Contenidos teóricos

Sitio: Agencia de Aprendizaje a lo largo de la Vida
Curso: CAC DJANGO 2023 1C
Libro: Contenidos teóricos

Imprimido por: Lucas Rodriguez
Día: martes, 7 de marzo de 2023, 11:16

Tabla de contenidos

1. Internet

Internet

Internet es una red informática que interconecta cientos de millones de dispositivos informáticos en todo el mundo. No hace mucho tiempo, estos dispositivos informáticos eran principalmente PC de escritorio tradicionales, estaciones de trabajo Linux y los llamados servidores que almacenan y transmiten información como páginas web y mensajes de correo electrónico. Sin embargo, cada vez más,
los sistemas finales de Internet no tradicionales, como computadoras portátiles, teléfonos inteligentes, tabletas, televisores, consolas de juegos, cámaras web, automóviles, dispositivos de detección ambiental, imagen los marcos y los sistemas eléctricos y de seguridad del hogar se están conectando a Internet. De hecho, el término red informática está empezando a sonar un poco anticuado, dados los muchos dispositivos no tradicionales que se están conectando a Internet. En la jerga de Internet, todos estos dispositivos se denominan hosts o sistemas finales. A partir de julio de 2011, había casi 850 millones de sistemas finales conectados a Internet [ISC 2012], sin contar los teléfonos inteligentes, computadoras portátiles y otros dispositivos que solo están conectados intermitentemente a Internet. En total, se estima que hay más de 2.000 millones de usuarios de Internet [UIT 2011]. Los sistemas finales están conectados entre sí por una red de enlaces de **comunicación** y **conmutadores de paquetes**. Veremos en la Sección 1.2 que hay muchos tipos de enlaces de comunicación, que se componen de diferentes tipos de medios físicos, incluidos cables coaxial, alambre de cobre, fibra óptica y espectro de radio. Diferentes enlaces pueden transmitir datos a diferentes velocidades, con la **velocidad de transmisión** de un enlace medida en bits/segundo. Cuando un sistema final tiene datos para enviar a otro sistema final, el sistema final emisor segmenta los datos y agrega bytes de encabezado a cada segmento. Los paquetes de información resultantes, conocidos como
paquetes en la jerga de las redes informáticas, se envían a través de la red al sistema final de destino, donde se vuelven a ensamblar con los datos originales. Un conmutador de paquetes toma un paquete que llega a uno de sus enlaces de comunicación entrantes y reenvía ese paquete a uno de sus enlaces de comunicación salientes. Los conmutadores de paquetes vienen en muchas formas y sabores, pero los dos tipos más prominentes en internet hoy en día son **los enrutadores** y **los conmutadores de capa de enlace**. Ambos tipos de conmutadores reenvían paquetes hacia sus destinos finales. Los conmutadores de capa de enlace se utilizan normalmente en las redes de acceso, mientras que los enrutadores se utilizan normalmente en el núcleo de la red. La frecuencia de los enlaces de comunicación y los conmutadores de paquetes atravesados por un paquete desde el sistema final de envío hasta el sistema final receptor se conoce como **ruta** o **camino** a través de la red. La cantidad exacta de tráfico que se transporta en Internet es difícil de estimar, pero Cisco [Cisco VNI 2011] estimaba que el tráfico global de Internet era de casi 40 exabytes por mes en 2012.Las redes de conmutación de paquetes (que transportan paquetes) son en muchos aspectos similares a las redes de transporte de autopistas, carreteras e intersecciones (que transportan vehículos). Considere, por ejemplo, una fábrica que necesita mover una gran cantidad de carga a algún almacén de destino ubicado a miles de kilómetros de distancia. En la fábrica, la carga se segmenta y se carga en una flota de camiones. Cada uno de los camiones viaja de forma independiente a través de la red de autopistas, carreteras e intersecciones hasta el almacén de destino. En el almacén de destino, la carga se descarga y se agrupa con el resto de la carga que llega del mismo envío. Por lo tanto, en muchos sentidos, los paquetes son análogos a los camiones, los enlaces de comunicación son análogos a las autopistas y carreteras, los conmutadores de paquetes son análogos a las intersecciones y los sistemas finales son análogos a los edificios. Así como un camión toma un camino a través de la red de transporte, un paquete toma un camino a través de una red de computadora. Los sistemas finales acceden a Internet a través **de proveedores de servicios de Internet (ISP),** incluidos los ISP residenciales, como las compañías locales de cable o telefonía; los ISP corporativos; los ISP universitarios; y los ISP que proporcionan acceso WiFi en aeropuertos, hoteles, coffeeshops y otros lugares públicos. Cada ISP es en sí mismo una red de conmutadores de paquetes y enlaces de comunicación. Los ISP proporcionan una variedad de tipos de acceso de red a los sistemas finales, incluido el acceso de banda ancha residencial, como el módem por cable o el acceso a la red de área local de alta velocidad DSL, el acceso inalámbrico y el acceso telefónico de módem de 56 kbps. Los ISP también proporcionan acceso a Internet a los proveedores de contenido, conectando sitios web directamente a Internet. Internet se trata de conectar los sistemas finales entre sí, por lo que los ISP que proporcionan acceso a los sistemas finales también deben estar interconectados. Estos ISP de nivel inferior están interconectados a través de ISP de nivel superior nacionales e internacionales, como Level 3 Communications, AT&T, Sprint y NTT. Un ISP de nivel superior consiste en enrutadores de alta velocidad interconectados con enlaces ópticos de fibra de alta velocidad. Cada red ISP, ya sea de nivel superior o inferior, se administra de forma independiente, ejecuta el protocolo IP (se verá luego) y se ajusta a ciertas convenciones de nomenclatura y dirección. Puede examinar los ISP y su interconexión más a fondo en la Sección 1.3 del libro que se brinda como referencia. Los sistemas finales, los conmutadores de paquetes y otras partes de Internet ejecutan **protocolos** que controlan el envío y la recepción de información dentro de Internet. El **Protocolo de Control de Transmisión (TCP)** y el **Protocolo de Internet (IP)** son dos de los protocolos más importantes de Internet. El protocolo IP especifica el formato de los paquetes que se envían y reciben entre los enrutadores y los sistemas finales. Los protocolos principales de Internet se conocen colectivamente como **TCP/IP**. Dada la importancia de los protocolos para Internet, es importante que todos estén de acuerdo sobre lo que hacen todos y cada uno de los protocolos, para que las personas puedan crear sistemas y productos que interoperen. Aquí es donde entran en juego los estándares.
Los estándares de Internet son desarrollados por el Grupo de Trabajo de Ingeniería de Internet (IETF) [IETF 2012]. Los documentos de estándares de IETF se denominan **solicitudes de comentarios (RFC).** Los RFC comenzaron como solicitudes generales de comentarios (de ahí el nombre) para resolver los problemas de diseño de redes y protocolos que enfrentó el precursor de Internet [Allman 2011]. Los RFC tienden a ser bastante técnicos y detallados. Definen protocolos como TCP, IP, HTTP (para la Web) y SMTP (para el correo electrónico). Actualmente hay más de 6.000 RFC. Otros organismos también especifican normas para los componentes de la red, en particular para los enlaces de red. El Comité de Estándares IEEE 802 LAN/MAN [IEEE 8022012], por ejemplo, especifica los estándares Ethernet y WiFi inalámbrico.

Bibliografía: James F. Kurose, Keith W. Ross. Computer Networking: A top-down approach. (6º ed.)

