últimos 16 bits para direccionar los nodos dentro de la red. La cantidad de redes posibles es 16384 con 65534 nodos cada una y van desde la dirección 128.0.0.0 hasta la 191.255.0.0.

La clase C se identifica por tener los 3 bits más significativos con la secuencia 110. Utiliza los primeros 3 bits fijos más los siguientes 21 para especificar la dirección de la red y los 8 bits restantes para identificar cada nodo. La cantidad máxima de direcciones de red clase C equivale a 2097152, con 254 nodos cada una. Las direcciones posibles que se pueden formar van de la 192.0.0.0 a la 223. 255.255.0.

La clase D se usa para especificar una dirección de red *multicast*, o multidifusión, y la clase E está reservada para usos futuros.

La dirección 0.0.0.0 la utilizan los dispositivos cuando están arrancando o no tienen una dirección asignada. Las direcciones 127.x.x.x se reservan para pruebas de retroalimentación y se les denomina *dirección de bucle local o loopback*.

Las direcciones IP son asignadas por la Corporación de Internet para Asignación de Nombres y Números (ICANN, por sus siglas en inglés). Las redes clase A se asignan a los gobiernos de los países, la clase B a empresas y la clase C a otro tipo de solicitantes.

4.6.3 Subredes

Las clases de direcciones de red mencionadas tienen longitudes fijas para el identificador de red y el identificador de host. Por ejemplo, una red clase C incluye 2097152 redes con 254 nodos cada una, según el formato mostrado en la Figura 4.9. Si se quisiera un formato de dirección que contuviera únicamente 60 nodos para cada dirección de red, la clase más aproximada sería la C, pero en este caso se tienen 254 nodos por red, y al sólo requerir 60, las demás direcciones se desperdiciarían.

Para darle un mejor aprovechamiento a la utilización del espacio de direcciones, se definen *subredes*, las cuales dividen el espacio de una dirección de red en redes con menor cantidad de nodos. Esto se logra

quitando bits al identificador de host y agregándolos al identificador de red de una dirección. Por ejemplo, para formar redes de 64 nodos cada una, una dirección clase C se puede dividir en cuattro subredes, dejando sólo 6 bits para el identificador de host y uniendo sus otros 2 bits a la parte del identificador de red. Cada una de estas subredes tendría como dirección de subred la dirección de la red (especificada en los primeros 24 bits) más la combinación formada por los 2 bits que se le quitaron al identificador de host (2 bits más significativos). Retomando el ejemplo de dirección IP clase C dado antes, las cuatro subredes resultantes serían: 194.163.25.0, 194.163.25.64, 194.163.25.128 y 194.163.25.192. Las direcciones de difusión de cada una de las subredes, equivaldrían a poner los 6 bits asignados al identificador de host a 1: 194.163.25.63, 194.163.25.127, 194.163.25.191 y 194.163.25.255. Los demás identificadores de host quedan libres para ser asignados a los nodos de la red, la cual podría incluir hasta 62 nodos.

Distribución de bits en el formato de una dirección	Byte 3								Byte 2								Byte1								Byte 0							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Formato de dirección Clase C	1	1	0	ID de red																					ID de host							
Subred 1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	x	x	x	x	x	x
	194								163								25								0							
Subred 2	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1	x	x	x	x	x	x
	194								163								25								64							
Subred 3	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	1	0	x	x	x	x	x	x
	194								163								25								128							
Subred 4	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	1	1	x	x	x	x	x	x
	194								163								25								192							

Figura 4.10 División de una dirección IP clase C en 4 subredes

Mientras mayor sea el número de bits que se le quiten al identificador de host, se pueden formar más subredes, por supuesto con menor número de nodos cada una.

4.6.4 Máscara de subred

Para diferenciar la parte de una dirección IP que corresponde al identificador de red de la parte que corresponde al identificador de host, se utiliza un término denominado máscara de *subred*. Se le llama máscara de subred ya que resulta útil cuando una dirección de red está dividida en subredes, de no ser así, la forma para saber cual parte identifica a la red y cual al host es mirar los primeros bits de la dirección e identificar de qué clase de red se trata, aunque también es válido usar una máscara.

La máscara de subred es un número entero de 32 bits con el mismo formato que la dirección IP. Los bits de la izquierda son unos y sirven para identificar la cantidad de bits que forman el identificador de red. Los bits de la derecha son ceros e indican la parte de la dirección que identifica al host. Es decir que si una dirección está formada por 26 bits de dirección de red y 6 de dirección de host, la máscara de subred debe tener los 26 bits más significativos iguales a uno y los 6 menos significativos iguales a cero, lo cual corresponde al valor 255.255.255.192.

Dirección IP	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0	0	1	0	1	1
	194								163								25								11							
Máscara de subred	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
	255								255								255								192							

Figura 4.11 Máscara de subred

Para extraer la dirección de red de una dirección IP a partir de su máscara de subred, se efectúa una operación AND entre ambas cantidades, esto provoca que los bits del identificador de red preserven su valor, mientras que los bits del identificador de host se vuelven cero:

Dirección IP	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1	0	0	1	0	1	1
Máscara de subred	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
AND	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0

Figura 4.12 Aplicación de la máscara de subred a una dirección IP

Las máscaras de subred correspondientes a las redes clase A, B y C son las siguientes:

Clase A	255.0.0.0
Clase B	255.255.0.0
Clase C	255.255.255.0

4.7 ICMP – Protocolo de Mensajes de Control de Internet

El protocolo de Mensajes de Control de Internet o ICMP (*Internet Control Messages Protocol*) es un mecanismo de reporte de errores que permite a los ruteadores informar sobre fallas en el proceso de entrega de datagramas a la fuente original. Se considera una parte fundamental del Protocolo de Internet.

Debido a que IP es un protocolo sin conexión y que la entrega de paquetes se realiza por caminos diferentes e independientes unos de otros, no se tiene control sobre la ruta que sigue cada paquete ni se sabe que ruteadores visitará. Por lo tanto, cuando surge un error, es difícil determinar en dónde se produjo y es imposible que un ruteador se comunique con el anterior, o con el que produjo el error, para poder corregirlo.

Algunos errores que podrían impedir la entrega exitosa de un datagrama a través de una red de redes son:

- Que el receptor se encuentre desconectado o apagado
- Que el tiempo de vida de un datagrama expire
- Que el ruteador se encuentre muy congestionado y no pueda procesar un paquete
- Que un datagrama con la bandera de *no fragmentar* intente cruzar una red con MTU menor a su tamaño

Los mensajes ICMP viajan encapsulados dentro de datagramas IP, ya que ICMP opera sobre una red de redes y utiliza los servicios de IP para direccionar mensajes de regreso al nodo que expidió el datagrama que no pudo ser enviado a su destino. Pese a esto, ICMP no se considera un protocolo de mayor nivel que IP, sino que operan en conjunto al mismo nivel. Para indicar que el datagrama IP acarrea un mensaje ICMP en su área de datos, el valor del campo *protocolo* del encabezado IP debe contener el valor de 1. Al utilizar IP para entregar mensajes ICMP, también pueden ocurrir errores en su entrega, en estos casos no se genera un nuevo mensaje de error.

4.7.1 Formato de mensajes ICMP

Existen diferentes tipos de mensajes ICMP relacionados con el tipo de acción que realizan. Cada uno tiene su propio formato, aunque todos contienen los mismos campos al inicio.

No. de byte	byte + 0								byte + 1								byte + 2								byte + 3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	Tipo								Código								Checksum																
--	Campos específicos del tipo de mensaje																															
--	•••																															
--	Encabezado IP + primeros 64 bits del datagrama																															
--	•••																															

Figura 4.13 Formato general de mensajes ICMP

A continuación se describe el contenido de los campos de los mensajes ICMP.

Tipo. Este campo es de 8 bits y sirve para identificar el tipo y formato de mensaje ICMP. Los tipos de mensajes pueden ser los presentados en la Tabla 4.4.

Tabla 4.4 Tipos de mensajes ICMP

Valor del campo TIPO	Tipo de mensaje ICMP
0	Respuesta de ECO
3	Destino Inalcanzable
4	Disminución de origen
5	Redireccionar (cambiar una ruta)
8	Solicitud de ECO
11	Tiempo extendido para un datagrama
12	Problema de parámetros en un datagrama
13	Solicitud de *timestamp*
14	Respuesta de *timestamp*
15	Solicitud de información (obsoleto)
16	Respuesta de información (obsoleto)
17	Solicitud de máscara de dirección
18	Respuesta de máscara de dirección

Código. Es un campo de 8 bits que contiene información adicional sobre el tipo de mensaje, en caso de que alguno de los tipos lo requiera. Por ejemplo el tipo 3 (Destino Inalcanzable) incluye en el campo *código* un valor numérico relacionado con la causa que impidió la entrega del paquete a su destino.

Checksum. La suma de verificación de 16 bits utiliza el mismo algoritmo que el checksum de IP y abarca todos los campos del mensaje ICMP.

Todos los mensajes ICMP incluyen los tres campos descritos seguidos de otro conjunto de campos específicos del tipo de mensaje. Algunos tipos de mensajes también incluyen el encabezado del datagrama IP en el que se detectó el error, así como los primeros 64 bits de sus datos. Esto ayuda al receptor a determinar la causa del error de manera más precisa.

Uno de los mensajes ICMP más usados es la solicitud y respuesta de ECO. Estos mensajes sirven para hacer un diagnóstico y probar si es posible establecer la comunicación entre dos nodos a nivel de red. El dispositivo que desea saber si se puede comunicar con otro dispositivo le envía a este último una *solicitud de ECO*, en la que puede incluir un conjunto de datos. Si la solicitud logra llegar hasta el destino, el receptor envía una *respuesta de ECO* al transmisor original e incluye los datos enviados en la solicitud, si los hubiera. Este mecanismo permite cerciorarse de que todo el proceso de comunicación en la red de redes funciona adecuadamente, pues implica que el transmisor y el receptor, así como los ruteadores que forman parte del sistema, son capaces de direccionar y encaminar paquetes de manera correcta a través de una red de redes, que el hardware de los dispositivos involucrados funciona adecuadamente y que el software que implementa los protocolos IP e ICMP también funciona de manera correcta.

En la Figura 4.14 se muestra el formato de los mensajes de solicitud y respuesta de ECO. El campo *tipo* debe contener el valor de 8 si el mensaje es de solicitud, o 0 si es de respuesta. El campo de *código* no se usa y debe contener el valor de cero. Los campos de *identificador* y *número de secuencia* se usan para identificar las solicitudes enviadas y sus correspondientes respuestas recibidas.

