Instituto Politécnico Nacional

Escuela Superior de Cómputo

Repaso protocolo HTTP

Martinez Olivares Vicente Jafet

Grupo: 3CV12

Web Application Development

Profesor: Sandra Ivette Bautista Rosales

Los Principios de las Aplicaciones de Red explican los servicios proporcionados por los protocolos de transporte de Internet. UDP es un protocolo de transporte ligero y simple que proporciona unos servicios mínimos, no orientado a la conexión y sin garantías de que el mensaje llegará al receptor o que lo hará en orden. Además, no incluye mecanismos de control de congestión. Por otro lado, TCP ofrece transferencia de datos fiable y se puede mejorar con SSL para proporcionar seguridad. Sin embargo, no hay garantías en cuanto a tasa de transferencia o temporización. Aún así, muchas aplicaciones sensibles al tiempo funcionan bien en Internet gracias a trucos de diseño. Por ejemplo, el correo electrónico, acceso remoto a terminales, la web y transferencia de archivos usan TCP para tener una transferencia fiable; mientras que aplicaciones como Skype prefieren usar UDP para tolerar cierto grado de pérdidas, pero usan TCP como alternativa si falla la comunicación por UDP. Finalmente, los protocolos de la capa de aplicación estructuran los mensajes y dan significado a cada uno de sus campos.

Las preguntas sobre cuando envían los procesos mensajes llevan al ámbito de los protocolos de la capa de aplicación. Estos protocolos definen cómo los procesos de una aplicación, que se ejecutan en distintos sistemas terminales, pasan mensajes entre sí. Específicamente, un protocolo de la capa de aplicación especifica los tipos de mensajes intercambiados (por ejemplo, solicitudes y respuestas); la sintaxis de los diferentes tipos de mensajes (los campos y la forma en que estos están delimitados); la semántica de los campos (el significado de la información contenida); y las reglas para determinar cuándo y cómo un proceso envía y responde a los mensajes. Algunos protocolos, como HTTP (HyperText Transfer Protocol - RFC 2616), son especificados en documentos RFC y por lo tanto son de dominio público; sin embargo, hay otros protocolos propietarios, como el usado por Skype. Es importante diferenciar entre aplicaciones de red y protocolos de la capa de aplicación. Un protocolo es solo un elemento importante de una aplicación, como se ve en el caso de la web, que consta del estándar para formatos de documentos (HTML), navegadores web, servidores web y el protocolo HTTP. Del mismo

modo, una aplicación para correo electrónico se compone también de varios componentes: servidores, clientes, estándar para la estructura de mensajes y el protocolo SMTP (Simple Mail Transfer Protocol - RFC 5321).

El Protocolo HTTP (Hypertext Transfer Protocol) es un protocolo de la capa de aplicación usado para enviar solicitudes y recibir respuestas entre un navegador y un servidor web. Esta sección explica cómo se abre una conexión TCP desde el navegador al servidor web, se envía el mensaje de solicitud HTTP y se recibe un mensaje de respuesta, incluyendo el archivo base HTML. También se explica cómo el navegador y el servidor deciden qué líneas de cabecera incluir en sus mensajes HTTP. Además, se explica la tecnología de las cookies, que permite a los sitios web identificar a los usuarios. Por ejemplo, si un usuario visita una página web comercial, esta puede usar cookies para identificar al usuario y mostrar contenido específico para él. Esta sección también menciona brevemente las otras aplicaciones importantes de Internet: correo electrónico, servicio de directorio, flujos de video y aplicaciones P2P.