2. Protocolos

El modelo de referencia TCP/IP

Pasemos ahora del modelo de referencia OSI al modelo de referencia que se utiliza en la más vieja de todas las redes de computadoras de área amplia: ARPANET y su sucesora, Internet. Aunque más adelante veremos una breve historia de ARPANET, es conveniente mencionar ahora unos cuantos aspectos de esta red. ARPANET era una red de investigación patrocinada por el **DoD** (**Departamento de Defensa de Estados Unidos**, del inglés *U.S. Department of the Defense*). En un momento dado llegó a conectar cientos de universidades e instalaciones gubernamentales mediante el uso de líneas telefónicas rentadas.
Cuando después se le unieron las redes de satélites y de radio, los protocolos existentes tuvieron problemas para interactuar con ellas, de modo que se necesitaba una nueva arquitectura de referencia. Así, casi desde el principio la habilidad de conectar varias redes sin problemas fue uno de los principales objetivos de diseño. Posteriormente esta arquitectura se dio a conocer como el **Modelo de referencia TCP/IP**, debido a sus dos protocolos primarios. Este modelo se definió por primera vez en Cerf y Kahn (1974); después se refinó y definió como estándar en la comunidad de Internet (Braden, 1989). Clark (1988) describe la filosofía de diseño detrás de este modelo.
Debido a la preocupación del DoD de que alguno de sus valiosos hosts, enrutadores y puertas de enlace de interredes pudieran ser volados en pedazos en cualquier momento por un ataque de la antigua Unión Soviética, otro de los objetivos principales fue que la red pudiera sobrevivir a la pérdida de hardware de la subred sin que se interrumpieran las conversaciones existentes. En otras palabras, el DoD quería que las conexiones permanecieran intactas mientras las máquinas de origen y de destino estuvieran funcionando, incluso aunque algunas de las máquinas o líneas de transmisión en el trayecto dejaran de funcionar en forma repentina. Además, como se tenían en mente aplicaciones con requerimientos divergentes que abarcaban desde la transferencia de archivos hasta la transmisión de voz en tiempo real, se necesitaba una arquitectura flexible.

La capa de enlace

Todos estos requerimientos condujeron a la elección de una red de conmutación de paquetes basada en una capa sin conexión que opera a través de distintas redes. La capa más baja en este modelo es la **capa de enlace**; ésta describe qué enlaces (como las líneas seriales y Ethernet clásica) se deben llevar a cabo para cumplir con las necesidades de esta capa de interred sin conexión. En realidad, no es una capa en el sentido común del término, sino una interfaz entre los hosts y los enlaces de transmisión. El primer material sobre el modelo TCP/IP tiene poco que decir sobre ello.

La capa de interred

Esta capa es el eje que mantiene unida a toda la arquitectura. Aparece en la figura 1-21 con una correspondencia aproximada a la capa de red de OSI. Su trabajo es permitir que los hosts inyecten paquetes en
cualquier red y que viajen de manera independiente hacia el destino (que puede estar en una red distinta).
Incluso pueden llegar en un orden totalmente diferente al orden en que se enviaron, en cuyo caso es responsabilidad de las capas más altas volver a ordenarlos, si se desea una entrega en orden. Tenga en cuenta que aquí utilizamos "interred" en un sentido genérico, aunque esta capa esté presente en la Internet.
La analogía aquí es con el sistema de correos convencional (lento). Una persona puede dejar una
secuencia de cartas internacionales en un buzón en un país y, con un poco de suerte, la mayoría de ellas se entregarán a la dirección correcta en el país de destino. Es probable que las cartas pasen a través de una o más puertas de enlace de correo internacionales en su trayecto, pero esto es transparente a los usuarios.
Además, los usuarios no necesitan saber que cada país (es decir, cada red) tiene sus propias estampillas, tamaños de sobre preferidos y reglas de entrega.

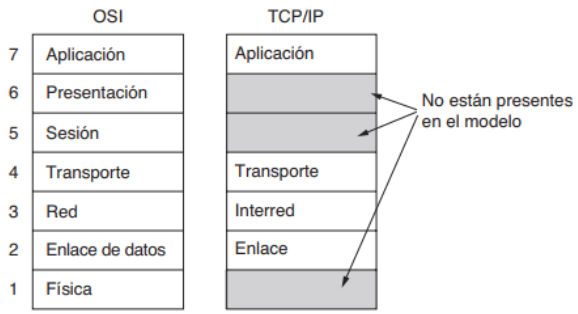


Figura 1-21. El modelo de referencia TCP/IP.

La capa de interred define un formato de paquete y un protocolo oficial llamado **IP** (**Protocolo de Internet**, del inglés *Internet Protocol*), además de un protocolo complementario llamado **ICMP** (**Protocolo de Mensajes de Control de Internet**, del inglés *Internet Control Message Protocol*) que le ayuda a funcionar. La tarea de la capa de interred es entregar los paquetes IP a donde se supone que deben ir.
Aquí el ruteo de los paquetes es sin duda el principal aspecto, al igual que la congestión (aunque el IP no ha demostrado ser efectivo para evitar la congestión)

La capa de transporte

Por lo general, a la capa que está arriba de la capa de interred en el modelo TCP/IP se le conoce como **capa de transporte**; y está diseñada para permitir que las entidades pares, en los nodos de origen y de destino, lleven a cabo una conversación, al igual que en la capa de transporte de OSI. Aquí se definieron dos protocolos de transporte de extremo a extremo. El primero, **TCP** (**Protocolo de Control de la Transmisión**, del inglés *Transmission Control Protocol*), es un protocolo confiable orientado a la conexión que permite que un flujo de bytes originado en una máquina se entregue sin errores a cualquier otra máquina en la interred. Este protocolo segmenta el flujo de bytes entrante en mensajes discretos y pasa cada uno a la capa de interred. En el destino, el proceso TCP receptor vuelve a ensamblar los mensajes recibidos para formar el flujo de salida. El TCP también maneja el control de flujo para asegurar que un emisor rápido no pueda

inundar a un receptor lento con más mensajes de los que pueda manejar.

El segundo protocolo en esta capa, **UDP** (**Protocolo de Datagrama de Usuario**, del inglés *User Datagram Protocol*), es un protocolo sin conexión, no confiable para aplicaciones que no desean la asignación de secuencia o el control de flujo de TCP y prefieren proveerlos por su cuenta. También se utiliza mucho en las consultas de petición-respuesta de una sola ocasión del tipo cliente-servidor, y en las aplicaciones en las que es más importante una entrega oportuna que una entrega precisa, como en la transmisión de voz o video. En la figura 1-22 se muestra la relación entre IP, TCP y UDP. Desde que se desarrolló el modelo, el IP se ha implementado en muchas otras redes.

La capa de aplicación

El modelo TCP/IP no tiene capas de sesión o de presentación, ya que no se consideraron necesarias. Las aplicaciones simplemente incluyen cualquier función de sesión y de presentación que requieran. La experiencia con el modelo OSI ha demostrado que esta visión fue correcta: estas capas se utilizan muy poco en la mayoría de las aplicaciones.

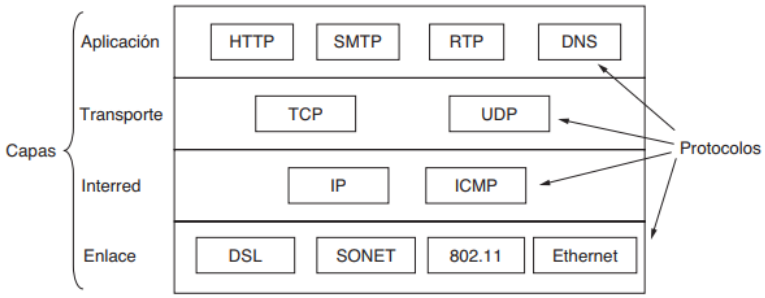


Figura 1-22. El modelo TCP/IP con algunos de los protocolos.