No. de byte	byte + 0								byte + 1								byte + 2								byte + 3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	Tipo (8 ó 0)								Código (0)								Checksum															
4	Identificador																Número de secuencia															
– –	Datos opcionales																															
– –	...																															

Figura 4.14 Formato de un mensaje ECO ICMP

Existe un comando disponible en los sistemas operativos para verificar la comunicación en una red. En la mayoría de estos sistemas el comando recibe el nombre de *Ping*, cuya función es enviar una solicitud de ECO a un destino específico, esperar la respuesta e imprimir información a cerca de la comunicación. Si la respuesta no regresa, es señal de que alguna parte de la red está fallando.

Debido a la utilidad de esta herramienta de prueba, el mensaje de solicitud y respuesta de ECO ICMP se implementa para la pila del microcontrolador presentada en esta tesis. En el capítulo 6 se muestra un ejemplo de la implementación del comando Ping. Los demás tipos de mensajes ICMP no son implementados.

4.8 UDP – Protocolo de Datagrama de Usuario

El Protocolo de Datagrama de Usuario o UDP (*User Datagram Protocol*) es un protocolo de la capa de transporte, no confiable, no orientado a conexión, que no garantiza la entrega de datos a través de una red, no ordena paquetes que lleguen en desorden, no proporciona detección y corrección de errores ni evita duplicaciones. Este protocolo es utilizado por los programas de aplicación y es responsabilidad de ellos proporcionar la seguridad necesaria en la transferencia de información. Para intercambiar información entre equipos de una red, UDP hace uso de los servicios proporcionados por el Protocolo de Internet (IP). Los datos manejados por el protocolo UDP reciben el nombre de *datagramas de usuario*.

Algunos protocolos de aplicación que utilizan UDP son: el Protocolo Trivial de Transferencia de Archivos (TFTP), el Sistema de Archivos de Red (NFS), el Protocolo de Administración Simple de Redes (SNMP), el Servidor de Nombres de Dominio (DNS) y el Protocolo de Configuración Dinámica de Servidor (DHCP), entre otros.

4.8.1 Puertos

En un dispositivo que se comunica a través de una red pueden existir varias aplicaciones que hacen uso de las funciones de los protocolos de red para enviar y recibir información y pueden estar siendo ejecutadas a la vez. Cuando los protocolos de red procesan información que sale o entra de los dispositivos, necesitan identificar a cada una de las aplicaciones que se están comunicando. Esta tarea le corresponde al protocolo de la capa de transporte, la cual necesita establecer algún mecanismo para identificar a todas las aplicaciones que hacen uso de ella. La idea anterior se ilustra en la Figura 4.15.

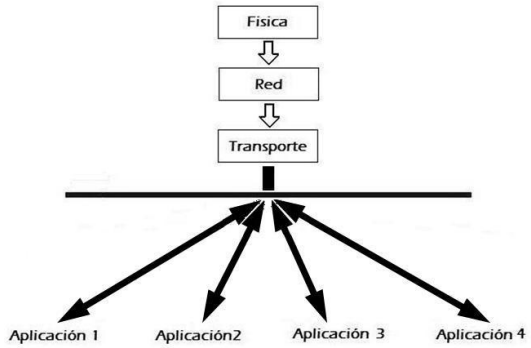


Figura 4.15 Comunicación de la capa de transporte con varias aplicaciones para enviar y recibir datos

En la capa de transporte el direccionamiento empleado se denomina *número de puerto* y es análogo a las direcciones empleadas en la capa de enlace y en la capa de red. De la misma manera en que el nivel de enlace hace uso de direcciones físicas para distinguir dispositivos conectados a una red y el nivel de red utiliza direcciones de red para distinguir entre diferentes redes y encaminar paquetes por una red de redes; el nivel de transporte utiliza los números de puerto para entregar información a aplicaciones específicas dentro de un dispositivo. Los puertos son la interfaz entre la capa de transporte y la capa de aplicación.

Cuando se genera información en un nodo origen, se debe especificar a qué aplicación del nodo destino está dirigida, indicando el número de puerto con el que se identifica la aplicación remota. El software de la capa de transporte en el lado receptor, asocia el número de puerto con una aplicación local, a la que hace entrega de la información. En la Figura 4.16 se ejemplifica cómo usa los números de puerto la aplicación destino.

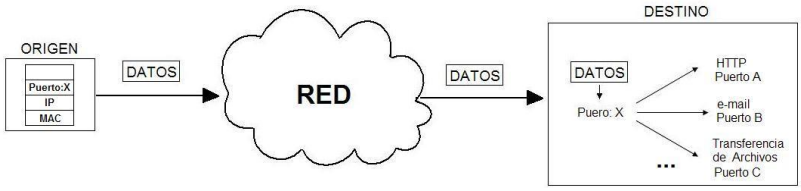


Figura 4.16 Uso de los números de puerto para identificar aplicaciones que corren dentro de un dispositivo

Para entender mejor la idea anterior imaginemos una aplicación X ejecutándose en un dispositivo que se comunica con tres aplicaciones A, B y C al mismo tiempo, y que cada una de ellas envía una información a, b y c, respectivamente, a la aplicación X, la cual enviará una respuesta a cada una de las aplicaciones emisoras. Los dispositivos que transmiten deben indicar a qué aplicación están enviando información, colocando en el paquete el número de puerto con el que se identifica la aplicación receptora, así como su propio número de puerto. Después de procesar la información la aplicación X genera una respuesta para cada una de las aplicaciones remotas y pasa las tres respuestas a la capa de transporte, indicando en cada una el número de puerto de la aplicación a la que le está respondiendo. La Figura 4.17 ilustra este ejemplo.

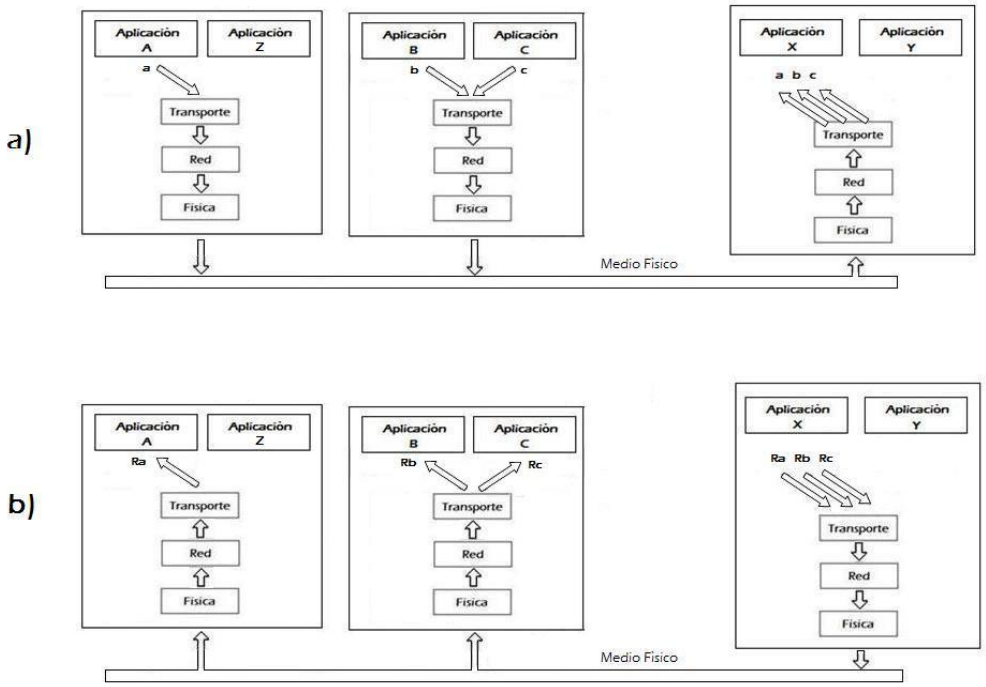


Figura 4.17 Ejemplo de uso de puertos para varias aplicaciones. a) Las aplicaciones que envían datos indican el número de puerto del receptor. b) La aplicación que envían una respuesta indica los números de puerto de cada una de las aplicaciones emisoras

La asignación de números de puerto depende del tipo de aplicación. Una aplicación que va a recibir información de otras aplicaciones siempre debe tener un número de puerto local asignado (que será visto como puerto remoto o puerto destino para las aplicaciones que le envíen información). Aquellas aplicaciones que son comúnmente usadas o estandarizadas bajo algún protocolo y conocidas por muchos usuarios, están asociadas a números de puerto conocidos (denominados puertos bien conocidos). Estos puertos son administrados por la Autoridad Internet de Números Asignados (IANA por sus siglas en inglés) y se encuentran documentados en el RFC 1700 "Números Asignados", el cual contiene una lista completa de puertos para servicios establecidos. En la Tabla 4.5 se muestran ejemplos de este tipo de aplicaciones. Las aplicaciones que no cuentan con un número de puerto establecido pueden ocupar números de puerto que no estén asociados a ninguna aplicación conocida y que se encuentren disponibles en el dispositivo al momento de la comunicación.

Los puertos son números enteros de 2 bytes (o 16 bits), por lo tanto, pueden abarcar desde el 0 hasta el 65535 y dividen de la siguiente manera:

- Los números del 0 al 1023 están reservados para aplicaciones conocidas y no pueden ser usados por otras aplicaciones (ejemplo HTTP puerto 80).
- Los números del 1024 al 49151 son puertos registrados (ejemplo MySQL[14] puerto 3306).
- Los números del 49152 al 65535 son puertos dinámicos que pueden usar las aplicaciones que se ejecutan dentro de un dispositivo para identificarse entre ellas, cada una debe solicitar al sistema (al sistema operativo por ejemplo) un número de puerto que no se encuentre en uso.

Tabla 4.5 Ejemplos de números de puerto asignados

Protocolo	Número de puerto	Aplicación
FTP	20	Transferencia de archivos
Telnet	23	Acceso remoto
SMTP	25	Correo electrónico
DNS	53	Servidor de nombres de dominio
HTTP	80	Transferencia de páginas web

Un ejemplo del empleo de puertos sería una aplicación de usuario que se comunica con un servidor HTTP, el puerto local se le solicita al sistema al momento de establecer la comunicación, supongamos que se obtiene el puerto 50000, y el puerto remoto corresponde al número 80 asociado a la aplicación HTTP.

