Instituto Politécnico Nacional

Escuela Superior de Cómputo

Ingeniería en Sistemas Computacionales

Aplicaciones para Comunicaciones en Red

Principios de las aplicaciones para comunicaciones en red

M. en C. Sandra Ivette Bautista Rosales

Principios de las aplicaciones de red

- El desarrollo de una aplicación de red implica escribir programas que se ejecuten en distintos sistemas terminales y que se comuniquen entre sí a través de la red.
- Por ejemplo, en una aplicación web se emplean dos programas diferentes que se comunican entre sí: el navegador que se ejecuta en el host del usuario (una computadora de escritorio, un portátil, una tableta, un teléfono inteligente, etc.) y el programa del servidor web que se ejecuta en el host que actúa como servidor web.

Principios de las aplicaciones de red

- Al desarrollar una aplicación se codifica un software que se ejecutará en múltiples sistemas terminales. Una cuestión importante es que no es necesario escribir software que se ejecute en los dispositivos del núcleo de la red, por ejemplo, los routers o switches de la capa de enlace. Incluso, aunque desearan escribir software de aplicación para estos dispositivos del núcleo de la red, no podrían hacerlo, ya que los dispositivos del núcleo de la red no operan en la capa de aplicación, sino en la capas situadas más abajo: la capa de red e inferiores.

- Este diseño básico (que confina el software de aplicación a los sistemas terminales), ha facilitado el rápido desarrollo e implementación de una gran variedad de aplicaciones de red.

Arquitecturas de las aplicaciones de red

- Antes de profundizar en la codificación del software, deberíamos disponer de una visión general de la arquitectura de la aplicación.
- Tengan en cuenta que la arquitectura de la aplicación es muy distinta de la arquitectura de la red (TCP/IP).
- Desde la perspectiva del desarrollador de aplicaciones, la arquitectura de la red es fija y proporciona un conjunto específico de servicios a las aplicaciones. Por otro lado, el desarrollador de aplicaciones diseña la **arquitectura de la aplicación**, que establece cómo debe estructurarse la aplicación en los distintos sistemas terminales. Al seleccionar la arquitectura de la aplicación, el desarrollador probablemente utilizará uno de los dos paradigmas arquitectónicos predominantes en las aplicaciones de red modernas: la arquitectura **cliente-servidor** o la arquitectura **P2P**.

Arquitecturas de las aplicaciones de red

- En la arquitectura **cliente-servidor** existe un host siempre activo, denominado servidor, que da servicio a las solicitudes de muchos otros host, que son los clientes. Un ejemplo clásico es la Web, en la que un servidor web siempre activo sirve las solicitudes de los navegadores que se ejecutan en los host clientes. Cuando un servidor web recibe una solicitud de un objeto de un host cliente, responde enviándole el objeto solicitado. Observe que, con la arquitectura cliente-servidor, los clientes no se comunican directamente entre sí; por ejemplo, en la aplicación web, dos navegadores no se comunican de forma directa.
- Otra característica de la arquitectura cliente-servidor, es que el servidor tiene una dirección fija y conocida, denominada dirección IP. Puesto que el servidor tiene una dirección fija y conocida, y siempre está activo, un cliente siempre puede contactar con él enviando un paquete a su dirección IP. Entre las aplicaciones más conocidas que utilizan la arquitectura cliente-servidor se encuentran la Web, FTP, Telnet y el correo electrónico.

Arquitecturas de las aplicaciones de red

- A menudo en una aplicación cliente-servidor un único host servidor es incapaz de responder a todas las solicitudes de sus clientes. Por ejemplo, el sitio de una red social popular puede verse rápidamente desbordado si solo dispone de un servidor para gestionar todas las solicitudes. Por esta razón, en las arquitecturas cliente-servidor suele utilizarse un **centro de datos**, que alberga un gran número de hosts, para crear un servidor virtual de gran capacidad.
- Los servicios de Internet más populares como los motores de búsqueda, sitios de comercio por Internet, el correo electrónico basado en la Web o las redes sociales, emplean uno o más centros de datos.
- Un centro de datos puede tener cientos de miles de servidores, a los que hay que suministrar energía y mantener. Además, los proveedores del servicio deben pagar los costes recurrentes de interconexión y de ancho de banda para poder enviar datos desde sus centros de datos.

Arquitecturas de las aplicaciones de red

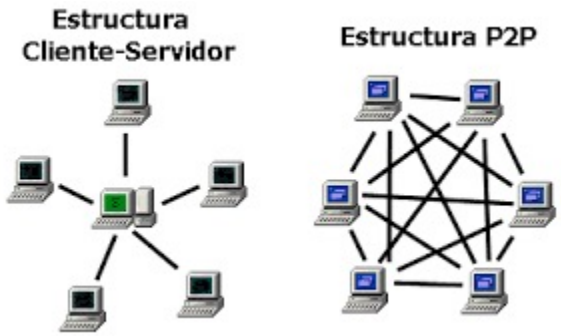
- En una arquitectura P2P existe una mínima (o ninguna) dependencia de una infraestructura de servidores dedicados situados en centros de datos. En su lugar, la aplicación explota la comunicación directa entre parejas de hosts conectados de forma intermitente, conocidos como pares (peers).
- Los pares no son propiedad del proveedor del servicio, sino que son computadoras de escritorio y portátiles controladas por los usuarios, encontrándose la mayoría de los pares en domicilios, universidades y oficinas.
- Puesto que los pares se comunican sin pasar por un servidor dedicado, la arquitectura se denomina arquitectura peer-to-peer (P2P). Muchas de las aplicaciones actuales más populares y con un mayor nivel de tráfico están basadas en arquitecturas P2P.

Arquitecturas de las aplicaciones de red

- Entre estas aplicaciones se incluyen la compartición de archivos (por ejemplo, BitTorrent), la aceleración de descarga con ayuda de pares (por ejemplo, Xunlei) y la telefonía y videoconferencia por Internet (por ejemplo, Skype).
- Conviene mencionar que algunas aplicaciones tienen arquitecturas híbridas, que combinan elementos cliente-servidor y P2P. Por ejemplo, en muchas aplicaciones de mensajería instantánea, los servidores se utilizan para hacer un seguimiento de las direcciones IP de los usuarios, pero los mensajes de usuario a usuario se envían directamente entre los hosts de dichos usuarios (sin pasar por servidores intermedios).

Arquitecturas de las aplicaciones de red

- Una de las características más convincentes de las arquitecturas P2P es su auto-escalabilidad. Por ejemplo, una aplicación de compartición de archivos P2P, aunque cada par genera una carga de trabajo solicitando archivos, también añade capacidad de servicio al sistema, distribuyendo archivos a otros pares. Las arquitecturas P2P también presentan una buena relación coste-prestaciones, ya que normalmente no requieren una infraestructura de servidores significativa ni un gran ancho de banda de servidor (a diferencia de los diseños cliente-servidor con centros de datos). Sin embargo, las aplicaciones P2P plantean problemas de seguridad, rendimiento y fiabilidad, debido a su naturaleza altamente descentralizada.