Encima de la capa de transporte se encuentra la **capa de aplicación**. Ésta contiene todos los protocolos de alto nivel. Entre los primeros protocolos están el de terminal virtual (TELNET), transferencia de archivos (FTP) y correo electrónico (SMTP). A través de los años se han agregado muchos otros protocolos. En la figura 1-22 se muestran algunos de los más importantes que veremos más adelante: el Sistema de nombres de dominio (DNS) para resolución de nombres de hosts a sus direcciones de red; HTTP, el protocolo para recuperar páginas de la World Wide Web; y RTP, el protocolo para transmitir medios en tiempo real, como voz o películas.

Bibliografía: Andrew S. Tanenbaum, David J. Wetherall. Redes de Computadoras (5º ed).

3. Arquitecturas

Arquitecturas

Cliente/Servidor

La mayoría de las empresas tienen una cantidad considerable de computadoras. Por ejemplo, tal vez una empresa tenga una computadora para cada empleado y las utilice para diseñar productos, escribir folletos y llevar la nómina. Al principio, algunas de estas computadoras tal vez hayan trabajado aisladas unas de otras, pero en algún momento, la administración podría decidir que es necesario conectarlas para distribuir la información en toda la empresa.
En términos generales, el asunto es **compartir recursos** y la meta es que todos los programas, equipo y en especial los datos estén disponibles para cualquier persona en la red, sin importar la ubicación física del recurso o del usuario. Un ejemplo obvio y de uso popular es el de un grupo de empleados de oficina que comparten una impresora. Ninguno de los individuos necesita realmente una impresora privada, por otro lado, una impresora en red de alto volumen es más económica, veloz y fácil de mantener que una extensa colección de impresoras individuales.
Pero, probablemente, compartir información sea aún más importante que compartir recursos físicos
como impresoras y sistemas de respaldo en cinta magnética. Las empresas tanto pequeñas como grandes dependen vitalmente de la información computarizada. La mayoría tiene registros de clientes, información de productos, inventarios, estados de cuenta, información fiscal y muchos datos más en línea. Si de repente todas sus computadoras se desconectaran de la red, un banco no podría durar más de cinco minutos. Una planta moderna de manufactura con una línea de ensamble controlada por computadora no duraría ni cinco segundos. Incluso una pequeña agencia de viajes o un despacho legal compuesto de tres personas son altamente dependientes de las redes de computadoras para permitir a los empleados acceder a la información y los documentos relevantes de manera instantánea.
En las empresas más pequeñas es probable que todas las computadoras se encuentren en una sola
oficina o tal vez en un solo edificio, pero en las empresas más grandes las computadoras y empleados se encuentran esparcidos en docenas de oficinas y plantas en muchos países. Sin embargo, un vendedor en Nueva York podría requerir acceso a una base de datos que se encuentra en Singapur. Las redes conocidas como **VPN** (**Redes Privadas Virtuales**, del inglés *Virtual Private Networks*) se pueden usar para unir las redes individuales, ubicadas en distintos sitios, en una sola red extendida. En otras palabras, el simple hecho de que un usuario esté a 15 000 km de distancia de sus datos no debe ser impedimento para que los utilice como si fueran locales. Podemos sintetizar este objetivo al decir que es un intento por acabar con la "tiranía de la geografía".
En términos más simples, imaginemos el sistema de información de una empresa como si estuviera
constituido por una o más bases de datos con información de la empresa y cierto número de empleados que necesitan acceder a esos datos en forma remota. En este modelo, los datos se almacenan en poderosas computadoras denominadas **servidores**. A menudo estos servidores están alojados en una ubicación central y un administrador de sistemas se encarga de su mantenimiento. Por el contrario, los empleados tienen en sus escritorios máquinas más simples conocidas como **clientes**, con las cuales acceden a los datos remotos, por ejemplo, para incluirlos en las hojas de cálculo que desarrollan (algunas veces nos referiremos al usuario humano del equipo cliente como el "cliente", aunque el contexto debe dejar en claro si nos referimos a la computadora o a su usuario). Las máquinas cliente y servidor se conectan mediante una red, como se muestra en la figura 1-1. Observe que mostramos la red como un óvalo simple, sin ningún detalle. Utilizaremos esta forma cuando hablemos de una red en el sentido más abstracto. Proveeremos los detalles según se requieran.
A esta disposición se le conoce como **modelo cliente-servidor**. Es un modelo ampliamente utilizado
y forma la base de muchas redes. La realización más popular es la de una **aplicación web**, en la cual el servidor genera páginas web basadas en su base de datos en respuesta a las solicitudes de los clientes que pueden actualizarla. El modelo cliente-servidor es aplicable cuando el cliente y el servidor se encuentran en el mismo edificio (y pertenecen a la misma empresa), pero también cuando están muy alejados. Por ejemplo, cuando una persona accede desde su hogar a una página en la World Wide Web se emplea el mismo
modelo, en donde el servidor web remoto representa al servidor y la computadora personal del usuario
representa al cliente. En la mayoría de las situaciones un servidor puede manejar un gran número (cientos o
miles) de clientes simultáneamente.
Si analizamos detalladamente el modelo cliente-servidor, podremos ver que hay dos procesos (es decir, programas en ejecución) involucrados: uno en la máquina cliente y otro en la máquina servidor. La comunicación ocurre cuando el

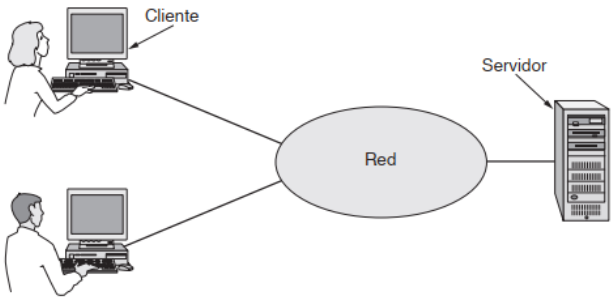


Figura 1-1. Una red con dos clientes y un servidor.

proceso cliente envía un mensaje a través de la red al proceso servidor.
El proceso cliente espera un mensaje de respuesta. Cuando el proceso servidor obtiene la solicitud, lleva a cabo la tarea solicitada o busca los datos solicitados y devuelve una respuesta. Estos mensajes se muestran
en la figura 1-2.
Un segundo objetivo al establecer una red de computadoras se relaciona con las personas y no con la información o con las computadoras. Una red de computadoras puede proveer un poderoso **medio de comunicación** entre los empleados. Ahora casi todas las empresas que tienen dos o más computadoras

usan el **email** (**correo electrónico**), generalmente para la comunicación diaria. De hecho, una de las quejas comunes que se escucha por parte de los empleados a la hora de sus descansos es la gran cantidad de correos electrónicos con la que tienen que lidiar, pues la mayoría son sin sentido debido a que los jefes han descubierto que pueden enviar el mismo mensaje (a menudo sin contenido) a todos sus subordinados con sólo oprimir un botón.
En algunos casos, las llamadas telefónicas entre los empleados se pueden realizar a través de la red de computadoras en lugar de usar la compañía telefónica. A esta tecnología se le conoce como **telefonía IP** o **Voz sobre IP** (**VoIP**) cuando se utiliza la tecnología de Internet. El micrófono y el altavoz en cada extremo pueden ser de un teléfono habilitado para VoIP o la computadora del empleado. Para las empresas ésta es una

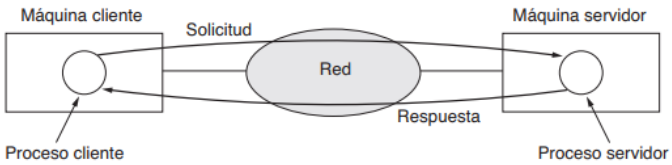


Figura 1-2. El modelo cliente-servidor implica solicitudes y respuestas.

