Capa de Transporte

Transcribed on July 14, 2025 at 2:37 PM by Minutes AI

Speaker 1 (00:08)

Hola a todos y bienvenidos a esta sesión del módulo 3 donde vamos a hacer una introducción a la capa de transporte.

Soy Javier Álvarez y voy a ser vuestro profesor en esta sesión.

A lo largo de esta clase veremos en qué consiste la capa de transporte.

Vamos a ver la capa de transporte tanto en el modelo TCPIP como en el modelo OSI y posteriormente, para terminar, vamos a ver qué protocolos existen dentro de esta capa de transporte.

Comenzamos entonces hablando de que la capa de transporte es conocida como la responsable de la entrega proceso a proceso y estos procesos se comunican usando el paradigma cliente servidor.

Cada proceso en el servidor escucha en un determinado puerto y el cliente define su propio puerto elegido al azar para escuchar las respuestas temporalmente.

Como este puerto, cuando recibe las respuestas dejará de estar activo, se le denomina puerto efímero.

Antes hablábamos de las direcciones Mac en la capa de enlace y de las direcciones IP en la capa de red, Ahora hablamos en la capa de transporte de puertos.

Los puertos son los que identifican el proceso concreto en el host y para identificar un puerto se utilizan normalmente 16 bits o 2 bytes y se trata de un número que VA desde el 0 hasta el 65535.

Normalmente se clasifican en tres grandes rangos definidos por la llana.

El primer rango es el de puertos bien conocidos que van desde el 0 hasta el 1023.

Estos puertos son asignados y controlados por la llana.

Luego tenemos puertos registrados del 1024 al 49151.

Estos puertos no son asignados ni controlados por la llana, solo pueden ser registrados por llana para evitar la duplicación.

Y por último tenemos los puertos dinámicos o privados que van desde el 49152 al 65535.

Al igual que antes no están controlados ni registrados por la llana y se usan por cualquier proceso.

Estos son los puertos efímeros que comentábamos.

En cuanto al modelo TCP IP y al modelo OSI que vemos aquí a la izquierda y derecha respectivamente, vemos que la capa de transporte representa en ambos un estrato o nivel, a diferencia de, por ejemplo, la capa de aplicación que comentamos siempre.

Antes hablábamos de trama en la capa de enlace y de paquetes o datagramas en la capa de Internet, Ahora que pasamos a la capa de transporte, vamos a hablar de segmentos, segmentos que tienen información como el puerto de origen, destino, algunos flash, números de secuencia y otras opciones que veremos con más detalle en cada uno de los protocolos.

Y hablando de protocolos, pues deciros que en la capa de transporte encontramos una serie de ellos fundamentales para el correcto funcionamiento de las comunicaciones y entre los más importantes destacan el TCP y el UDP.

El TCP, Transmission Control Protocol es un protocolo orientado a conexión, confiable, proporciona entrega de datos sin errores, controla el flujo, controla congestión y es muy utilizado en aplicaciones como navegadores web, correo electrónico y transferencia de archivos.

Y justo en el lado opuesto tenemos AUDP, User Datagram Protocol, se trata de un protocolo que no se ha orientado a conexión y es menos confiable que TCP, pero a diferencia ofrece una entrega más rápida de datos, pero sí que no garantiza ni su entrega ni su orden.

Este protocolo se utiliza en aplicaciones donde la velocidad es prioritaria como videojuegos, streaming multimedia y servicios de transmisión por voz y vídeo en tiempo real.

En las próximas clases, como comentábamos, veremos con más detalles ambos protocolos.

Y con esto terminamos esta sesión que ha sido muy rápida.

A lo largo de esta clase veremos qué es el protocolo de control de transmisiones TCP, cómo es el formato de los segmentos TCP, para qué se utiliza y cómo funciona.

Comenzamos entonces hablando de qué es el protocolo de control de transmisiones, y es que este protocolo es un estándar que lo que nos permite es intercambiar datos de forma confiable entre dispositivos.

Ha sido diseñado para garantizar una transmisión segura y eficiente de información a través de redes, proporcionando una serie de características esenciales que vamos a aumentar a continuación.

