PROTOCOLOS DE COMUNICACIÓN DE RED

CLASE 5: CAPA DE TRANSPORTE

Lic. **Rubén G. Apolloni**

Área de Sistemas de Computación
Universidad Nac. De San Luis
(5700) San Luis – San Luis

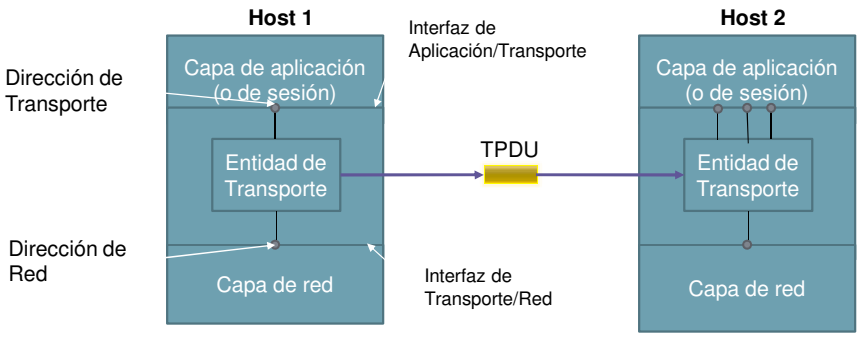
Objetivo

- Proporcionar transporte de datos confiable, eficiente y económico de la máquina origen a la máquina destino
- Independientemente de la red o redes físicas en uso.
- Utilizando los servicios proporcionados por la capa de red.
- Brindar servicio a los procesos de la capa de aplicación.
- Permitir a los programas de aplicación emplear un conjunto estándar de primitivas.
- Posibilitar que los programas funcionen en una amplia variedad de redes, sin preocuparse por las diferencias de las subredes y transmisiones no confiables.

Entidades de Transporte

- Software o Hardware de la capa de transporte encargado de realizar las tareas de la capa.
- Se puede implementar en el kernel del sistema operativo, en un proceso de aplicación o en un paquete de biblioteca.
- Unidad de Datos del Protocolo de Transporte (TPDU): mensajes enviados de una unidad de transporte a otra.

Entidades de Transporte



Servicios de Transporte

- **Servicio orientado a la conexión:** proporciona el establecimiento, mantenimiento y cierre de una conexión lógica entre aplicaciones de usuario.
- **Servicio no orientado a la conexión o servicio de datagramas**.
- Son muy similares a los servicios que brinda la capa de red.

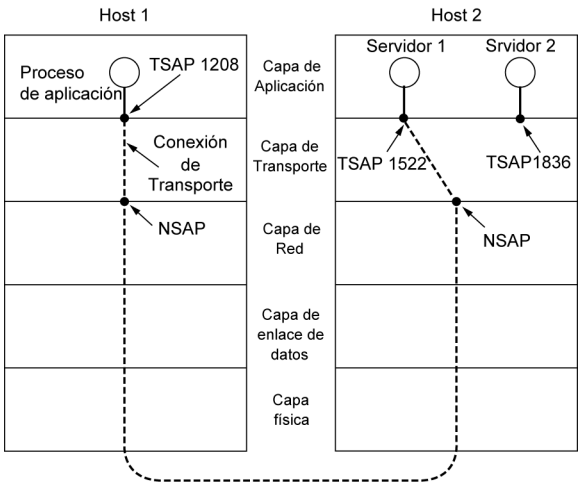
Servicios de Transporte

- Pero los códigos de transporte se ejecutan en la máquina del usuario.
- Entonces el usuario es el encargado de controlar y mejorar la calidad del servicio.
- La capa de transporte permite escribir programas de aplicación empleando un conjunto estándar de primitivas.
- La capa cumple la función de aislar a las capas superiores de la tecnología, el diseño y las imperfecciones de la subred.

Direccionamiento de la Capa de Transporte

- Los procesos de aplicación (clientes, servidores) se conectan a un punto de acceso para establecer una conexión.
- Cada proceso servidor define una dirección de transporte, desde la cual escucha las solicitudes de conexión.
- Cuando un proceso de aplicación debe conectarse con un proceso de aplicación remoto debe especificar a cuál se conectará.
- En Internet se denominan **puertos**.
- En general se denominan **Puntos de Acceso al Servicio de Transporte (TSAP)**.

Direccionamiento de la Capa de Transporte



Direccionamiento de la Capa de Transporte

- ¿Cómo conoce el cliente el TSAP del proceso servidor?
- Estáticos:
  - El TSAP se emplea por años, entonces los clientes lo conocen.
  - Se establece y se conoce de antemano.
- Dinámicos:
  - **Protocolo inicial de conexión:** el servidor tiene un servidor de proceso especial, el cual escucha todas las peticiones y lo deriva al proceso de servicio que corresponde.
  - **Servidor de nombre:** el cliente se conecta con el servidor enviando el nombres del servicio, al cual se desea conectar. Entonces el servidor le responde con el TSAP correspondiente al servicio.

