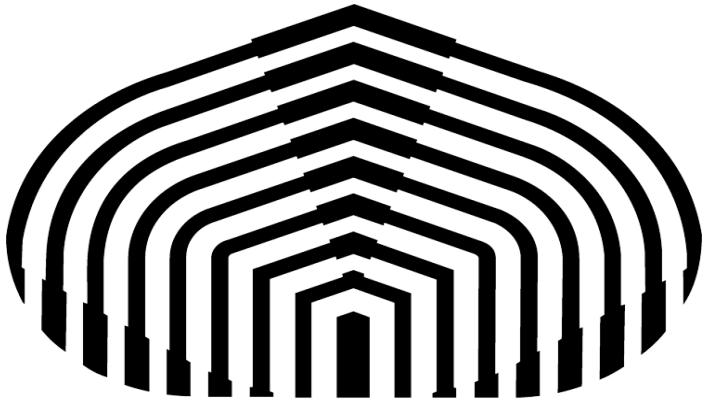
REDES DE COMPUTADORAS

EC5751



USB

Capa de Transporte

Prof: Mariano Arias

LA CAPA TRANSPORTE

Contenido

- Servicios de transporte
- Elementos de los protocolos de transporte
- Controles de Flujo/Congestión
- Protocolos de internet – UDP
- Protocolos de internet – TCP
- Redes tolerantes al retardo (Long Fat)
- GATEWAYS

LA CAPA TRANSPORTE

Es la responsable de entregar los datos a través de las redes con la confiabilidad y calidad requerida

Servicios ofrecidos a la capa superior:

- Primitivas de servicios de transporte
- Sockets de Berkeley

Ejemplo de socket: servidor de archivos

La capa de transporte añade confiabilidad a la capa de red

• Ofrece servicios sin conexión (UDP) y orientados a conexión (el mas famoso es TCP) a la capa de aplicaciones

LA CAPA TRANSPORTE

Primitivas

Las **Primitivas** de los servicios de transporte son aquellas que las aplicaciones pueden llamar para enviar data en un servicio simple orientado a conexión, hay dos tipos:

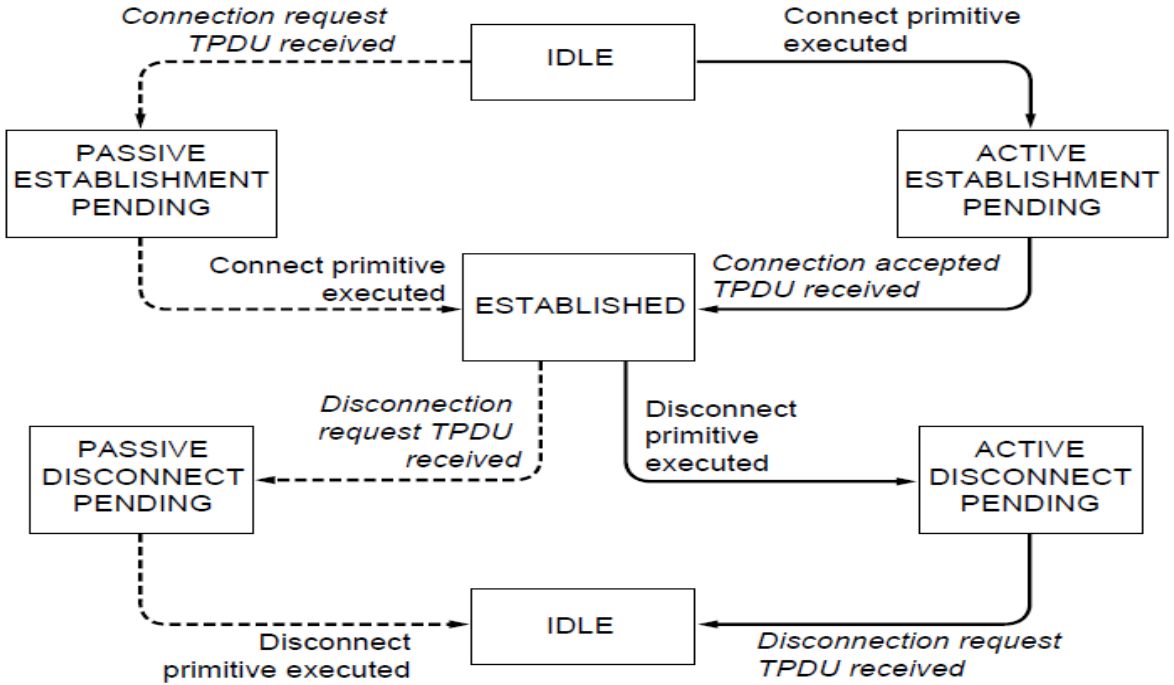
- Cliente: CONNECT, SEND, RECEIVE, DISCONNECT
- Servidor: LISTEN, RECEIVE, SEND, DISCONNECT

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

LA CAPA TRANSPORTE

Primitivas

Diagrama de estados para un servicio simple orientado a conexión



Las líneas sólidas (derecha) muestran
la secuencia de estados del cliente.

Las líneas discontinuas (izquierda) muestran la secuencia de estados del servidor.

En cursiva se muestran las transiciones debido a la llegada de segmentos

LA CAPA TRANSPORTE

Sockets de Berkeley

Son unas primitivas ampliamente usadas que comenzaron con el TCP en UNIX

- Los “sockets” son como puntos finales de transporte
- Conjunto de primitivas simples pero añadiendo SOCKET, BIND y ACCEPT.

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

LA CAPA TRANSPORTE

Ejemplo de socket – Servidor de archivos de internet (Cliente)

```
if (argc != 3) fatal("Usage: client server-name file-name");
h = gethostbyname(argv[1]);                    /* look up host's IP address */
if (!h) fatal("gethostbyname failed");

s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
if (s <0) fatal("socket");
memset(&channel, 0, sizeof(channel));
channel.sin_family= AF_INET;
memcpy(&channel.sin_addr.s_addr, h->h_addr, h->h_length);
channel.sin_port= htons(SERVER_PORT);

  c = connect(s, (struct sockaddr *) &channel, sizeof(channel));
  if (c < 0) fatal("connect failed");

  /* Connection is now established. Send file name including 0 byte at end. */
  write(s, argv[2], strlen(argv[2])+1);

  /* Go get the file and write it to standard output. */
  while (1) {
       bytes = read(s, buf, BUF_SIZE);          /* read from socket */
       if (bytes <= 0) exit(0);                 /* check for end of file */
       write(1, buf, bytes);                    /* write to standard output */
  }
}

fatal(char *string)
{
  printf("%s\n", string);
  exit(1);
}
```

Obtiene dirección IP del servidor

Crea un socket

Trata de conectarse

Escribe data (equivalente a enviar)

Lazo de lectura (equivalente a recibir) hasta que no queden datos

Salida implica cerrar la conexión

LA CAPA TRANSPORTE

Ejemplo de socket – Servidor de archivos de internet (Server)

```
/* Build address structure to bind to socket. */
memset(&channel, 0, sizeof(channel));       /* zero channel */
channel.sin_family = AF_INET;
channel.sin_addr.s_addr = htonl(INADDR_ANY);
channel.sin_port = htons(SERVER_PORT);

/* Passive open. Wait for connection. */
s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);    /* create socket */
if (s < 0) fatal("socket failed");
setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

b = bind(s, (struct sockaddr *) &channel, sizeof(channel));
if (b < 0) fatal("bind failed");

l = listen(s, QUEUE_SIZE);                  /* specify queue size */
if (l < 0) fatal("listen failed");
/* Socket is now set up and bound. Wait for connection and process it. */
while (1) {
    sa = accept(s, 0, 0);                   /* block for connection request */
    if (sa < 0) fatal("accept failed");

    read(sa, buf, BUF_SIZE);                /* read file name from socket */

    /* Get and return the file. */
    fd = open(buf, O_RDONLY);               /* open the file to be sent back */
    if (fd < 0) fatal("open failed");

    while (1) {
        bytes = read(fd, buf, BUF_SIZE); /* read from file */
        if (bytes <= 0) break;              /* check for end of file */
        write(sa, buf, bytes);              /* write bytes to socket */
```