Una de sus características principales es la orientación a conexión.

TCP antes de comenzar a transferir datos, lo que hace es un proceso de tres pasos llamado handshake o apretón de manos, donde se establece una comunicación entre el emisor y el receptor.

Además, también se dice que TCP es confiable, garantiza que los datos enviados se reciban correctamente y para ello utiliza técnicas como la confirmación de recepción mediante ACKs y la retransmisión de los datos para asegurar que no se pierda información durante la transmisión.

También gracias a TCP tenemos control de flujo y control de congestión, esto es, que TCP ajustará la velocidad de transmisión de datos para evitar que la red se sature.

Para ello utiliza algoritmos como el control de congestión de ventana deslizante.

Con este protocolo controlamos la cantidad de datos que un emisor puede enviar antes de recibir una confirmación del receptor.

Se dice también que TCP es fulltuplex.

Esto significa que permite la comunicación bidireccional de forma simultánea, lo que significa que ambos extremos pueden enviar y recibir datos al mismo tiempo.

Esto es importante porque mientras el emisor nos está enviando datos, también el receptor nos puede ir enviando annules para confirmarnos que ha recibido correctamente la información.

También tenemos comprobación de errores y recuperación.

Esto significa que TCP va a verificar constantemente si los datos han llegado correctamente al destino.

Si detecta errores o que ha habido un timeout o cualquier otro caso, lo que va a hacer es retransmitir esos datos perdidos para garantizar que la entrega ha sido precisa y completa.

Y por último tenemos segmentación y reensamblaje.

Al final, TCP va a dividir los datos en segmentos más pequeños antes de enviarlos a través de la red.

Esto lo que facilita es la transmisión y permite que los datos luego se reensamblen correctamente en el destino, incluso si se transmiten a través de redes con diferentes capacidades de tamaño del paquete.

Esto también es posible con TCP.

Vamos a ver ahora cómo es el formato de los segmentos.

TCP, como podéis ver, contiene los siguientes campos.

Comenzamos con el primero de ellos, que es el puerto origen.

El servidor va a tener que devolvernos información cuando haya recibido una petición nuestra para saber dónde nos tiene que enviar.

Este es el puerto que usará en las respuestas.

Después tenemos el puerto destino.

Por ejemplo, en aplicaciones web el puerto es el 80 o el 443, pero si estamos comunicándonos con un servidor FCP el puerto será normalmente el 21.

También tenemos el número de secuencia y aquí hay dos opciones.

Si el flash sync que veis abajo representado como sin, está activo, está 1, este campo va a tomarse como el número de secuencia inicial de la transmisión.

Sin embargo, si el Flashsync que comentábamos está a cero, está desactivado.

El número de secuencia representará los bytes de datos que se han enviado y se sumará el número de bytes que se envían en cada nueva secuencia.

Todo esto lo veremos con más tranquilidad y más detalle a continuación.

También tenemos el número de acknowledge el número de ACK.

Este campo está presente cuando el flag ACK lo está e indica el número del próximo byte que el emisor espera recibir del receptor.

También tenemos el Data Offset que nos indica el tamaño de la cabecera TCP en palabras de 32 bytes.

Luego tenemos un campo reservado que normalmente está a cero y que es para uso futuro.

Y Después contamos con 8 flags que comentábamos de un bit cada uno, que nos indican qué y que no está activado en cada momento.

No vamos a pararnos a ver cada uno de ellos, pero sí deciros que tenemos el flag ACK para confirmar las recepciones, tenemos el flash SYNC para empezar una conexión, el flash reset para cerrarla o el flag thin para terminar también una conexión de forma exitosa.

También cuando queremos enviar datos ya se ha realizado una conexión, pues vamos a usar el flag PSH o FLAG para enviar dichos datos.

A continuación tenemos el tamaño de la ventana deslizante.

Esta especifica el número de unidades que el receptor está dispuesto a recibir antes de enviar una confirmación.

Y por último tenemos el Urgent Pointer o puntero de urgencia.