Establecer una Conexión

- Objetivos:
  - Asegurar que cada extremo existe.
  - Negociación de parámetros.
  - Reserva de recursos de la entidad de transporte.

Establecer una Conexión

- Para establecer la conexión se envía un CONNECTION REQUEST al destino y se espera una respuesta CONNECTION ACCEPTED.
- Pero la red puede perder, almacenar o duplicar paquetes.
- Cuando la red esta congestionada las confirmaciones pueden no llegar a tiempo.
- El principal problema es la posible existencia de duplicados retrasado.

Establecer una Conexión

- Direcciones de transporte desechable: Cada vez que se requiere una nueva conexión se asigna una dirección diferente.
- Asignar a cada conexión un identificador de conexión: número secuencial, incrementado con cada conexión.

- Ambos esquemas requieren almacenar historia de las conexiones.
- Si se cae se pierde la historia.

Establecer una Conexión

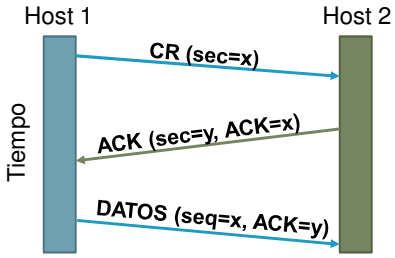
- Evitar que los paquetes permanezcan en la red por tiempo indeterminado y en algún momento lleguen a destino.
- Limitar el tiempo que permanecen los paquetes en la red:
  - Diseño de subred restringida: algún método para evitar que los paquetes hagan ciclos, retardos por congestión, etc.
  - Colocar un contador de saltos en cada paquete.
  - Marcar el tiempo en cada paquete.
- Garantizar que se limita el tiempo de permanencia en la red de las confirmaciones de recepción.

Establecer una Conexión

- Para solucionar la perdida de la historia de los paquetes enviados y recibidos ante la caída de un host es introducir la hora del día en cada paquete.
- Cada host contiene un reloj con la hora actual.
- Es un contador binario que se incrementa a intervalos uniformes.
- El reloj debe seguir funcionando ante una caída del host.
- Nunca puede estar pendiente al mismo tiempo dos TPDUs de número idéntico.
- Cuando se establece una conexión, los n bits menos significativos del reloj se estampan como número inicial de secuencia.

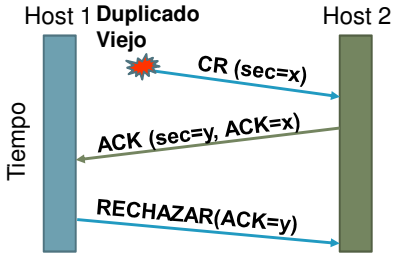
Establecer una Conexión

- Los TPDUs de control también pueden sufrir retrasos.
- Acuerdo de tres vías: para establecer una conexión.



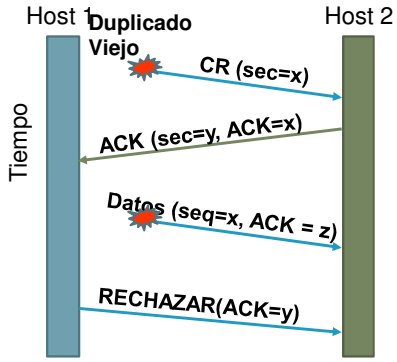
Establecer una Conexión

- Acuerdo de tres vías:
- Problema: paquetes de control (Connection Request) duplicados con retraso.



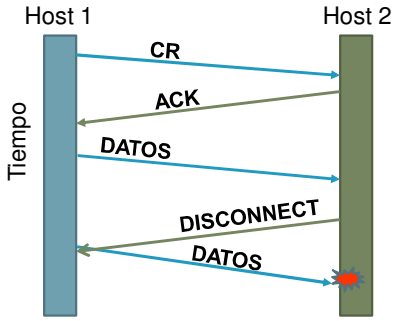
Establecer una Conexión

- Acuerdo de tres vías:
- Problema: paquetes de control (Connection Request y ACK) duplicados con retraso.



Liberar una Conexión

- **<u>Liberación asimétrica</u>**: cualquiera de los host puede finalizar la conexión.
- Cuando un cierra la conexión, la conexión se interrumpe.
- El cierre puede provocar la pérdida de datos.