Tanto el protocolo UDP como el protocolo TCP emplean los números de puerto como mecanismo de direccionamiento entre la capa de transporte y la capa de aplicación.

En el protocolo de datagramas de usuario el uso del número de puerto local no es obligatorio pero el número de puerto remoto sí lo es. Debido a su naturaleza sin conexión el protocolo UDP puede enviar información en un sólo sentido sin esperar confirmaciones ni algún otro tipo de información de respuesta. En tal caso, la aplicación emisora puede omitir su número de puerto local, ya que la aplicación receptora no lo necesitara porque nunca le enviará ninguna información de regreso. Si las necesidades de la aplicación requieren que el envío de la información sea en ambas direcciones, cada una de las aplicaciones emisora y receptora debe contar con un número de puerto que las identifique.

4.8.2 Formato de datagramas UDP

En la Figura 4.18 se muestra el formato de un datagrama de usuario con 4 bytes en cada renglón y a continuación se describe cada uno de sus campos.

No. de byte	byte + 0								byte + 1								byte + 2								byte + 3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	Puerto UDP origen																Puerto UDP destino															
4	Longitud del mensaje UDP																Checksum UDP															
– –	Datos																															

Figura 4.18 Formato de datagramas de usuario

Puerto UDP de origen. Es un campo de 2 bytes y especifica el número de puerto que el emisor está empleando para enviar información de una aplicación local a una remota. El número de puerto origen es opcional, por ejemplo, si una aplicación sólo necesita enviar información sin esperar respuesta, su número de puerto local no necesariamente debe estar definido, puede ser cero. En cambio, si se espera que el receptor envíe una respuesta, debe especificarse un número de puerto de origen para que el dispositivo destino sepa hacia qué puerto direccionar la respuesta.

Puerto UDP destino. Es un campo de 2 bytes en el que se especifica el número de puerto asociado a la aplicación receptora.

[14] Servicio de base de datos

Longitud del mensaje UDP. Es un campo formado por 2 bytes que contiene el número de bytes del datagrama de usuario completo, es decir, incluye los campos del encabezado y los datos provenientes de la aplicación. La longitud mínima del datagrama es 8 bytes, correspondiente a la longitud del encabezado.

Checksum. Como en otros niveles, el valor de la suma de verificación es un número de 2 bytes. Se emplea el mismo mecanismo de verificación de datos que en el Protocolo de Internet: se hace una suma de los datos en complemento a uno, considerándolos cantidades de 16 bits, y se obtiene el complemento a uno del valor resultante. En el caso del protocolo UDP se decidió que el cálculo de la suma de verificación fuera opcional, si no se realiza, este campo debe contener el valor de cero. En el caso de realizar el cálculo, deben incluirse, además de los datos del encabezado UDP y los datos de la aplicación, algunos campos que en realidad forman parte del encabezado IP. A estos datos tomados del encabezado IP y agregados al datagrama UDP para obtener el checksum se le denomina *pseudo–encabezado*.

El *pseudo–encabezado* incluye la dirección IP de destino, la de origen, el número de protocolo que interactua con IP (17 para UDP), un campo de 16 bits que contiene la longitud del datagrama UDP (sin considerar el *pseudo–encabezado*) y un campo de 1 byte que contiene sólo ceros, usado para ajustar la longitud de los datos a un múltiplo de 16 bits. En la Figura 4.19 se muestra la distribución de los campos del *pseudo–encabezado* utilizado por UDP.

No. de byte	byte + 0								byte + 1								byte + 2								byte + 3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	Dirección IP origen																															
4	Dirección IP destino																															
8	CERO												Protocolo												Longitud UDP							

Figura 4.19 Formato del pseudo-encabezado UDP

4.9 TCP – Protocolo de Control de Transmisión

El Protocolo de Control de Transmisión o TCP (*Transmission Control Protocol*) es un protocolo Full–Duplex, orientado a conexión, proporciona un servicio confiable que garantiza la entrega de datos extremo a extremo y mantiene una comunicación entre el emisor y el receptor, sin preocuparse de cómo un paquete cruza una red para llegar a su destino, ya que de esto se encarga el Protocolo de Internet sobre el cual trabaja TCP, aunque en realidad es un protocolo independiente y puede implementarse sobre cualquier otro sistema de red. Debido a que el nivel de red no es confiable, TCP se encarga de asegurar la integridad de los datos, garantizar la transmisión de manera exacta y en la secuencia apropiada, detectar y corregir errores, retransmitir datos que se pierden en el camino, descartar paquetes que por algún error lleguen duplicados y ordenar paquetes que lleguen en desorden. Al proporcionar todas estas funciones a la comunicación, la implementación del protocolo se vuelve compleja, pero libera a las aplicaciones de la tarea de asegurar la entrega de los datos y en consecuencia su trabajo se ve reducido.

Las aplicaciones que requieren un alto grado de confiabilidad en la entrega de datos pueden hacer uso de un servicio de conexión fiable en la capa de transporte. Este servicio fue diseñado para ahorrar trabajo a los programadores de aplicaciones, de manera que sea posible que muchas aplicaciones puedan ser construidas sobre una misma capa de transporte que proporcione una interfaz uniforme para el servicio de transferencia de flujo que evite que los programas tomen la responsabilidad de solucionar problemas de seguridad y separe las funciones de los programas de aplicación de los detalles del trabajo de la red.

Entre los protocolos de aplicación que se basan en TCP se pueden mencionar el Protocolo de Transferencia de Archivos (FTP), el Protocolo Simple de Transferencia de Correo (SMTP), el Protocolo de Oficina Postal (POP), Telnet y el Protocolo de Transferencia de Hipertexto (HTTP), entre otros.

Entre la capa de transporte y la de aplicación, la transferencia de información se define como *flujo de bits,* donde se agrupa a los bits en conjuntos de ocho a los que se denomina octetos o bytes. El flujo que una aplicación recibe de la capa de transporte en el destino, corresponde exactamente al flujo que una aplicación entrega a la capa de transporte en el origen. La unidad de datos manejada por TCP se denomina *segmento*. Cada segmento viaja por la red encapsulado dentro de un datagrama IP.

TCP acomoda los bytes del flujo que generan las aplicaciones dentro de espacios temporales de memoria en donde se acumulan y permanecen esperando hasta que puedan ser enviados. El protocolo TCP organiza la transferencia de información en bloques, de manera transparente para la aplicación. En el lado receptor también se tiene una memoria temporal para almacenar ordenadamente los datos que van llegando y ahí esperan hasta ser entregados a la aplicación correspondiente.

TCP divide el flujo de datos para ensamblar segmentos, si la memoria de transmisión no cuenta con suficientes datos, espera hasta recibir más para enviarlos todos en un mismo segmento, en lugar de enviar cada porción en un segmento separado que contenga pocos datos. Esto logra que el desempeño de TCP sea eficiente, pero también puede causar conflictos con ciertas aplicaciones. Por ejemplo en una aplicación en la que se capturan caracteres tecleados por un usuario y se transmiten a una aplicación remota que realiza alguna acción vinculada al carácter que recibe. En este caso, la entrega de un carácter se debe realizar al momento que éste se genera, enviando un segmento por cada carácter sin esperar a que en la memoria de transmisión se acumulen varios caracteres para enviarlos en un solo segmento. Las aplicaciones con características similares a esta pueden solicitar un *empuje* (o *push*) de los datos, lo cual le indica al transmisor que los datos deben enviarse sin demora y en el extremo receptor que la entrega a la aplicación también debe ser inmediata.

Por el lado contrario, cuando el flujo de información es grande, los datos se acumulan en la memoria de transmisión y se dividen para entregarse en segmentos separados. Cada byte en la memoria debe permanecer ahí hasta que sea su turno de ser enviado. Si una aplicación genera datos que deben ser transmitidos de manera urgente sin esperar dentro de la memoria hasta que se envíen los datos que están antes, lo solicita mediante un *señalamiento urgente*. En el extremo receptor dicho señalamiento indica que los datos deben entregarse a la aplicación tan pronto como se reciben independientemente de su posición en la memoria de recepción, es decir, si hay datos esperando a ser entregados los datos con la condición de urgentes tienen prioridad y se entregan primero.

4.9.1 Confiabilidad en la entrega y recepción de datos

TCP utiliza números de secuencia en todos los segmentos transmitidos para asegurar la entrega de todos ellos y ponerlos en el orden adecuado antes de pasarlos a la aplicación. Por cada segmento transmitido TCP espera recibir un segmento de confirmación que contiene un número de acuse de recibo, el cual indica que el segmento previamente enviado fue recibido y correctamente validado en el extremo receptor. Por cada segmento que se envía con un número de secuencia x, la confirmación contiene el número siguiente x+1, indicando que el segmento x (y por ende todos los anteriores) se recibió de manera exitosa y la próxima vez se espera recibir el segmento siguiente (x+1). Cada vez que se transmite un segmento que contiene un número de confirmación, se indica en el encabezado TCP mediante un bit denominado ACK (bit de confirmación o *Acknowledge*) que contiene el valor de uno.

TCP maneja una comunicación Full–Duplex, lo que permite el intercambio de información entre dos dispositivos de una red en ambas direcciones y en cualquier instante. Debido a lo anterior, cada uno de los dispositivos cuenta con un número de secuencia propio e independiente. Por ejemplo, un dispositivo empieza la secuencia con el número *x,* continúa con *x+1, x+2* …x+n y el otro inicia con el número de secuencia *y*, continúa con *y+1*, *y+2*…y+n.

En cada segmento en el que se envían datos se puede incluir confirmación, esto es, en un mismo segmento se envía información y se incluye la confirmación de datos previamente recibidos. En la Figura 4.20 se ilustra el procedimiento de envíos y confirmaciones entre dos nodos de una red. Las líneas verticales representan el transcurso del tiempo [1]. El primer evento se genera en el nodo A que envía un segmento con secuencia *x* al nodo B. Al llegar el segmento al nodo B, éste envía a A un segmento con secuencia *y* en el que incluye una confirmación del segmento *x* recibido, indicándole al nodo A que el próximo segmento que envíe debe estar numerado con la secuencia *x+1*. El procedimiento se repite mientras los nodos tengan datos que enviar. El número indicado como *ack* en la Figura 4.20 no debe confundirse con el bit de confirmación que indica que el segmento contiene una confirmación en su encabezado; los números *sec* y *ack* de la corresponden a los valores de los campos *Número de Secuencia* y *Número de Confirmación* del encabezado TCP. Por ejemplo, el primer segmento que envía A no contiene ningún número de confirmación, por lo tanto, el bit ACK estaría puesto a cero.