Crea un socket

Asigna una dirección

Se prepara para direcciones

Bloquea esperando siguiente conexión

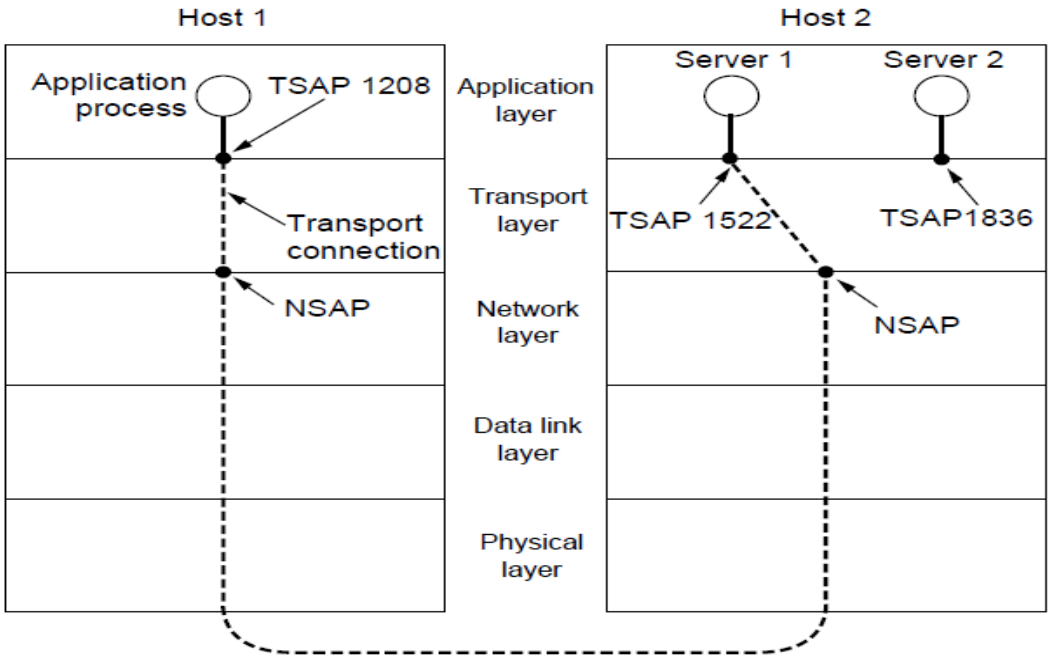
Lee el pedido

Almacena lo pedido en un archivo

LA CAPA TRANSPORTE

Elementos: Direccionamiento (TSAP o puertos)

- La capa de transporte añade TSAP (Transport Service Access Points
- Múltiples clientes y servidores pueden ejecutarse en una máquina con una sola dirección (IP)
- TSAPs son puertos para TCP/UDP
- Un NSAP (Network Service Access Point) pide acceso a los TSAP

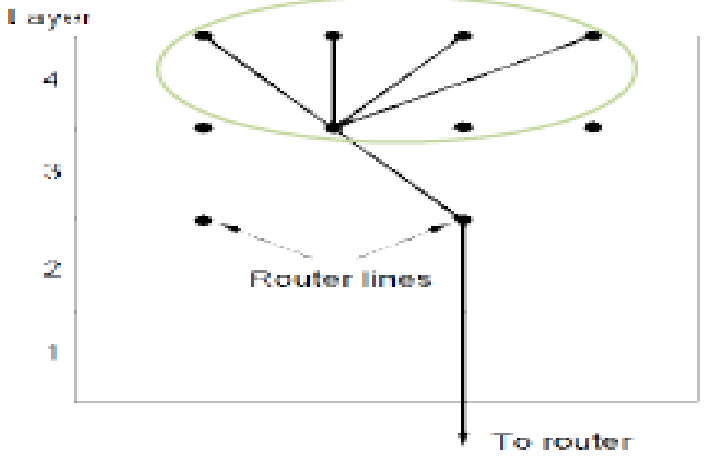


LA CAPA TRANSPORTE

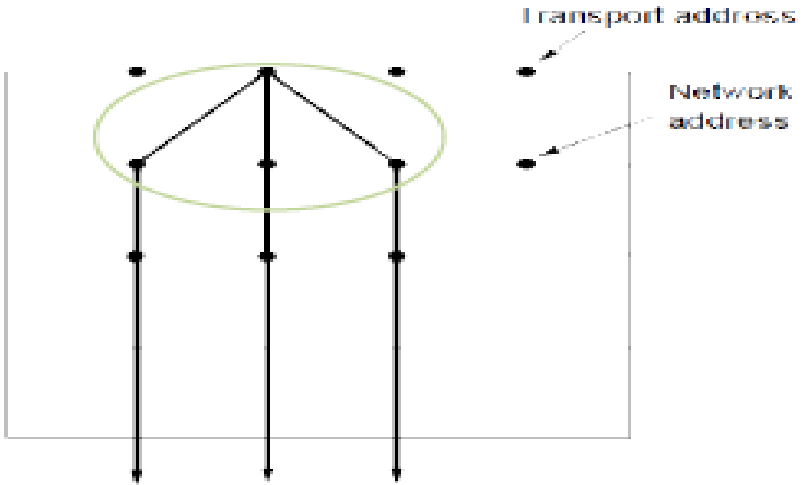
Elementos: Multiplexado

Tipos de transporte/red distribución que pueden existir:

- Multiplexado: conexiones comparten una dirección de red
- Multiplexado reverso: varias direcciones comparten una conexión



Multiplexado



Multiplexado reverso

LA CAPA TRANSPORTE

Elementos: Establecimiento de conexiones

• El problema **clave es asegurar la confiabilidad** aún cuando los paquetes puedan perderse, corromperse, retrasarse y duplicarse. (No confundir un paquete viejo o duplicado con uno nuevo)

- Usar ARQ y sumas de chequeos and checksums para detectar pérdidas o datos corruptos

Como:

• Utilizar Three-way handshake para establecer conexiones.

• Agrega números de secuencia a los TPDU (mensajes).

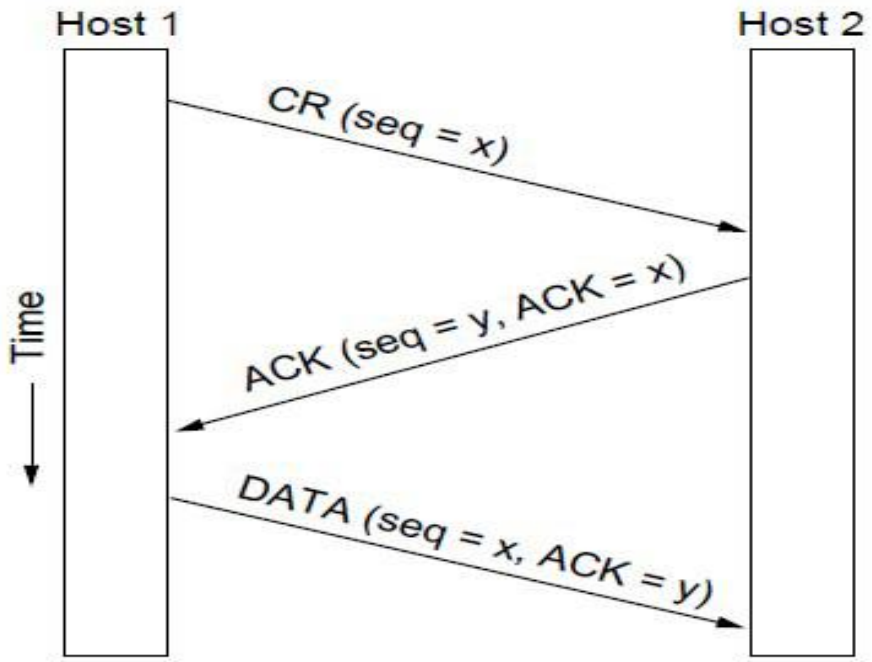
• Usar un espacio de números de secuencia lo suficientemente grande, para que no se encuentren dentro del mimo MSL (Maximum Segment Lifetime) de 2T=240 segundos y así no se encuentren.