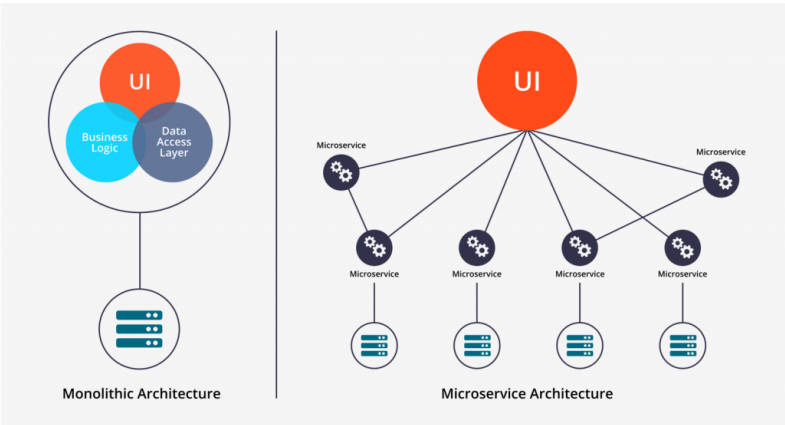
maravillosa forma de ahorrar en sus cuentas telefónicas.

Las redes de computadoras hacen posibles otras formas de comunicación más completas. Se puede agregar video al audio de manera que los empleados en ubicaciones distantes se puedan ver y escuchar mientras sostienen una reunión. Esta técnica es una poderosa herramienta para eliminar el costo y el tiempo dedicados a viajar. Los **escritorios compartidos** permiten a los trabajadores remotos ver una pantalla gráfica de computadora e interactuar con ella. Gracias a ello es posible que dos o más personas que trabajan a distancia lean y escriban en un pizarrón compartido, o escriban juntos un informe. Cuando un empleado realiza una modificación en un documento en línea, los demás pueden ver esa modificación de inmediato, en vez de tener que esperar varios días para recibir una carta. Dicha agilización facilita la cooperación entre los grupos remotos de personas, lo cual antes hubiera sido imposible. Hasta ahora se están empezando a utilizar formas más ambiciosas de coordinación remota como la telemedicina (por ejemplo, el monitoreo remoto de pacientes), lo cual puede tomar aún más importancia en un futuro cercano. En ocasiones se dice que la comunicación y el transporte están en constante competencia, y quien resulte ganador hará que el perdedor se vuelva obsoleto.
Un tercer objetivo para muchas empresas es realizar negocios electrónicamente, en especial con los clientes y proveedores. A este nuevo modelo se le denomina **e-commerce** (**comercio electrónico**) y ha crecido con rapidez en los años recientes. Las aerolíneas, librerías y otros vendedores han descubierto que a muchos clientes les gusta la conveniencia de comprar desde su hogar. En consecuencia, muchas empresas proveen catálogos de sus artículos y servicios en línea, e incluso reciben pedidos en línea. Los fabricantes de automóviles, aeronaves y computadoras entre otros, compran subsistemas de una variedad de proveedores y después ensamblan las piezas. Mediante el uso de redes de computadoras, los fabricantes pueden colocar los pedidos en forma electrónica según sea necesario. Esto reduce la necesidad de tener grandes inventarios y mejora la eficiencia.

Microservicios

La arquitectura de microservicios (del inglés Micro Services Architecture) es un método de desarrollo de software que consiste en construir una aplicación como un conjunto de pequeños servicios, con operaciones bien definidas e independientes entre sí.

Cada microservicio ejecuta su propio proceso y se encarga de implementar una funcionalidad completa del negocio. Puede estar programado en distintos lenguajes y usar diferentes tecnologías de almacenamiento de datos. A la hora de hacer un despliegue, cada servicio se hace de forma independiente.



Las principales diferencias entre una estructura tradicional y una de microservicios, es que la tradicional se realiza de forma monolítica, es decir, todas las partes de la aplicación que podían implementarse se encontraban en esa única aplicación de forma integral. Al contrario, una de microservicios descompone una aplicación en sus funciones principales. Cada función se denomina microservicio y se puede diseñar e implementar de forma independiente. Esto permite que funcionen de forma aislada a los demás servicios de la plataforma.

La tendencia se ha hecho popular en los últimos años a medida que las empresas buscan ser más ágiles y avanzar hacia un DevOps y pruebas continuas. Los microservicios pueden ayudar a crear software más rápido, que sea escalable y lograr un modelo nativo de la nube.

Redes de igual a igual (peer to peer o p2p)

No todos pueden establecer una CDN con 1000 nodos en ubicaciones en todo el mundo para distribuir su contenido (en realidad no es difícil rentar 1000 máquinas virtuales alrededor del mundo, debido a la industria de hospedaje tan bien desarrollada y competitiva. Sin embargo, el proceso de establecer una CDN apenas empieza cuando se obtienen los nodos). Por suerte hay una alternativa para el resto de nosotros que es simple para usar

y puede distribuir una enorme cantidad de contenido. Nos referimos a la red P2P (igual a igual).
Las redes P2P empezaron a funcionar a partir de 1999. La primera aplicación extendida fue para el crimen en masa: 50 millones de usuarios de Napster intercambiaban canciones protegidas por derechos de autor sin el permiso de los propietarios de esos derechos, hasta que la corte cerró Napster en medio de una gran controversia. Sin embargo, la tecnología de igual a igual tiene muchos usos interesantes y legales. Otros sistemas continuaron su desarrollo, con tanto interés por parte de los usuarios que el tráfico P2P pronto eclipsó al tráfico web. En la actualidad, BitTorrent es el protocolo P2P más popular. Se usa tanto para compartir videos (con licencia y del dominio público) al igual que otro tipo de contenido, que es responsable de una gran parte de todo el tráfico de Internet. Lo analizaremos en esta sección.
La idea básica de una red de compartición de archivos P2P (Igual a Igual, del inglés *Peer-to-Peer*)
es que muchas computadoras se conecten entre sí y reserven sus recursos para formar un sistema de distribución de contenido. Con frecuencia las computadoras son sólo domésticas. No necesitan ser máquinas en centros de datos de Internet. A las computadoras se les llama iguales debido a que cada una de ellas puede actuar de manera alternativa como cliente para otro igual y obtener su contenido, y como servidor para proveer contenido a otros iguales. Lo que hace interesantes a los sistemas de igual a igual es que no hay una infraestructura dedicada, a diferencia de una CDN. Todos participan en la tarea de distribuir contenido, y a menudo no hay punto central de control.