Este campo se utilizará junto al indicador de urgencia, junto a ese flag que vemos de URG, y lo que nos indica es la posición del último byte de datos urgentes en el segmento.

Como campo adicional tenemos el campo Options, que es un campo variable y que nos proporciona información adicional o parámetros de configuración que pueden ser necesarios para una conexión TCP específica.

Vamos a ver algunos ejemplos de usos comunes de TCP y por qué se utilizan aquí y no se utiliza, por ejemplo, UDP.

El primero de ellos es la transferencia de archivos.

TCP se utiliza ampliamente para transferir archivos a través de la red, ya sea mediante protocolos como FTP, SFTP, Samba o simplemente a través de conexiones seguras HTTPs.

TCP también se utiliza en el correo electrónico, y es que los servidores de correo utilizan TCP tanto para enviar como para recibir dichos correos.

Esto es debido a que al final TCP nos permite que los correos se entreguen de una manera confiable y en el orden correcto, que al final es algo esencial para la comunicación.

Otro de los usos de TCP es la navegación web.

Cada vez que visitamos un sitio web en nuestro navegador, estamos utilizando por debajo TCP para establecer una conexión con el servidor web y descargar los datos necesarios para mostrar la página que estamos visitando.

También cuando, por ejemplo, nos conectamos mediante acceso remoto a una máquina o mediante una red privada virtual, una VPN.

TCP también se utiliza para proporcionarnos una conexión segura y confiable entre los dispositivos remotos y en general siempre queremos transferir datos sensibles y donde no haya una tolerancia a la pérdida de paquetes y por tanto a la pérdida de información, por ejemplo, transacciones financieras, escritura y lectura de datos en una base de datos, pues en estos casos se va a utilizar siempre TCP.

Vamos ahora a pasar a ver cómo funciona el protocolo TCP de forma técnica, es decir, cómo es el intercambio de paquetes entre el cliente y el servidor.

Este protocolo, este funcionamiento, se divide en tres pasos principales.

El primero de ellos es el establecimiento de una conexión, el segundo la transferencia de datos y el último el cierre de la conexión que hemos iniciado anteriormente.

Comenzamos con el establecimiento de conexión.

Normalmente este proceso se llama un proceso de tres vías y es llamado como un handshake o como un apretón de manos.

Este handshake se inicia por parte del cliente donde envía un segmento de tipo TCP con el flagsync activado.

El número de secuencia de este segmento TCP es el 8000.

Por qué el 8000 al final es un número de secuencia que se elige de forma totalmente aleatoria por el cliente o por el servidor en cada momento.

Por tanto aquí la hemos representado como el 8000, pero podría ser cualquier otro.

A este segmento TCP con el flashing activado responde el servidor con otro segmento TCP y en este caso el flagsync y ACK están activados.

El flagsync porque dice que sí que nos queremos conectar, que queremos sincronizarnos y realiza ese handsake y el ACK para responder que hemos recibido previamente su SYN.

El número de secuencia en este segmento también es elegido de forma totalmente aleatoria por el servidor, pero sí que el número de ACK es el número de secuencia del cliente que hemos recibido anteriormente, el 8000 incrementado en 1.

El servidor con esto lo que hace es confirmar la recepción del SYNC del cliente y le dice que ha recibido los bytes hasta el 8000 y que espera recibir a partir del 8001.

A continuación, una vez recibido ese sin ACK, finalmente el cliente responde con ACK.

Aquí el número de secuencia en este segmento es el número de ACK que recibió del servidor, es decir el 8001 y el número de ACK es el número de secuencia del servidor incrementado en 1 y fijaros que se incrementa en 1 aunque no se envíen ningún byte ni ningún contenido, debido a que las peticiones con el flagsync activado consumen un número de secuencia.

La petición es únicamente como ACK, por ejemplo, la última que ha enviado el cliente.

Esta petición no consume ningún número de secuencia y si el servidor comenzara ahora a enviar datos podría utilizar como el siguiente número de secuencia el 8001 y no el 8002.

Pues vamos a ello.

