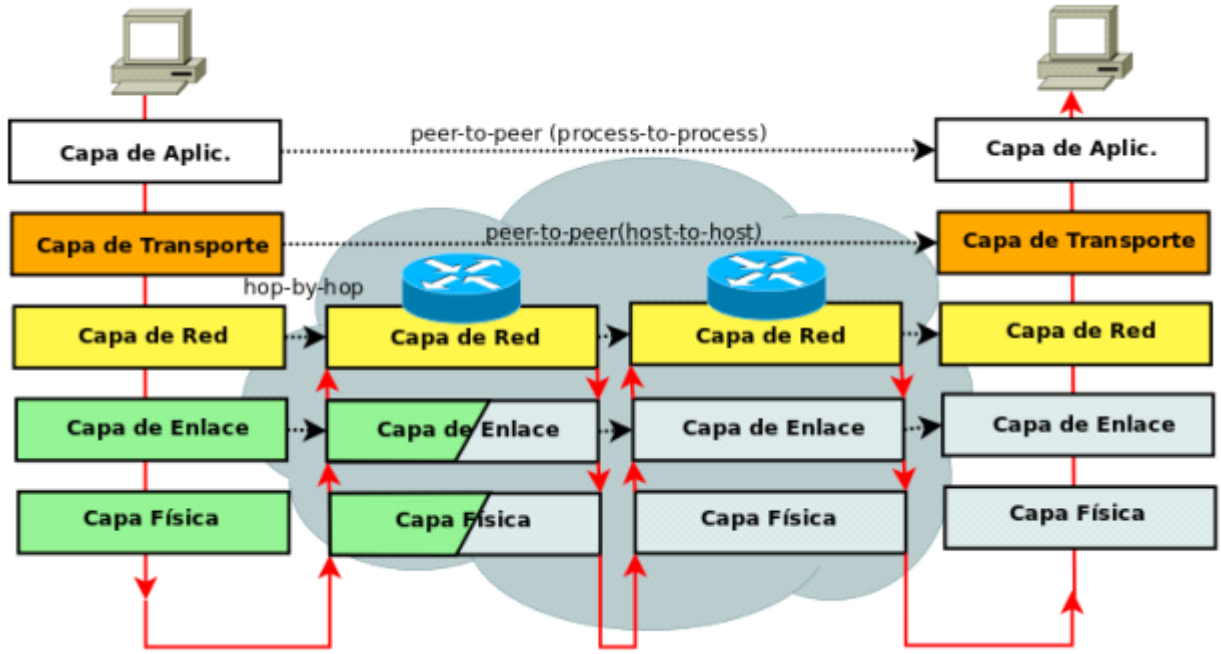
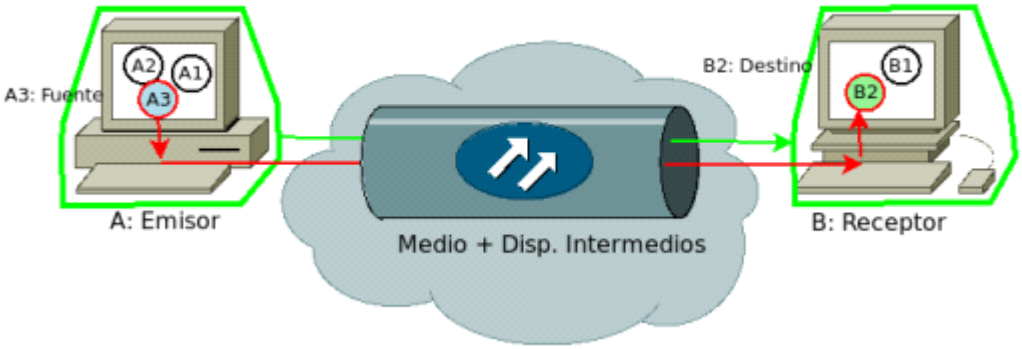
Clase 7- Intro tcp-udp

miércoles, 16 de octubre de 2024 09:52

Protocolo ip (capa de red)-> da servicios a capa de transporte
Ip es un servicio debil, no resuelve problemas pero es eficiente. Puede ser que los paquetes se desordenen, descarten, dupliquen, etc y no se ocupa.
IP usa protocolo hop to hop: de un host a otro, pasando por los intermediarios que va pasando la información.
A nivel de transporte la comunicación es end to end: de aplicación a aplicación. Comunica procesos



Funcionalidad capa de transporte:
Identificar cada aplicación. Hacia qué aplcacion de las que está corriendo el otro dispositivo va la data?