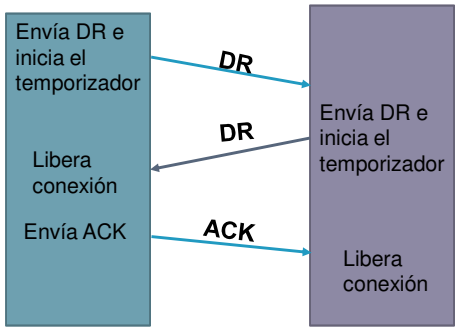


Liberar una Conexión

- **Liberación simétrica**: cada parte debe cerrar la conexión.
- Cada conexión se trata como dos conexiones unidireccionales.
- Empleada cuando cada proceso conoce la cantidad exacta de datos por enviar.
  - El host puede continuar recibiendo datos cuando ya envió el TPDU DISSCONNECT.

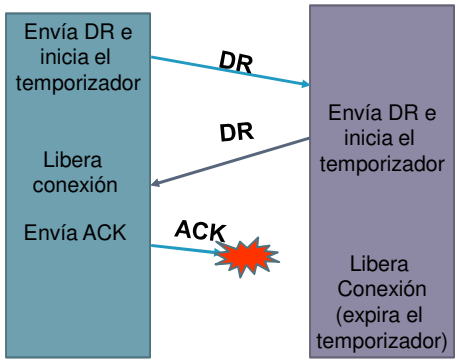
Liberar una Conexión

- **Liberación simétrica**: usando un protocolo de acuerdo de tres vías.



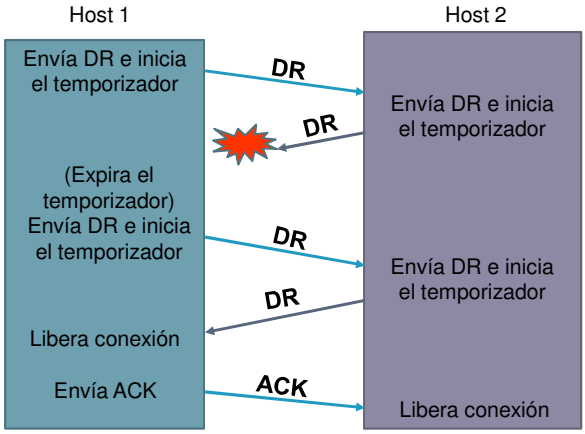
Liberar una Conexión

- **Liberación simétrica**: acuerdo de tres vías
- Problema: se pierde o retrasa el ACK.



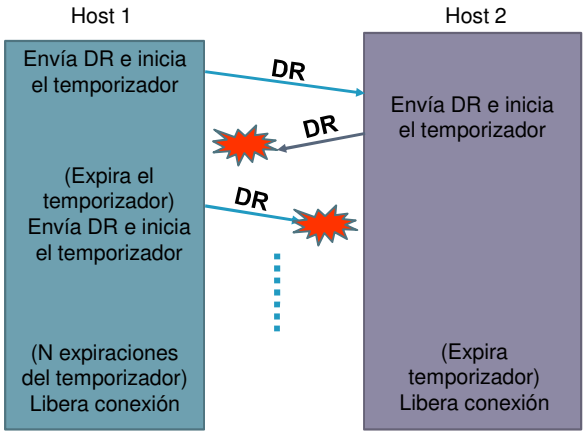
Liberar una Conexión

- **Liberación simétrica**: acuerdo de tres vías
- Problema: se pierde o retrasa el DR del host 2.



Liberar una Conexión

- **Liberación simétrica**: acuerdo de tres vías
- Problema: se retrasan o extravían todos los DR.



Control de Flujo

- Similitudes con el control de flujo en la capa de enlace de datos.
- Diferencias respecto a la cantidad de conexiones que se administran en la capa de transporte.
- Mucho retardo entre las transmisiones de los segmentos.
- Muy variable los tiempos de retardo.
- Técnicas:
  - No hacer nada.
  - Rechazar nuevos segmentos provenientes de la capa de red.
  - Ventana deslizante.
  - Esquema de créditos.

Control de Flujo

- Esquema de créditos:
- Cada octeto individual de datos que se transmite tiene un número de secuencia único.
- Cada segmento tiene una cabecera que incluye tres campos:
  - Número de Secuencia (SN) .
  - Número de Confirmación (AN).
  - Tamaño de la ventana (W).
- En cada transmisión de un segmento se incluye el número de secuencia del primer octeto del campo de datos.

Control de Flujo

- Esquema de créditos:
- El host servidor confirma un segmento recibido con el retorno de un segmento que incluye (AN = i , W = j):
  - Se confirman los i-1 octetos
  - Se concede permiso para enviar una ventana de j octetos.

Multiplexación

- Multiplexación hacia arriba:
  - Muchos procesos (aplicaciones) emplean el mismo protocolo de transporte.
  - se utilizan los puertos (TSAP) para distinguir a quien corresponde el segmento.
- Multiplexación hacia abajo:
  - Solo se dispone de una sola dirección de red.
  - Todas la conexiones deberán utilizar la misma dirección.