Entonces ya hemos establecido la conexión y vamos a pasar a enviar datos.

¿Y cómo los enviamos?

Con segmentos de tipo TCP con el flag push activado.

Fijaros que el primer segmento de tipo TCP tiene el número de secuencia 8001.

Es lo que comentábamos, como el último mensaje era de tipo ACK, este no incrementa el número de secuencia en ningún valor y por tanto es el mismo que el anterior.

En este primer segmento TCP el cliente está enviando datos desde el byte 8001 hasta el 9000 y posteriormente vuelve a enviar de nuevo datos desde el 9001 hasta el 10000.

De esta forma el número de secuencias ha incrementado en 1000 debido a que anteriormente habíamos enviado 1000 bytes de datos y el número ACK sigue completamente igual porque no hemos recibido ningún dato por parte del servidor.

El tercer mensaje en esta comunicación nos confirma que el servidor ha recibido correctamente hasta el byte 10000.

¿Y cómo vemos esto?

Porque este servidor nos ha enviado una CK con 10.001, es decir, he recibido hasta el 10.000, espero a partir del 2001 en adelante.

Además el servidor nos está enviando datos, fijaros que nos envía desde el BAEC hasta el 17.000.

Por tanto ahora tenemos que responderle que lo hemos recibido.

¿Y cuál va a ser nuestro número de secuencia?

El 10001, porque anteriormente en el último mensaje fue el 9001 y habíamos enviado 1000 bytes de datos.

Por tanto lo implementamos el 1000 y tenemos 10001.

¿Y el ACK cuál va a ser?

Pues acabamos de recibir datos por parte del servidor hasta el byte 17000, por tanto el ACK que esperamos es del 17001 en adelante.

Pues ya hemos transmitido los datos correctamente y lo que vamos a hacer ahora es cerrar la conexión.

Aquí el primer mensaje que envía por parte del cliente va a ser un segmento TCP con el flag pin activo.

El número de secuencia va a ser el 10001.

¿Por qué?

Porque el último mensaje que enviábamos llevaba este número de secuencia y como era de tipo ACK no ha consumido ningún número de secuencia más y podemos utilizar el mismo.

Y como tampoco hemos recibido ningún mensaje por parte del servidor, también tenemos el mismo número de ACK, el 17001.

El flag pin lo utilizamos con el objetivo de finalizar la comunicación entre el cliente y el servidor y este mensaje ahora mismo no lleva datos, pero podría llevarlos y sí que consume un número de secuencia, al igual que el flag SYNC.

Después de este segmento TCP, el servidor nos responde con un FLAG PIN y ACK, es decir, nos confirma que ha recibido nuestro flag PIN y además nos dice que también desea finalizar la comunicación.

Aquí el número de secuencia va a ser el 17001, es decir, el número que esperábamos y además nos dice que espera recibir el mensaje 10002, es decir, el 10001 1 de ese mensaje Fin.

Al igual que antes, este mensaje pues no lleva datos, podría llevarlos y sí que consume también un número de secuencia porque tenemos el flap PIN activado.

Por último el cliente ha recibido ese mensaje y le confirma al servidor que sí que lo ha recibido con un ACK.

Para ello indica el número de secuencia 10.002, es decir, 10.001 1, porque el flat fin consumía un dato, un número de secuenza y pone el ACK 17.002 del 17.001 del número de secuenCIA de ese fin ACK por parte del servidor 1 pone el número de ACK, pero realmente no espera recibir nada porque la comunicación ya se termina.

Y este último paquete, como comentábamos, pues este sí que no puede llevar datos y tampoco consume número en secuencia.

Y con esto terminamos esta clase donde hemos hecho un REP.

Hola a todos y bienvenidos a esta sesión del módulo 3 donde vamos a hablar del protocolo de datagramas de usuario UDP.

Soy Javier Álvarez y voy a ser vuestro profesor en esta sesión.

A lo largo de esta clase veremos qué es el User Datagram Protocol, el protocolo UDP, veremos qué estructura tiene el datagrama UDP, veremos también para qué se utiliza el protocolo UDP y por último veremos cómo funciona UDP.