LA CAPA TRANSPORTE

Elementos: Establecimiento de conexiones

Three-way handshake

- No hay estado anterior
- Ambos equipos contribuyen con un número de secuencia nuevo
- CR = Connect Request



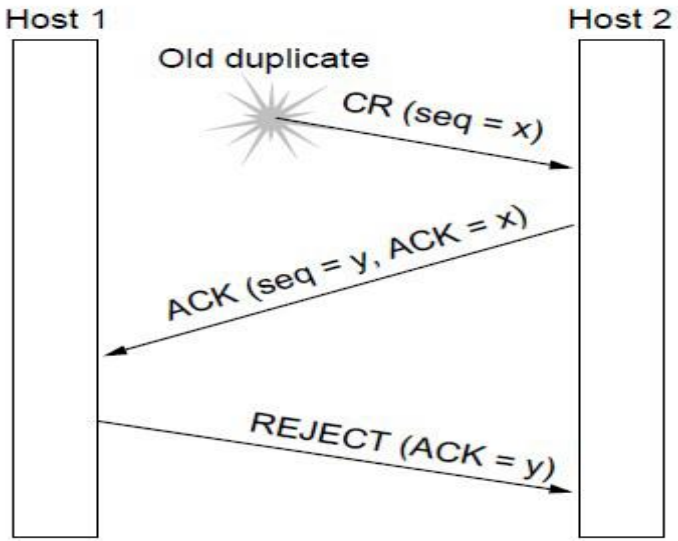
LA CAPA TRANSPORTE

Elementos: Establecimiento de conexiones

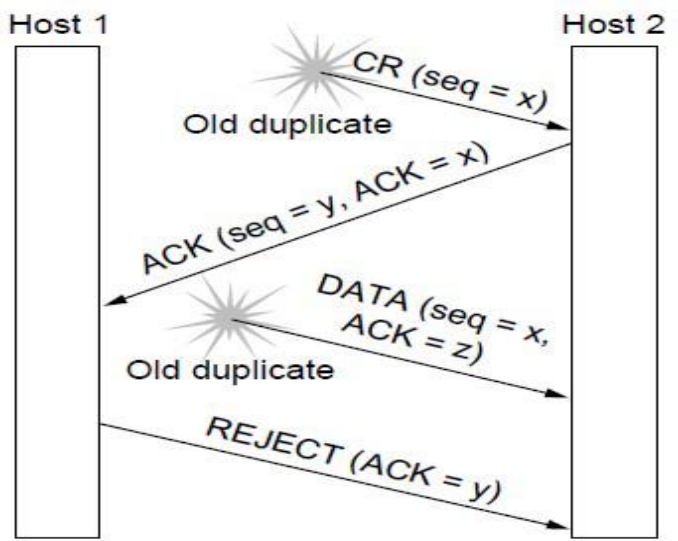
Three-way handshake protege ante casos extraños:

a) CR. engañosos ACK duplicados no conectan

b) CR y DATA duplicados.



a



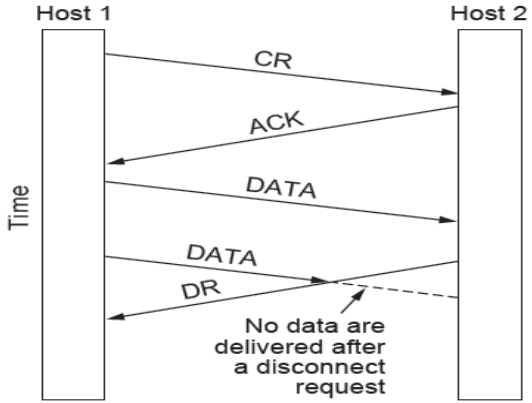
b

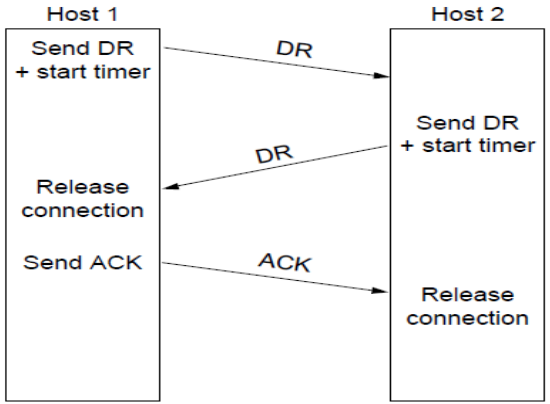
LA CAPA TRANSPORTE

Elementos: Liberación de conexiones

- El problema clave es tener confiabilidad al liberar la conexión
- Liberación asimétrica (cuando un lado termina la conexión) es abrupta y puede conducir a pérdida de datos
- Secuencia de liberación normal, iniciada por el usuario de transporte en Host 1

Ambos DRs son ACKed por la otra parte



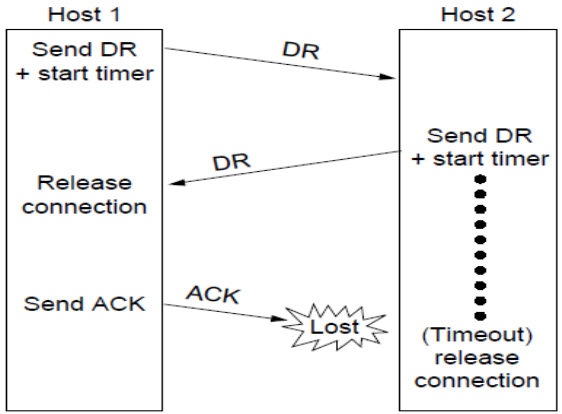


- DR = Disconnect Request

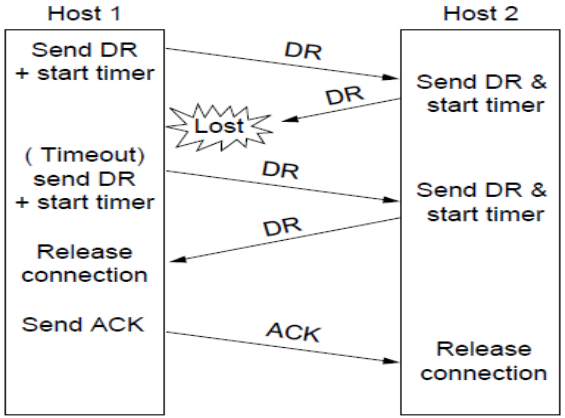
LA CAPA TRANSPORTE

Elementos: Liberación de conexiones

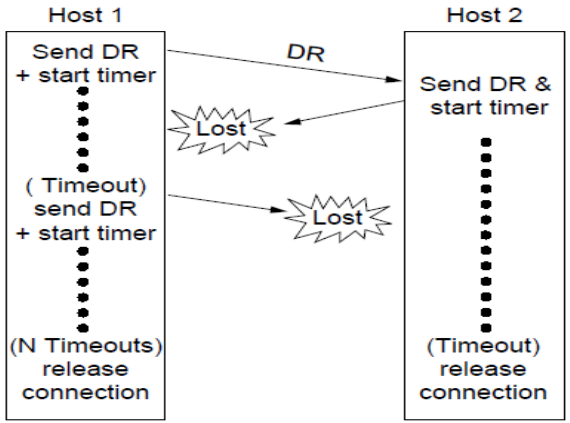
Ejemplos: Casos de error son manejados con temporizadores y retransmisión



ACK final perdido, Host 2 temporizador se acaba



DR perdido
Causa retransmisiones



Extremo: muchos DRs perdidos causan timeouts en ambos hosts

LA CAPA TRANSPORTE

Elementos: Control de errores y flujo

- La base para el control de errores es la utilizar la ventana deslizante (desde la capa de enlace) con sumas de chequeo y retransmisiones
- El control de flujo administra el encolado en emisor y receptor
- La cuestión es que los datos van a/desde la red a las aplicaciones en tiempos diferentes
- La ventana avisa a el emisor del tamaño de la cola del receptor
- Se usa una ventana deslizante de tamaño variable.

