Protocolos

Arquitectura y Sistemas Operativos.
Tecnicatura Superior en Programación.
UTN-FRA

Autores: *Prof. Martín Isusi Seff*

Revisores: *Prof. Marcos Pablo Russo*

Versión: 1

¿Qué es un protocolo?

Los protocolos son mecanismos que definen las reglas y convenciones para la comunicación entre dispositivos. Los protocolos incluyen tanto los mecanismos para identificar y realizar conexiones dentro de una red, así como también especifican las reglas y el formato con el que deben transmitirse los datos. Existen protocolos que, además, incluyen soporte para la compresión de los datos, acuso de recibo de mensajes, etc.

Hoy en día, los protocolos de red, generalmente utilizan un mecanismo llamado conmutación de paquetes para enviar y recibir los mensajes en forma de paquetes. Esto significa, mensajes divididos en piezas más pequeñas, que luego de enviadas, son reensambladas en el dispositivo destino.

Al hablar de protocolos, es importante mencionar el modelo OSI. Este modelo define un estándar para las comunicaciones entre distintos dispositivos. Siete capas son las que forman partes de este modelo:

- Capa de aplicación
- Capa de presentación
- Capa de sesión
- Capa de transporte
- Capa de red
- Capa de datos
- Capa física

El propósito de este apunte es desarrollar teórica y prácticamente algunos protocolos comunes de la capa de aplicación (HTTP, FTP, etc.). Esta capa define todos los protocolos que describen la comunicación entre aplicaciones (cómo transmitir archivos, cómo realizar una consulta a una página web, etc.). Puede decirse que la capa de aplicación es la que se encuentra en un "nivel más alto" con respecto al hardware, ya que no importa de qué manera se transmitan a bajo nivel los paquetes, estos protocolos funcionan de igual manera.

Funciones de los protocolos de la capa de aplicación

Tanto el dispositivo emisor como el receptor utilizan protocolos de capa de aplicación durante una comunicación. Para que la comunicación sea exitosa, todos los dispositivos que participan deben comunicarse con el mismo protocolo, esto quiere decir que, quién envía utilizará un protocolo para formar los paquetes y, el receptor, deberá utilizar el mismo protocolo para recrear el mensaje enviado.

Los protocolos de la capa de aplicación deben realizar las siguientes tareas:

- Establecer reglas consistentes para el intercambio de datos entre las aplicaciones y servicios cargados en el emisor y receptor.
- Especificar cómo se estructuran los datos, y los tipos de mensajes que pueden enviarse los

participantes de la comunicación.

Muchos tipos diferentes de aplicaciones se comunican a través de la red. Es por eso que, la capa de aplicación debe implementar múltiples protocolos para los distintos tipos de comunicaciones. Cada protocolo tiene un propósito específico y contiene las características requeridas por tal propósito.

Las aplicaciones y servicios pueden utilizar múltiples protocolos a lo largo de una sola conversación. Por ejemplo, un protocolo puede especificar cómo establecer la comunicación, mientras que otro describe el proceso para transferir los datos.

Modelo cliente-servidor

En el modelo cliente-servidor, dos dispositivos forman parte de la comunicación. Un dispositivo que solicita información llamado ***cliente***, y un dispositivo que provee tal información llamado ***servidor***. El cliente inicia la comunicación o intercambio, enviando un pedido (***request***). Este pedido puede contener información requerida por el servidor para generar la respuesta (***response***).

Protocolo HTTP

El protocolo HTTP (HyperText Transfer Protocol) es un protocolo ***sin estado***, que forma parte de la capa de aplicación y permite la comunicación entre sistemas distribuidos. Este protocolo es la base de la web como la conocemos ahora.

HTTP permite la comunicación entre clientes y servidores bajo distintas configuraciones de red. Esto es posible ya que no se necesita conocer ningún parámetro particular o configuración de los sistemas que se están comunicando. Se dice que es "sin estado", ya que durante una comunicación en la que se transmiten distintos mensajes, ninguno de estos mensajes comparte un estado en común, esto quiere decir que son independientes entre sí.

Dado que un dispositivo generalmente posee una sola comunicación física con la red, necesita existir una manera de diferenciar las distintas conexiones que se producen de distintas aplicaciones en un mismo equipo. Esta distinción se realiza a través de la asignación de ***puertos*** a cada una de las aplicaciones. Las aplicaciones que trabajan con el protocolo HTTP utilizan de manera estándar el puerto 80, aunque puede haber excepciones.

La comunicación entre el servidor y el cliente ocurre a través de un par ***request/response***.