Aplicación de éxito en Internet. El correo electrónico es más complejo que la Web, ya que usa varios protocolos de la capa de aplicación. Después del correo electrónico, abordaremos el Sistema de Nombres de Dominio (DNS), que proporciona un servicio de directorio a Internet. La mayoría de los usuarios no interactúan directamente con el DNS; en su lugar, invocan indirectamente al DNS a través de otras aplicaciones (entre las que se incluyen las aplicaciones web, de transferencia de archivos y de correo electrónico). El DNS ilustra cómo se puede implementar en la capa de aplicación de Internet un elemento básico de funcionalidad de red (la traducción entre nombres de red y direcciones de red). Después examinaremos las aplicaciones P2P para compartir archivos y completaremos nuestro estudio sobre las aplicaciones analizando los flujos de video bajo demanda, incluyendo la distribución de video almacenado a través de redes de distribución de contenido. A principios de la década de 1990 apareció la World Wide Web (WWW).

Esta fue la primera aplicación en atraer la atención del público general. Cambió drásticamente cómo las personas interactúan dentro y fuera del trabajo. Lo que atrae a los usuarios es que opera bajo demanda: reciben lo que desean cuando lo desean, lo cual es muy diferente a la radio y televisión, que obligan a los usuarios sintonizar programas cuando el proveedor tiene contenido disponible. Además, es fácil publicar información en la Web (todo el mundo puede convertirse en editor con costos extremadamente bajos) y existen hipervínculos y motores de búsqueda para navegar por el océano web. El corazón de la Web es el Protocolo para la Transferencia Hipertexto (HTTP), definido en los documentos RFC 1945 y RFC 2616. Está implementado por dos programas: un cliente y un servidor, que se comunican intercambiando mensajes HTTP. Una página web consta de objetos (como texto, imágenes, videos, etc.) que son descargados desde el servidor al cliente usando HTTP.

Un objeto es un archivo (como un archivo HTML, una imagen JPEG, un applet Java o un clip de video) que se puede direccionar mediante un único URL. La mayoría de las páginas web están compuestas por un archivo base HTML y varios objetos referenciados. Por ejemplo, si una página web contiene texto HTML y cinco imágenes JPEG, entonces la página web contiene seis objetos: el archivo base HTML y las cinco imágenes. El archivo base HTML hace referencia a los otros objetos contenidos en la página mediante los URL de los objetos. Cada URL tiene dos componentes: el nombre de host del servidor que alberga el objeto y el nombre de la ruta al objeto. Por ejemplo, en el URL http://www.unaEscuela.edu/unDepartamento/imagen.gif , www.unaEscuela.edu corresponde a un nombre de host y /unDepartamento/imagen.gif es el nombre de la ruta. HTTP define cómo los clientes web solicitan páginas web a los servidores y cómo estos servidores transfieren esas páginas a los clientes. Cuando un usuario solicita una página web (por ejemplo, haciendo clic en un hipervínculo), el navegador envía al servidor mensajes de solicitud HTTP, pidiendo los objetos contenidos en la página. El servidor recibe las solicitudes y responde con mensajes de respuesta HTTP que contienen los objetos. HTTP utiliza TCP como su protocolo de transporte subyacente (en lugar

de ejecutarse por encima de UDP). El cliente HTTP primero inicia una conexión TCP con el servidor y luego ambos procesos acceden a TCP a través de sus interfaces de socket para enviar mensajes de solicitud y respuesta que serán garantizados por TCP para llegar intactos al destinatario. Es importante observar que el servidor envía los archivos solicitados a los clientes sin almacenar información acerca del estado del cliente. Si un determinado cliente pide el mismo objeto dos veces en un espacio de tiempo corto, el servidor no responderá diciendo que acaba de servir dicho objeto.