LA CAPA TRANSPORTE

Protocolos: Internet Control Message Protocol (ICMP-RFC792)

- Es uno de los principales mecanismos que permiten la correcta operación de la capa de Transporte y Red en Internet.
- Solo maneja información de control y errores.
- Usa UDP como transporte.
- PING y TRACEROUTE se envían sobre ICMP.

ICMP Type	ICMP Code	Description
0	0	Echo Reply (used by ping)
3	0	Destination Network Unreachable
3	1	Destination Host Unreachable
3	3	Destination Port Unreachable
8	0	Echo Request (used by ping)
11	0	TTL Expired (used by traceroute)

LA CAPA TRANSPORTE

Protocolos: UDP (User Datagram Protocol - RFC768)

CARACTERISTICAS

***Unreliable*:** Cuando se envía un mensaje no se valida si alcanza su destino.

No ordenado: Se desconoce el orden en que llegaran los paquetes.

Liviano: solo los campos origen, destino, longitud, data y validación de errores.

Datagramas: Cada paquete es individual, solo se verifica si esta bien en el destino, en caso de daño se descarta.

No tiene mecanismos de control de congestión

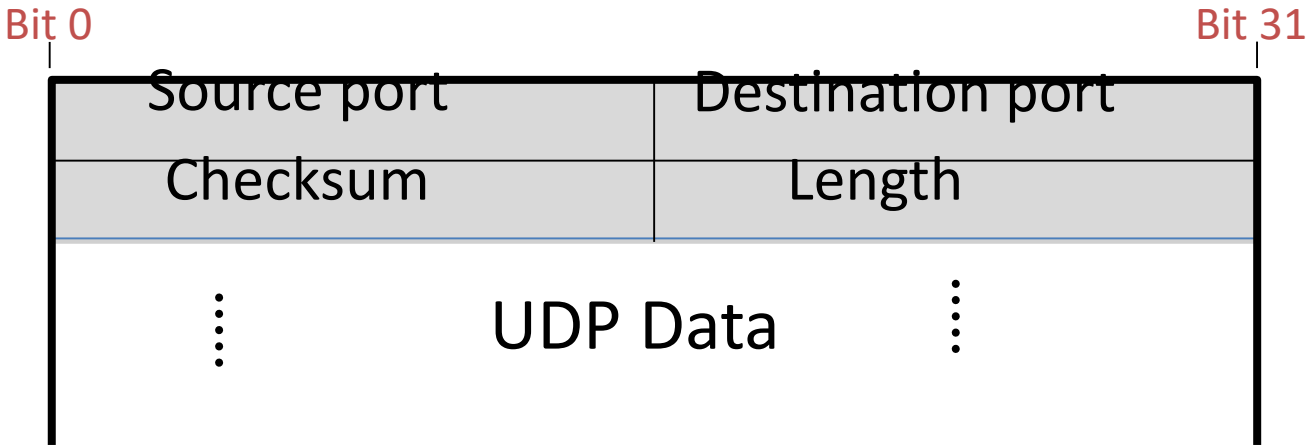
Broadcasts: por ser sin conexión se puede hacer broadcast asi todos los nodos de la red lo reciben.

Adaptable: fácilmente a IPv4 e IPv6, solo ampliando los campos.

LA CAPA TRANSPORTE

Protocolos: User Datagram Protocol (UDP)

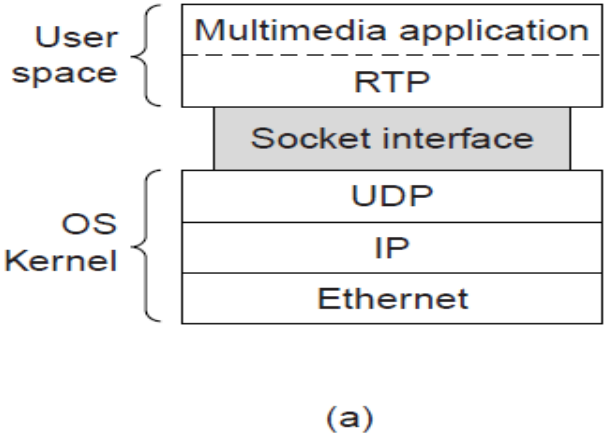
- Diseñado en los 80 por David P Reed
- Su principal uso es para transmitir aplicaciones que no requieren verificación o validan perdida de paquetes. Un ejemplo es voz y video sobre IP

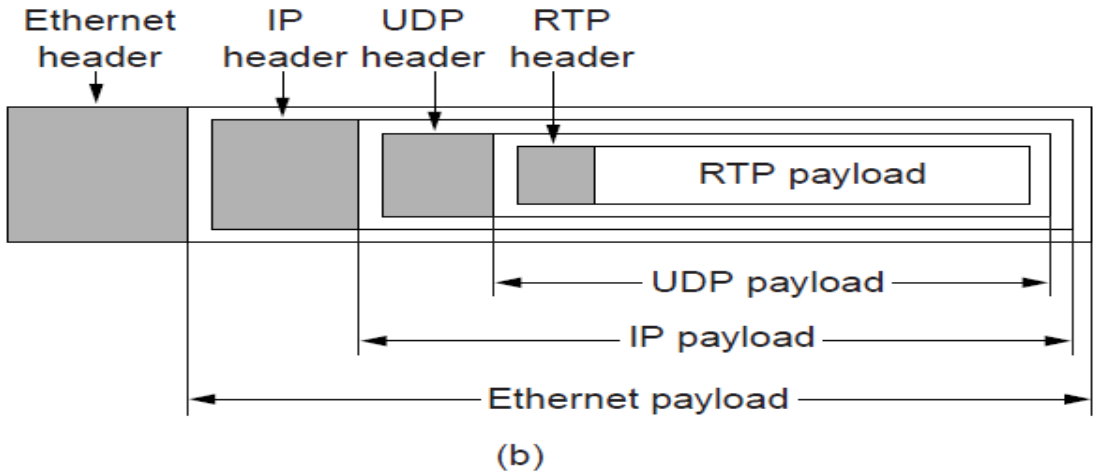


LA CAPA TRANSPORTE

Protocolos: Real-time Transport Protocol (RTP - 3550)

- Provee soporte para enviar datos en tiempo real sobre UDP
- A menudo es implementado como parte de la aplicación
- También es usado por protocolos como SONET

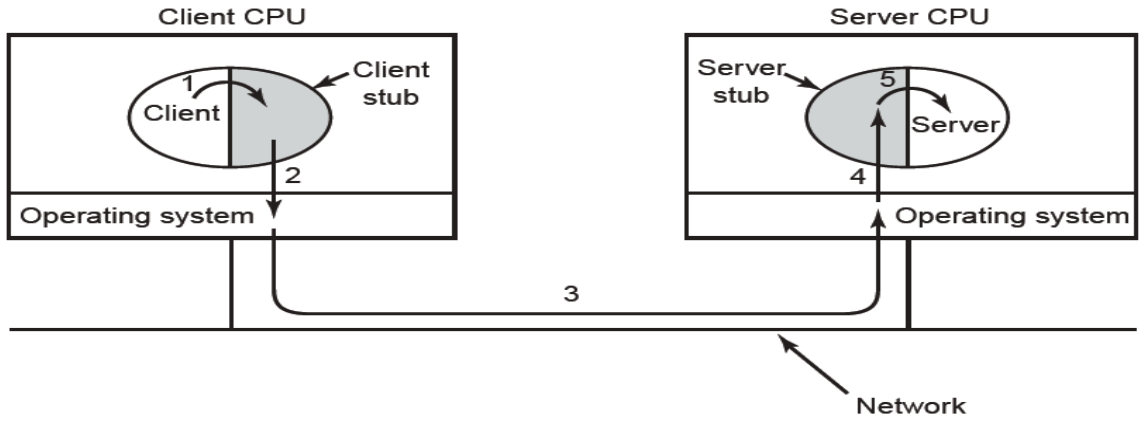




LA CAPA TRANSPORTE

Protocolos: Remote Procedure Call (RPC – RFC5531 y 7861)