Muchas personas están entusiasmadas sobre la tecnología P2P, ya que se ve como algo que otorga
poder al débil. La razón no sólo es que se necesita de una empresa grande para operar una CDN, mientras que cualquiera con una computadora se puede unir a una red P2P. Resulta ser que las redes P2P tienen una capacidad formidable para distribuir contenido que puede rivalizar con el mayor de los sitios web.
Considere una red P2P compuesta de N usuarios promedio, cada uno de ellos con conectividad de
banda ancha a 1 Mbps. La capacidad de envío agregada de la red P2P, o tasa a la que los usuarios pueden enviar tráfico a Internet, es de N Mbps. La capacidad de descarga, o tasa a la que los usuarios pueden recibir tráfico, es también de N Mbps. Cada usuario puede enviar y descargar al mismo tiempo, ya que tienen un enlace de 1 Mbps en cada dirección.
No es obvio que esto deba ser cierto, pero resulta que toda la capacidad se puede usar de manera
productiva para distribuir contenido, incluso para el caso de compartir una sola copia de un archivo con todos los demás usuarios. Para ver cómo puede hacerse esto, imagine que los usuarios están organizados en un árbol binario, en donde cada usuario que no es nodo hoja envía datos a otros dos usuarios. El árbol transportará la única copia del archivo a todos los demás usuarios. Para usar el ancho de banda de envío de todos los usuarios posibles en todo momento (y por ende, distribuir el archivo grande con poca latencia), necesitamos canalizar la actividad de los usuarios en la red. Imagine que el archivo se divide en 1000 piezas. Cada usuario puede recibir una nueva pieza desde cualquier parte en un nivel superior del árbol y enviar la pieza recibida previamente hacia abajo por el árbol al mismo tiempo. De esta forma, una vez que se inicia la canalización y después de enviar un pequeño número de piezas (equivalente a la profundidad del árbol), todos los usuarios que no sean nodos hoja estarán ocupados enviando el archivo a otros usuarios. Como hay en promedio $N/2$ usuarios que no son nodos hoja, el ancho de banda de envío de este árbol es de $N/2$ Mbps. Podemos repetir este truco y crear otro árbol que use los otros $N/2$ Mbps de ancho de banda de envío si intercambiamos los roles de los nodos hoja y los que no lo son. En conjunto, esta construcción utiliza toda la capacidad.
Este argumento significa que las redes P2P son autoescalables. Su capacidad de envío útil aumenta
junto con las demandas de descarga que pueden tener sus usuarios. Siempre son lo "bastante grandes" en cierto sentido, sin la necesidad de una infraestructura dedicada. Por el contrario, hasta la capacidad de un sitio web grande es fija. Considere un sitio con sólo 100 clústeres, cada uno capaz de transmitir 10 Gbps. Esta enorme capacidad no ayuda cuando hay una cantidad pequeña de usuarios. El sitio no puede transmitir información a N usuarios a una tasa más rápida que N Mbps debido a que el límite está en los usuarios y no en el sitio web. Y cuando hay más de un millón de usuarios de 1 Mbps, el sitio web no puede transmitir datos con la suficiente rapidez como para mantener a todos los usuarios ocupados descargando. Esto puede parecer una gran cantidad de usuarios, pero las redes BitTorrent extensas (como Pirate Bay) afirman tener más de 10 000 000 usuarios. ¡Esto es más aproximado a 10 terabits/seg en términos de nuestro ejemplo! Hay que tomar estas cifras con mucho cuidado, debido a que simplifican más de la cuenta la situación.
Un reto considerable para las redes P2P es usar bien el ancho de banda cuando puede haber usuarios de todo tipo y tamaño, y pueden tener distintas capacidades de envío y descarga. Sin embargo, estas cifras indican el enorme potencial de P2P.
Hay otra razón por la que las redes P2P son importantes. Las redes CDN y otros servicios de operación central ponen a los proveedores en la posición de tener un tesoro de información personal sobre muchos usuarios, desde sus preferencias de navegación y en donde compran las personas en línea, hasta sus ubicaciones y direcciones de correo electrónico. Esta información se puede utilizar para proveer un mejor servicio y más personalizado, o se puede usar para invadir la privacidad de las personas. Esto último puede ocurrir en forma intencional (por ejemplo, como parte de un nuevo producto) o por medio de una divulgación accidental que comprometa esa información. En los sistemas P2P no puede haber un solo proveedor que sea capaz de monitorear todo el sistema completo. Esto no necesariamente significa que los sistemas P2P provean privacidad, ya que los usuarios confían entre sí hasta cierto grado. Sólo significa que pueden proveer una manera distinta de privacidad a la de los sistemas administrados en forma central. En la actualidad se están explorando los sistemas P2P para ofrecer servicios más allá de la compartición de archivos (por ejemplo, almacenamiento, flujo continuo); el tiempo dirá si esta ventaja es importante o no.
La tecnología P2P ha seguido dos trayectorias relacionadas en el transcurso de su desarrollo. En el lado más práctico están los sistemas que se utilizan a diario. Los más conocidos de estos sistemas se basan en el protocolo BitTorrent. En el lado más académico, ha existido un intenso interés en los algoritmos DHT (Tabla de Hash Distribuida) que permiten a los sistemas P2P funcionar bien como un todo, pero que para nada dependen de componentes centralizados.

4. World Wide Web

World Wide Web

Desde el punto de vista del usuario, la web consiste en una enorme colección de contenido en forma de **páginas web**, por lo general, conocidas simplemente como **páginas**. Cada una puede contener vínculos a otras páginas en cualquier lugar del mundo. Para seguir un vínculo, los usuarios pueden hacer clic en él, y a continuación los llevará a la página apuntada. Este proceso se puede repetir de manera indefinida. La idea de hacer que una página apunte a otra, lo que ahora se conoce como **hipertexto**, fue inventada por un profesor visionario de ingeniería eléctrica del MIT, Vannevar Bush, en 1945 (Bush, 1945). Esto fue mucho antes de que se inventara Internet. De hecho, fue antes de que existieran las computadoras comerciales, aunque varias universidades habían producido prototipos que ocupaban habitaciones extensas, pero tenían menos poder que una moderna calculadora de bolsillo. Por lo general, las páginas se ven mediante un programa llamado **navegador**. Firefox, Internet Explorer y Chrome son ejemplos de navegadores populares. El navegador obtiene la página solicitada, interpreta el contenido y despliega la página en pantalla con el formato adecuado. El contenido en sí puede ser una mezcla de texto, imágenes y comandos de formato, ya sea en forma de un documento tradicional u otras formas de contenido, como un video o programas que produzcan una interfaz gráfica con la que puedan interactuar los usuarios.

En la parte superior izquierda de la figura 7-18 se muestra la imagen de una página. Ésta es la página del Departamento de Ciencias Computacionales e Ingeniería, de la Universidad de Washington. Esta

página muestra texto y elementos gráficos (que en su mayoría son muy pequeños como para leerlos). Algunas partes de la página están asociadas con vínculos a otras. A una pieza de texto, icono, imagen u otro elemento asociado con otra página se le conoce como hipervínculo. Para seguir un vínculo, el usuario coloca el cursor del ratón en la parte vinculada del área de la página (lo cual hace que el cursor cambie de forma) y hace clic. Seguir un vínculo es simplemente una forma de decir al navegador que obtenga otra página. En los primeros días de la web, los vínculos se resaltaban con texto subrayado y coloreado para que pudieran sobresalir. Hoy en día, los creadores de las páginas web tienen formas de controlar la apariencia de las regiones vinculadas, por lo que un vínculo podría aparecer como un icono o cambiar su apariencia cuando el ratón pase sobre él. Queda a criterio de los creadores de la página hacer que los vínculos se puedan distinguir en forma visual para proveer una interfaz que se pueda usar. Los estudiantes en el Departamento pueden aprender más si siguen un vínculo a una página con información especial para ellos. Para acceder a este vínculo deben hacer clic en el área encerrada en un círculo.

A continuación el navegador obtiene la nueva página y la despliega en pantalla, como se muestra de manera parcial en la parte inferior izquierda de la figura 7-18. Hay docenas de páginas adicionales a las que se puede acceder mediante vínculos desde la primera página, además de este ejemplo.