Protocolo de Transporte de Internet TPC

- Diseñado para proporcionar un flujo de bytes confiable de extremo a extremo a través de una interred no confiable.
- Una interred difiere de una sola red debido a que diversas partes podrían tener diferentes topologías, ancho de banda, retardos, tamaños de paquetes y otros parámetros.
- Maneja flujos TCP e interactúa con la capa IP.

Protocolo de Transporte de Internet TPC

- Maneja flujos TCP e interactúa con la capa IP.
- Acepta flujos de datos de usuario de procesos locales y los divide en fragmentos que no exceden los 64 KB.
- En la practica 1460 bytes de datos para que se ajusten en una sola trama Ethernet con los encabezados IP y TCP.
- Cada fragmento es enviado como un datagrama IP independiente.

Protocolo de Transporte de Internet TPC

- Maneja flujos TCP e interactúa con la capa IP.
- Acepta flujos de datos de usuario de procesos locales y los divide en fragmentos que no exceden los 64 KB.
- En la practica 1460 bytes de datos para que se ajusten en una sola trama Ethernet con los encabezados IP y TCP.
- Cada fragmento es enviado como un datagrama IP independiente.

Protocolo de Transporte de Internet TPC

- Cuando llega al destino.
- IP se lo pasa a la entidad TCP.
- Reconstruye los flujos de bytes originales.
- Funcionalidades que corresponde realizar a la entidad TCP:
  - Reensambar los datagramas en el orden correcto.
  - Terminar los temporizadores.
  - Retransmitir segmentos necesarios.

Modelo de Servicio TCP

- El servicio TCP se obtiene al hacer que el servidor y el cliente creen puntos terminales, llamados *sockets.*
- Establecer de manera explícita una conexión entre un *socket* en la máquina emisora y uno en la máquina receptora.
- Cada *socket* tiene una dirección:
  - Dirección NSAP (IP).
  - Dirección TSAP: un número de 16 bits denominado puerto en TCP.
- Un socket permite múltiples conexiones al mismo tiempo.
- Las conexiones se identifican mediante los identificadores de socket de los dos extremos.

Modelo de Servicio TCP

- Los números de puerto menores de 1024 se denominan puertos bien conocidos y se reservan para servicios estándar.

Puerto	Protocolo	Uso
21	FTP	Transferencia de archivos.
23	Telnet	Inicio remoto de sesión.
25	SMTP	Correo electrónico.
69	TFTP	Protocolo de transferencia de archivos trivial.
79	Finger	Búsqueda de información de usuarios.
80	HTTP	Protocolo de WWW.
110	POP-3	Acceso remoto al correo electónico.
119	NNTP	Noticias USENET.

Modelo de Servicio TCP

- Cada proceso servidor (demonio) puede estar bloqueado esperando la llegada de nuevas solicitudes de servicio.
- Inconveniente: permanecerían la mayor parte del tiempo inactivos, ocupando memoria y otros recursos del sistema.
- **Súper demonio:** (utilizada en los sistemas UNIX) se mantiene en ejecución un único demonio, **inetd** (demonio de Internet).
- Se conecta a todos los puertos habilitados en el sistema y espera la próxima conexión entrante.
- Cuando una solicitud de conexión llega, inetd crea un nuevo proceso y ejecuta el demonio apropiado.
- El nuevo proceso se encarga de resolver el pedido de servicio entrante.

Modelo de Servicio TCP

- Las conexiones TCP son:
  - dúplex total: el tráfico puede viajar en ambas direcciones al mismo tiempo.
  - Punto a punto: cada conexión tiene dos extremos finales, no soporta multidifusión.
- Las conexiones son de flujos de bytes: los límites de los mensajes no se preservan a través de la ruta.
- Cuando una aplicación entrega los datos a TCP, estos pueden ser enviados inmediatamente o almacenados en un buffer.
- **PUSH:** Los programas de aplicación puede indicar que TCP no retrase las transmisiones.
- **URGENT:** TCP interrumpe el encolamiento de datos y transmite inmediatamente todo lo que almacena para la conexión.

Protocolo TCP.

- La entidad emisora y receptora TCP intercambian datos en forma de segmentos.
- Tamaño de los segmento TCP: encabezado fijo de 20 bytes + parte opcional + 0 o más bytes de datos.
- El software TCP decide el tamaño de los segmentos:
  - Puede acumular varias datos para completar un segmento.
  - Puede dividir el segmento en varios datos.
- Límites del tamaño de un segmento TCP:
  - Carga útil de un paquete IP (65515 bytes) incluido el encabezado TCP.
  - Unidad Máxima de Transferencia (MTU) de la red (1500 bytes en Ethernet).