- Es un protocolo que pueden usar los programas, para solicitar un servicio a otro programa ubicado en una computadora remota en la red, sin necesidad de entender detalles de la red. También se le conoce como llamado a subrutina.
- Se envía sobre UDP



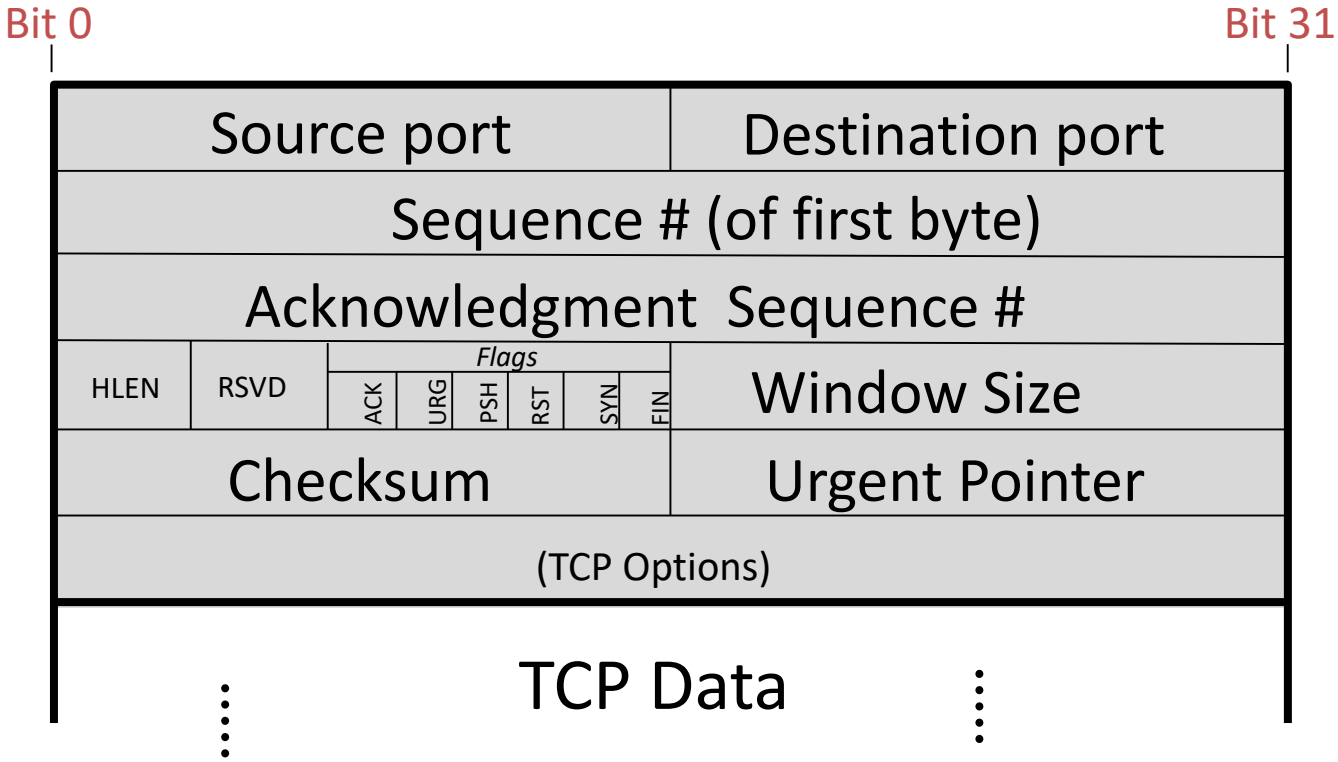
LA CAPA TRANSPORTE

Protocolos: Transmission Control Protocol (TCP – RFC793)

- Principal protocolo de la Internet. Se origino en conjunto con el protocolo de red IP por ello comúnmente se le conoce por TCP/IP.
- Provee a las aplicaciones un flujo confiable de bytes entre procesos, evitando la saturación de la red .
- Asegura que los datos emitidos sean recibidos sin errores y en el mismo orden que fueron emitidos.
- Es orientado a conexión, ambos extremos deben aceptar la conexión antes de comenzar a transmitir, luego deben finalizarla.
- Proporciona la vía para distinguir distintas aplicaciones (Multiplexar) en una misma máquina, a través del concepto de **Puerto (TSAP)**
- Segmenta los datos de forma adecuda para "entregarlos" al protocolo IP.
- Creado entre los años 1973 y 74 por Vint Cerf y Robert Kahn.

LA CAPA TRANSPORTE

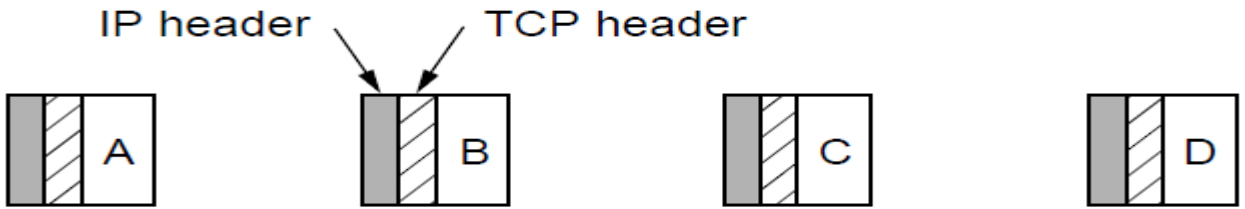
Protocolos: TCP Header



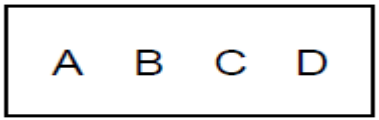
LA CAPA TRANSPORTE

Protocolos: TCP

Las aplicaciones que usan TCP ven únicamente flujos de bytes (a) y no los segmentos (b) como paquetes IP separados



a) Cuatro segmentos, cada uno con 512 bytes de datos y llevados en un paquete IP



b) 2048 bytes de datos entregados a la aplicación en un simple llamado de lectura

LA CAPA TRANSPORTE

Protocolos: TCP - Detalles de la Conexión

- TCP establece conexiones con un three-way handshake.
- Utiliza secuencia de paquetes.
- Maneja Control de Flujo en Ventana Deslizante.
- Mecanismos Control de Congestion (Slow start/Additive Increase/ Retransmisión de paquetes).
- Finaliza la conexión con 4 way Handshake.
- Maneja Puertos (TSAP) para el envió de información.