Ya os adelanto que es bastante más sencillo que el protocolo TCP que hemos visto en la anterior clase.

Creo que la mejor forma de definir el protocolo UDP, protocolo que como sabéis es de la capa de transporte, es comparándolo de tú a tú con el protocolo TCP visto anteriormente.

De esta forma sabremos qué diferencias existen.

Comenzamos entonces esta comparación viendo el tipo de conexión.

En el caso de TCP es un protocolo orientado a conexión, significa que tenemos entonces ahí un proceso de handshaking, un establecimiento de conexión antes de transmitir datos.

En UDP todo lo contrario, es un protocolo sin conexión y significa que no tenemos ni requiere un hash ni establece una conexión antes de transmitir los datos.

Luego tenemos el apartado de fiabilidad.

TCP es fiable ya que gestiona la confirmación de mensajes, la retransmisión y los tiempos de espera.

UDP no es fiable, no gestiona nada de esto.

En cuanto al orden de los mensajes, TCP garantiza el orden y UDP no garantiza dicho orden.

Luego tenemos un apartado del peso del protocolo.

Podríamos decir que el TCP es un protocolo pesado porque requiere un proceso de configuración antes de las transmisiones, tiene control de flujo, control de congestión y todo esto hace que sea más pesado que por ejemplo el protocolo UDP que se considera ligero ya que no requiere ningún tipo de configuración previa.

En cuanto al apartado de transmisión de datos, tenemos que TCP transmite datos como un flujo de bytes sin indicación de los límites de los mensajes.

En UDP transmitimos los datos en paquetes individuales con límites de mensajes definido.

Luego tenemos el control de congestión o control de flujo y aquí TCP gestiona ese control gracias a las ventanas deslizante a ese protocolo que veíamos anteriormente.

En UDP no tenemos ningún mecanismo para controlar la congestión o el flujo de los mensajes.

Y por último TCP no soporta ni broadcast ni multicast, a diferencia de UDP que sí que lo soporta.

Y una vez hecha esa comparación, vamos a ver cómo es la cabecera del datagrama UDP.

Ya veis que es bastante sencilla, tenemos únicamente el puerto de origen, el puerto destino, la longitud de la cabecera y para terminar un checksum para comprobar que el mensaje se ha recibido igual que se ha enviado.

En cuanto al uso principal de UDP, se utiliza para protocolos como DHCP, DNS y otros protocolos en los que el intercambio de paquetes de la conexión desconexión son mayores o no son rentables con respecto a la información transmitida, es decir, si establecer la conexión, cerrar la conexión, controlar la congestión, el flujo y todo esto es mayor que al final la información que queremos enviar pues no merece la pena utilizar TCP, para ello utilizaremos directamente UDP.

Otro uso muy extendido de UDP es la transmisión de audio y vídeo en tiempo real.

Aquí no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tienen en estos casos, y es por ello que se utiliza UDP también en videojuegos online, pues al igual estamos transmitiendo voz y vídeo en tiempo real y por tanto tampoco nos podemos permitir retransmitir el mensaje que no le ha llegado a la persona.

Y en cuanto a cómo funciona UDP, los programas pueden enviar mensajes datagramas a través de la red sin necesidad de establecer una conexión previa, es decir, no necesitamos un handshake y los datagramas UDP se envían con direcciones IP de origen y destino, así como con los números de puerto para identificar tanto la aplicación que envía como la aplicación que lo recibe.

UDP es utilizado normalmente en aplicaciones donde, como comentábamos, la velocidad y la simplicidad son más importantes que la fiabilidad y control de flujo.

Como decíamos, para transmisión de vídeo y audio en tiempo real, juegos en línea, por ejemplo, o servicios de transmisión de datos donde la pérdida ocasional de paquetes no es crítica, pues debemos utilizar UDP.

Y con esto terminamos esta sesión donde hemos visto qué es UDP, las diferencias que contiene con respecto al protocolo TCP, cómo se utiliza y el formato de los datagramas UDP.

Sin más, me despido y nos vemos en la próxima sesión.