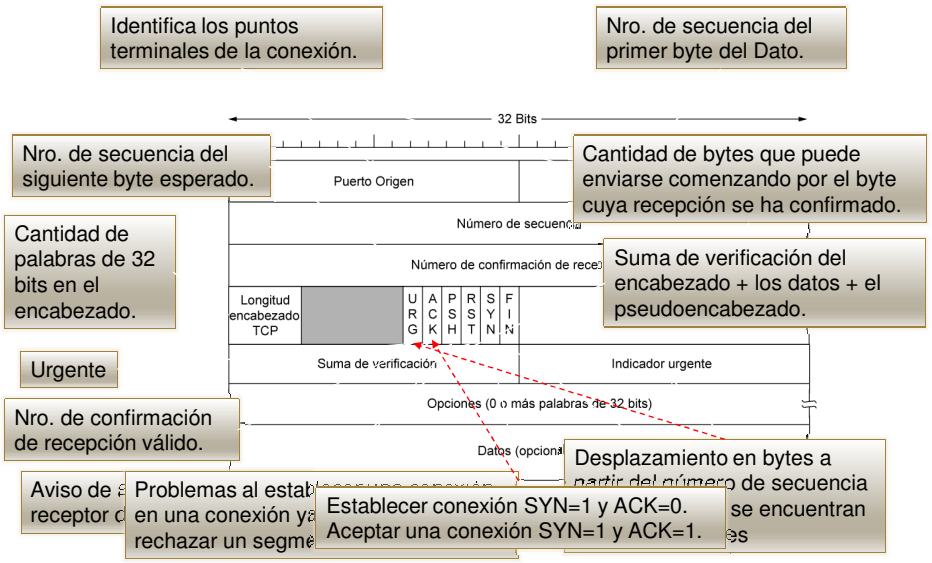
Protocolo TCP

- Emplea ventana deslizante:
  - Cuando un transmisor envía un segmento, inicializa un temporizador.
  - Cuando un segmento llega al destino, el receptor TCP devuelve un segmento (con datos, si existen) con el número de confirmación de recepción igual al siguiente número de secuencia que espera recibir.
  - Si el temporizador del emisor expira antes de recibir la confirmación, el segmento es reenviado.
  - La retransmisión podría incluir rangos de bytes diferentes a los de la transmisión original.

TCP: Formato del segmento.

- 20 bytes de formato fijos para el encabezado.
- 20 bytes opcionales.
- 65,495 bytes (65535 – 20 – 20, 20 bytes del encabezado TCP y 20 bytes del encabezado IP) de datos.
- El campo datos puede contener 0 bytes (no contiene datos).
- Utilizados para segmentos de confirmación de recepción y mensajes de control.

TCP: Formato del segmento.



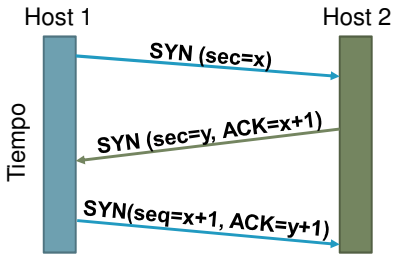
TCP: Pseudoencabezado.

- Incluir el pseudoencabezado en la suma de verificación permite detectar paquetes mal entregados.
- Incluye:
  - Dirección IP origen y destino.
  - Número de Protocolo TCP (6).
  - Cantidad de bytes del segmento TCP (incluido el encabezado).
- Viola el modelo de encapsulamiento.

TCP: Establecimiento de una Conexión

- Emplea el acuerdo de tres vías.
- El servidor espera una conexión entrante, en algún puerto.
- El cliente ejecuta la primitiva CONNECT especificando la dirección IP del servidor y el puerto del proceso servidor.
- La primitiva CONNECT envía un segmento TCP con el bit *SYN* encendido y el bit *ACK* en 0.
- Al llegar el segmento al destino, TCP revisa si existe un proceso con el número de puerto abierto.
- Si no existe, responde con un segmento con el bit *RST* en 1 para rechazar la conexión.
- Si existe, el proceso recibe el segmento y puede aceptar o rechazar la conexión; si acepta devuelve un segmento de confirmación de la conexión.