Cada aplicación usa un puerto distinto.
Ip maneja paquetes autocontenidos-> datagramas, necesito algo que me permita mandar datos de otros tamaños
Necesito protocolo que me ayude a ver errores.
Necesito ver cuándo me conviene mandar esos datos?

Modelo confiable TCP:
Las aplicaciones mediante la api network socket pueden pedirle o recibir cositas de la capa de transporte

Modelo no confiable UDP. Lo mínimo para poder usar ip:
User datagram protocol

- Menos overhead, best- effort. Pueden llegar los datos pero se pueden perder
- Mensajes—> datagramas son autocontenidos. No tienen un tamaño arbitrario
- Solo provee ultiplexación/ demultiplexación
- Tiene básicamente set reset. Mandá estos datos, recibí estos datos.

- Se usa en aplicaciones ágiles. Video, streaming voz

TCP transport control protocol:

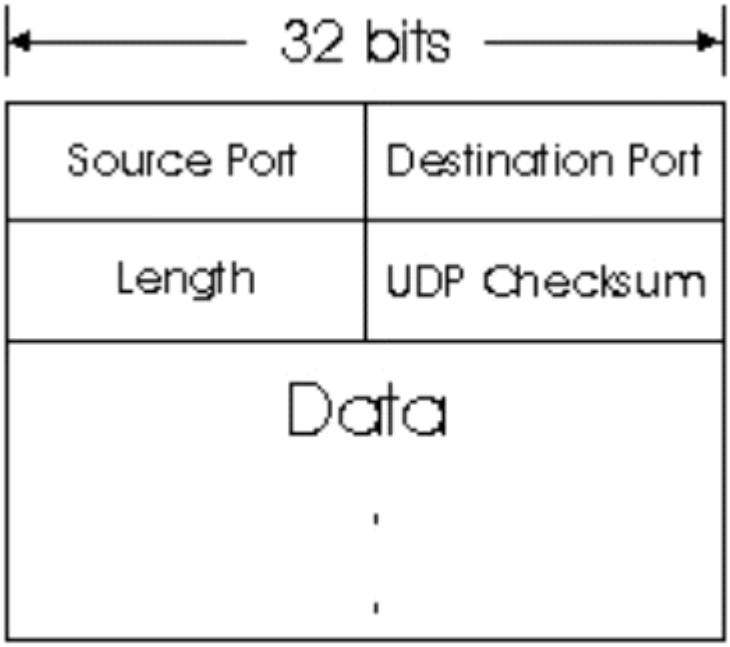
- Confiable, tiene control de flujo y congestión. Mejor uso de la red
- Orientado a streams. Puedo mandar tamaños de datos arbitrarios
- Cuando mando algo llega en secuencia ordenada
- Se mandan Segmentos. Se segmentan las secuencias de bytes.
- + overhead
- Hay que especificar de dónde a dónde se hace la comunicación