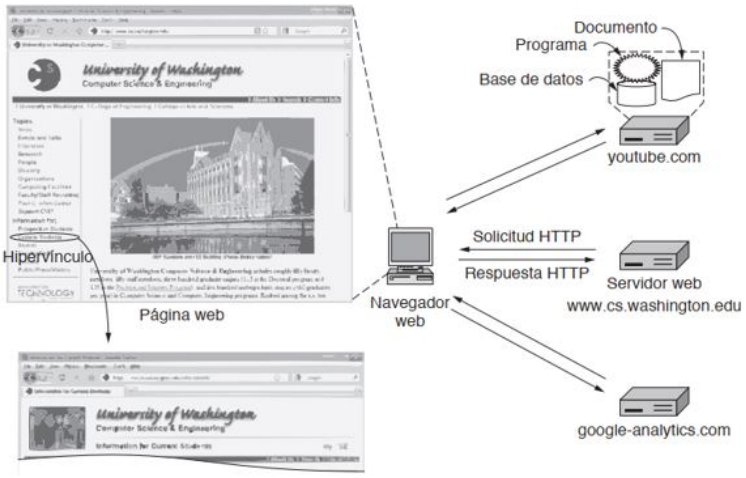


Figura 7-18. Arquitectura de la web.

Cualquier otra página puede estar compuesta de contenido en la(s) misma(s) máquina(s) que la primera página, o en máquinas en alguna parte del mundo. El usuario no puede distinguir esto. El navegador se encarga del proceso de obtención de las páginas, sin ninguna ayuda del usuario. De esta forma, el proceso de desplazarse entre máquinas al momento de ver contenido es transparente para el usuario.

En la figura 7-18 se muestra el modelo básico de la forma en que se despliegan las páginas en pantalla. El navegador despliega una página web en la máquina cliente. Para obtener cada página, se envía una solicitud a uno o más servidores, los cuales responden con el contenido de la página. El protocolo de solicitud-respuesta para obtener páginas es un protocolo simple basado en texto que se ejecuta sobre TCP, como en el caso de SMTP. Este protocolo se llama **HTTP** (**Protocolo de Transferencia de HiperTexto**, del inglés *HyperText Transfer Protocol*). El contenido puede ser simplemente un documento que se lea de un disco, o el resultado de una consulta en una base de datos y la ejecución de un programa. La página se considera una **página estática** si es el mismo documento cada vez que se despliega en pantalla. Por el contrario, si se generó bajo demanda mediante un programa o contiene uno, es una **página dinámica**. Una página dinámica se puede presentar de manera distinta cada vez que se despliega en pantalla. Por ejemplo, la página principal de una tienda electrónica puede ser distinta para cada visitante. Si el cliente de una librería ha comprado novelas de misterio en sus anteriores visitas, es probable que la próxima vez le aparezcan las nuevas novelas de misterio en la página de inicio; mientras que un cliente enfocado hacia lo culinario podría ser recibido con nuevos libros de cocina. La manera en que el sitio web mantiene el registro de los gustos de cada cliente es algo que veremos en breve. En sí, la respuesta involucra el uso de *cookies*.

En la figura anterior, el navegador se contacta con tres servidores para obtener las dos páginas: *cs.washington.edu*, *youtube.com* y *google-analytics.com*. El contenido de estos distintos servidores se integra para que el navegador lo despliegue. La visualización conlleva un rango de procesamiento que depende del tipo de contenido. Además de generar texto y gráficos, puede implicar la reproducción de un video o la ejecución de una secuencia de comandos (*script*) que presente su propia interfaz de usuario como parte de la página.

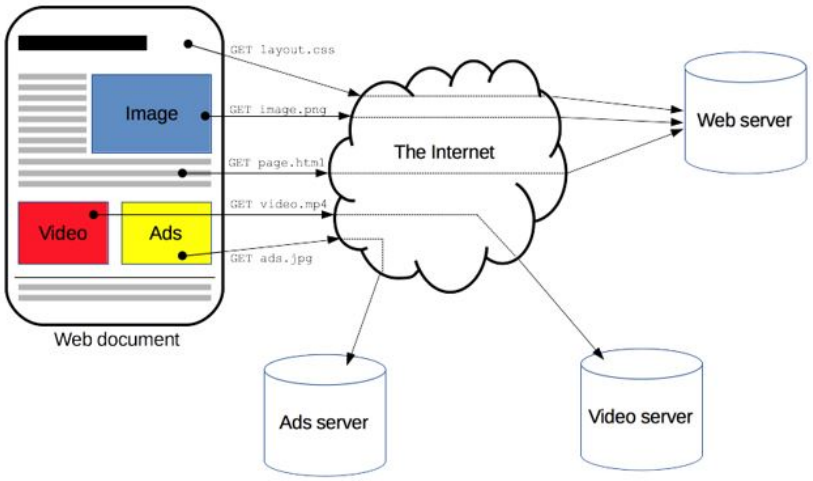
En este caso, el servidor *cs.washington.edu* suministra la página principal, el servidor *youtube.com* provee un video incrustado y el servidor *google-analytics.com* no suministra nada que el usuario pueda ver, pero rastrea los visitantes del sitio.

Bibliografía: Andrew S. Tanenbaum, David J. Wetherall. Redes de Computadoras (5º ed).

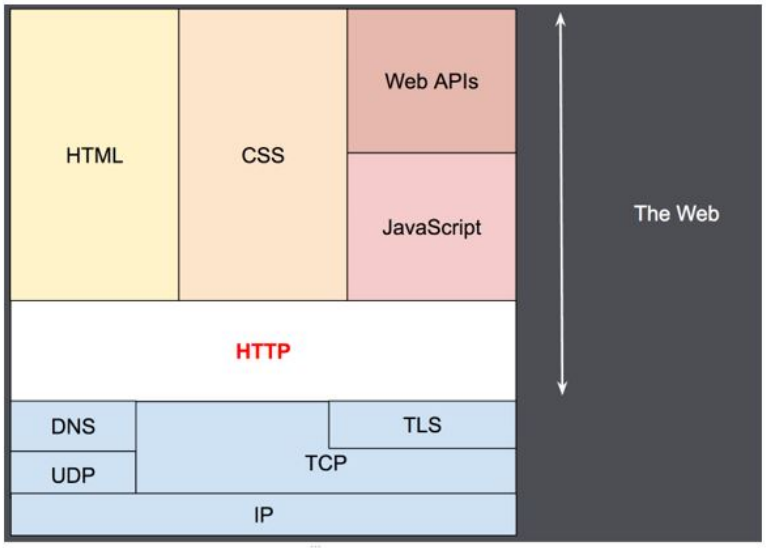
5. HTTP

HTTP

Hypertext Transfer Protocol (HTTP) (o *Protocolo de Transferencia de Hipertexto en español)* es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta de este. Se trata de un protocolo sin estado, lo que significa que el servidor no guarda ningún dato (estado) entre dos peticiones. Aunque en la mayoría de casos se basa en una conexión del tipo TCP/IP, puede ser usado sobre cualquier capa de transporte segura o de confianza, es decir, sobre cualquier protocolo que no pierda mensajes silenciosamente, tal como UDP.



Clientes y servidores se comunican intercambiando mensajes individuales (en contraposición a las comunicaciones que utilizan flujos continuos de datos). Los mensajes que envía el cliente, normalmente un navegador Web, se llaman *peticiones*, y los mensajes enviados por el servidor se llaman *respuestas*.



Diseñado a principios de la década de 1990, HTTP es un protocolo ampliable, que ha ido evolucionando con el tiempo. Es lo que se conoce como un protocolo de la capa de aplicación, y se transmite sobre el protocolo TCP, o el protocolo encriptado TLS (en-US), aunque teóricamente podría usarse cualquier otro protocolo fiable. Gracias a que es un protocolo capaz de ampliarse, se usa no solo para transmitir documentos de hipertexto (HTML), si no que además, se usa para transmitir imágenes o vídeos, o enviar datos o contenido a los servidores, como en el caso de los formularios de datos. HTTP puede incluso ser utilizado para transmitir partes de documentos, y actualizar páginas Web en el acto.