TCP: Establecimiento de una Conexión

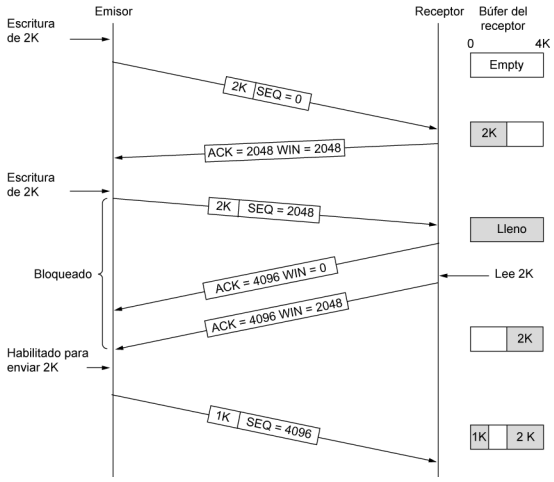


TCP: Liberación de una Conexión

- Las conexiones se deben liberar en ambos sentidos, del cliente al servidor y del servidor al cliente.
- Cualquiera de las partes puede liberar la conexión, enviando un segmento con el bit *FIN* encendido.
- Al confirmarse el segmento de *FIN*, la conexión en ese sentido se apaga.
- El otro sentido de la conexión se mantiene habilitada, por el cual se puede seguir enviando segmentos.
- Cuando ambos sentidos se apagan, la conexión se libera.

TCP: Política de Transmisión.

- No se requiere que el emisor envíe datos tan pronto como llegan de la aplicación.
- No se requiere que el receptor envíe confirmación de recepción tan pronto como sea posible.



TCP: Política de Transmisión.

- Reducir el tráfico desde el receptor: Retardo del las confirmaciones de recepción y de las actualizaciones de ventana durante 500 mseg. Para agregar las confirmaciones a segmentos de datos.
- **Algoritmo de Neagle**: Retrasar el envío de datos hasta que llegue la confirmación de los bytes enviados y pendientes de confirmar. Esto da tiempo para recolectar más datos.
- **Algoritmo de Clark:** Enviar una actualización del tamaño de la ventana cuando pueda contener un segmento o hasta que el buffer tenga la mitad de su capacidad.

TCP: Control de la Congestión.

- Responsable principal del problema.
- Manipulación dinámica de la ventana: no transmitir más paquetes hasta que se confirmen los anteriores.
- Cuando expira el temporizador es porque se retraso.
- Esto le indica a TCP que se esta produciendo congestiones en la red (en la mayoría de los casos).
- Evitar la congestión:
  - Al establecer la conexión se establece el tamaño de la ventana adecuado.
  - El receptor establece la ventana al tamaño de su búfer.
  - Si el emisor se ajusta al tamaño de la ventana no se provoca congestión.
- Pero puede producirse por problemas de congestión en la red.

TCP: Control de la Congestión.

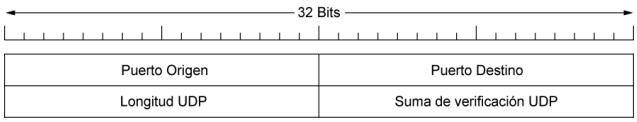
- Técnica del Arranque Lento:
  - Cada emisor define el tamaño de dos Ventanas: ventana acordada con el receptor y ventana de congestión.
  - Al establecer una conexión el emisor establece la ventana de congestión al tamaño máx. del segmento usado por la conexión.
  - Si se recibe la confirmación antes de que expire el temporizador, se amplía el tamaño de la ventana de congestión el equivalente al tamaño de un segmento.
  - Ahora envía 2 segmentos.
  - Si recibe las 2 confirmaciones antes de que expire el temporizador, amplia el equivalente a 2 segmentos.
  - Continua este proceso (crecimiento exponencial) hasta que no se confirme algún paquete o se alcance el tamaño de la ventana del receptor.

TCP: Control de la Congestión.

- Congestión en Internet:
- Agrega un tercer parámetro: Umbral (64 KB).
- Al expirar un temporizador, se establece el umbral a la mitad del tamaño de la ventana de congestión.
- La ventana de congestión se restablece a un segmento.
- Se aplica arranque lento para determinar lo que puede manejar la red. Pero cuando se alcanza el umbral se corta el crecimiento exponencial y se comienza a crecer de a un paquete (crecimiento lineal).

Protocolo de Datagramas de Usuario: UDP

- Protocolo de Transporte No Orientado a la Conexión.
- Permite que las aplicaciones envíen datagramas IP.
- Segmento: encabezado de 8 bytes y la carga útil.



- Puerto Origen, Puerto Destino: identifica los puntos terminales dentro de las máquinas de origen y destino.
- Cuando un paquete arriba al destino, su carga útil se entrega al proceso que está enlazado al puerto de destino.
- Longitud: del encabezado + los datos.
- Suma de verificación UDP: es opcional (valor 0).

Protocolo de Datagramas de Usuario: UDP

- No realiza control de flujo.
- No realiza control de errores.
- No realiza retransmisión.

- Es especialmente aplicable al modelo cliente servidor. El cliente envía solicitudes cortas al servidor y espera respuestas cortas.
- El Sistema de Nombres de Dominio (DNS) emplea el protocolo UDP.
- Se envía un paquete UDP al servidor DNS, conteniendo el nombre de host remoto y espera como respuesta un paquete UDP con la dirección IP.