La *request* o *petición* es la iniciadora de la comunicación, y la realiza el cliente a través de una dirección ***URL*** (Uniform Resource Locator). Un ejemplo de URL es:

http://www.dominio.com:8080/pagina/archivo.php?a=1&b=2

Este ejemplo podemos descomponerlo en los siguientes componentes:

Protocolo	Servidor o *host*	Puerto	Ruta al recurso	Consulta
http://	***www.dominio.com***	***8080***	***/pagina/archivo.php***	***?a=1&b=2***

Las URL son las identidades del *host* con el que nos queremos comunicar. Cuando utilizamos el protocolo HTTP, con especificar la URL no es suficiente. Además de la URL para "ubicar" el *host*, es necesario especificar el ***método*** que queremos utilizar. El *método* especifica la acción que debe realizar el *host* en base a la petición del cliente. Los cuatro *métodos* más comunes son:

- **GET**. Este método está pensado para obtener un recurso existente del servidor, aunque no es exclusivamente para ello. Cuando se utiliza el método GET para una petición, toda información extra que necesite ser enviada al servidor se codificará en la URL. En el siguiente ejemplo se hace una petición al recurso *traerUsuario.php* del *host usuarios.com*, y se pasan, como información extra, dos parámetros: *nombre* y *apellido*.
  http://usuarios.com/traerUsuario.php?nombre=Marcelo&apellido=Rodriguez
- **POST**. Este método está pensado para la creación de un nuevo recurso en el servidor. Al igual que el método GET, el método POST suele utilizarse pasando parámetros al servidor, pero a diferencia de GET, en POST los parámetros no se codifican como parte de la URL, sino que viajan en el ***cuerpo*** de la petición. Esto hace que sea un método más seguro, ya que la información no es visible directamente para el usuario.
- **PUT**. Este método se utiliza para actualizar un recurso. Suele ir acompañado de información al igual que GET y POST.
- **DELETE**. Se utiliza para eliminar recursos existentes.

Suele decirse que los métodos PUT y DELETE son una versión específica del método POST, ya que este puede utilizarse también para la actualización y eliminación de recursos.

Respuestas y códigos de status

Una vez que el cliente realiza la petición (*request*), el servidor realiza las operaciones correspondientes y devuelve una respuesta (*response*). Las respuestas del servidor llevan consigo un código que representa cómo se completó la operación, y de qué manera la misma debe ser interpretada por el cliente.

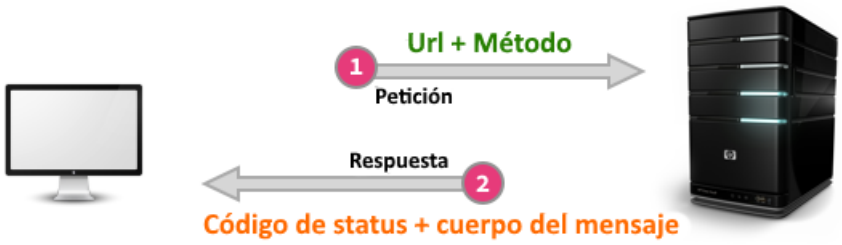

Figura 1

Códigos 2xx

Los códigos 2xx indican que la petición fue procesada correctamente por parte del servidor. El código más común cuando una petición se procesa correctamente es el código ***200.*** En el caso

que el usuario haya realizado una petición por un recurso (utilizando el método GET), el recurso pedido será devuelto en el ***cuerpo*** de la respuesta. Otros códigos que indican un procesamiento correcto son:

- **202**. La petición de un recurso fue aceptada, pero el mismo puede no estar incluido en la respuesta.
- **204**. Representa una respuesta que no tienen ningún contenido.
- **205**. Indica al cliente que vuelva a mostrar el contenido de cero.
- **206**. Indica que la respuesta contiene solo una parte de la respuesta.

Códigos 3xx

Los códigos 3xx indican al cliente que debe realizar alguna acción extra. El caso más común es que el servidor indique al cliente que debe redireccionarse a una dirección distinta.

- **301**. El recurso fue movido permanentemente (cambia su URL).
- **303**. El recurso fue movido temporalmente y el usuario debe redirigirse. Como parte de la respuesta, se envía la URL temporal.
- **304**. El servidor determina que el recurso requerido por el cliente no se ha modificado desde la última vez que lo consultó, por lo tanto, indica que debe utilizar la copia almacenada en *caché*.

