

La función de la capa de transporte es conectar procesos de la capa de aplicación. Tiene dos posibles modelos, acompañados del protocolo de internet (IP), UDP y TCP. Siendo UDP el menos confiable ya que trabaja bajo el mismo lema de best effort que IP, por lo tanto no tiene un sistema muy complejo de detección y control de errores, el cual TCP si tiene, además de control de flujo y de congestión.



La estructura de UDP consta de una cabecera y de los datos de la aplicación. Las cabeceras son:

**Número de puerto de origen y número de puerto de destino**: Los numeros de puerto permiten al host de destino pasar los datos de la aplicación al proceso apropiado que está ejecutandose en el sistema terminal de destino.

**Longitud**: especifica el numero de bytes del segmento (cabecera + datos)

**Suma de comprobación**: se utiliza para evaluar posibles errores del segmento.

Por otro lado, la estructura de un segmento TCP es mucho más compleja. El segmento TCP consta de campos de cabecera y un campo de datos. Al igual que con UDP, la cabecera incluye los números de puerto de origen y de destino, que se utilizan para multiplexar y demultiplexar los datos de y para las aplicaciones de la capa superior. También, al igual que UDP, la cabecera incluye un campo de suma de comprobación. La cabecera de un segmento TCP también contiene los siguientes campos:

**Número de secuencia** y **número de reconocimiento**: son utilizados por el emisor y el receptor de TCP para implementar un servicio de transferencia de datos fiable.

**Ventana de recepción**: se utiliza para el control de flujo.

**Longitud de cabecera**: especifica la longitud de la cabecera TCP en palabras de 32 bits.

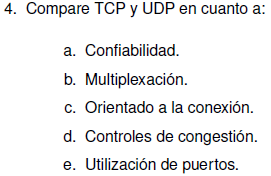
**Opciones**: es opcional y de longitud variable.

**Campos indicadores**: tiene 6 bits. Los indicadores son: ACK, RST, SYN, FIN, PSH y URG.

**Puntero de datos urgentes**



El objetivo de utilizar puertos en el modelo TCP/IP es diferenciar los distintos procesos dentro de la misma máquina.



|  |  |  |
| --- | --- | --- |
| **Característica** | **UDP** | **TCP** |
| Confiabilidad | No es fiable. | Es fiable ya que usa técnicas de control de flujo, números de secuencia, temporizadores y mensajes de reconocimiento. |
| No garantiza que los segmentos lleguen al proceso destino, tampoco que lleguen en orden o se conserve la integridad de los datos. | Garantiza que los datos transmitidos por el proceso emisor sean entregados al proceso receptor, correctamente y en orden. Continuará reenviando un segmento hasta que la recepción del mismo haya sido confirmada por el destino. |
| Multiplexación | Utiliza multiplexación y demultiplexación SIN conexión. | Utiliza multiplexación y demultiplexación orientada a la conexión. |
|  | Creamos sockets indicando el número de puerto. | El socket, en TCP se identifica por una tupla de cuatro elementos: Dirección IP Origen, Nro. Puerto Origen, Dirección IP Destino, Nro. Puerto Destino. |
|  | El segmento (de UDP + IP) identifica el socket destino mediante dos campos de cabecera: Dirección IP Destino y Nro. Puerto Destino. Ademas el segmento tiene una “dirección de retorno”. Por si el receptor desea devolver un segmento al emisor. | Dos segmentos TCP entrantes con direcciones IP de origen o números de puerto de origen diferentes (con la excepción de un segmento TCP que transporte la solicitud original de establecimiento de conexión) serán dirigidos a dos sockets distintos. |
|  | Cabeceras de 8 bytes. | Cabecera de 20 bytes (es variable). |
| Orientado a la conexión |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Controles de congestión |  |  |
|  |  |  |
|  |  |  |
| Utilización de puertos |  |  |



Los paquetes de la capa de transporte se definen como *segmentos* y los de capa de red *datagrama*. Sin embargo, en algunos RFC tambien se emplea la palabra *segmento* haciendo referencia al PDU de TCP y *datagrama* haciendo referencia al PDU de UDP.



Suponga que hay un proceso en ejecución en un host (cliente) que desea iniciar una conexión con otro proceso que se ejecuta en otro host (servidor). El proceso de aplicación cliente informa en primer lugar al cliente TCP que desea establecer una conexión con un proceso del servidor. A continuación, el protocolo TCP en el cliente establece una conexión TCP con el protocolo TCP en el servidor de la siguiente manera:

1- TCP del lado del cliente envia un segmento especial al TCP del lado del servidor. Este segmento no tendrá datos de la capa aplicación. La cabecera tendrá el **flag SYN** puesto en 1 y un número de secuencia inicial aleatorio *cliense\_nsi*. Este segmento se encapsula dentro de un datagrama IP y se envía al servidor.

2- Una vez que el datagrama IP llego al host servidor (suponiendo que llega), el servidor extrae dicho segmento SYN del datagrama, asigna los buffers y variables TCP a la conexión y envía un segmento de conexión concedida al cliente. Este segmento de conexión concedida tampoco contiene datos de la capa de aplicación. Sin embargo, contiene tres fragmentos de información importantes de la cabecera del segmento: el **bit SYN** se pone a 1, el campo reconocimiento de la cabecera del segmento TCP se hace igual al número de secuencia inicial del cliente *cliense\_nsi* + 1 y un número de secuencia inicial elegida por el servidor *servidor\_nsi*. Este segmento se lo conoce como **segmento SYNACK**.