Cada vez que se envía un segmento por parte de alguno de los participantes, el software del protocolo TCP espera hasta que el otro extremo envíe la confirmación, activando un contador. Si el tiempo del contador se agota y no se ha recibido la confirmación, TCP asume que el segmento se extravió o dañó en el camino y lo

vuelve a enviar. Esta es la característica fundamental de TCP que garantiza la entrega de datos; si un paquete no se recibe, la confirmación tampoco será entregada, si se recibe un paquete defectuoso, se desecha y no se envía confirmación. En cualquiera de estos casos, el transmisor envía nuevamente el paquete. El uso de los números de secuencia también sirve para evitar la duplicidad, si un segmento de confirmación se extravía en el camino, el transmisor asume que el paquete no fue recibido y lo retransmite, el receptor reconoce que el paquete está duplicado y lo desecha.

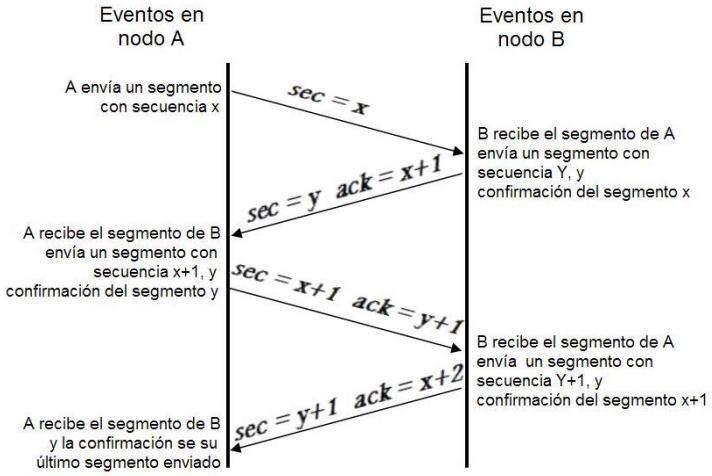


Figura 4.20 Procedimiento de envíos y confirmaciones de segmentos TCP

4.9.2 Etapas de la comunicación

El proceso de comunicación TCP consta de tres etapas: 1) establecimiento de una conexión, 2) intercambio de información y 3) cierre de la conexión.

4.9.2.1 Establecimiento de la conexión TCP

Antes de comenzar el intercambio de información en el nivel de transporte, se debe establecer una *conexión lógica*, llamada así porque no es una conexión directa por hardware, pero se toma como si lo fuera. El dispositivo que tiene necesidad de enviar o solicitar información (cliente) contacta al dispositivo con el cual desea comunicarse (servidor) y solicita que se establezca una conexión entre ambos. Si el servidor acepta comienza el intercambio de información a través de dicha conexión.

Al establecimiento de la conexión entre dos dispositivos se le denomina *proceso de sincronización*. La forma de indicar que la comunicación se encuentra en el estado de sincronización es mediante un bit del encabezado TCP que se denomina SYN. Este bit debe ser puesto a 1 cada vez que se intercambia un segmento de sincronización. El establecimiento de la conexión se realiza en tres pasos, ya que se intercambian tres segmentos TCP entre un cliente que desea establecer comunicación y un servidor que la acepta:

- El cliente envía una solicitud de conexión, para lo cual genera y transmite un segmento TCP con el bit de sincronización puesto a 1 y un número de secuencia inicial *x* que le indica al servidor en qué número comenzará la secuencia de sus segmentos.

- El servidor registra el número de secuencia inicial del cliente, si acepta la comunicación, genera y transmite un segmento de confirmación que contiene los bits SYN y ACK puestos a uno. El número de confirmación que el servidor incluye en el segmento de respuesta corresponde al siguiente número de secuencia que debe enviarle el cliente (x+1). También incluye en el segmento de confirmación el número de secuencia inicial que utilizará para enviar datos a través de la conexión, por ejemplo el número *y*.

- El cliente envía un segmento de confirmación al servidor con el bit ACK puesto a uno y el valor del número de confirmación correspondiente al siguiente número de secuencia que espera recibir del servidor, es decir, *y+1*. En esta etapa ya se ha establecido la conexión TCP, nótese que en los

segmentos intercambiados durante el establecimiento de la misma no se incluye ningún dato, sólo los valores específicos del encabezado TCP necesarios para establecer la comunicación. A este tipo de segmentos se les denomina *segmentos de control TCP*. Una vez establecida la conexión los segmentos de datos pueden intercambiarse entre el cliente y el servidor.

Para que el esquema de conexión mencionado se pueda llevar a cabo, es necesario establecer dos tipos de conexiones, una corresponde a la que establece el servidor y la otra la establece el cliente. Un servidor espera recibir conexiones a través de un puerto específico, para lo cual abre una *conexión pasiva* y se dice que el servidor está *escuchando* (o esperando) conexiones entrantes. La *conexión activa* es realizada por el cliente al enviar una solicitud para establecer comunicación con un servidor que espera recibir peticiones por un puerto.

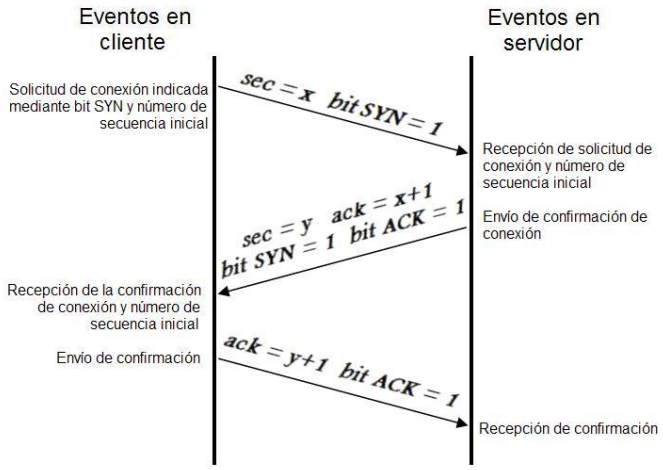


Figura 4.21 Establecimiento de la conexión en tres pasos

4.9.2.2 Intercambio de información y control de flujo

En la etapa de intercambio de información tanto el cliente como el servidor envían segmentos que contienen datos, mismos que se almacenan en espacios temporales de memoria. En cada extremo de un enlace, TCP mantiene buffers de recepción y transmisión. El buffer de transmisión se llena con los datos que generan las aplicaciones y contiene dos conjuntos: datos que ya fueron enviados pero aún no han sido confirmados y datos que están esperando para ser enviados. El primer conjunto no se puede descartar hasta que la confirmación correspondiente sea recibida, ya que si la confirmación no llega, deben ser reenviados. Una vez confirmados, los datos se borran de la memoria temporal para liberar espacio. El buffer de recepción también contiene dos conjuntos de datos: datos recibidos que ya fueron ordenados y están listos para ser entregados a la aplicación y datos recibidos y validados correctamente pero que aún necesitan ordenarse para pasarse a la aplicación.

Con el propósito de hacer más eficiente el desempeño de TCP, los segmentos de datos se envían en grupos en lugar de enviarse de uno en uno y esperar las confirmaciones entre uno y otro. La cantidad de bytes que se pueden enviar en algún momento está dada por el tamaño de una *ventana*, tal como lo define el algoritmo de ventana deslizante que se explicó en el capítulo 2. En la Figura 4.22 se reproduce el mecanismo de ventana deslizante de la Figura 2.21.

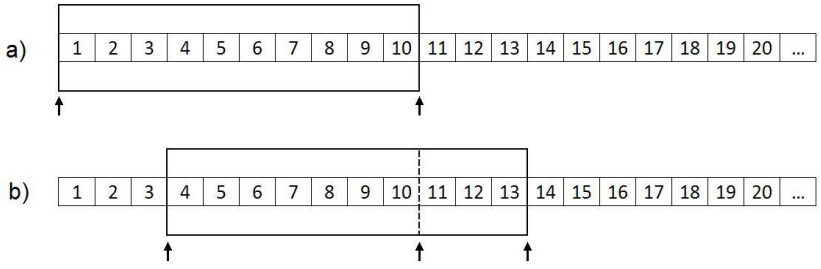


Figura 4.22 Ejemplo de ventana deslizante

Con el fin de controlar el envío de los datos, TCP mantiene apuntadores a las localidades del buffer (representados con flechas en la Figura 4.22) que indican en dónde empiezan los datos transmitidos que aún no son confirmados (flecha izquierda), a partir de dónde se puede comenzar a enviar más datos (flecha en medio) y hasta dónde se termina de enviar datos (flecha derecha).

El algoritmo de ventana deslizante se implementa en TCP con una mejora: el tamaño de la ventana no es fijo como se muestra en la Figura 4.22, sino que es dinámico, es decir, la ventana de transmisión cambia su tamaño con el tiempo, basado en un tamaño de ventana anunciado por el receptor.

El receptor debe indicarle al transmisor el tamaño de su ventana de recepción cada vez que envía un segmento. El emisor ajusta el tamaño de su ventana de transmisión de acuerdo al dato especificado por el receptor. Si retomamos el ejemplo de la Figura 4.22 podemos observar que el transmisor tiene una ventana de transmisión de tamaño 10 paquetes (aunque el valor se especifica en bytes). Supongamos que desde el extremo receptor se anuncia una ventana de recepción de tamaño 5. Bajo esta situación el transmisor cambia su tamaño de ventana a 5 para enviar solamente los 5 paquetes que el receptor puede aceptar. Conforme el espacio del buffer de recepción se va llenando o liberando, el tamaño de la ventana de recepción se va ajustando dinámicamente y, en consecuencia, también el de la ventana de transmisión. Esta característica permite mantener un control de flujo, enviando sólo la cantidad de información que un dispositivo puede almacenar en un momento dado.

Cuando un buffer de recepción se llena, se envía al transmisor un segmento con el valor de tamaño de ventana igual a cero, lo cual obliga al emisor a detener su transmisión. Esta situación genera que el buffer de transmisión del emisor también se llene, ya que seguirá recibiendo datos de las aplicaciones sin poder liberar espacio. En este punto el transmisor bloquea a la aplicación impidiendo que genere más datos hasta que la comunicación se restablezca. Cuando se libera espacio en el buffer de recepción del nodo receptor, éste envía un segmento con un tamaño de ventana diferente de cero, con lo que se logra que el emisor continúe su transferencia y comience a liberar espacio en su buffer de transmisión, que a su vez permite que la aplicación siga generando datos.