SCTP stream control transmission protocol

- Está en el medio de tcp y udp

Cada protocolo está identificado por un número de puerto.
Udp:17
Tcp:6

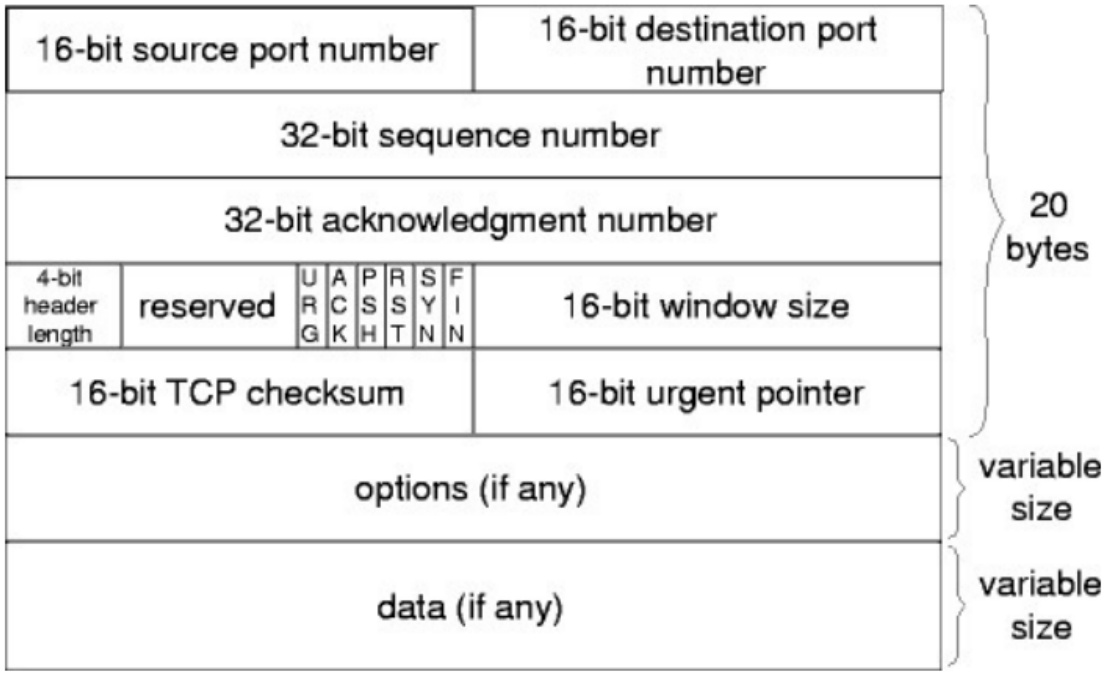
Mensajes UDP:



Origen/destino
Checksum para ver si se corrompieron datos
Lenght para ver la longitud
Datos

Checksum se calcula de una manera mambosa espero que esto no entre en nningún lado s

Mensajes TCP:



Tcp usa timers para saber si tiene que reenviar información

- Campos de Sesiones: Flags: SYN(Synchronize), FIN(Finish), RST(Reset).
- Campo de Detección de Errores: Checksum.
- Campos de Control de Errores: ACK, Num. Sec (\#Seq), Num. Ack(#Ack).
- Campo de Control de Flujo: a los de control de errores se agrega,Win.
- Campos de Control de Congestión: se agregan flags si participa la red.

En tcp hay parte variable, se pueden negociar

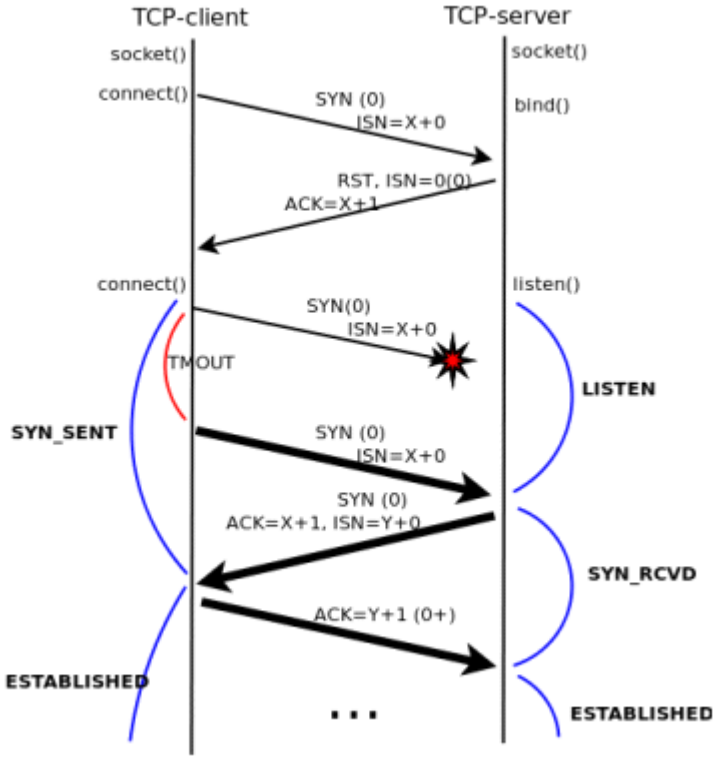
Asociación entre la entrega de datos a la aplicación y de la aplicación a tcp. Aplicación <> tcp