Comunicación entre procesos

- Antes de crear una aplicación de red, es necesario tener los conocimientos básicos sobre cómo se comunican entre sí los programas que se ejecutan en varios sistemas terminales.
- En el ámbito de los sistemas operativos, realmente los que se comunican no son los programas, sino procesos. Un proceso puede interpretarse como un programa que se ejecuta dentro de un sistema terminal. Cuando los procesos se ejecutan en el mismo sistema terminal, pueden comunicarse entre sí mediante sistemas de comunicación inter-procesos, aplicando reglas gobernadas por el sistema operativo del sistema terminal.
- Los procesos de dos sistemas terminales diferentes se comunican entre sí intercambiando mensajes a través de la red de computadoras. Un proceso emisor crea y envía mensajes a la red; un proceso receptor recibe estos mensajes y posiblemente responde devolviendo otros mensajes.

Procesos cliente y servidor

- Una aplicación de red consta de parejas de procesos que se envían mensajes entre sí a través de una red. Normalmente, en una pareja de procesos que están comunicándose, designamos a uno de los procesos como el cliente y al otro como el servidor. En la Web, un navegador es un proceso cliente y el servidor web es un proceso servidor. En la compartición de archivos P2P, el par que descarga el archivo se designa como el cliente y el host que está cargando el archivo se designa como el servidor.
- En algunas aplicaciones, tales como la compartición de archivos P2P, un proceso puede ser tanto un cliente como un servidor. De hecho, un proceso en un sistema de compartición de archivos P2P puede cargar y descargar archivos. No obstante, en el contexto de cualquier sesión de comunicación específica entre una pareja de proceso, seguimos pudiendo designar a uno de los procesos como el cliente y al otro como el servidor.

Procesos cliente y servidor

- En el contexto de una sesión de comunicación entre una pareja de procesos, definimos los procesos cliente y servidor como sigue:
  - El proceso que inicia la comunicación en el mismo o diferente nodo (es decir, que inicialmente se pone en contacto con el otro proceso al principio de la sesión) se designa como el cliente.
    - Las peticiones están originadas por la necesidad de acceder al recurso que gestiona el servidor.
  - El proceso que espera a ser contactado para comenzar la sesión es el servidor.
    - Su función es gestionar el acceso a un determinado recurso.
- En la compartición de archivos P2P, cuando un par A pide a un par B que le envíe un determinado archivo, el A es el cliente y el B es el servidor en el contexto de esta sesión de comunicación concreta. En ocasiones también se emplea el término “lado del cliente y lado del servidor de una aplicación”.

Proceso servidor

- Está continuamente esperando peticiones de servicio.
- Cuando se produce una petición, el servidor despierta y atiende al cliente.
- Cuando el servicio concluye, el servidor vuelve al estado de espera.

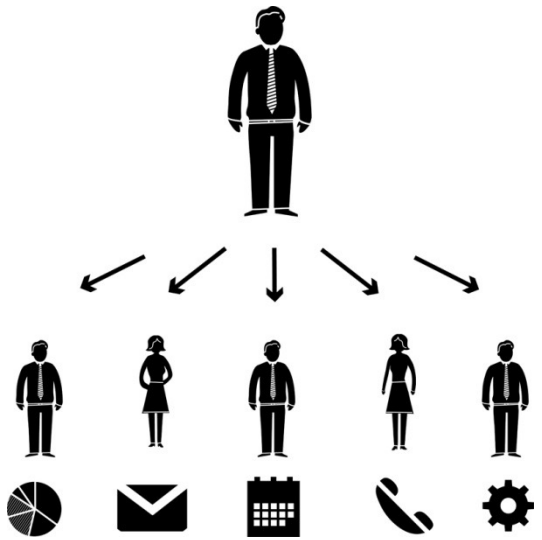
Almacena y valida datos, realiza cálculos complejos, contesta las respuestas

Proceso servidor

- De acuerdo con la forma de prestar servicio , se pueden considerar 2 tipos de servidores:

- SERVIDORES INTERACTIVOS: El servidor no solo recoge la petición de servicio, sino que él mismo se encarga de atenderla.
  - Inconveniente: Si el servidor es lento en atender a los clientes y hay una demanda de servicio muy elevada, se van a originar tiempos de espera muy grandes.
  - Solo pueden procesar una petición a la vez

Proceso servidor



- SERVIDORES CONCURRENTES:
  - El servidor recoge cada un de las peticiones de servicio y crea otros procesos para que se encarguen de atenderlas.
  - Este tipo de servidores solo es aplicable en sistemas multiproceso.
  - Ventaja: El servidor puede recoger peticiones a muy alta velocidad, porque esta descargado de la tarea de atención al cliente.
  - En aplicaciones donde los tiempos de servicio son variables, es recomendable implementar este tipo de servidores.

Proceso servidor

- Su papel es pasivo en el establecimiento de la comunicación.
  - Para esto dispone de un socket de escucha, enlazado al puerto TCP correspondiente al servicio, sobre el que espera las peticiones de conexión.
  - Cuando llega al sistema una petición de este tipo, se despierta el proceso servidor y se crea un **nuevo socket**, que se llama *socket de servicio*, el cual se conecta al cliente.
- Entonces el servidor podrá:
  - Delegar el trabajo necesario para la realización del servicio a un nuevo proceso, que utilizará entonces la conexión.
  - Volverá al socket de escucha.

Proceso cliente

- Es la entidad activa en el establecimiento de una conexión, puesto que es el que toma la iniciativa de la demanda de conexión a un servidor.
  - Esta demanda se realiza por medio de la primitiva **connect()** solicitando el establecimiento de una conexión que será conocida por los dos extremos.
  - Además el cliente está informado del éxito o el fracaso del establecimiento de la conexión.

Captura los datos, los muestra, solicita consultas, mantiene la interfaz de usuario.

Procesos cliente y servidor

- La comunicación entre cliente y servidor puede ser orientada a la conexión o bien sin conexión.

Servicios orientados a la conexión y no orientados a la conexión

Servicio orientado a la conexión

- Para usar un servicio de red orientado a la conexión, el usuario:
  - Establece una conexión
  - La utiliza
  - La abandona

Servicios orientados a la conexión

- Conexión: el emisor empuja objetos (bits) en un extremo y el receptor los toma en el otro extremo.
  - Generalmente se conserva el orden de los bits
- Al establecer la conexión, el emisor, el receptor y la subred utilizan una negociación sobre los parámetros que se van a utilizar, por ejemplo:
  - Tamaño máximo del mensaje
  - Calidad del servicio solicitado
- Uno hace la propuesta y el otro la acepta, rechaza o hace una contrapropuesta.