Características clave del protocolo HTTP

HTTP es sencillo

Incluso con el incremento de complejidad, que se produjo en el desarrollo de la versión del protocolo HTTP/2, en la que se encapsularon los mensajes, HTTP esta pensado y desarrollado para ser leído y fácilmente interpretado por las personas, haciendo de esta manera más facil la depuración de errores, y reduciendo la

curva de aprendizaje para las personan que empieza a trabajar con él.

HTTP es extensible

Presentadas en la versión HTTP/1.0, las cabeceras de HTTP, han hecho que este protocolo sea fácil de ampliar y de experimentar con él. Funcionalidades nuevas pueden desarrollarse, sin más que un cliente y su servidor, comprendan la misma semántica sobre las cabeceras de HTTP.

HTTP es un protocolo con sesiones, pero sin estados

HTTP es un protocolo sin estado, es decir: no guarda ningún dato entre dos peticiones en la misma sesión. Esto crea problemáticas, en caso de que los usuarios requieran interactuar con determinadas páginas Web de forma ordenada y coherente, por ejemplo, para el uso de "cestas de la compra" en páginas que utilizan en comercio electrónico. Pero, mientras HTTP ciertamente es un protocolo sin estado, el uso de HTTP cookies, si permite guardar datos con respecto a la sesión de comunicación. Usando la capacidad de ampliación del protocolo HTTP, las cookies permiten crear un contexto común para cada sesión de comunicación.

HTTP y conexiones

Una conexión se gestiona al nivel de la capa de trasporte, y por tanto queda fuera del alcance del protocolo HTTP. Aún con este factor, HTTP no necesita que el protocolo que lo sustenta mantenga una conexión continua entre los participantes en la comunicación, solamente necesita que sea un protocolo fiable o que no pierda mensajes (como mínimo, en todo caso, un protocolo que sea capaz de detectar que se ha pedido un mensaje y reporte un error). De los dos protocolos más comunes en Internet, TCP es fiable, mientras que UDP, no lo es. Por lo tanto HTTP, se apoya en el uso del protocolo TCP, que está orientado a conexión, aunque una conexión continua no es necesaria siempre.

En la versión del protocolo HTTP/1.0, habría una conexión TCP por cada petición/respuesta intercambiada, presentando esto dos grandes inconvenientes: abrir y crear una conexión requiere varias rondas de mensajes y por lo tanto resultaba lento. Esto sería más eficiente si se mandaran varios mensajes.

Para atenuar estos inconvenientes, la versión del protocolo HTTP/1.1 presentó el 'pipelining' y las conexiones persistentes: el protocolo TCP que lo transmitía en la capa inferior se podía controlar parcialmente, mediante la cabecera 'Connection'. La versión del protocolo HTTP/2 fue más allá y usa multiplexación de mensajes sobre un única conexión, siendo así una comunicación más eficiente.

Todavía hoy se sigue investigando y desarrollando para conseguir un protocolo de transporte más conveniente para el HTTP. Por ejemplo, Google está experimentado con QUIC, que se apoya en el protocolo UDP y presenta mejoras en la fiabilidad y eficiencia de la comunicación.

Flujo HTTP

Cuando el cliente quiere comunicarse con el servidor, tanto si es directamente con él, o a través de un proxy intermedio, realiza los siguientes pasos:

Abre una conexión TCP: la conexión TCP se usará para hacer una petición, o varias, y recibir la respuesta. El cliente pude abrir una conexión nueva, reusar una existente, o abrir varias a la vez hacia el servidor.

Hacer una petición HTTP: Los mensajes HTTP (previos a HTTP/2) son legibles en texto plano. A partir de la versión del protocolo HTTP/2, los mensajes se encapsulan en franjas, haciendo que no sean directamente interpretables, aunque el principio de operación es el mismo.