Códigos 4xx

Los códigos 4xx son indican al cliente que hay un error en la petición. Esto puede suceder cuando se solicita un recurso que no existe o hay algún problema en la petición misma. Algunos de los códigos 4xx más comunes son:

- **400**. Indica que la petición está mal realizada (se conoce como *400 Bad Request*).
- **401**. Indica la necesidad de una autenticación del cliente.
- **403**. Indica que el servidor niega al cliente el acceso al recurso solicitado.
- **404**. Indica que el recurso solicitado no fue encontrado. (*404 not found*)
- **405**. Indica que el método utilizado en la petición (GET, POST, etc.) no está permitido, o el servidor no lo soporta.

Códigos 5xx

Estos códigos son utilizados para indicar un fallo en el servidor durante el procesamiento de la petición.

- **500.** Indica un error interndo en el servidor (*Internal server error*)
- **501.** Indica que el servidor todavía no soporta la operación pedida por cl cliente (*Not implemented*)
- **503.** Este código se devuelve cuando existe un problema interno en el servidor, o cuando el mismo está sobrecargado. En estos casos, suele suceder que el servidor

directamente no devuelve ninguna respuesta. Cuando el servidor no da respuesta, se consume el tiempo de la petición y se produce un ***timeout***.

Protocolo FTP

El protocolo FTP (*File Transfer Protocol*) es uno más dentro de la capa de aplicación, y se utiliza para la transferencia de archivos a través de la red.

Un servidor que ofrezca un servicio FTP, nos permitirá acceder a directorios y archivos en un determinado directorio, o la raíz de directorios completa, dependiendo de los permisos que tengamos. Dicho esto, es necesario mencionar que para conectarnos a un servidor FTP, es necesario contar con un usuario y contraseña. Si bien existe la posibilidad de configurar el servidor de tal manera que acepte conexiones ***anónimas***, esto no es recomendado por motivos de seguridad. Si se configura un usuario anónimo, será necesario prestar atención a los permisos que se conceden al mismo.

Así como HTTP tiene un puerto por defecto (puerto 80), el protocolo FTP trabajará por defecto en el ***puerto 21***.

Existen múltiples maneras de conectarse a un servidor FTP.

- **Conexión desde un navegador.** Para conectarse a un servidor FTP a través de un navegador, tendremos que utilizar una URL, al igual que en HTTP. La URL debe tener la siguiente forma:

  `ftp://usuario@dominioftp.gov/`

  A diferencia de una URL para acceder a un recurso a través de HTTP, aquí el protocolo que se especifica en la dirección es `ftp://`. Luego será necesario especificar el nombre de usuario con el que se desea realizar la conexión, seguido del caracter **@** y el dominio del servidor.
- **Conexión desde la línea de comandos.** En la mayoría de los sistemas operativos existe un comando para establecer conexiones FTP.
- **Conexión a través de una interfaz gráfica.** Además de la línea de comandos y el navegador, existen aplicaciones gráficas que nos permiten acceder a un servidor FTP y navegar los archivos. Generalmente estas aplicaciones permiten otras operaciones como modificación de los archivos en el servidor, copia, eliminación, etc. Si bien esto es posible realizarlo íntegramente desde la línea de comandos, una interfaz gráfica nos simplificará las tareas.

Protocolo SSH

El protocolo SSH (*Secure Shell*), es un método para iniciar una sesión segura, en una computadora remota, utilizando distintos métodos como autenticación con passwords, llaves SSH, etc. Una vez iniciada la sesión, la información transmitida desde el servidor hacia el cliente, y viceversa.

El protocolo SSH puede utilizarse para:

- Proveer acceso a usuarios y procesos automáticos a un servidor remoto.
- Transferencia de archivos. Si bien se puede utilizar FTP para ello, SSH ofrece una alternativa más segura.
- Ejecutar comandos de manera remota.

Existen varias muchas maneras de conectarse a un servidor utilizando el protocolo SSH. En GNU/Linux o sistemas operativos basados en Unix, se puede utilizar el comando ***ssh*** especificando el servidor al que se desea conectar. En Microsoft Windows, existen distintas aplicaciones que permiten establecer una comunicación SSH.

Redes P2P

Las redes P2P, son aquellas que conectan múltiples clientes entre sí (también llamados ***pares*** o ***peers***), de una manera determinada y con un fin específico. Este tipo se utiliza generalmente para compartir archivos; aplicaciones, archivos de video, imágenes, documentos, etc. A diferencia de las redes tradicionales cliente-servidor, donde hay siempre un servidor (o múltiples servidores, pero que sirven para un mismo propósito), y múltiples clientes, en las redes P2P, cada nodo (dispositivo conectado a la red), tiene el rol de cliente y servidor al mismo tiempo. Esto hace que cada cliente pueda estar consumiendo recursos de otro cliente o que esté sirviendo recursos para un cliente remoto.