Servicios No orientados a la conexión

- Cada mensaje (carta) lleva la dirección destino y cada una se enruta a través del sistema, independientemente de las demás.

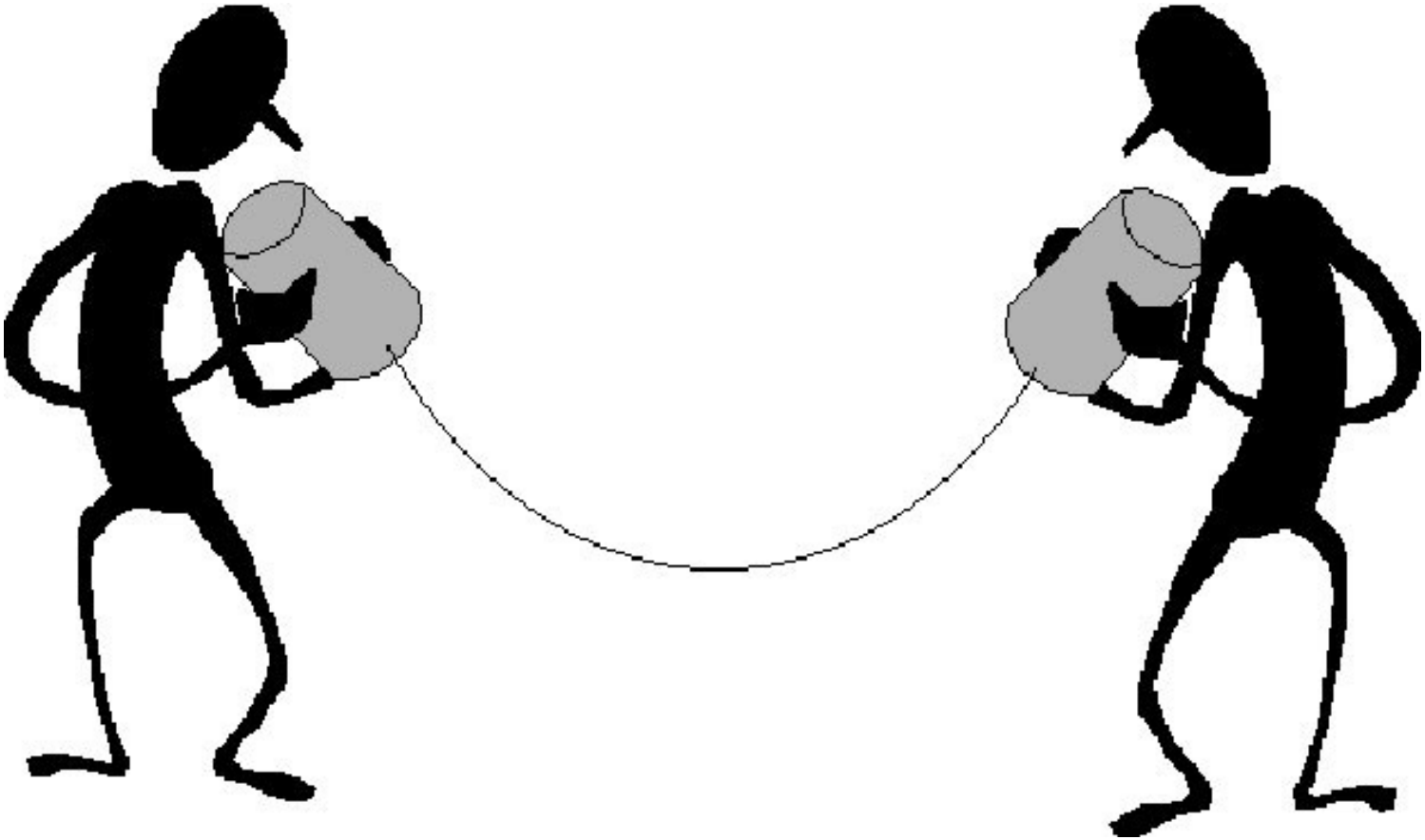
Servicios No orientados a la conexión

- Generalmente, cuando se envían dos mensajes al mismo destino el primero que se envíe será el primero en llegar.
  - Es posible que el primero se dilate y llegue primero el segundo mensaje.

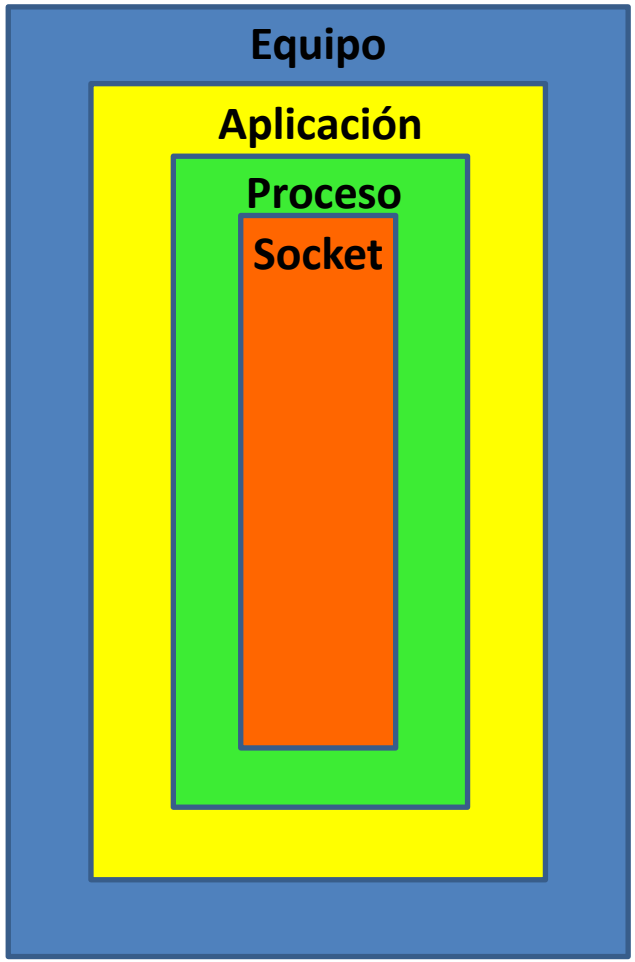
Servicios orientados y no orientados a la conexión

	Servicio	Ejemplo
Orientado a la conexión	Flujo confiable de mensajes	Secuencia de páginas
	Flujo confiable de bytes	Inicio de sesión remoto
	Conexión no confiable	Voz digitalizada
No orientado a la conexión	Datagrama no confiable	Correo electrónico basura
	Datagrama confirmado	Correo certificado
	Solicitud-respuesta	Consulta de base de datos