UDP: Llamada a procedimiento remotos (RPC).

- El propósito esencial es hacer que las llamadas a procedimientos remotos sean lo más parecido posible a invocaciones de funciones y procedimientos locales.
- Posibilita invocar procedimientos alojados en hosts remotos, con uno o mas parámetros. El proceso se bloquea esperando la respuesta.
- El host remoto ejecuta el proceso invocado, con el conjunto de parámetros y devuelve el resultado final al host de origen.
- Los pasajes de mensajes es transparente para los usuarios. La información se transportar desde el invocador al proceso invocado en los parámetros.
- Servidor: proceso invocado, que ejecuta el procedimiento invocado
- Cliente: proceso invocador, que solicita la ejecución de un proceso remoto.

UDP: Llamada a procedimiento remotos (RPC).

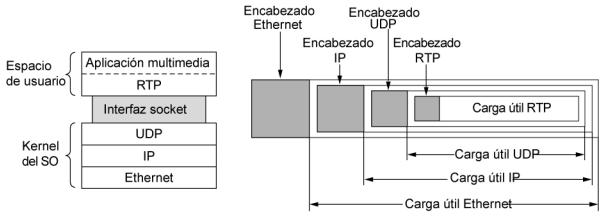
- Se emplea paquetes UDP.
- En algunos casos se emplean paquetes TCP, en el caso que los parámetros o resultados son más grandes que el tamaño máximo del paquete UDP.

UDP: Protocolo de transporte en Tiempo Real (RTP)

- Se emplea ampliamente en Radio en Internet, telefonía en Internet, música bajo demanda, vídeos conferencias, videos bajo demanda y otras aplicaciones multimedia.
- La principal función de RTP es multiplexar varios flujos de datos en tiempo real en un solo flujos de paquetes UDP.
- El flujo UDP se puede enviar en un único destino (unidifusión), o a múltiples destinos (multidifusión).

UDP: Protocolo de transporte en Tiempo Real

- Las aplicaciones multimedia consisten en múltiples flujos de audio, vídeo, texto, etc.
- Éstos se colocan en la biblioteca RTP, la cual multiplexa los flujos y los codifica en paquetes RTP, que después los colocas en un socket.
- Se generan uno o mas paquetes UDP para ser encapsulados en paquetes IP.



UDP: Protocolo de transporte en Tiempo Real

- El protocolo no implementa control de flujo, control de errores, confirmación de recepción y mecanismos para solicitar retransmisiones.
- A cada paquete enviado en un flujo RTP se le asigna un número más grande que a su predecesor. Permitiendo detectar la perdida de algún paquete.
- En caso de perderse (o se retrasarse) un paquete, el protocolo lo podría subsanar calculando el valor o valores faltantes mediante la interpolación.

Primitivas del Servicio de Transporte

- La capa de transporte proporciona una interfaz para los servicios de transporte.
- Permite a los usuarios acceder a los servicios de transporte.
- Los servicios de transporte orientados a la conexión ofrecen un servicios confiables en una red no confiable.

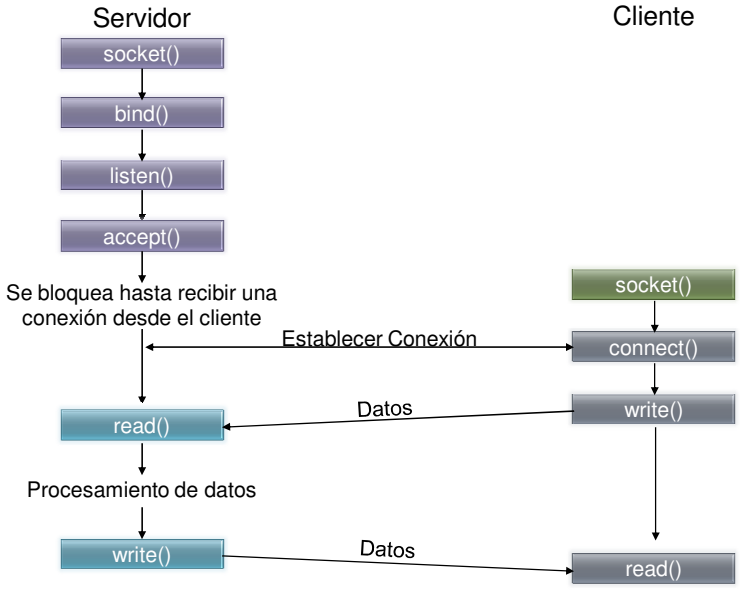
Socket de Berkeley

- Funciones ejecutadas por el programa Servidor:
  - ***socket()***: crea un nuevo punto de comunicación y le asigna espacio en las tablas de la entidad de transporte.
  - ***bind()***: asigna la dirección del socket.
  - ***listen()***: asigna espacio para poner en cola las llamadas entrantes. Puede ocurrir que varios clientes intenten conectarse al mismo tiempo.
  - ***accept()***: el servidor se bloquea esperando nuevas solicitudes de conexión. Cuando llega una TPDU solicitando una conexión, la entidad de transporte crea un nuevo socket con las mismas propiedades que el original. El proceso original regresa a su estado bloqueado a la espera de nuevas solicitudes.
  - ***send() y receive()***: para transmitir y recibir datos a través de la conexión dúplex total.
  - ***close()***: liberación de la conexión. La liberación de la conexión es simétrica.