La arquitectura de la Web es cliente-servidor. Un servidor web siempre está activo, con una dirección IP fija, y brinda servicio a solicitudes procedentes de, potencialmente, millones de navegadores diferentes. Esta interacción cliente-servidor se realiza sobre TCP, y el desarrollador de la aplicación debe decidir si cada par solicitud/respuesta se envía a través de una conexión TCP separada o si todas las solicitudes y sus correspondientes respuestas se envían a través de la misma conexión TCP. Si se usa el primer método, se dice que la aplicación utiliza conexiones no persistentes; si se usa la segunda opción, entonces se habla de conexiones persistentes. Para ilustrar cómo funcionan las conexiones no persistentes, supongamos que un cliente quiere transferir una página web desde un servidor, que consta de un archivo base HTML y 10 imágenes JPEG. El proceso cliente HTTP inicia una conexión TCP con el servidor en el puerto 80 predeterminado para HTTP. Luego, el cliente envía un mensaje de solicitud al servidor a través del socket. El servidor recibe el mensaje a través del socket, recupera el objeto requerido del almacenamiento y lo encapsula en un mensaje de respuesta. El servidor indica entonces a TCP que cierre la conexión TCP. El cliente recibe el mensaje de respuesta y extrae el archivo del mensaje para examinarlo en busca de referencias a los 10 objetos JPEG restantes. Los cuatro primeros pasos se repiten entonces para cada objeto JPEG referenciado hasta que el navegador recibe la página web y la muestra al usuario.

Las especificaciones HTTP (RFC 1945 y RFC 2616) definen el protocolo de comunicación entre un programa cliente HTTP y un programa servidor HTTP. Por ejemplo, cuando un usuario solicita una página web, se generan 11 conexiones TCP para transportar exactamente un mensaje de solicitud y un mensaje de respuesta. La mayoría de los navegadores abren entre 5 y 10 conexiones TCP en paralelo para gestionar transacciones solicitud-respuesta. Si el usuario lo prefiere, el número máximo de conexiones en paralelo puede establecerse en uno, en cuyo caso se establecerán diez conexiones en serie. Un cálculo aproximado estima que el tiempo de respuesta total es aproximadamente igual a dos RTT (tiempo de ida y vuelta) más el tiempo de transmisión del archivo HTML por parte del servidor. Las conexiones no persistentes presentan algunos inconvenientes como la sobrecarga significativa del servidor web, ya que puede estar sirviendo solicitudes de cientos de clientes distintos simultáneamente. Por esta razón, se crearon las conexiones persistentes, las cuales permiten mantener la misma conexión TCP para todos los objetos solicitados. Esto reduce significativamente el tiempo de respuesta.

HTTP 1.1 usa conexiones persistentes para mejorar el rendimiento. Esto significa que, en lugar de cerrar la conexión TCP después de que el servidor envía una respuesta, el servidor deja la conexión abierta para permitir que el cliente envíe solicitudes de objetos adicionales. Por ejemplo, cuando un cliente solicita una página web, el servidor envía primero el archivo HTML base y luego 10 imágenes a través de la misma conexión TCP persistente. Esto significa que se ahorran dos Round-Trip Times (RTT): un RTT para establecer la conexión TCP y otro RTT para solicitar y recibir un objeto. Además, varias páginas web del mismo servidor se pueden enviar al mismo cliente a través de una única conexión TCP persistente. Estas solicitudes de objetos se realizan una tras otra sin esperar a recibir las respuestas a las solicitudes pendientes (procesamiento en cadena). HTTP/2 [RFC 7540] ha ampliado la funcionalidad de HTTP 1.1, permitiendo entrelazar múltiples solicitudes y respuestas en la misma conexión y proporcionando también un mecanismo para priorizar los mensajes HTTP dentro de dicha conexión. Además, el formato de los mensajes HTTP consiste en dos tipos:

mensajes de solicitud y mensajes de respuesta. Los mensajes de solicitud constan normalmente de cinco líneas, incluyendo la línea de solicitud (con los campos Método, URL y Versión HTTP) y las líneas de cabecera (por ejemplo, Host:, Connection:, User-agent:, etc.).