Comunicación entre procesos



Comunicación entre procesos



Mecanismos de comunicación

Internos

Externos

Mecanismos de comunicación

- Internos (Entre procesos dentro de una misma máquina)
  - Básicos
    - Señales: Informan de sucesos pero no sirven para enviar o recibir datos adicionales
    - Ficheros con bloqueos: Relativamente lento → accesos a disco. Se puede decir que es una «pseudo Base de datos»
    - Tuberías: FIFO internas (pipe) y FIFO externas con nombre (memoria RAM, más rápida pero capacidad limitada)
  - Mecanismos IPC (Inter-Process Communication) de UNIX
    - Semáforos: Mecanismos de sincronización
    - Colas de mensajes: Intercambio ordenado de cantidades discretas de datos
    - Memoria compartida: Acceso compartido a un mismo espacio de memoria

Mecanismos de comunicación

Colas de mensajes: pueden ser descritas como una serie de listas enlazadas situadas dentro del espacio de direcciones del kernel. Los procesos pueden enviar mensajes a una cola y retirarlos en un orden u otro de la misma. Cada cola de mensajes tiene un identificador único.

Semáforos: están implementados como contadores que sirven para controlar el acceso a recursos compartidos por varios procesos. Se suelen usar como un mecanismo de bloqueo que impide a otros procesos el acceso a un recurso en particular en tanto haya algún proceso realizando operaciones sobre el mismo. Los semáforos son de los tres mecanismos el más complejo de entender.

Memoria compartida: Es posible que dos procesos compartan un mismo segmento de memoria, si el segmento es mapeado al espacio de direcciones de cada proceso que la comparta. Por eso es el mecanismo IPC más rápido porque no hay ningún intermedio, ni tuberías, ni colas. La información es escrita directamente en la memoria mapeada a todos los procesos que inmediatamente la pueden ver. Un segmento así de memoria compartida es creado por un proceso y a partir de ahí otros la leen y la escriben.

Mecanismos de comunicación

- Externos (Entre procesos de distintas máquinas conectadas mediante una red)
  - Comunicación basada en sockets
    - AF_UNIX:
      - Sockets internos de UNIX (Sockets del sistema de archivos)
      - Similar a los fifos con nombre pero bidireccionales
    - AF_INET:
      - Protocolos de Internet ARPA (TCP/IP)→ Protocolos TCP y UDP de la capa de transporte.
      - Sockets de redes de UNIX

Comunicación entre procesos

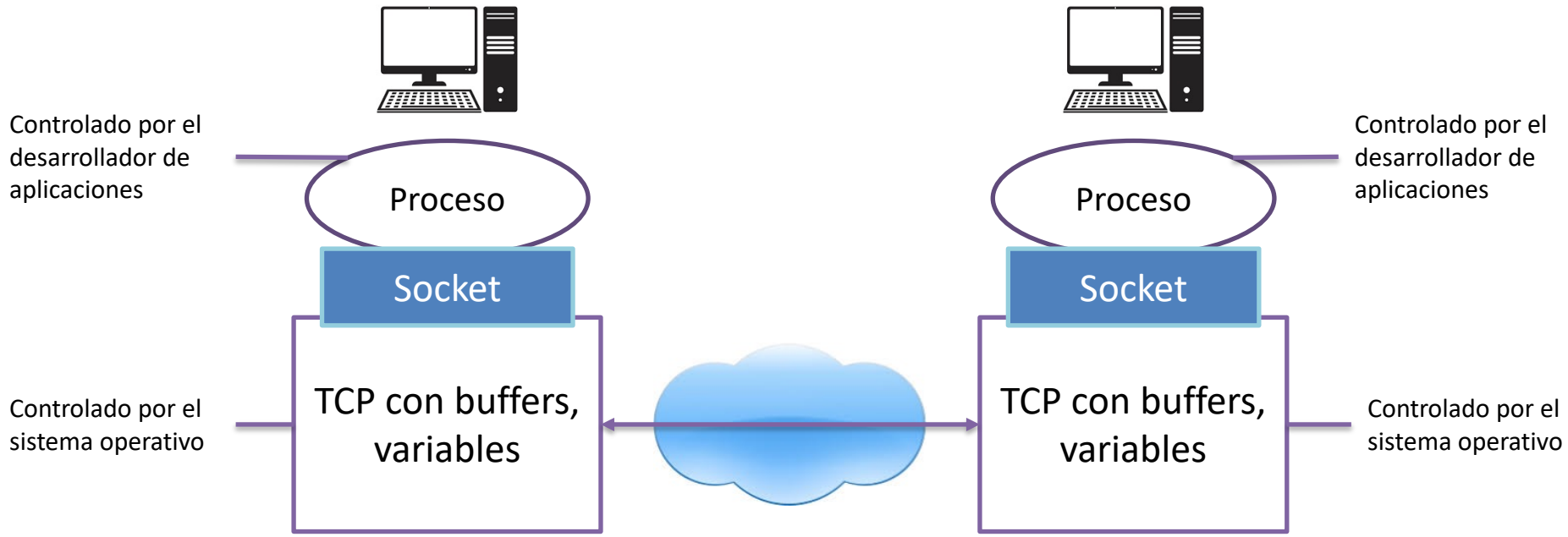
- Intercambio de mensajes (emisor-receptor(es))
  - Uno a uno (unicast)
  - Uno a muchos (multicast y broadcast)

- Esquema típico: mecanismo petición-respuesta
  - Distintos niveles de abstracción
  - Ejemplos: interfaz sockets, mecanismos RPC (llamada a procedimiento remoto)

Interfaz entre el proceso y la red de computadoras

- La figura de la siguiente diapositiva ilustra la comunicación mediante sockets entre dos procesos que se comunican a través de Internet (donde se supone que el protocolo de transporte subyacente utilizado por los procesos es el protocolo TCP de Internet).
- Como se muestra en la figura, un socket es la interfaz entre la capa de aplicación y la capa de transporte de un host. También se conoce como Interfaz de programación de aplicaciones (API, Application Programming Interface) entre la aplicación y la red, ya que el socket es la interfaz de programación con la que se construyen las aplicaciones de red.

