Redes de computadoras

Capa de transporte – TCP

Las diapositivas están basadas en en libro:

“Redes de Computadoras – Un enfoque descendente”
de James F. Kurose & Keith W. Ross

Transmission Control Protocol

Protocolo de la capa de transporte fiable y orientado a la conexión.

- Mecanismos de detección de errores
- Retransmisiones
- Reconocimientos acumulativos
- Temporizadores
- Números de secuencia y de reconocimiento

RFC’s [793, 1122, 1323, 2018, 2581]

Transmission Control Protocol

Orientado a la conexión:

Antes de que se intercambien los datos de la aplicación se realiza una comunicación donde se ajustan variables de estado.

La conexión TCP no es un circuito terminal a terminal con multiplexación TDM ni FDM.

El estado se mantiene en los hosts, los elementos intermedios (routers y switches) son inconscientes de las conexiones TCP.

Transmission Control Protocol

Una vez realizada la conexión los procesos de aplicación pueden enviarse datos mutuamente.

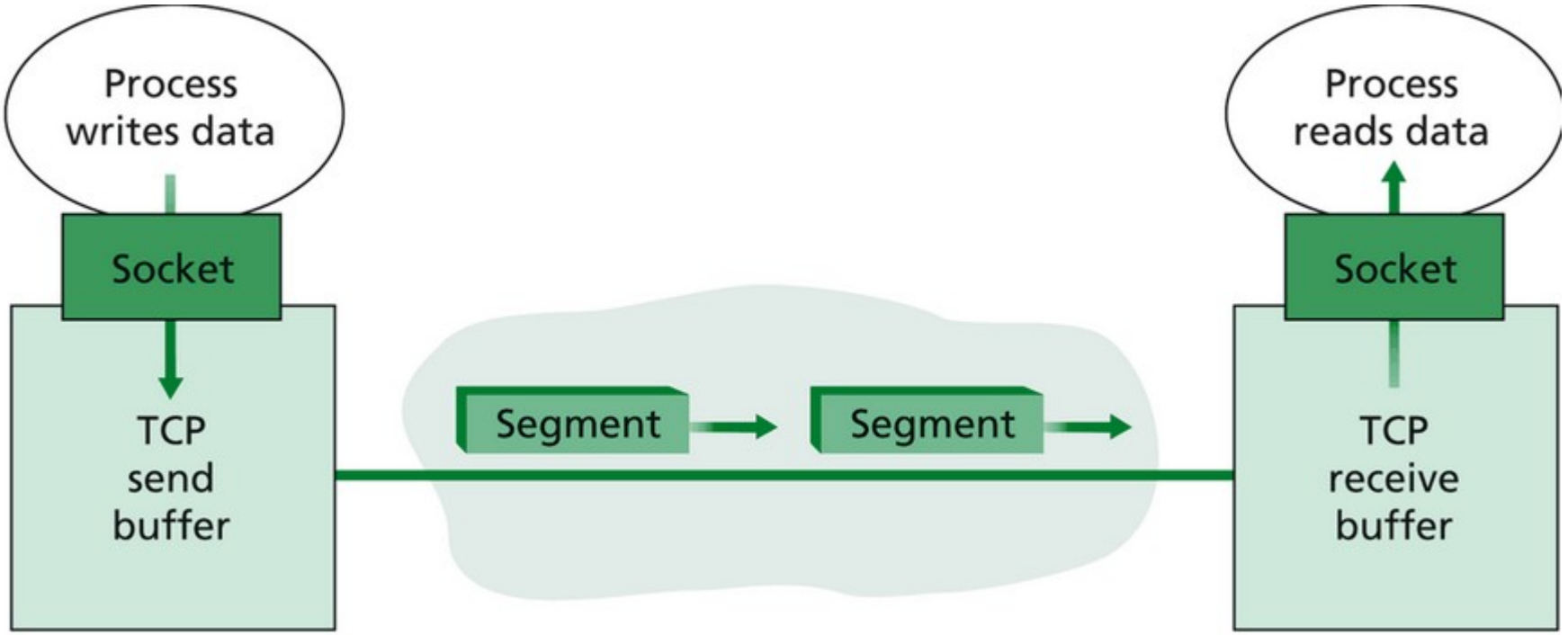
Los procesos pasan flujos de datos a través del socket y a partir de ahí se encargará el protocolo TCP.

TCP dirige los datos al **buffer de emisión** el cual se define al momento de la conexión.

De allí cada tanto tomará datos los cuales “envolverá” formando un segmento.

Transmission Control Protocol

TCP envía y recibe buffers



Transmission Control Protocol

Al establecerse la conexión se acuerda la únidad máxima de transmisión, **MTU** (Maximum Transmission Unit)

- Los datos pasan a través de múltiples routers y lo ideal es que cada segmento de datos pase por cada router sin ser fragmentado.

A partir de esta unidad se calcula el **MSS** (Maximum Segment Size) Que es el tamaño máximo de datos que pueden colocarse en un segmento.

MMS = MTU – cabecera TCP – cabecera IP

Three way handshake (Acuerdo en tres fases)

Un proceso en el host A desea iniciar una conexión con otro proceso que se ejecuta en el host B

- El proceso cliente se comunica con la capa de transporte para que envíe al host B un primer paquete con el campo **SYN**chronize.

- El host B recibe el paquete y responde con un paquete **SYN**chronize-**ACK**nowledgement el cual es recibido por el host A.

- Al recibir el SYN-ACK, A envía un **ACK**nowledge, el cual podrá contener carga útil y al ser recibido por **B** dejará establecida una conexión de sockets TCP.