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!DOCTYPE html... (here comes the 29769 bytes of the requested web page)
```

- Leer la respuesta enviada por el servidor
- Cierre o reuso de la conexión para futuras peticiones.

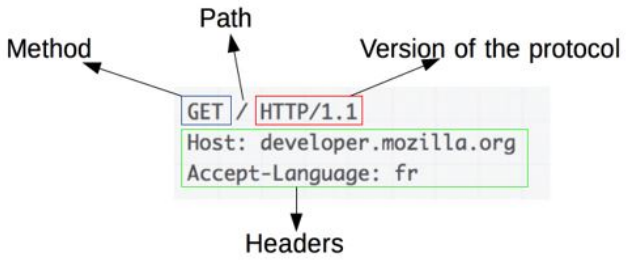
Si está activado el HTTP *pipelining*, varias peticiones pueden enviarse sin tener que esperar que la primera respuesta haya sido satisfecha. Este procedimiento es difícil de implementar en las redes de computadores actuales, donde se mezclan software antiguos y modernos. Así que el HTTP *pipelining* ha sido substituido en HTTP/2 por el multiplexado de varias peticiones en una sola trama.

Mensajes HTTP

En las versiones del protocolo HTTP/1.1 y anteriores los mensajes eran de formato texto y eran totalmente comprensibles directamente por una persona. En HTTP/2, los mensajes están estructurados en un nuevo formato binario y las tramas permiten la compresión de las cabeceras y su multiplexación. Así pues, incluso si solamente parte del mensaje original en HTTP se envía en este formato, la semántica de cada mensaje es la misma y el cliente puede formar el mensaje original en HTTP/1.1. Luego, es posible interpretar los mensajes HTTP/2 en el formato de HTTP/1.1.

Existen dos tipos de mensajes HTTP: peticiones (requests) y respuestas (responses), cada uno sigue su propio formato.

Peticiones (requests)



Un ejemplo de una petición HTTP:

Una petición de HTTP, está formado por los siguientes campos:

Un método HTTP, normalmente pueden ser un verbo, como: GET, POST o un nombre como: OPTIONS (en-US) o HEAD (en-US), que defina la operación que el cliente quiera realizar. El objetivo de un cliente, suele ser una petición de recursos, usando GET, o presentar un valor de un formulario HTML, usando POST, aunque en otras ocasiones puede hacer otros tipos de peticiones.

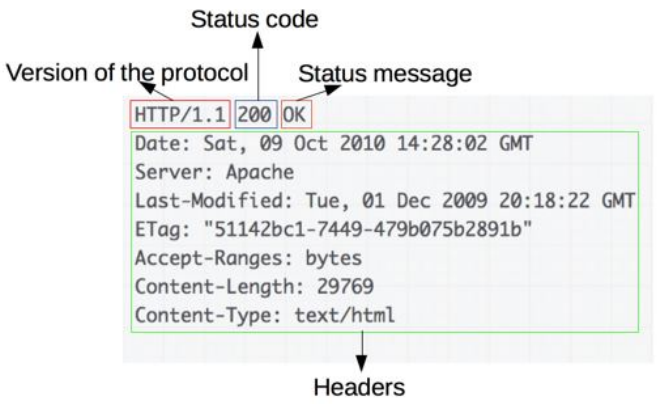
La dirección del recurso pedido; la URL del recurso, sin los elementos obvios por el contexto, como pueden ser: sin el protocolo (http://), el dominio (aquí developer.mozilla.org), o el puerto TCP (aquí el 80).

La versión del protocolo HTTP.

Cabeceras HTTP opcionales, que pueden aportar información adicional a los servidores.

O un cuerpo de mensaje, en algún método, como puede ser POST, en el cual envía la información para el servidor.

Respuestas (response):



Un ejemplo de una respuesta:

Las respuestas están formadas por los siguientes campos:

La versión del protocolo HTTP que están usando.

Un código de estado, indicando si la petición ha sido exitosa, o no, y debido a que.

Un mensaje de estado, una breve descripción del código de estado.

Cabeceras HTTP, como las de las peticiones.

Opcionalmente, el recurso que se ha pedido.

Conclusión

El protocolo HTTP es un protocolo ampliable y fácil de usar. Su estructura cliente-servidor, junto con la capacidad para usar cabeceras, permite a este protocolo evolucionar con las nuevas y futuras aplicaciones en Internet.

Aunque la versión del protocolo HTTP/2 añade algo de complejidad, al utilizar un formato en binario, esto aumenta su rendimiento, y la estructura y semántica de los mensajes es la misma desde la versión HTTP/1.0. El flujo de comunicaciones en una sesión es sencillo y puede ser fácilmente estudiado e investigado con un simple monitor de mensajes HTTP.

Referencias:

Mozilla MDN - https://developer.mozilla.org/es/docs/Web/HTTP/Overview

6. WSGI

WSGI

WSGI son las siglas de Web Server Gateway Interface. Es una especificación que describe cómo se comunica un servidor web con una aplicación web, y cómo se pueden llegar a encadenar diferentes aplicaciones web para procesar una solicitud/petición (o *request*).

WSGI es un estándar Python que está descrito en detalle en la especificación PEP 3333.

¿Por qué se necesita WSGI?

Intentemos explicarlo con el siguiente ejemplo.

Web server

Imagina que eres un Web server y tu trabajo consiste en:

Estar sentado tranquilamente esperando a recibir una petición (*request*) de un amable cliente.

Cuando el cliente manda una petición, la recibes gustosamente.

Entonces, coges dicha petición y se la pasas a alguien llamado **Aplicación Python** (la web app) diciéndole: "*¡Ey! Despierta y mira esta petición que envía este cliente tan importante. ¡Por favor, haz algo con ella!*"

Unos ms después, obtienes una respuesta de **Aplicación Python** y devuelves la respuesta al cliente raudo y veloz.

Como ves, tu única misión en la vida es esa: servir a clientes. No sabes nada más del contenido de esas peticiones. Sin embargo, eres muy muy bueno en lo que haces; si en un momento determinado recibes millones de peticiones, eres capaz de duplicarte y escalar para resolver todas esas peticiones sin problemas.

Web App (Aplicación Python)

Ahora imagina que eres la **Aplicación Python**. En este caso, sólo existes cuando el web server (que siempre está activo) te despierta. Es decir, sólo existes en tiempo de ejecución. Tu trabajo sería algo como esto:

El **web server** te despierta y te manda la petición.

Coges la petición, la analizas y ejecutas ciertos comandos que tú sabes hacer muy bien.

Devuelves el resultado de esas ejecuciones al web server y te vuelves a dormir.

Finalmente, el web server entrega tus respuestas al ansioso cliente.

Vienen los problemas

De repente un día que había amanecido soleado, se nubla y llegan los problemas. Las conversaciones entre el web server y la "Aplicación Python" siempre han fluido perfectamente, porque son en Español:

"¡Ey! Despierta y mira esta petición que envía este cliente tan importante. ¡Por favor, haz algo con ella! "

Sin embargo, ese día te encuentras con:

"Эй! Проснись и посмотри на этот запрос, отправленный этим важным клиентом. Пожалуйста, сделай что-нибудь с ним!"

Y las cosas empiezan a ir fatal porque no hay forma de entenderse ¿ves por donde voy?

El web server se puede comunicar con la Aplicación Web de muchísimas formas distintas, así que ésta tiene que aprender cientos de idiomas diferentes para entender lo que le llegue del web server.

Antes de los WSGI, los desarrolladores de las "Aplicaciones Python", tenían que adaptar el código para adaptarse a los requerimientos del web server. Además, había que escribir diferentes códigos para cada clase de web server. Los desarrolladores están para escribir código, pero no para estar peleando con este tipo de cosas.

El super héroe WSGI ya está aquí

WSGI se pensó para solucionar este tipo de problemas. ¿Cómo solucionarlo? Fácil, igual que hacemos los humanos para comunicarnos. Si se definen **una serie de reglas**, todo queda claro y no encontraremos problemas.

Si quieres hablar con la "Aplicación Python", aprende a hablar con *estas palabras*; si algo va mal, comunícalo de *esta otra forma*. Y de la misma manera, el web server tendrá que aprender *esas palabras*, saber cómo comunicarse en caso de que *algo vaya mal*, etc.

El unicornio verde

Actualmente **Gunicorn** es el servidor HTTP WSGI más usado por los Pythonistas. El nombre viene de la abreviatura **G**reen **Unicorn**. Está escrito en Python y su uso es extremadamente simple.

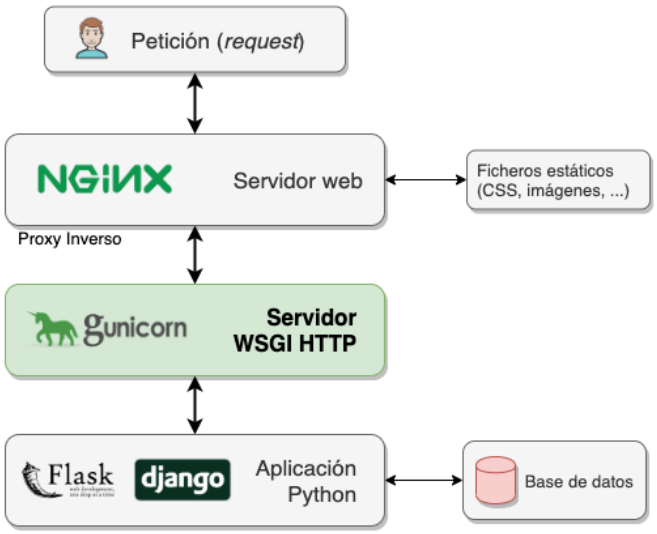
Uno de sus desarrolladores explica porqué se necesita usar "...algo como Gunicorn" en este post:

https://serverfault.com/questions/331256/why-do-i-need-nginx-and-something-like-gunicorn

A producción

Llegados a este punto, para poner tu aplicación Python Django/Flask/(*framework* que use WSGI) a dar servicio tienes varias formas de hacerlo. Si preguntas a amigos o lees por internet, encontrarás descripciones de los stacks que suele usar la gente. Te cuento una de esas formas.

Pese a que tanto Django como Flask tienen servidores web integrados que se suelen usar para pruebas, sólo se recomienda el uso de estos precisamente para eso: entornos de pruebas. Cuando queramos poner nuestra aplicación en producción, generalmente se usa Nginx para hacer de proxy inverso y Gunicorn como servidor WSGI. Gunicorn pude escalar con varios workers trabajando en paralelo y conectándose con nuestra aplicación Django/Flask. Quedaría así:



Por tanto, un WSGI nos ayuda a...

Darnos flexibilidad: los desarrolladores pueden cambiar entre Gunicorn, uWSGI, etc sin cambiar el código.

Escalabilidad: gracias al uso de Gunicorn podemos servir miles de peticiones resolviendo contenido dinámico.

Aprender más sobre WSGI

Si quieres aprender más sobre WSGI, te recomiendo que sigas los tutoriales y explicaciones en la página wsgi.org

Referencia: Alonso, Ignacio: https://medium.com/@nachoad/que-es-wsgi-be7359c6e001