Interfaz entre el proceso y la red de computadoras

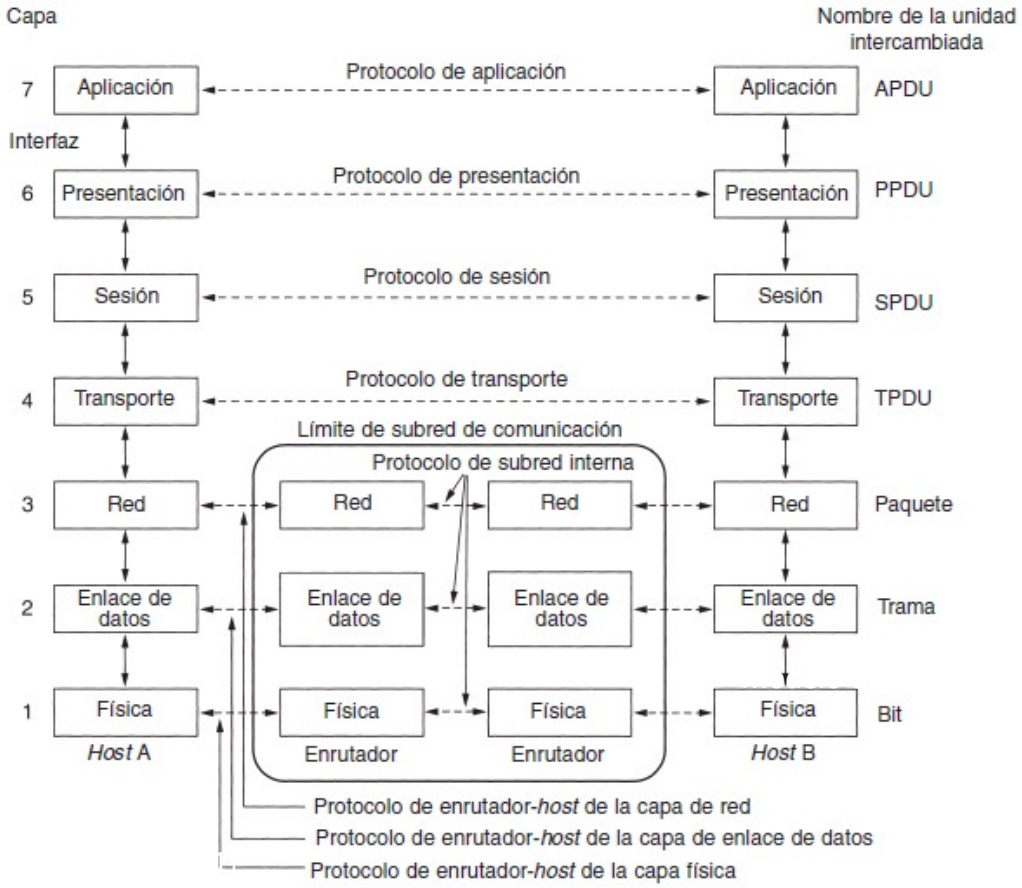


Interfaz entre el proceso y la red de computadoras

- El desarrollador de la aplicación tiene el control sobre todos los elementos del lado de la capa de aplicación del socket, pero apenas tiene control sobre el lado de la capa de transporte del socket. El único control que tiene el desarrollador de la aplicación sobre el lado de la capa de transporte es:
  - La elección del protocolo de transporte
  - Quizá la capacidad de fijar unos pocos parámetros de la capa de transporte, por ejemplo, el tamaño máximo del buffer y de segmento.
- Una vez que el desarrollador de la aplicación ha seleccionado un protocolo de transporte, la aplicación se construye utilizando los servicios de la capa de transporte proporcionados por dicho protocolo.

1.1 Servicios definidos en la capa de transporte

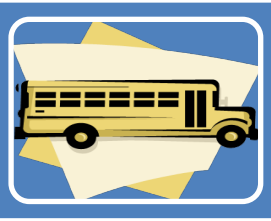
Modelo OSI


Capa
Nombre de la unidad intercambiada
7
Aplicación
Protocolo de aplicación
Aplicación
APDU
Interfaz
6
Presentación
Protocolo de presentación
Presentación
PPDU
5
Sesión
Protocolo de sesión
Sesión
SPDU
4
Transporte
Protocolo de transporte
Transporte
TPDU
Límite de subred de comunicación
Protocolo de subred interna
3
Red
Red
Red
Red
Paquete
2
Enlace de datos
Enlace de datos
Enlace de datos
Enlace de datos
Trama
1
Física
Física
Física
Física
Bit
Host A
Enrutador
Enrutador
Host B
Protocolo de enrutador-host de la capa de red
Protocolo de enrutador-host de la capa de enlace de datos
Protocolo de enrutador-host de la capa física

Arquitectura TCP/IP

Aplicación

- HTTP, FTP, TFTP, etc.

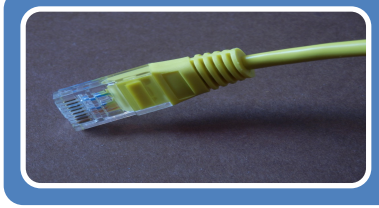


Transporte

- TCP
- UDP

Internet

- IP
- IGMP



Acceso a la red

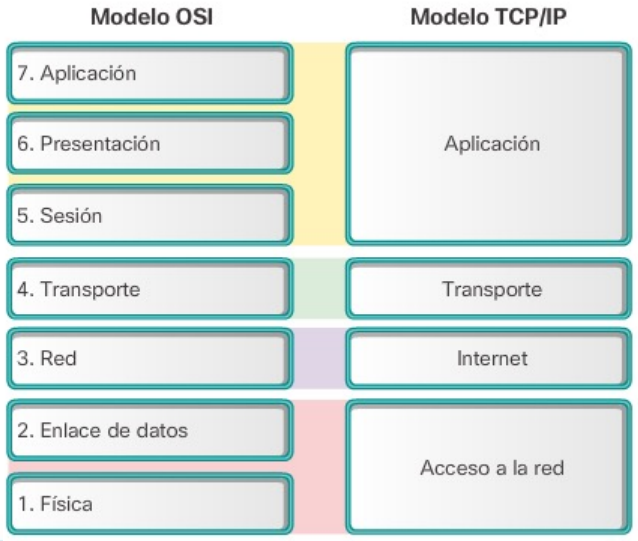
- LLC
- MAC

Funciones principales de la capa de transporte

- Recibe los datos de la capa de aplicación, los prepara para un manejo eficiente para que se entreguen de manera rápida o no.
- Hace uso de la capa de Internet para enviar los datos.
- Ofrece a los usuarios de sus servicios (usuarios o capas superiores) un transporte extremo a extremo de los datos para garantizar la comunicación entre aplicaciones.
- Se encarga de los requerimientos de las aplicaciones.

Funciones principales de la capa de transporte

- Seguimiento de comunicaciones individuales.
- Segmentación de datos y gestión de cada porción (encapsulado).
- Re ensamblaje de segmentos.
- Identificación de las diferentes aplicaciones.



Funciones principales de la capa de transporte

- El nivel de red se encarga de la entrega de paquetes individuales desde el origen al destino y no conoce ninguna relación entre los paquetes individuales, trata a cada uno de forma independiente, como si pertenecieran a mensajes diferentes.
- El nivel de transporte, asegura que el mensaje completo llega intacto y en orden, supervisando el control de errores y el control de flujo en el nivel origen a destino.
  - Igual que el nivel de enlace de datos, la capa de transporte puede ser responsable del control de flujo y de control de errores, con la diferencia de que se realiza en los extremos y no en cada enlace individual.