LA CAPA TRANSPORTE

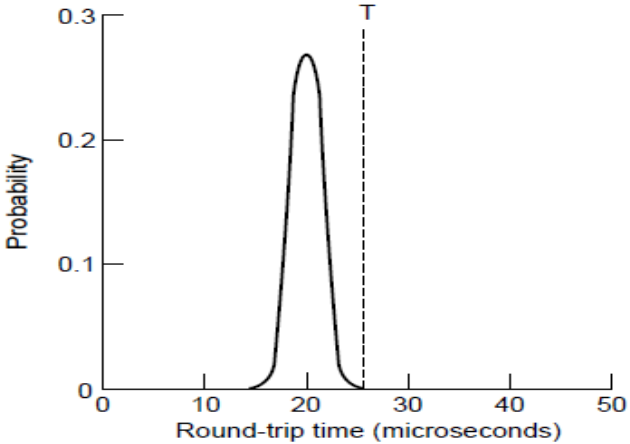
- Demo control de Flujo TCP

LA CAPA TRANSPORTE

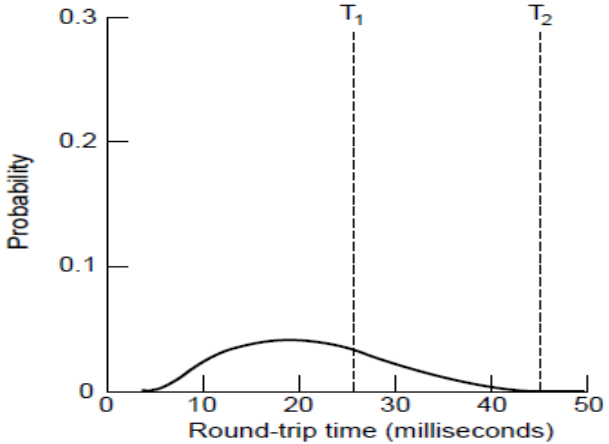
Protocolos: TCP – Manejo de temporizadores

Se estima el tiempo de retransmisión de los segmentos de RTTs

- Lleva registro de tanto el promedio como la varianza (para el caso de internet)
- El Timeout es ajustado a el promedio mas 4 veces la varianza



LAN



Internet

LA CAPA TRANSPORTE

Protocolos: TCP - Puertos

• Se usa el concepto de **número de puerto** para identificar a las aplicaciones emisoras y receptoras.

• Cada lado de la conexión TCP tiene asociado un número de puerto (de 16 bits sin signo, con lo que existen 65536 puertos posibles)

• Los puertos son clasificados en tres categorías:

- Bien conocidos: asignados por la *Internet Assigned Numbers Authority* (IANA), del 0 al 1023.
- Registrados: empleados por las aplicaciones de usuario de forma temporal, van del 1024 al 49151.
- Dinámicos/privados: también pueden ser usados por las aplicaciones de usuario, pero no tienen significado fuera de la conexión TCP, van del 49152 al 65535.

LA CAPA TRANSPORTE

Protocolos: TCP - Servicios

Algunos servicios populares se ejecutan en puertos bien conocidos

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

LA CAPA TRANSPORTE

Protocolos: TCP – Estrategias para mejorar el rendimiento

• Cargas inesperadas a menudo interaccionan con los protocolos para causar problemas de rendimiento.

•El software mal implementado (o mal diseñado) de un huésped puede producir la disminución del rendimiento de la red.

• Acelerar el caso común con un camino rápido.

• Los campos de las cabeceras son a menudo el mismo de un paquete a otro de un flujo; se pueden copiar para aumentar la velocidad de procesado.

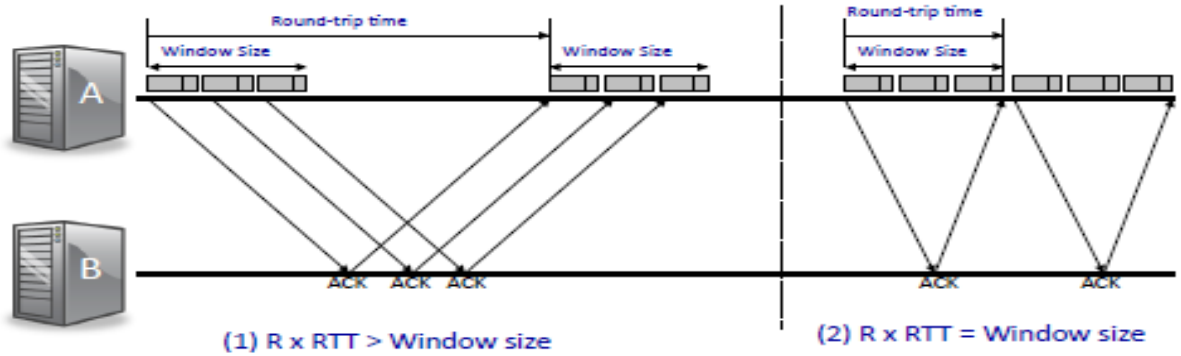
• Usa compresión de cabeceras, omitir campos que no cambian o lo hacen de manera predecible.

•La medición es la llave para entender los problemas de rendimiento.

LA CAPA TRANSPORTE

Protocolos: TCP – Control de Congestión (AIMD)

- TCP utiliza control de congestión desde el Host origen:
  - Evalúa eventos en el canal como: Perdida de paquetes y retardos.
  - Utiliza el mecanismo de ventanas deslizantes variando el tamaño de la ventana para garantizar que los TPDU atraviesen la red.
  - La variación se hace utilizando AIMD (Additive increse, Multiplicative decrease)



LA CAPA TRANSPORTE

Protocolos: TCP – Control de Congestión (AIMD)

- El tamaño de la ventana cambia entre:

$$\text{Window size} = \min\{\underbrace{\text{Advertised window}}_{\text{Receiver}}, \underbrace{\text{Congestion Window}}_{\text{Transmitter (“cwnd”)}}\}$$

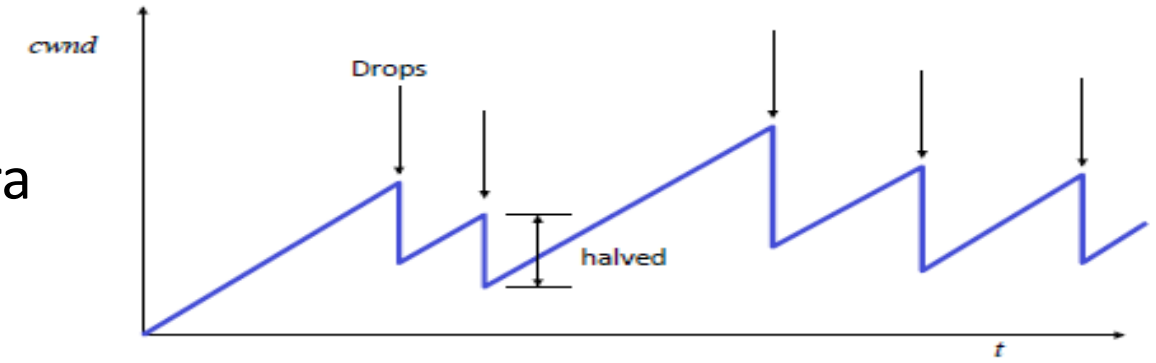
Si los TPDU llegan al destino la venta se incrementa según:

$$W \leftarrow W + \frac{1}{W}$$

Si los TPDU se pierden se decrementa según:

$$W \leftarrow \frac{W}{2}$$

Diente de sierra AIMD



LA CAPA TRANSPORTE

Protocolos: TCP – Control de Congestión (AIMD)

- Flujo Único en el router:
  - La rata de transmisión R es constante (**R = W/Rtt**).
  - El tamaño del Buffer debe ser **T=Rtt x C** (c: velocidad del link)
  - En operación estable hay un diente de sierra en cwnd

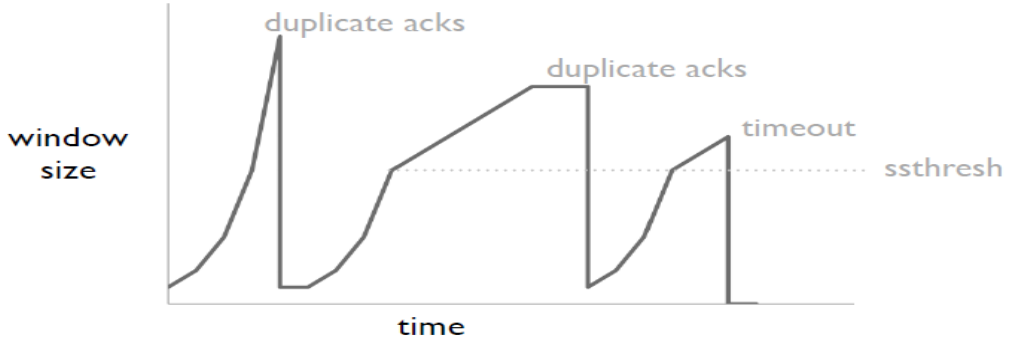
- Múltiples flujos en el router (mayoría de los casos en Internet)
  - Rtt es constante para todos los flujos
  - El tamaño del Buffer debe ser **T=Rtt x C**
  - Cada flujo tendrá una R distinta y variable
  - Cada flujo mantiene un diente de sierra.