Segmento TCP

- puerto origen y destino
- checksum
- numero de secuencia
- ventana de recepción
- longitud de cabecera (data offset)
- opciones
- indicador
  - URG
  - ACK
  - PSH
  - RST
  - SYN
  - FIN

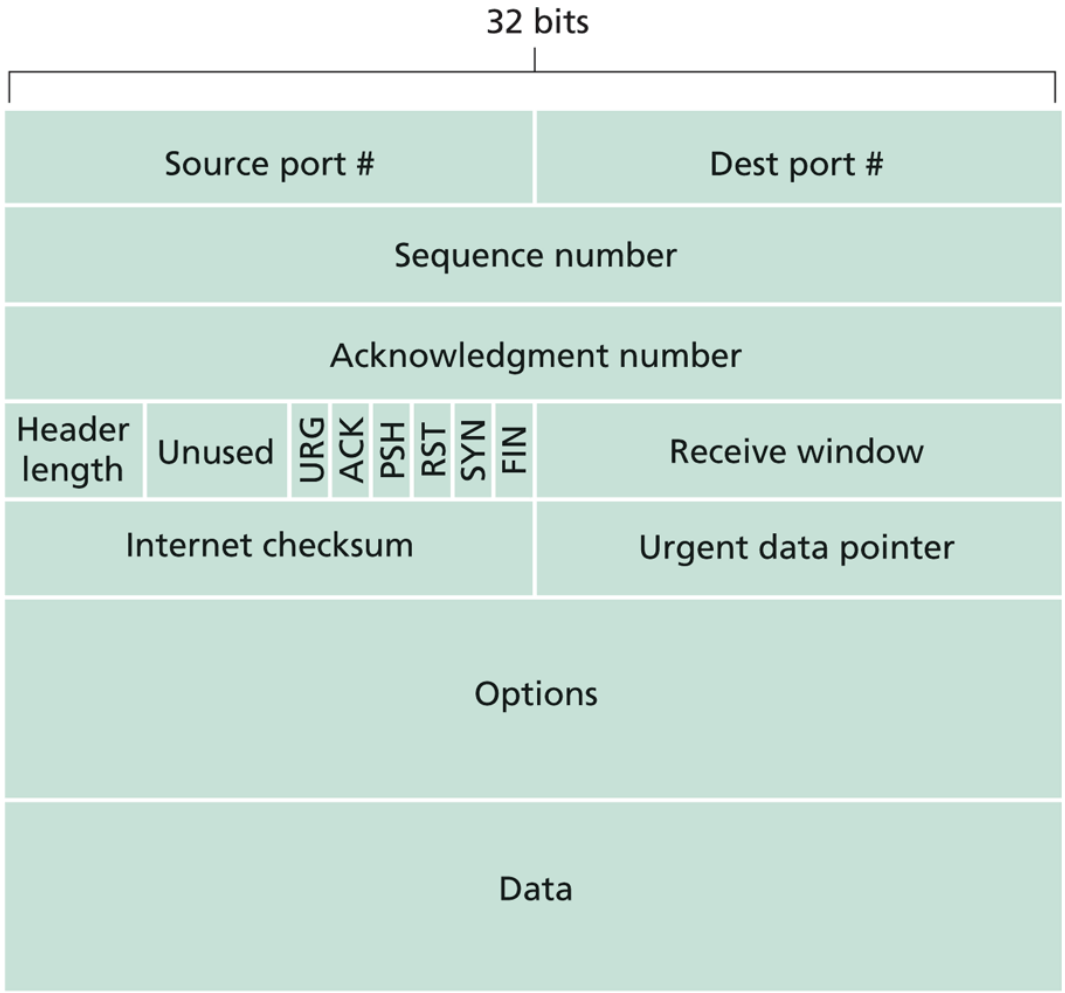


Figure 3.29 ♦ TCP segment structure

Segmento TCP

Número de secuencia

Número del primer byte del segmento dentro del flujo de bytes.

Si el host A envía vía TCP un archivo de 500.000 bytes, siendo el MSS bytes numerando el primer byte como 0.

TCP construirá 500 segmentos a partir del flujo de datos.

- primer segmento 0
- segundo segmento 1000
- tercer segmento 2000

Segmento TCP

Acknowledgement number

(Número de reconocimiento)

Número de secuencia del siguiente byte que el host espera recibir.

Host A recibe el segmento 0 a 999, responderá con un ACK en el que el número de reconocimiento será 1000.

En caso de que reciba el 2000 a continuación de todos modos enviará un ACK number 1000, dado que TCP maneja **reconocimientos acumulativos**.

¿Qué hace el host cuando recibe segmentos en forma desordenada?

Estimación RTT

Para recuperarse de la pérdida de segmentos se debe contar con un mecanismo de retransmisión, y para ello con un temporizador.

¿Que intervalo de tiempo utilizar en dicho temporizador?

Mayor al tiempo de ida y vuelta (RTT), ¿pero cuanto?

¿cómo estimar el RTT?

¿un temporizador por cada segmento?

Estimación RTT

Se genera un **RTTMuestra** a partir de un segmento transmitido.

Esta muestra cambiará cada RTT segundos, y no se calcula en para las retransmisiones.

En base a esta muestra se calcula un **RTTEstimado.**

El temporizador será mayor o igual que RTTEstimado,

pero no menor, ya que se generarían retransmisiones innecesarias

ni mucho mayor para poder retransmitir un segmento sin mayor demora.

Enlaces

TCP

http://www.inetdaemon.com/tutorials/internet/tcp/index.shtml