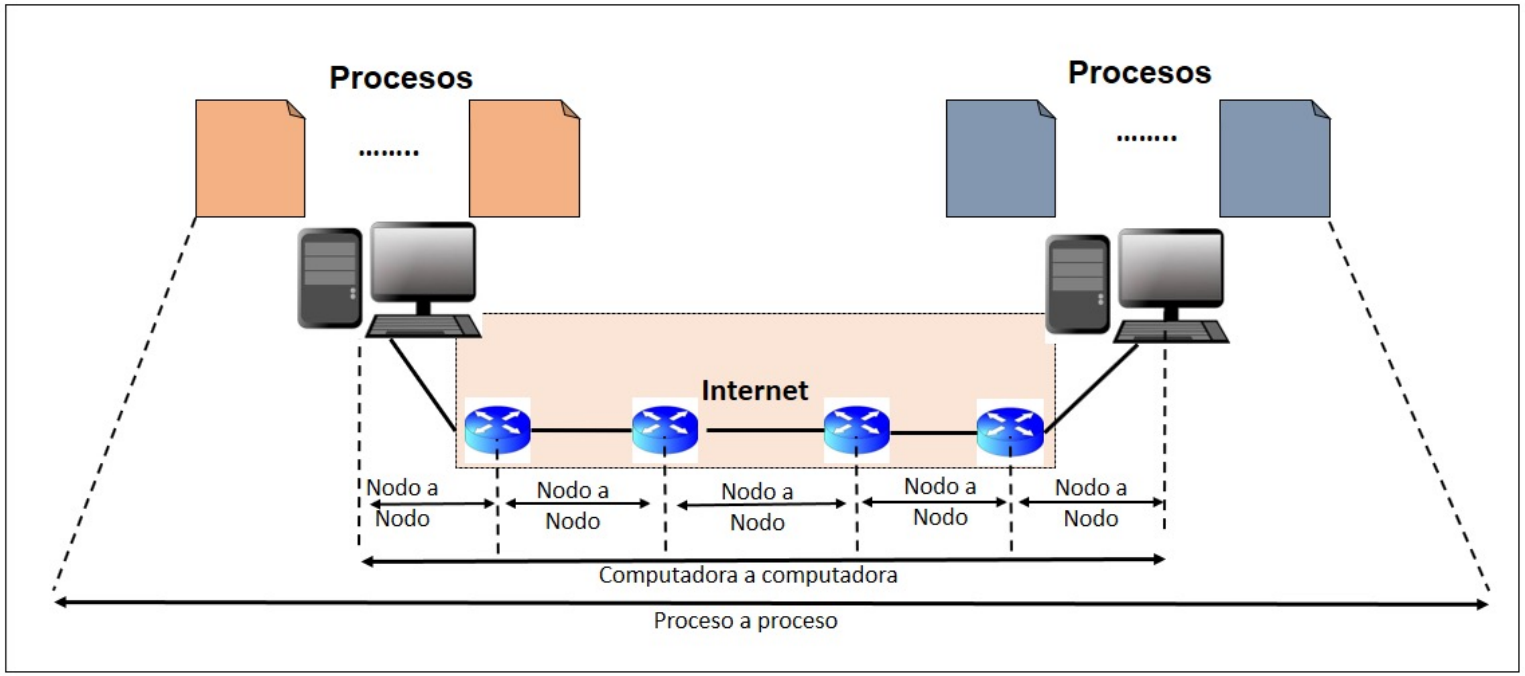
Funciones principales de la capa de transporte

- Este transporte se realiza mediante un protocolo o diálogo también extremo a extremo con la entidad homóloga de la capa de Transporte en el nodo destinatario.
- La cabecera del nivel de transporte, debe incluir un tipo de dirección denominado “dirección de punto de servicio” en OSI y “número de puerto” o “dirección de puerto” en Internet y la familia de protocolos TCP/IP.
- Si ese servicio es fiable, la capa de Transporte será responsable del establecimiento, control y liberación de las conexiones de transporte para los usuarios del servicio. Aunque, es posible dar un servicio no fiable, sin conexiones.

Comunicación proceso a proceso

- Las computadoras normalmente ejecutan varios programas a la vez, por eso la entrega origen destino, significa la entrega no sólo de una computadora a la siguiente sino también de un proceso concreto a otro.
  - Un proceso es un programa de aplicación que se ejecuta en una estación.
- El nivel de enlace es responsable de la entrega de tramas entre dos nodos vecinos en un enlace, a esto se le llama “comunicación nodo a nodo”
- El nivel de red es responsable de la entrega de datagramas entre dos computadoras, a esto se le llama “comunicación computadora a computadora”

Comunicación proceso a proceso



Tipos de comunicación de datos y sus dominios

Comunicación proceso a proceso

- La comunicación real tiene lugar entre dos procesos (programas de aplicación), por lo que se necesita una comunicación de este tipo y el nivel de transporte es el responsable de esto: la entrega de un paquete, parte de un mensaje, de un proceso a otro.

Funciones principales de la capa de transporte

Protocolo orientado a conexión

- Establece una conexión con el nivel de transporte de la máquina destino antes de entregar los paquetes. Una vez transmitidos los datos se finaliza la conexión.
- TCP y SCTP: crea una relación entre los segmentos utilizando números de secuencia.
- Utiliza ventana deslizante para el control de flujo y confirmaciones para el control de errores.

Protocolo NO orientado a conexión

- Trata a cada segmento como un paquete independiente y lo entrega al nivel de transporte en la máquina destino.
- UDP: trata cada segmento de forma independiente.
- No realiza control de errores ni de flujo.

En el nivel de transporte, un mensaje normalmente se divide en segmentos.

Sincronización en mecanismos de paso de mensajes

¿Qué es…?

Primitiva de servicio:

Un servicio está definido por un conjunto de operaciones más sencillas llamadas PRIMITIVAS.

En general, las primitivas se utilizan para realizar alguna acción o para informar de un suceso ocurrido en una entidad par.