4.9.2.3 Cierre de la conexión

Cuando un dispositivo no tiene más datos que transmitir, notifica al otro dispositivo que desea terminar la conexión. El dispositivo que solicita el término de la comunicación cierra la conexión de su lado con el envío un segmento de finalización y ya no se le permite enviar más datos, aunque continúa recibiendo datos y enviando confirmaciones si el otro dispositivo aún tiene datos que enviar. Cuando el segundo dispositivo termina de enviar su información, envía su segmento de finalización, terminando así la comunicación. La forma de identificar un segmento de finalización es mediante un bit del encabezado TCP denominado FIN, el cual debe contener el valor de uno (ver Figura 4.23).

La secuencia de finalización de la conexión se describe a continuación, donde se asume que existe una conexión establecida entre un nodo A y un nodo B, que han estado intercambiando datos y que el nodo A es el que termina de enviar sus datos antes que B (podría ser al revés):

- El nodo A envía un segmento de finalización con el bit FIN puesto a 1 y su número de secuencia correspondiente.
- El nodo B envía un segmento para confirmar la finalización de la conexión por parte de A. En este momento A no puede seguir enviando más datos a B, ahora la conexión está establecida en una sola dirección.
- B continúa enviando datos hacia A con su número de secuencia; A continúa enviando segmentos de confirmación a B con el bit ACK puesto a 1 y el número de confirmación correspondiente.
- B no tiene más datos que enviar, así que envía un segmento de finalización con el bit FIN puesto a 1 y su número de secuencia correspondiente.
- A confirma la desconexión de B y envía un segmento de confirmación con el número de confirmación y el bit ACK puesto a 1. En este momento la conexión entre A y B está completamente cerrada.

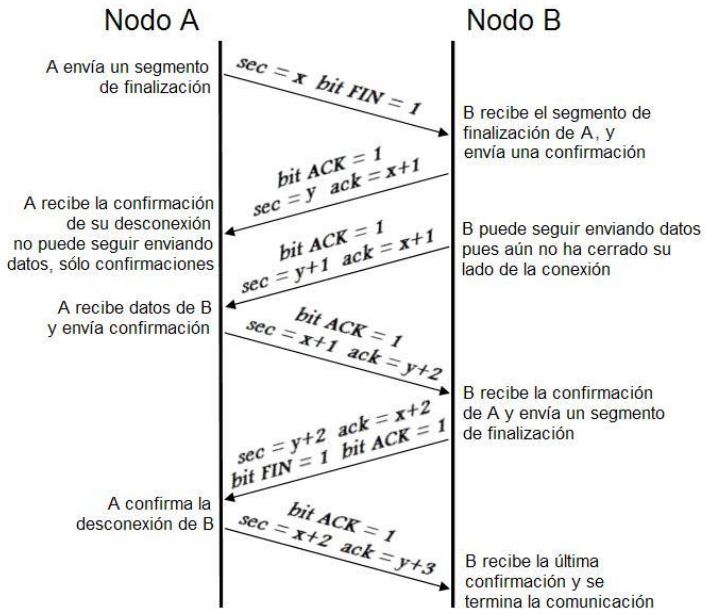


Figura 4.23 Esquematización del proceso de finalización de una conexión

4.9.2.4 Ejemplo de comunicación TCP en sus tres etapas

En la Figura 4.24 se ejemplifica el proceso de comunicación TCP entre dos dispositivos: un cliente y un servidor. El ejemplo fue tomado del libro *Comunicación entre computadoras y tecnología de redes* de los autores Michael A. Gallo y William M. Hancock [3]. El proceso de establecimiento de comunicación de tres pasos que se usa para establecer una conexión se ilustra en los pasos del 1 al 3 y la terminación del enlace está ilustrada en los pasos del 7 al 12. Los autores manejan las confirmaciones como reconocimientos en algunos puntos del esquema y a los bits de sincronización y confirmación los llaman *banderines*.

1. A envía un segmento de sincronización a B con el cual indica su deseo de establecer una conexión y que su número de secuencia inicial es 300. Esto significa que el primer segmento de datos que A envíe será numerado 301.
2. B recibe el segmento de sincronización de A y envía un segmento de sincronización y confirmación. El número de secuencia inicial de B es 800, lo que significa que el primer segmento de datos de B será numerado 801.
3. A recibe el segmento de sincronización y confirmación de B y envía una confirmación. En esta etapa queda establecida una conexión TCP entre A y B.
4. A transmite el segmento de datos 301 e informa a B que está esperando el segmento numerado 801.
5. B recibe el segmento de A y envía el segmento de datos 801. Este segmento también confirma la recepción del segmento de A, informando que espera recibir el segmento 302.
6. A recibe el segmento y envía a B el segmento de datos 302, que también confirma la recepción del segmento de B.
7. A envía a B un segmento de término que informa que A está cerrando su lado de la conexión TCP.
8. B recibe los dos últimos segmentos de A y envía el segmento de datos 802. Este segmento también confirma la recepción de las transmisiones previas de A fijando el número de confirmación en 304. En esta etapa A no puedo transmitir ningún segmento nuevo de datos, pero continúa transmitiendo segmentos de confirmación.
9. A confirma la última transmisión de B.
10. B recibe el segmento de confirmación de A y envía el segmento de datos 803.
11. B envía a A un segmento de término que informa que está cerrando su lado de la conexión TCP.
12. A recibe las últimas transmisiones de B y las confirma. En esta etapa el enlace está terminado, ya que ni A ni B tienen más datos que transmitir.

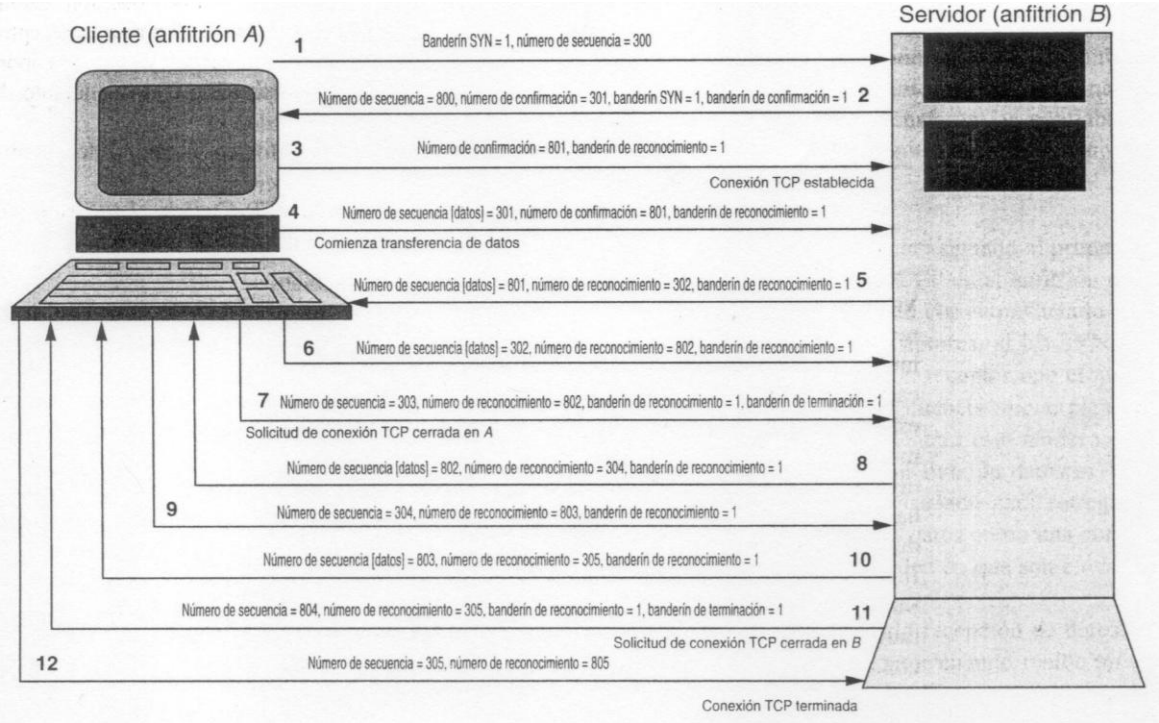


Figura 4.24 Descripción gráfica de una conexión TCP

4.9.4 Formato de segmentos TCP

En la Figura 4.25 se muestra el formato de un segmento TCP en el cual se representan 4 bytes por cada renglón y a continuación se define el contenido de cada uno de los campos del encabezado.

No. de byte	byte + 0								byte + 1								byte + 2								byte + 3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	Puerto TCP origen																Puerto TCP destino															
4	Número de secuencia																															
8	Número de confirmación																															
12	HLEN				Reservado						URG	ACK	PSH	RST	SYN	FIN	Ventana															
16	Checksum																Señalador de urgencia															
20–60	Opciones TCP (opcional)																								Relleno							
--	Datos																															

Figura 4.25 Formato de segmento TCP

Puerto TCP origen. Campo de 2 bytes. Número de puerto que identifica a la aplicación que envía un mensaje.

Puerto TCP destino. Campo de 2 bytes. Número de puerto que identifica la aplicación que va a recibir un mensaje.

Número de secuencia. Campo de 32 bits que especifica el número de secuencia de los segmentos transmitidos. El número de secuencia inicial se indica en el establecimiento de la conexión, puede ser arbitrario, pero generalmente se usa el 0. Tanto el receptor como el transmisor mantienen sus propios números de secuencia independientes. En los segmentos TCP se enumeran los bytes de datos. Si el segmento es de control la secuencia se incrementa en uno, correspondiente al número de secuencia asignado al segmento. Si el segmento contiene datos, el número de secuencia se refiere a la numeración del primer byte de los datos y se incrementa en función de la cantidad de bytes que contenga el segmento.

Número de confirmación. Campo de 32 bits que usa un receptor para enviar confirmaciones de segmentos recibidos, su valor depende del número de secuencia usado por el transmisor. Cuando se recibe un segmento que contiene datos, su número de secuencia indica el valor asignado al primer byte de los datos. El receptor cuenta la cantidad de bytes recibidos exitosamente, incrementa el valor del número de secuencia y lo asigna a su número de confirmación. Por ejemplo, si se recibe un segmento con número de secuencia 3 y contiene 5 bytes, el número de confirmación es 8, lo que indica al transmisor que los bytes anteriores al 8 se recibieron exitosamente y en la próxima entrega espera encontrar el número de secuencia 8.