$$R = \sqrt{\frac{3}{2}} \frac{1}{RTT\sqrt{p}}$$

LA CAPA TRANSPORTE

Protocolos: TCP – TAHOE

- **Slow Start:**
  - Inicia la ventana con el Maximun Segment Size (MSS)
  - Incrementa en forma exponencial (sube MSS cada vez, n x MSS)
  - Si hay un ack duplicado pasa a
- **Congestion Avoidance:**
  - Inicia como Slow Start pero al llega a la mitad Incrementa con AIMD
  - Con un doble ack repite el ciclo pero se mantienen en este estado.
  - Si hay timeout vuelve a Slow Star en el estado inicial
- Utiliza **“Fast Retransmit”**



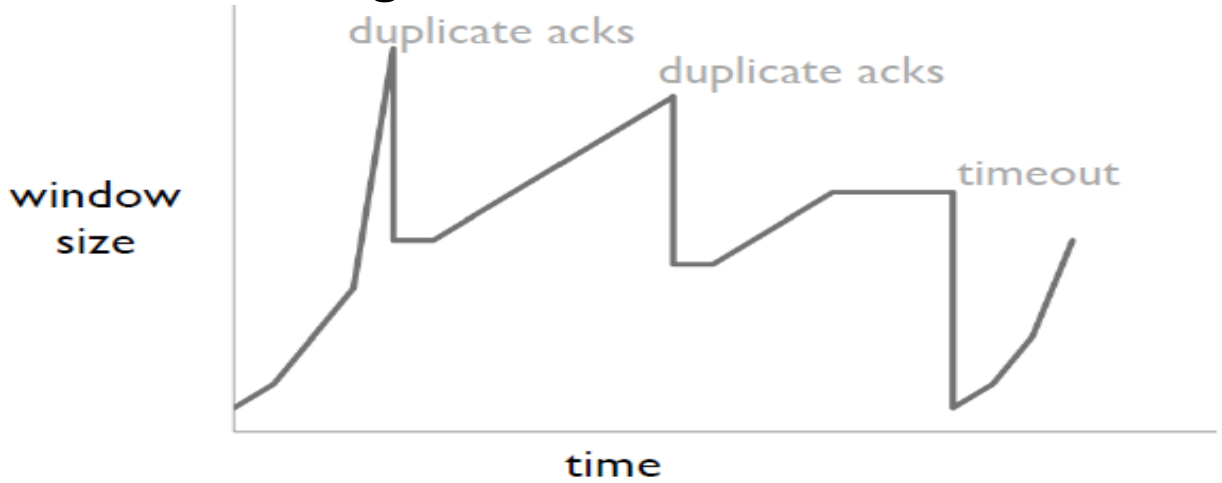
1987-8: Van Jacobson

RFC 5681

LA CAPA TRANSPORTE

Protocolos: TCP – RENO

- Al inicio se comporta como TAHOE (Slow Star)
  - Si hay doble ACK pasa a
- Fast Recovery:
  - Baja a la mitad del MSS y empieza a incrementar con AIMD
  - Solo en time OUT regresa a Slow Star.



1990

RFC 6582

DEMO

LA CAPA TRANSPORTE

Protocolos: Para redes “Long Fat”

• Redes con alto ancho de banda (“Fat”) y alto retardo (“Long”) pueden ”guardar” mucha información dentro de la red.

• Requieren protocolos con amplio encolamiento y pocos RTTs, en vez de reducir los bits en el cable.

Enviando 1 Mbit
San Diego - Boston

20ms después del comienzo

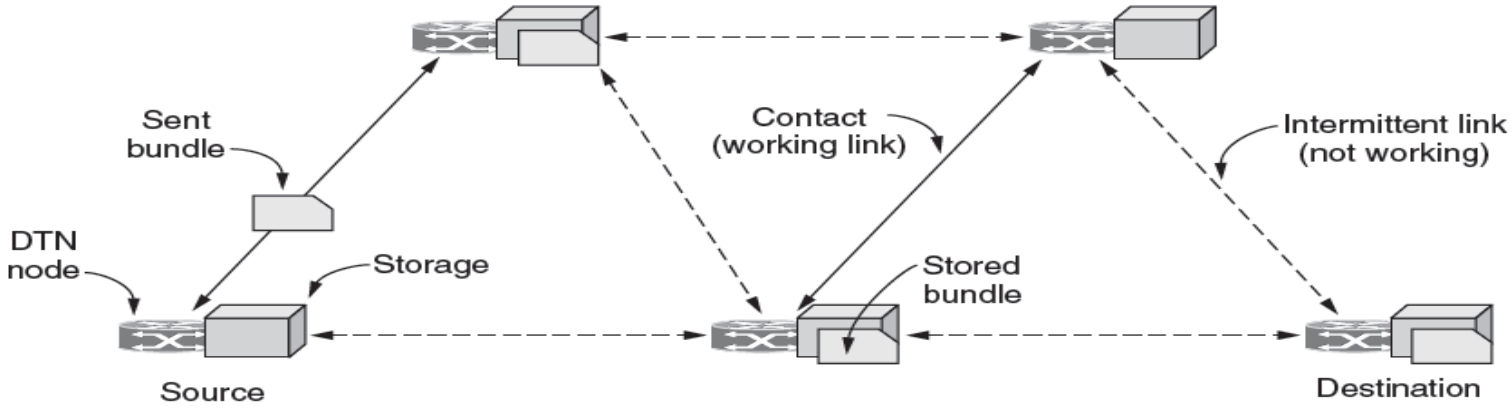
40ms después del comienzo

Puedes comprar mas ancho de banda pero no menor delay

LA CAPA TRANSPORTE

Protocolos: Para redes “Long Fat”

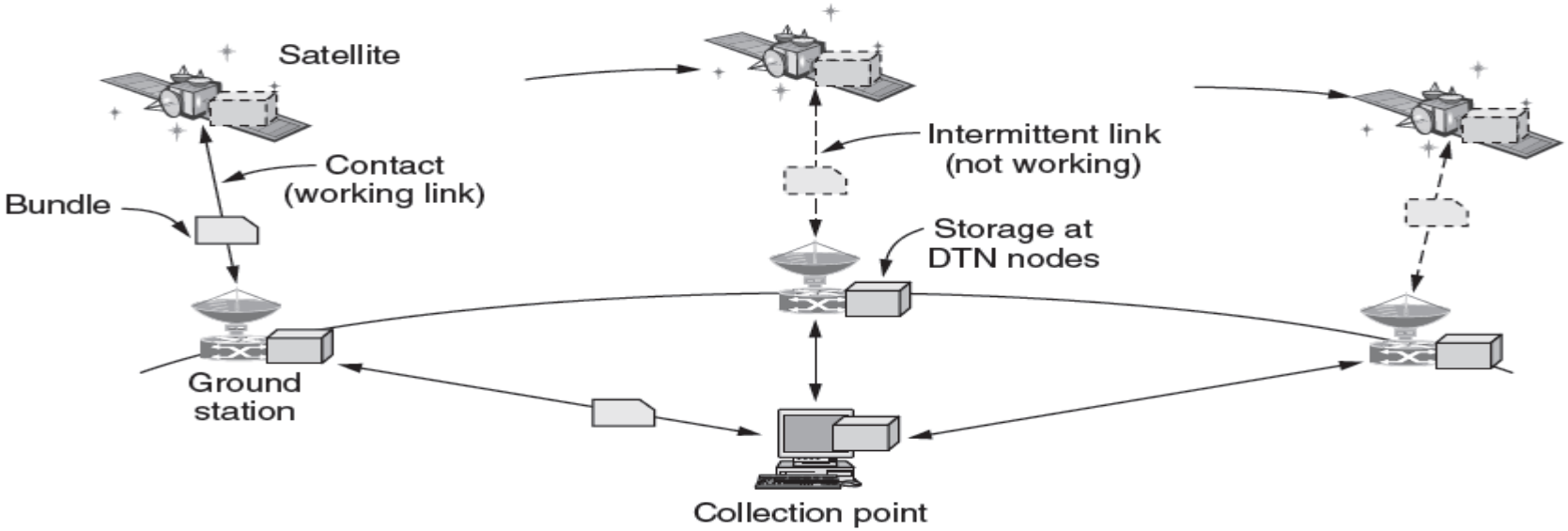
- DTNs (Delay Tolerant Networks RFC-5050) guardan mensajes (Bundles) en los nodos de red (DTN) hasta que pueden ser entregados
- Los Bundles pueden esperar **horas** en los enrutadores
- Puede no haber un camino de punto a punto en ningún momento