Sincronización en mecanismos de paso de mensajes

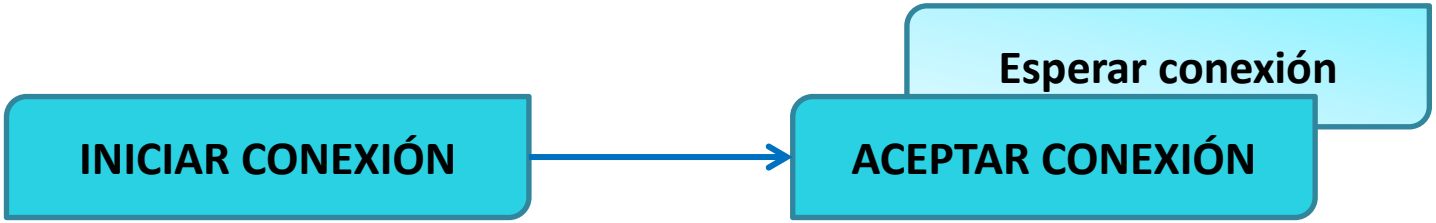
- Conceptualmente todo mecanismo de paso de mensajes contará con las siguientes primitivas básicas:
  - Enviar: proceso emisor transmite datos a un proceso receptor.
  - Recibir: proceso receptor acepta los datos de un emisor.
  - Iniciar conexión: (opcional, en sistemas orientados a conexión) un proceso indica que desea iniciar una conexión con otro
    - Proceso activo, típicamente un cliente

Sincronización en mecanismos de paso de mensajes

- Esperar conexión: (opcional, en sistemas orientados a conexión) un proceso indica que está dispuesto a recibir conexiones
  - Proceso pasivo, típicamente un servidor
- Aceptar conexión: (opcional, en sistemas orientados a conexión) un proceso acepta la comunicación con otro
  - Proceso pasivo, típicamente un servidor

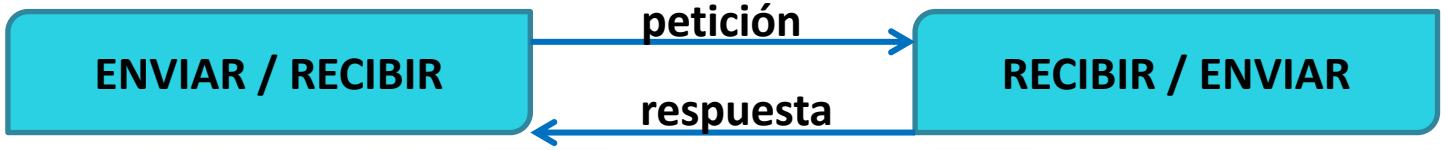
Sincronización

- Para asegurar el establecimiento de la conexión:



- Para asegurar la transferencia de un mensaje:

- Esquemas de petición+respuesta (2 mensajes):



Sincronización

- Dependiendo del modo en que se implementen las primitivas anteriores, existen diversos esquemas de sincronización entre procesos:
  - Primitivas bloqueantes
  - Primitivas no bloqueantes

→ Archivo de ejemplos

Direccionamiento de procesos

- Los sistemas operativos actuales soportan entornos multiusuario y multiproceso. Una computadora remota puede ejecutar varios programas al mismo tiempo. Para la comunicación, es necesario definir lo siguiente:
  - Computadora local
  - Proceso local
  - Computadora remota
  - Proceso remoto
- Cada vez que es necesario enviar algo a un destino específico entre muchos otros, es necesario tener una dirección.
- En el nivel de enlace se usa la dirección MAC para distinguir entre varios nodos si la conexión no es punto a punto. Una trama del nivel de enlace necesita una dirección MAC destino para su envío y una dirección MAC de origen para permitir la respuesta.

Direccionamiento de procesos

- A nivel de red se necesita una dirección IP para elegir un dispositivo entre millones. Un datagrama de nivel de red necesita una dirección IP del destino para enviar y la dirección IP de origen para la respuesta del destinatario. En Internet (TCP/IP), el host se identifica mediante su dirección IP (en el caso de IPv4, tiene magnitud de 32 bits).
- A nivel de transporte es necesario tener una dirección de nivel de transporte, denominada número de puerto, para elegir entre múltiples procesos que se ejecutan en el dispositivo destino. El número de puerto de destino, es necesario para la entrega; el número de puerto de origen es necesario para la respuesta.
- En el modelo de Internet, los números de puerto son enteros de 16 bits entre 0 y 65,535.

Direcciones de sockets

- La comunicación proceso a proceso necesita dos identificadores, dirección IP y número de puerto, en cada extremo para poder crear una conexión. La combinación de una dirección IP y un número de puerto se denomina **dirección de socket**. La dirección del socket cliente define al proceso cliente de forma única, al igual que la del socket de servidor define al proceso servidor.
- Un protocolo de nivel de transporte necesita un par de direcciones de sockets: socket cliente y socket servidor. Estas cuatro piezas de información son parte de la cabecera IP y de la cabecera del protocolo de nivel de transporte. La cabecera IP contiene las direcciones IP; la cabecera UDP o TCP contiene los números de puerto.

Direccionamiento de procesos

Multiplexación y demultiplexación

- El mecanismo de direccionamiento permite multiplexar y demultiplexar las direcciones en el nivel de transporte.
- Multiplexación: En el lado del emisor, puede haber varios procesos que necesitan enviar paquetes. Sin embargo, hay un único nivel de transporte. Existe una relación muchos a uno y necesita multiplexación. El protocolo acepta mensajes de distintos procesos diferenciados por los números de puerto que tienen asignados. Después de añadir la cabecera, el nivel de transporte pasa los paquetes al nivel de red.
- Demultiplexación: En el lado del receptor, la relación es uno a muchos y necesita demultiplexación. El nivel de transporte recibe datagramas del nivel de red. Después de comprobar si hay errores y quitar la cabecera, el nivel de transporte entrega cada mensaje al proceso apropiado basándose en el número de puerto.

Servicios de transporte disponibles para las aplicaciones

- Recordemos que un socket es la interfaz entre el proceso de la aplicación y el protocolo de la capa de transporte. La aplicación del lado del emisor envía los mensajes a través del socket. En el otro lado del socket, el protocolo de la capa de transporte tiene la responsabilidad de llevar los mensajes hasta el socket del proceso receptor.
- Muchas redes, incluyendo Internet, proporcionan más de un protocolo de transporte. Para desarrollar una aplicación, se debe elegir uno de los protocolos de la capa de transporte disponibles. ¿Cómo elegir? Seleccionando el protocolo que proporcione los servicios que mejor se adapten a las necesidades de la aplicación.
- Se pueden clasificar los posibles servicios de forma bastante general según cinco parámetros: **servicio orientado o no a conexión**, **transferencia de datos fiable**, **tasa de transferencia**, **temporización** y **seguridad**.

Servicios de transporte disponibles para las aplicaciones

- Servicio sin conexión vs servicio orientado a conexión
  - En un servicio sin conexión, los paquetes son enviados de una parte a la otra sin necesidad de establecer o liberar una conexión. Los paquetes no están numerados; pueden retrasarse, perderse o llegar fuera de orden. No hay ningún tipo de confirmación.
  - En un servicio orientado a conexión, se establece primero una conexión entre emisor y receptor. Los datos se transfieren. Al final, se libera la conexión.