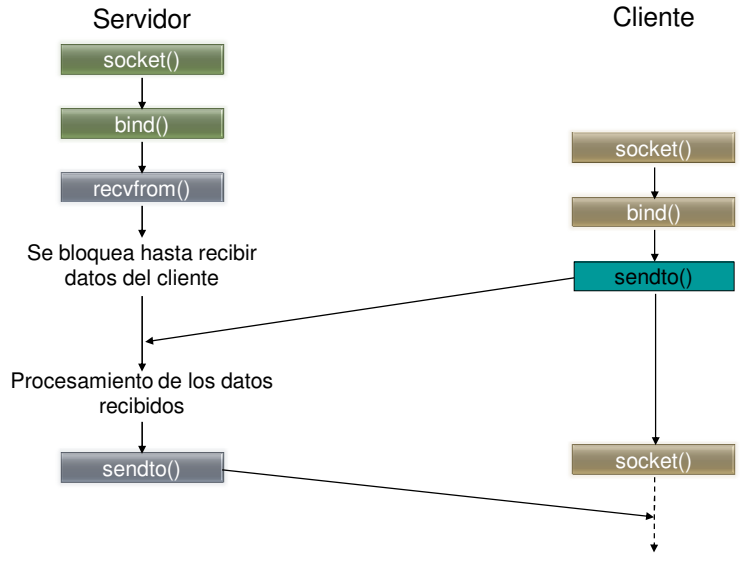
Socket de Berkeley

- Funciones ejecutadas por el programa Cliente:
  - ***socket()***: crea un nuevo punto de comunicación y le asigna espacio en las tablas de la entidad de transporte.
  - ***connect()***: bloquea el proceso y comienza activamente el proceso de conexión. Al completarse el proceso cliente se desbloquea y se establece la conexión.
  - ***send() y receive()***: para transmitir y recibir datos a través de la conexión dúplex total.
  - ***close()***: liberación de la conexión. La liberación de la conexión es simétrica.

Socket: Protocolo con conexión



Socket: Protocolo sin conexión



Socket: Ejemplo TCP

Servidor.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>

int main() {
   int s_sock, c_sock;
   int server_len, client_len;
   struct sockaddr_in server_address;
   struct sockaddr_in client_address;
   char str_send[30]= "Hola, Cliente remoto  !!!";
   char str_read[30];

   s_sock= socket(AF_INET, SOCK_STREAM, 0);

   server_address.sin_family= AF_INET;
   server_address.sin_addr.s_addr= inet_ad
   server_address.sin_port= 9734;
   server_len= sizeof(server_address);
```

Familia de Proto

Tipo de Protocolo:
•SOCK_STREAM
•SOCK_DGRAM

Crea el extremo de la conexión.

Socket de Berkeley: Ejemplo

Servidor.c

```
bind(s_sock, (struct sockaddr *)&server_address, server_len);

listen(s_sock, 5);

while(1) {
   printf("server waiting\n");
   c_sock= accept(s_sock, (struct sockaddr *)&client_address,  &client_len);

   read(c_sock, str_read, sizeof(str_read));
   printf("Server:  msg resivido: %s", str_read);

   write(c_sock, str_send, sizeof(str_send));
   close(c_sock);
  }
}
```

Logitud de cola de conexiones pendientes.

Conecta el socket *(s_sock)* con la dirección *server_address.*

Dispos

Se bloquea esperando una nueva conexión.

Socket de Berkeley: Ejemplo

Cliente.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>

int main() {
   int sock;
   int len;
   struct sockaddr_in address;
   int result;
   char str[30]= "Hola, Servidor remoto !!!!";

   sock= socket(AF_INET, SOCK_STREAM, 0);

   address.sin_family= AF_INET;
   address.sin_addr.s_addr= inet_addr(" 172.210.56.96");
   address.sin_port= 9734;
   len= sizeof(address);
```

Socket de Berkeley: Ejemplo

Cliente.c

```
   connect(sock, (struct sockaddr *)&address, len);

   write(sock, str, strlen(str));

   read(sock, str, strlen(str));
   printf("Cliente:\n  Msg recivido: %s\n", str);

   close(sock);
}
```