HLEN (Header Length). En este campo de 4 bits se indica la longitud total que abarca el encabezado del segmento TCP, ya que el último campo es de longitud variable y además opcional. La longitud se mide en palabras de 4 bytes. Si el segmento no incluye el último campo, la longitud de la cabecera es de 20 bytes (o 5 palabras), como todos los demás campos son de longitud fija, esta es la longitud mínima del encabezado. La longitud máxima del campo opcional es de 40 bytes (10 palabras), por lo que la longitud máxima del encabezado TCP es de 60 bytes (15 palabras).

Reservado. Los 6 bits del campo reservado no se usan, están reservados para uso futuro y deben contener el valor de cero.

Control. Los siguientes 6 bits se denominan bits de codificación y conforman un campo de control. Estos bits se usan para indicar qué tipo de información contiene el segmento. Se les suele llamar *banderas* y son válidas cuando contienen el valor de uno. A continuación se describe la función que realizan:

- **URG** – Indicador de segmento urgente, indica que el campo *señalador de urgencia* contiene datos válidos.
- **ACK** – Indica que el campo de *número de confirmación* contiene una confirmación válida.
- **PUSH** – Indica que el segmento solicita una operación de empuje, los segmentos con este bit activado se deben entregar a la aplicación destino tan pronto como son recibidos. Un evento de empuje lo solicita la aplicación origen.
- **RST** – Se usa cuando surge un evento que causa una desconexión, se envía un segmento con el bit de RST y el receptor aborta la transmisión.
- **SYN** – Sincroniza números de secuencia iniciales (establece una conexión).
- **FIN** – Indica que no se tienen más datos por enviar y se cierra la conexión. Cada extremo debe cerrar la conexión de manera independiente.

Ventana. Campo de 2 bytes en donde se indica la cantidad de bytes disponibles en el buffer de recepción. La cantidad de datos que envía el transmisor no debe exceder el tamaño de ventana establecido por el receptor.

Checksum. Campo de suma de verificación de 2 bytes. El algoritmo de verificación es el mismo que se ha empleado en los otros niveles: se realiza la suma en complemento a uno de todos los datos involucrados tomándolos como enteros contiguos de 16 bits, al resultado se le saca su complemento a uno. El cálculo se realiza en el extremo emisor y se agrega al campo *checksum*, mientras que en el extremo receptor se vuelve a calcular y se compara el resultado con el realizado por el emisor.

La suma de verificación incluye el encabezado TCP, los datos y adicionalmente se le añaden algunos campos del encabezado IP, que incluyen las direcciones IP de origen y destino para verificar que los datos se entregaron al destino correcto. A los datos añadidos sobre el encabezado TCP para realizar la suma de verificación se le denomina *pseudo encabezado* y tienen el mismo formato que el *pseudo encabezado* de los datagramas UDP.

No. de byte	byte + 0								byte + 1								byte + 2								byte + 3							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	Dirección IP origen																															
4	Dirección IP destino																															
8	CERO								Protocolo								Longitud TCP															

Figura 4.26 Pseudo-encabezado TCP

El campo *protocolo* indica qué protocolo de la capa de transporte va a procesar el datagrama IP, en el caso de TCP el valor corresponde a 6. El campo *Longitud TCP* especifica la longitud total del segmento TCP incluyendo el encabezado y los datos, pero no los campos del *pseudo encabezado*. El campo con etiqueta

CERO corresponde a un byte con el valor de cero que se usa para ajustar la longitud del *pseudo encabezado* a un múltiplo de 16 bits con el fin de realizar la suma de números enteros de 16 bits.

Señalador de urgencia. Es un campo de 2 bytes que se usa cuando un segmento contiene datos urgentes, indicados por el bit URG. En este campo se indica la posición de los datos del segmento en donde terminan los datos urgentes y continúan los datos normales.

Opciones TCP. Este campo es opcional y tiene una longitud variable. El tamaño máximo es de 40 bytes y depende de la opción que se incluya en el segmento. Las opciones añaden funcionalidad al protocolo pero no siempre son utilizadas.

Relleno. El campo de relleno se utiliza cuando se usa el campo *opciones* y sirve para ajustar el tamaño total del encabezado a un múltiplo de 4 bytes para que la longitud en bytes pueda ser dividida entre 4 y se indique en el campo HLEN como palabras de 4 bytes. En caso de que este campo sea usado su contenido se llena con ceros.

4.10 Sockets

Como se mencionó anteriormente, los protocolos de la capa de transporte hacen uso de cantidades denominadas *números de puerto* o *puertos de protocolo* para identificar el destino final dentro de una máquina, el cual corresponde a una aplicación en específico. TCP utiliza las direcciones de puerto de manera similar a como lo hace UDP para identificar a las aplicaciones que utilizan sus servicios. Sin embargo, la idea de puerto por sí sola no explica por completo las funciones que realiza TCP en la interfaz entre la capa de transporte y la de aplicación.

En UDP basta con informar al protocolo a qué aplicación se dirige un mensaje. En TCP se identifican las conexiones más que los puertos por sí mismos. TCP maneja varias conexiones en una misma máquina de manera independiente. Cada conexión se identifica por un par de *puntos finales* [1].

Un *punto final* se define como un par dispositivo–puerto, mediante dos números enteros que identifican, respectivamente, la dirección IP de un dispositivo y el número de puerto de una aplicación que se ejecuta dentro de dicho dispositivo:

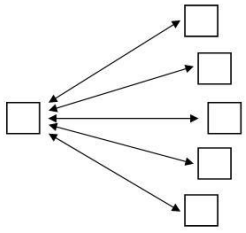
$$Punto\ Final \rightarrow (IP, puerto)$$

Por ejemplo, el punto final (194.163.25.11, 22) especifica el puerto 22 del dispositivo identificado con la dirección IP 194.163.25.11.

Una conexión siempre está definida por un par de puntos finales: una aplicación de una máquina comunicándose con otra aplicación en otra máquina.

La gran ventaja de las conexiones TCP es que no sólo es posible comunicar puntos finales uno a uno, sino que también se permiten conexiones uno a muchos, esto es, varios puntos finales pueden comunicarse con un mismo punto final a la vez, identificándose cada uno con una conexión propia. Estas conexiones permiten que un mismo puerto sea compartido por múltiples conexiones en la misma máquina.

a) Conexión uno a uno



b) Conexión uno a muchos

Figura 4.27 Capacidad de TCP para comunicar varios puntos finales con un mismo punto final

Si dos o más puntos finales se comunican con la misma máquina a través del mismo puerto, TCP asocia los mensajes entrantes con las conexiones en vez de asociarlos directamente con el número de puerto, de hacerlo así resultaría ineficiente ya que todas las conexiones comparten el mismo número de puerto de destino y la aplicación no sabría de dónde proviene cada mensaje que recibe ni a quién le debe contestar.

Esta característica de las conexiones se aprovecha para establecer el esquema de conexiones *cliente–servidor.* Las aplicaciones configuradas como servidores aceptan peticiones de conexión en un puerto por parte de aplicaciones configuradas como clientes, tienen la capacidad de manejar múltiples conexiones de manera simultánea y diferenciar a cada uno de los clientes a partir de dichas conexiones, sin la necesidad de establecer números de puerto únicos para cada cliente. Un ejemplo de este esquema es el servicio de correo electrónico, en el que una máquina se configura como servidor y muchas máquinas denominadas clientes pueden acceder al servicio en el momento que lo deseen, incluso simultáneamente. Los clientes, por su parte, pueden conectarse a varios servidores al mismo tiempo desde la misma aplicación y cada uno de los servidores se comunica con el cliente por medio del mismo número de puerto destino. Esto no crea ningún conflicto puesto que cada servidor posee un identificador diferente, que unido al identificador del cliente crea una conexión única, aun cuando todas ellas comparten el número de puerto del cliente.

En la implementación la idea de puntos finales se abstrae mediante el concepto de *socket*. Un *socket* es un elemento abstracto que manipula la entrada y salida de flujo de información entre los programas de aplicación y la capa de transporte. Para las aplicaciones el socket es la puerta de entrada y salida de la información. Un socket se define por la dirección IP de un dispositivo y el número de puerto de una aplicación que se ejecuta en dicho dispositivo.

Una conexión mantiene un socket en cada extremo del enlace mientras, que en cada aplicación las diferentes conexiones se identifican mediante su socket asociado. En una conexión los sockets en los extremos proporcionan una ruta de comunicación bidireccional para ambas aplicaciones y el enlace que forman es único en la red.

En la implementación de los protocolos de la capa de transporte de la pila TCP/IP se definen dos tipos de sockets. El primero maneja flujos de datos de manera bidireccional y secuencial y proporciona conexiones confiables. Este es el tipo de socket utilizado por el protocolo TCP. El segundo tipo de socket es el que emplea UDP y proporciona entrega de datagramas no confiable en ambas direcciones. A pesar de que UDP no realiza conexiones, en este protocolo también se implementa el uso de sockets, ya que esto ofrece una interfaz de fácil manejo para las aplicaciones. En TCP la dirección y el número de puerto de la aplicación destino se asocian al socket y después se envía y recibe información a través del socket hasta que se cierre la conexión. Por otro lado, en los sockets UDP no es posible asociar al socket con los datos del dispositivo remoto, ya que este protocolo no establece conexiones y no se compromete con ningún destino en particular, cada vez que se envía información por medio de un socket UDP se debe especificar la dirección y número de puerto del destino.

Los sockets proporcionan a la capa de aplicación una *Interfaz de Programación de Aplicaciones* (API), que es un conjunto de funciones de la capa de transporte que se usan para crear aplicaciones que se comuniquen por la red a través de un socket.

El conjunto de funciones que integran la API es diferente en cada implementación particular, pero en general todas las implementaciones son muy similares y cumplen las mismas funciones. Por ejemplo en sistemas operativos Windows existe una función que recibe ciertos argumentos y crea un socket, y es muy similar a la función que usan los sistemas UNIX para realizar la misma acción. Diferentes lenguajes de programación también cuenta con su propia API de manejo de sockets. Las funciones que deben realizar los sockets son las siguientes:

- Realizar una conexión.
- Asociarse con un puerto.
- Esperar una conexión.
- Aceptar una conexión.
- Enviar datos.
- Recibir datos.
- Cerrar una conexión.