LA CAPA TRANSPORTE

Protocolos: Para redes “Long Fat”

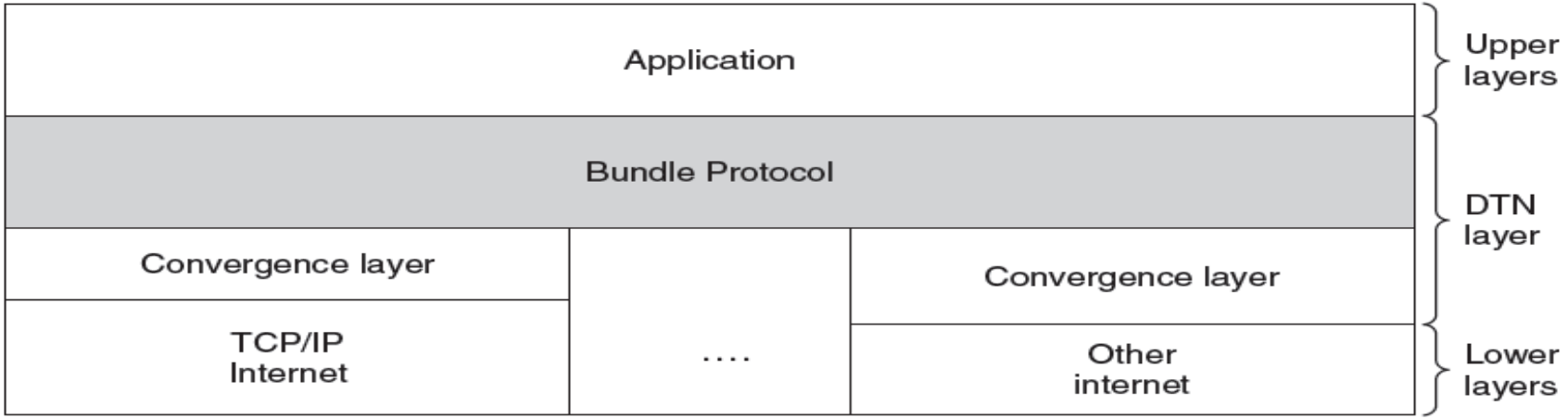
Ejemplo de red DTN: red de satélites a una serie de puntos



LA CAPA TRANSPORTE

Protocolos: Para redes “Long Fat”

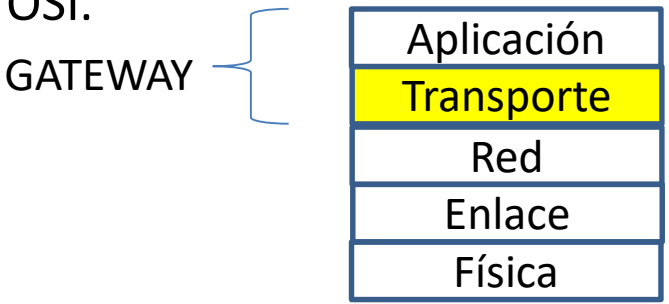
- El protocolo Bundle se coloca sobre la capa de TCP u otro transporte y provee servicio DTN a aplicaciones
- Dest./source añade direcciones de alto nivel (no puertos/IP)
- Custodia los cambios de transferencias de responsabilidad de entrega



LA CAPA TRANSPORTE

GATEWAYS (Compuerta o Pasarela)

- Un Gateway consiste en una computadora u otro dispositivo que actúa **como traductor** entre dos sistemas que no utilizan los mismos protocolos de comunicaciones, formatos de estructura de datos, lenguajes y/o arquitecturas.
- Un Gateway no es como un puente, que simplemente transfiere la información entre dos sistemas sin realizar conversión. Un Gateway modifica el empaquetamiento de la información o su sintaxis para acomodarse al sistema destino. Su trabajo está dirigido a los niveles más altos de la referencia OSI.



LA CAPA TRANSPORTE

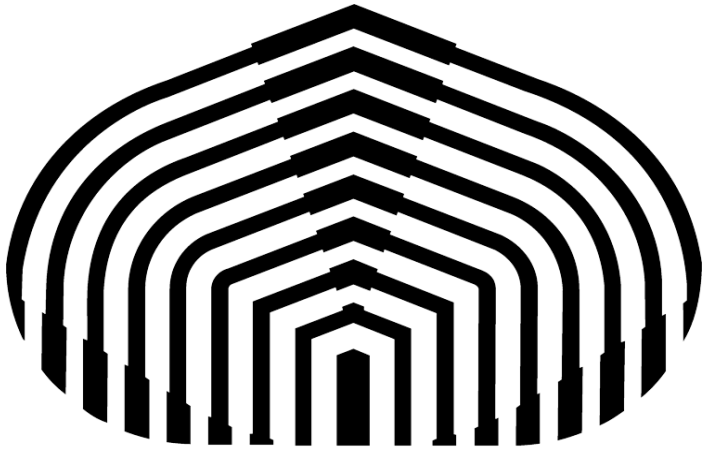
GATEWAYS (Compuerta o Pasarela)

- Son dispositivos y/o programas que permiten la comunicación entre redes que pueden ser diametralmente distintas. Por ello su ubicación será siempre en los extremos -entradas y salidas- de las redes.
- También se les denomina traductores de protocolos. Aunque la denominación más común en lengua castellana es la de “pasarelas”. Los protocolos con los que más frecuentemente trabajan son TCP/IP, IPX, ATM, Ethernet, RDSI, SONET, Token Ring, xDSL, y ARCNET.
- Para realizar esta traducción tienen duplicada la pila de protocolos OSI, de modo que estas dos pilas sean de los 2 protocolos a traducir, es valido aunque uno de los protocolos no sea conforme a la norma OSI. Esta traducción implica una ralentización en la transmisión de paquetes de una red a otra.

LA CAPA TRANSPORTE

GATEWAYS: Funcionalidades

- **Firewall:** como los gateway se colocan a la entrada de las redes, son un punto estratégico para situar un firewall, con el que controlar que no entren códigos nocivos en la red de destino.
- **Servidor proxy:** En este caso la función realizada es la de acceder a documentos externos a una red cuando alguno de los equipos que sí están dentro de esa red lo solicita. Pueden tener una cache en la que almacenar los documentos recientemente consultados, lo cual acelera los accesos repetidos a los mismos.
- **Servidores de dominios de nombre (DNS).**
- **VPN (Virtual Private Networks):** Por medio de una técnica llamada "tunneling", se simulan conexiones punto a punto en redes de difusión.
- **Correo electrónico (SNMP, POP3).**
- **Servidores de web (HTTP/1.1).**



USB