La linea de cabecera del host es un elemento clave de los mensajes de solicitud HTTP, que se utilizan para comunicarse entre el navegador y el servidor web. La linea de cabecera Connection: close le indica al servidor que no desea molestarse en trabajar con conexiones persistentes, sino que desea que el servidor cierre la conexión después de enviar el objeto solicitado. La linea de cabecera User-agent: especifica el agente de usuario (por ejemplo, Mozilla/5.0, un navegador Firefox). Esta linea resulta útil porque el servidor puede enviar versiones diferentes del mismo objeto a los distintos tipos de agentes de usuario (todas las versiones tienen la misma dirección URL). Por último, la linea de cabecera Accept-language: indica que el usuario prefiere recibir una versión en francés del objeto, si tal objeto existe en el servidor; en caso contrario, el servidor debe enviar la versión predeterminada. El formato general de un mensaje de solicitud HTTP es muy similar al usado en el ejemplo anterior. Sin embargo, después de las lineas de cabecera (y el retorno de carro y el salto de línea adicionales) se incluye un cuerpo de entidad. Este campo queda vacío cuando se utiliza el método GET, pero no cuando se usa el método Post. A menudo, un cliente HTTP utiliza el método Post cuando el usuario completa un formulario; por ejemplo, cuando especifica términos para realizar una búsqueda utilizando un motor de búsqueda. Si el valor del campo de método es Post, entonces el cuerpo de la entidad contendrá lo que el usuario haya introducido en los campos del formulario. El mtodo HEAD es similar al mtodo GET y suele utilizarse para labores de depuracion. El mtodo PUT suele utilizarse junto con herramientas de publicación web para cargar objetos en un servidor web determinado. El mtodo DELETE permite a un usuario o a una aplicación borrar un objeto de un servidor web. Los mensajes de respuesta HTTP son generados por los servidores web como respuesta a las solicitudes HTTP. Estos mensajes contienen información sobre la petición realizada y si ésta ha sido satisfactoria o

no. Por ejemplo, si la petición ha sido satisfactoria entonces aparecerá la línea HTTP/1.1 200 OK en la respuesta.

Este mensaje de respuesta contiene tres secciones: una linea de estado inicial, seis lineas de cabecera y el cuerpo de entidad, el cual es el objeto solicitado. La linea de estado indica que el servidor está utilizando HTTP/1.1 y que todo es correcto (OK). La linea de cabecera Connection: close indica al cliente que el servidor va a cerrar la conexión TCP después de enviar el mensaje. La linea Date: indica la hora y fecha en la que fue creado y enviado el mensaje. Por otro lado, la linea Last-Modified: especifica la hora y fecha en que el objeto fue modificado por última vez. El campo Content-Length: indica el número de bytes del objeto que se está enviando. Finalmente, el campo Content-Type: indica que el objeto incluido en el cuerpo de entidad es texto HTML. Además, los códigos de estado y sus frases asociadas indican el resultado de la solicitud. Por ejemplo, un código 200 OK significa que la solicitud se ha ejecutado con éxito y se ha devuelto la información en el mensaje de respuesta; mientras que un 404 Not Found significa que el documento solicitado no existe en este servidor. Para ver un mensaje de respuesta HTTP real, se puede establecer una conexión Telnet con un servidor web favorito e ingresar un mensaje de solicitud para obtener algún objeto almacenado en él. Por ejemplo: "telnet gaia.cs.umass.edu 80 GET /kurose_ross/interactive/index.php HTTP/1.1 Host: gaia.cs.umass.edu". Resumiendo, este mensaje de respuesta contiene tres secciones: una linea de estado inicial, seis lineas de cabecera y el cuerpo de entidad (el objeto solicitado). Estas lineas contienen información como los códigos de estado y sus frases asociadas para mostrar el resultado de la solicitud, la hora y fecha en la que fue creado y enviado el mensaje, así como cuando fue modificado por última vez; además del número de bytes del objeto y su tipo (texto HTML). Para ver un mensaje real se puede establecer una conexión Telnet con un servidor web favorito e ingresar un mensaje de solicitud para obtener algún objeto almacenado en él.