Una parte de las funciones del API se utiliza en las aplicaciones cliente mientras que otra parte sirve para programar las aplicaciones servidor. Algunas funciones son comunes para los clientes y los servidores.

Una aplicación cliente debe obtener un socket utilizando la función apropiada para ello y puede ser un socket de flujo de datos (TCP) o un socket de datagramas (UDP). La función encargada de crear el socket solicita un número de puerto local al sistema operativo (si estamos hablando de una computadora) y el programador de la aplicación no tiene que especificar este atributo. Los clientes realizan conexiones activas, lo cual significa que envían una petición de conexión a algún servidor indicando la dirección IP y número de puerto con el que se identifica la aplicación servidor. Para esto el cliente cuenta con una función c*onectar* en la que se indican los datos del servidor. Si el socket es UDP no se usa la función *conectar*.

Los servidores también deben obtener un socket para poderse comunicar con las aplicaciones cliente. Después de obtener un socket, un servidor debe asociarlo con un número de puerto local. Esta acción a diferencia de los clientes, la debe realizar el programador de la aplicación indicando el número de puerto en el que el servidor acepta conexiones. Para esto debe existir una función propia de los servidores que permita asociar al socket con el número de puerto local después de haber sido creado.

El socket del servidor debe realizar una conexión pasiva, lo que significa que se configura en modo de espera de conexiones activas por parte de los clientes. A esta acción se le denomina *escuchar por un puerto* y se realiza con una función específica para servidores. Desde el momento en que se ejecuta la función *escuchar* el servidor permanece esperando a que algún cliente le envíe una petición de conexión. Si una petición de conexión llega, el servidor debe ejecutar una función que acepte la conexión y en el momento que esto ocurre la conexión entre ambos queda establecida. Si el socket es UDP no se realiza ninguna de las funciones anteriores.

Cuando el cliente y el servidor ya están conectados (en el caso de los sockets TCP) o el socket está creado (en el caso de los sockets UDP), se utilizan funciones de lectura y escritura para intercambiar información entre las aplicaciones. Las funciones que se utilizan para leer y escribir son las mismas para los clientes y los servidores, pero no son las mismas para los sockets TCP y los sockets UDP, ya que los primeros sólo envían información dado que el socket ya conoce el destino final de la misma, pero los segundos deben especificar el destino cada vez que envían información.

Al terminar el intercambio de información se cierra la conexión, ya sea por iniciativa del cliente o por parte del servidor y se libera el socket.

El manejo de los sockets UDP es muy sencillo, las funciones que realizan se describen a continuación:

Funciones del cliente y del servidor:

- Obtener un socket.
- Enviar y recibir datos.
- Liberar socket cuando ya no sea necesario.

Los sockets TCP, como es natural, realizan mas funciones. A continuación se resumen las operaciones de un cliente y un servidor programados sobre TCP.

Funciones del cliente:

- Crear un socket.
- Conectarse a un servidor.
- Enviar y recibir información.
- Cerrar la conexión.
- Liberar al socket.

Funciones del servidor:

- Crear el socket.
- Asociarlo a un número de puerto.
- Esperar conexiones.
- Aceptar conexión.
- Enviar y recibir información.
- Cerrar la conexión.

- Regresar al tercer punto y repetir la secuencia con nuevas conexiones.
- Liberar el socket cuando ya no sea necesario.

4.11 HTTP – Protocolo de Transferencia de Hipertexto

El Protocolo de Transferencia de Hipertexto o HTTP (*Hyper Text Transfer Protocol*) es un protocolo de aplicación usado para la transferencia de recursos vía web (www – *World Wide Web*), por ejemplo una página de internet. Su funcionamiento se basa en el esquema cliente–servidor mediante peticiones (por parte de los clientes) y respuestas (por parte de los servidores). La información que se transmite por parte de un servidor se llama recurso y se identifica a través de un URL que el cliente indica en su petición.

Los recursos entregados por un servidor pueden ser archivos de texto, gráficos, audio, el código de una página, la consulta a una base de datos, el resultado de un programa ejecutado en el servidor, imágenes, videos o cualquier tipo de información que se pueda representar digitalmente y almacenar en un dispositivo conectado a una red. Cualquier tipo de recurso viaja por la red codificado como una secuencia de caracteres. Los mensajes de petición y respuesta, incluyendo el recurso, son sólo flujos de bits que TCP transporta de un lugar a otro.

El proceso de transferencia de un recurso web es como sigue:

- Un servidor realiza una conexión pasiva, esperando conexiones a través de un puerto, para HTTP se emplea el puerto establecido número 80.
- Un cliente realiza una conexión activa al servidor.
- Una vez establecida la comunicación, el cliente envía un mensaje de petición indicando el recurso que solicita.
- El servidor recibe e interpreta la petición, envía un mensaje de respuesta y si es capaz de entender la solicitud y encontrar el recurso, lo envía en el cuerpo del mensaje, si no, envía un error.
- Cuando el recurso termina de transferirse la conexión se cierra.

El proceso anterior se repite para cada petición que se realiza, atendiendo cada una en una conexión por separado. Por ejemplo, si una página contiene tres enlaces y se accede a cada uno de ellos, el proceso de solicitud y respuesta se lleva a cabo tres veces.

4.11.1 Identificadores Uniformes de Recursos (URI) y Localizadores Uniformes de Recursos (URL)

Los recursos que se transfieren en una red se encuentran almacenados en un servidor denominado *servidor de origen* (ya que es el origen de almacenamiento de la información). Para identificarlos se usa un Identificador Uniforme de Recursos o URI (*Uniform Resource Identifier*), el cual define un conjunto universal de nombres y direcciones de todos los recursos independientemente de su ubicación.

El Localizador Uniforme de Recursos o URL (*Uniform Resource Locator*) es un ejemplo específico de URI, se usa para indicar la ubicación de algún recurso accesible en una red. Su formato es como se muestra a continuación:

esquema: esquema–datos específicos

El esquema se refiere al protocolo mediante el cual se accede al recurso. La segunda parte depende del protocolo utilizado. Por ejemplo, para HTTP la forma del URL es de la siguiente manera:

http://host:port/url–path

Este formato especifica la dirección IP del servidor donde reside el recurso (*host*), el número de puerto con el que se identifica la aplicación en una conexión TCP (*port*) y la ubicación del recurso solicitado dentro del servidor (*url–path*).

Si el puerto no se especifica, se toma por omisión el número 80, correspondiente al protocolo HTTP. En el caso de que la trayectoria URL (url–path) no esté especificada, se considera la solicitud de la página de inicio del servidor, identificada normalmente con el nombre *index.html* o *index.htm*.

4.11.2 Método de acceso al servidor

El método es un comando que le indica al servidor la acción que debe ejecutar sobre el recurso que solicita el cliente. En la Tabla 4.6 se describen algunos métodos usados para acceder a un servidor.

Tabla 4.6 Métodos de acceso al servidor

Método	Definición
Get	Indica que se requiere un recurso del servidor.
Head	Solicita información sobre algún objeto del servidor, por ejemplo tamaño o fecha de modificación. Cuando se recibe un comando HEAD el servidor solamente envía los encabezados de un objeto.
Post	Se usa para enviar información al servidor por parte de un cliente, por ejemplo, los datos contenidos en un formulario.
Put	Se usa para enviar un recurso del cliente al servidor, el servidor lo almacena e identifica mediante un URL que también se indica en el mensaje PUT.
Delete	Indica al servidor que borre el objeto identificado con un URL.

El programa cliente selecciona el método adecuado para cualquier solicitud, no es necesario que un usuario del servicio especifique el método, de hecho ni siquiera tiene la responsabilidad de saber qué es un método y cómo se usa.

La solicitud de páginas web y los enlaces contenidos en ellas siempre ocupan el método GET. El envío de datos de formularios utiliza POST, pero también puede utilizar GET.

4.11.3 Mensajes de solicitud y respuesta

Los mensajes de solicitud y respuesta siguen un formato según el estándar documentado en el RFC 822. Cada mensaje HTTP contiene los elementos que se muestran en la Figura 4.28 y se describen a continuación.

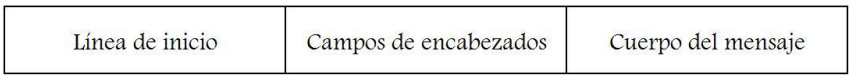


Figura 4.28 Formato general de mensaje HTTP

Línea de inicio. En los mensajes de solicitud es una *línea de solicitud* y en los mensajes de respuesta es una *línea de estado.*

Campos de encabezados. Son campos opcionales, se dividen de a cuerdo a la información que proporcionen, como se muestra en la Tabla 4.7.

Cuerpo del mensaje. En el mensaje de respuesta el cuerpo contiene el recurso, al contenido del recurso se le llama *entidad*. En el mensaje de solicitud el cuerpo es opcional, se usa por ejemplo cuando se envía un documento del cliente al servidor.

Tabla 4.7 Encabezados de mensajes de solicitud y respuesta HTTP

Campos del encabezado	Función	Ejemplo
Campos de solicitud	Los usan los clientes para añadir información adicional sobre ellos o la solicitud que envían al servidor.	- Tipo de recurso que el cliente acepta (imagen, audio, etc.). - Tipo de caracteres y codificación que el cliente acepta. - Sistema operativo usado por el cliente. - Navegador.
Campos de respuesta	Los usan los servidores para añadir información adicional sobre ellos o	- Software que utiliza el servidor (por ejemplo Apache, IIS, etc).

	el recurso entregado al cliente.	- Tiempo que transcurre desde que la respuesta se genera.
Campos generales	Se usan en todos los mensajes de solicitud y respuesta para adjuntar información.	- Fecha en que se origina el mensaje - Codificación empleada en el cuerpo del mensaje.
Campos de entidad	Definen el contenido del recurso en aquellos mensajes que contienen un cuerpo donde se transfiere dicho recurso.	- Método que soporta el recurso. - Codificación empleada en el contenido de la entidad. - Tamaño del contenido de la entidad.

4.11.3.1 Solicitud HTTP

En la Figura 4.29 se aprecia el formato de una solicitud HTTP y a continuación se explica su contenido, CR–LF indica retorno de carro y salto de línea.