- TCP entrega y envía lo datos agrupados o separados de forma dis-asociada de la aplicación:
  - La aplicación puede enviar 300 bytes en un write y TCP lo podría enviar en 3 segmentos separados de 100 bytes c/u.
  - La aplicación puede enviar 100 bytes y luego otros 200 y TCP esperar para enviarlos todos juntos.
  - La aplicación puede intentar leer 200 bytes del buffer y TCP solo entregar 150 bytes y luego el resto.

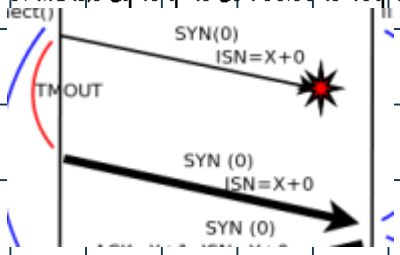
Establecimiento de conexión

Tienen que ponerse de acuerdo ambas partes en qué mandar y cómo hacerlo.

3 way handshake

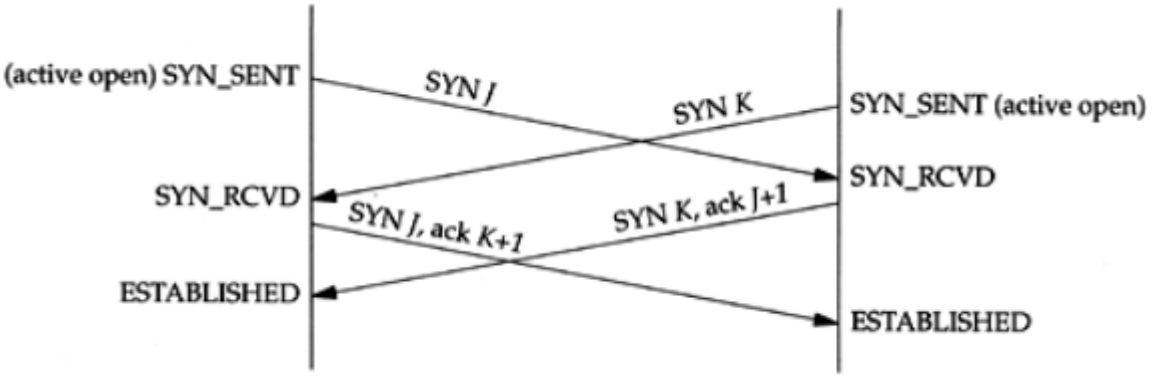


- Se inicia de forma activa la conexión:: llamada a la api socket CONNECT —> segmento con flag SYN (sync) prendido y numero de secuencia que determina tcp para el sistema de conexión. (Numeración
- Si del otro lado no hay nadie esperando la conexión, se manda RST y confirmación del sync.
- Si mando sync y no se recibe lo voy a volver a mandar después de un rato



Primero se manda syn y isn. Se recibe y se devuelve ack , syn y ack. Se manda ack para avisar que se estableció la conexión

También se puede dar Open simultáneo: ambos empiezan la conexión de forma activa



Envio de datos: una vez que se estableció la sesión

Envio en porciones (segmentos). Cada uno con headers y nro de secuencia

Cierre es 4 ways close