3- Al recibir el segmento SYNACK, el cliente asigna buffers y variables, y envía otro segmento al servidor. Este segmento será la confirmación de la conexión. El bit de SYN se pone en 0, ya que la conexión está establecida. Esta tercera etapa del proceso de acuerdo en tres fases puede transportar datos del cliente al servidor dentro de la carga útil del segmento.

TODO: SEGUIR…







<https://phoenixnap.com/kb/ss-command>

<https://unix.stackexchange.com/questions/410949/understanding-ss-output>

**a. Para listar las comunicaciones TCP establecidas.**

ss -t o ss --tcp

**b. Para listar las comunicaciones UDP establecidas.**

ss --udp o ss -u

**c. Obtener sólo los servicios TCP que están esperando comunicaciones**

ss -l -t

**d. Obtener sólo los servicios UDP que están esperando comunicaciones.**

ss -l -u

**e. Repetir los anteriores para visualizar el proceso del sistema asociado a la conexión.**

ss <opción> | gep “pid” ¿?? DUDA

**f. Obtenga la misma información planteada en los items anteriores usando el comando netstat**

a. netstat -t -p TCP

b. netstat -u -p UDP

c. netstat -t -l -p TCP

d. netstat -u -l -p UDP



Si un host recibe un paquete TCP SYN cuyo puerto de destino es, por ejemplo, el número 80, y el host no está aceptando conexiones en dicho puerto (es decir, no está ejecutando un servidor web en el puerto 80). Entonces, el host enviará al origen un segmento especial de reinicio. Este segmento TCP tiene el bit indicador RST puesto en 1. Por lo tanto, cuando un host envía un segmento de reinicio, le está diciendo al emisor “No tengo un socket para ese segmento. Por favor, no reenvies el segmento.”

**a. Utilice hping3 para enviar paquetes TCP al puerto destino 22 de la máquina virtual con el flag SYN activado.**

*hping3 -p 22 -S localhost*: con p indico el puerto y con S indico que el flag SYN esta en 1.

**b. Utilice hping3 para enviar paquetes TCP al puerto destino 40 de la máquina virtual con el flag SYN activado.**

*hping3 -p 40 -S localhost*

**c. ¿Qué diferencias nota en las respuestas obtenidas en los dos casos anteriores? ¿Puede explicar a qué se debe? (Ayuda: utilice el comando ss visto anteriormente).**

*ss -t -n -a* (n de number para ver los números de puerto)

Hay un socket escuchando en el puerto 22 pero no en el 40.



**a. Utilice hping3 para enviar datagramas UDP al puerto destino 5353 de la máquina virtual.**

*hping3 -2 -p 5353 -S localhost*: con -2 indico el modo UDP.

*hping3 localhost --udp -p 5353*

No devuelve nada.

**b. Utilice hping3 para enviar datagramas UDP al puerto destino 40 de la máquina virtual.**

*hping3 localhost --udp -p 40*

Respuesta: ICMP Port Unreachable from ip=127.0.0.1 name=localhost

**c. ¿Qué diferencias nota en las respuestas obtenidas en los dos casos anteriores? ¿Puede explicar a qué se debe? (Ayuda: utilice el comando ss visto anteriormente).**

*ss -nau*

No puedo observar lo que devuelve el inciso a pero estimo que los datagramas se envian bien ya que hay un socket escuchando en el puerto 5353, no se recibe respuesta alguna debido a que UDP no manda segmentos ACK. En cambio como no hay socket escuchando en el puerto 40, se recibe el paquete ICMP.

Utilizando el comando ss podemos observar que hay un socket escuchando el puerto 5353 pero no hay uno escuchando el 40.



<https://users.cs.northwestern.edu/~agupta/cs340/project2/TCPIP_State_Transition_Diagram.pdf>



**a. En ambos equipos inspeccionar el estado de las conexiones y mantener abiertas ambas ventanas con el comando corriendo para poder visualizar los cambios a medida que se realiza el ejercicio. Ayuda: watch -n1 ’ss -nat’.**

**b. En Servidor, utilice la herramienta ncat para levantar un servicio que escuche en el puerto 8001/TCP. Utilice la opcion -k para que el servicio sea persistente. Verifique el estado de las conexiones.**

*ncat -l -k 8001*

**c. Desde CLIENTE1 conectarse a dicho servicio utilizando también la herramienta ncat. Inspeccione el estado de las conexiones.**

*ncat 10.0.0.20 8001*

**d. Iniciar otra conexión desde CLIENTE1 de la misma manera que la anterior y verificar el estado de las conexiones. ¿De qué manera puede identificar cada conexión?**

Podemos identificar las conexiones porque en el cliente están en puertos distintos.

**e. En base a lo observado en el item anterior, ¿es posible iniciar más de una conexión desde el cliente al servidor en el mismo puerto destino? ¿Por qué? ¿Cómo se garantiza que los datos de una conexión no se mezclarán con los de la otra?**

Es posible iniciar más de una conexión ya que desde el cliente las conexiones los puertos serán distintos, de esta manera no se mezclarán los datos de una conexión con los de la otra y los sockets serán diferentes.

**- Analice en el tráfico de red, los flags de los segmentos TCP que ocurren cuando:**

**i. Cierra la última conexión establecida desde CLIENTE1. Evalúe los estados de las conexiones en ambos equipos.**

**ii. Corta el servicio de ncat en el servidor (Ctrl+C). Evalúe los estados de las conexiones en ambos equipos.**

**iii. Cierra la conexión en el cliente. Evalúe nuevamente los estados de las conexiones.**