Línea de solicitud			Línea en blanco	Encabezados de solicitud	Línea en blanco	Cuerpo de solicitud
Método	URL	Versión	CR–LF	cero o más campos	CR–LF	Documento

Figura 4.29 Formato de solicitud HTTP

Línea de solicitud. A la línea de inicio de un mensaje de solicitud se le denomina *línea de solicitud*. Esta línea indica: el método de acceso al servidor, la dirección del recurso solicitado indicado mediante su URL y la versión del protocolo usada por el cliente. Cada uno de estos elementos está separado por un espacio en blanco:

- Método.
- Dirección URL.
- Versión del protocolo. La más usada actualmente es la 1.1, y se indica mediante la cadena HTTP/1.1.

Campos del encabezado de solicitud. Los encabezados son opcionales y cada uno se escribe en una línea. Los campos contienen un nombre seguido de dos puntos y un valor. En el siguiente ejemplo se ilustra la sintaxis de dos campos del encabezado de una solicitud, en los cuales se indica el tipo de información que acepta el cliente y una fecha condicional que le informa al servidor que no debe entregar el recurso si no ha sido modificado desde dicha fecha:

```
Accept : Text/html
If-Modified-Since : Saturday, 15-January-2009 14:37:11 GMT
```

El final de los campos de encabezados se indica mediante una línea en blanco después de la cual viene el cuerpo de la solicitud.

Cuerpo de la solicitud. Es un conjunto de líneas opcionales que permiten enviar datos específicos para algún tipo de solicitud, por ejemplo en un mensaje POST los datos de un formulario se envían en el cuerpo.

Ejemplo de petición HTTP

Los mensajes de solicitud se generan en un programa cliente, comúnmente un navegador web como Internet Explorer, aunque puede ser cualquier otro programa que realice una conexión activa a un servidor. El navegador web contiene un espacio en donde los usuarios insertan la dirección URL del recurso que solicitan. El programa cliente genera el mensaje de solicitud y lo envía al servidor. El usuario no ingresa una petición con el formato especificado en los párrafos anteriores, sólo debe conocer la dirección de un recurso y el navegador construye el mensaje a partir de dicha dirección. Al recibir la respuesta del servidor, el navegador la interpreta mostrando su contenido en un visualizador.

Si un usuario desea ingresar a la página de la Facultad de Ingeniería, debe teclear en el área de direcciones de algún navegador web la dirección **www.ingenieria.unam.mx**. A este formato de dirección se le llama *nombre de dominio* y permite a los usuarios recordar de manera más fácil la dirección de un servidor, en vez de usar las direcciones IP especificadas con números. Todos los nombres de dominio tienen su

correspondiente IP numérica, la transformación entre el nombre de dominio y la dirección IP se realiza a través de *Servidores de Nombres de Dominio* (DNS). En el caso del ejemplo anterior, la IP del servidor en donde reside la página de la Facultad de Ingeniería es 132.248.54.13, cuando un usuario teclea **www.ingenieria.unam.mx** se está conectando al servidor cuya IP es 132.248.54.13. De hecho si se teclea la dirección IP en lugar del nombre de dominio, se observa el mismo resultado. La transformación del nombre de dominio a la dirección IP correspondiente es transparente para el usuario, aunque en realidad el cliente se conecta primero al servidor de nombres de dominio, en donde se resuelve la dirección para que después el cliente pueda conectarse al servidor destino.

Una vez ingresada la dirección de un recurso en el navegador se genera un mensaje de petición que se ve de la siguiente manera:

```
GET http://www.ingeniería.unam.mx HTTP/1.1
Accept : Text/html
User-Agent : Mozilla/4.0 (compatible; MSIE 7.0;
Windows NT 5.1; Windows 95)
Port: 2989
Connection: keep-alive
```

Al analizar la solicitud mostrada en el recuadro anterior se puede observar que la primera línea corresponde a la línea de solicitud, formada por el método, la dirección URL del recurso y la versión del protocolo, seguida de las líneas de encabezados, las cuales incluyen el tipo de información que acepta el cliente, el tipo de agente de usuario, el puerto usado por el cliente y el estado de la conexión. En esta solicitud no hay un cuerpo del mensaje, si lo hubiera tendría que haber una línea en blanco entre él y la última línea de encabezados.

El ejemplo anterior no incluye el número de puerto ni una trayectoria URL que indique la ubicación de un archivo dentro del servidor. En el siguiente ejemplo un cliente se conecta a la página de la Facultad de Ingeniería solicitando una edición de la gaceta que se publica en dicha página. La línea de solicitud se muestra a continuación:

```
GET http://132.248.54.13:80/paginas/gaceta/2009/gaceta12_2009.pdf HTTP/1.1
```

La dirección URL del recurso incluye el protocolo de acceso, la dirección IP del servidor donde reside el recurso, el número de puerto empleado para la conexión y la trayectoria URL del recurso, la cual indica que el recurso que se solicita se encuentra dentro de una carpeta llamada *paginas,* que contiene otra carpeta con el nombre *gaceta*, en cuyo interior se encuentra la carpeta *2009* que contiene el documento `gaceta12_2009.pdf` que se solicita.

4.11.3.2 Respuesta HTTP

En el mensaje de respuesta un servidor envía el recurso solicitado por un cliente y añade información adicional. El formato de los mensajes de respuesta se muestra en la Figura 4.30.

Línea de estado			Línea en blanco	Encabezados de respuesta	Línea en blanco	Cuerpo de respuesta
Versión	Código	Descripción	CR–LF	cero o más campos	CR–LF	Entidad

Figura 4.30 Formato de respuesta HTTP

Línea de estado. La línea de inicio de un mensaje de respuesta se denomina *línea de estado* y contiene tres elementos separados por un espacio en blanco: la versión del protocolo usada por el servidor, un código numérico que indica el estado de la transacción y un campo con una descripción del estado indicado con el código:

- Versión del protocolo
- Código de estado
- Descripción del código de estado

Campos del encabezado de respuesta. Los encabezados son opcionales y cada uno se escribe en una línea. Los campos contienen un nombre, seguido de dos puntos y un valor. En el siguiente ejemplo se ilustra la sintaxis de cuatro campos del encabezado de una respuesta, en los cuales se indica el software de servidor usado, el tipo de contenido enviado por el servidor, la longitud del contenido del cuerpo y la fecha de la última modificación del documento.

```
Server : Microsoft-IIS/2.0
Content-Type : text/HTML
Content-Length : 1245
Last-Modified : Fri, 14 Jan 2009 08:25:13 GMT
```

El final de los campos de encabezados se indica mediante una línea en blanco, después de la cual viene el cuerpo de la respuesta.

Cuerpo de la respuesta. El recurso solicitado se envía en el cuerpo del mensaje utilizando la codificación indicada en los encabezados para que el cliente la pueda interpretar. Por ejemplo, si se trata de una página web, generalmente se utiliza código HTML.

Códigos de estado

Los códigos de estado son números enteros de tres dígitos que el servidor envía al cliente para informarle el estado de la transacción. El primer dígito indica el tipo de respuesta que se envía, se dividen en mensajes informativos, de operación exitosa, de redirección, de error debido al cliente o error debido al servidor. En la Tabla 4.8 se ilustran algunos de los códigos usados en las respuestas.

Tabla 4.8 Códigos de estado HTTP

CÓDIGO	TIPO DE MENSAJE	DESCRIPCIÓN
1XX	Mensaje informativo	Solicitud recibida, proceso continuado
100	Continuar	Indica al cliente que su solicitud fue recibida y puede continuar con el envío.
2XX	Operación exitosa	La transferencia se realizó exitosamente
200	OK	La solicitud tuvo éxito.
201	Creada	Se creó un nuevo recurso en el servidor identificado con el URL que se envía de regreso al cliente.
202	Aceptada	La solicitud fue aceptada pero no se ha terminado de procesar.
204	Ningún contenido	No hay información que entregar, el mensaje de respuesta no contiene un cuerpo.
3XX	Redirección	El recurso ya no se encuentra en el servidor
300	Opciones múltiples	El recurso solicitado no tiene una representación única.
301	Cambiado permanentemente	El recurso solicitado tiene un nuevo URL permanente, que se indica en la respuesta para poder solicitar el recurso de manera adecuada en futuras ocasiones.
302	Encontrado	El recurso solicitado tiene un URL diferente de manera temporal, en futuras solicitudes se debe especificar el URL original.
4XX	Error debido al cliente	La solicitud es incorrecta
400	Mala solicitud	El servidor no lo logró entender la solicitud debido a una mala sintaxis.
401	No autorizada	La solicitud requiere autenticación del usuario.
403	Prohibido	El servidor no puede mostrar el recurso solicitado ya que está

		prohibido.
404	No encontrado	El servidor no encuentra el recurso identificado con la URL especificada en la solicitud.
405	Método no permitido	El método especificado no es permitido para el recurso solicitado.
5XX	Error debido al servidor	Error interno en el servidor
500	Error interno del servidor	Se presentó una condición inesperada en el servidor que le impidió responder la solicitud.
501	No implementada	El servidor no soporta la funcionalidad requerida para la solicitud.
503	Servicio no disponible	El servidor no puede procesar la solicitud de manera momentánea.
505	Versión HTTP no soportada	El servidor no soporta la versión del protocolo usada por el cliente.

Ejemplo de respuesta HTTP

Cuando un servidor atiende una solicitud, antes de enviar el recurso, envía al cliente algunos campos de información como parte del protocolo. Siguiendo con el ejemplo mostrado del cliente que se conecta al servidor de la Facultad de Ingeniería, la respuesta enviada tiene el siguiente formato:

```
HTTP/1.1 200 OK
Date: Tue, 15 Sep 2009 02:42:02 GMT
Server: Apache
Connection: close
Content-Type: text/html

<HTML><HEAD>…
…
… </HED><HTML>
```

En la línea de estado se indica la versión del protocolo usada por el servidor y el código de estado número 200, seguido de la explicación textual "OK", que indica que la solicitud fue recibida y procesada con éxito. A partir de la segunda línea viene una serie de campos de encabezados, que indican la fecha en la que se generó el mensaje de respuesta, el software de servidor empleado, el estado de la conexión y el tipo de contenido usado en el cuerpo del mensaje. La línea en blanco separa los encabezados del cuerpo del mensaje, el cual contiene el recurso solicitado. En el ejemplo, el recurso es el código de la página de inicio de la Facultad de Ingeniería, codificado en lenguaje HTML.

Los conceptos referentes a los protocolos de la pila TCP/IP definidos en el presente capítulo son la base del desarrollo de la pila implementada en el capítulo 5 y son las normas de diseño en las que se basa la implementación.