Servicios de transporte disponibles para las aplicaciones

- Transferencia de datos fiable
  - En una red de computadoras pueden perderse paquetes. Por ejemplo, un paquete puede desbordar el buffer de un router, o podría ser descartado por un host o un router después de comprobar que algunos de sus bits están corrompidos.
  - En muchas aplicaciones (como el correo electrónico, la transferencia de archivos, el acceso remoto a hosts, la transferencia de documentos web y las aplicaciones financieras) la pérdida de datos puede tener consecuencias catastróficas. Por tanto, para dar soporte a estas aplicaciones, es preciso hacer algo para garantizar que los datos enviados desde un terminal de la aplicación sean todos ellos entregados correcta y completamente al otro terminal de la aplicación.

Servicios de transporte disponibles para las aplicaciones

- Transferencia de datos fiable
  - Si un protocolo proporciona un servicio de entrega de datos garantizado, se dice que proporciona una **transferencia de datos fiable**.
  - Un servicio importante que un protocolo de la capa de transporte puede potencialmente proporcionar a una aplicación es la transferencia de datos fiable proceso a proceso.
  - Cuando un protocolo de transporte suministra este servicio, el proceso emisor puede pasar sus datos al socket y saber con certidumbre absoluta que los datos llegarán sin errores al proceso receptor.

Servicios de transporte disponibles para las aplicaciones

- Transferencia de datos fiable
  - Si un protocolo de la capa de transporte no proporciona una transferencia de datos fiable, los datos enviados por el proceso emisor pueden no llegar nunca al proceso de recepción. Esto puede ser aceptable para las aplicaciones tolerantes a pérdidas; por ejemplo, la mayor parte de las aplicaciones multimedia, como las de audio/video en tiempo real, pueden tolerar que cierta cantidad de datos se pierda. En estas aplicaciones multimedia, la pérdida de datos puede dar lugar a una pequeña interrupción al reproducir el audio/video, lo que no constituye un problema fundamental.

Servicios de transporte disponibles para las aplicaciones

- Transferencia de datos fiable
  - A menudo se hacen esta pregunta. Si el nivel de enlace es fiable y tiene control de flujo y error ¿También se necesita un nivel de transporte? La respuesta es sí. La fiabilidad a nivel de enlace es entre dos nodos; se necesita fiabilidad entre ambos extremos. Debido a que el nivel de red en Internet no es fiable (envío con mayor esfuerzo), es necesario implementar la fiabilidad a nivel de transporte.

Servicios de transporte disponibles para las aplicaciones

- Tasa de transferencia
  - En el contexto de una sesión de comunicaciones entre dos procesos a lo largo de una ruta de red, es **la tasa a la que el proceso emisor puede suministrar bits al proceso de recepción**.
  - Puesto que otras sesiones compartirán el ancho de banda a lo largo de la ruta de red y puesto que esas otras sesiones se iniciarán y terminarán aleatoriamente, la tasa de transferencia disponible puede fluctuar con el tiempo.
  - Estas observaciones nos llevan de forma natural a otro servicio que un protocolo de la capa de transporte podría proporcionar: una **tasa de transferencia disponible garantizada a una cierta velocidad especificada**.

Servicios de transporte disponibles para las aplicaciones

- Tasa de transferencia
  - Con un servicio así, la aplicación podría solicitar una tasa de transferencia garantizada de n bits/segundo y el protocolo de transporte podría entonces garantizar que la tasa de transferencia disponible sea siempre al menos de n bits/segundo. Un servicio que ofreciera una tasa de transferencia garantizada como ésta, resultaría atractivo para muchas aplicaciones.
  - Por ejemplo, si una aplicación de telefonía por Internet codifica voz a 32 kbps, tendrá que enviar datos a la red y tendrá que entregar los datos a la aplicación receptora a esa velocidad. Si el protocolo de transporte no puede proporcionar esa tasa de transferencia, la aplicación tendrá que realizar la codificación a una velocidad menor (y recibir una tasa de transferencia adecuada como para mantener esa velocidad de codificación más lenta) o bien tendrá que renunciar, puesto que recibir a la mitad de la tasa de transferencia necesaria no tiene ninguna utilidad para esta aplicación de telefonía por Internet.

Servicios de transporte disponibles para las aplicaciones

- Tasa de transferencia
  - Las aplicaciones con requisitos de tasa de transferencia se conocen como **aplicaciones sensibles al ancho de banda**. Muchas aplicaciones multimedia actuales son sensibles al ancho de banda, pero algunas de ellas pueden emplear técnicas de codificación adaptativa para codificar la voz o el video digitalizados a una velocidad que se adapte a la tasa de transferencia disponible en cada momento.
  - Mientras que las aplicaciones sensibles al ancho de banda tienen requisitos específicos de tasa de transferencia, las denominadas **aplicaciones elásticas** pueden hacer uso de la tasa de transferencia, mucha o poca, que haya disponible. El correo electrónico, la transferencia de archivos y las transferencias web son todas ellas aplicaciones elásticas. Por supuesto, cuanto mayor sea la tasa de transferencia, mejor.

Servicios de transporte disponibles para las aplicaciones

- Temporización
  - Un protocolo de la capa de transporte también puede proporcionar garantías de temporización. Al igual que con las tasas de transferencia garantizadas, las garantías de temporización también pueden darse de diversas formas. Un ejemplo de garantía podría ser que cada bit que el emisor empuja por su socket llegue al socket del receptor en no más de 100 milisegundos. Un servicio así sería atractivo para las aplicaciones interactivas en tiempo real, como la telefonía por Internet, los entornos virtuales, la teleconferencia y los juegos multijugador, todas las cuales requieren restricciones de temporización muy estrictas sobre la entrega de datos para ser efectivas.

Servicios de transporte disponibles para las aplicaciones

- Temporización
  - Por ejemplo, los retardos largos en la telefonía por Internet tienden a dar lugar a pausas antinaturales en una conversación; en un juego multijugador o en un entorno virtual interactivo, un retardo largo entre la realización de una acción y la visualización de la respuesta del entorno (por ejemplo, de otro jugador que se encuentra en el otro extremo de una conexión extremo a extremo) hace que la aplicación parezca menos realista. En las aplicaciones que no se ejecutan en tiempo real, siempre es preferible un retardo pequeño que uno grande, pero no se aplican restricciones estrictas a los retardos extremo a extremo.

Servicios de transporte disponibles para las aplicaciones

- Seguridad
  - Un protocolo de transporte puede proporcionar a una aplicación uno o más servicios de seguridad. Por ejemplo, en el host emisor, un protocolo de transporte puede cifrar todos los datos transmitidos por el proceso emisor, y en el host receptor puede descifrar los datos antes de entregarlos al proceso receptor. Un servicio así proporcionaría confidencialidad entre los dos procesos, incluso aunque los datos sean observados de alguna manera entre los procesos emisor y receptor.
  - Un protocolo de transporte también puede proporcionar otros servicios de seguridad además de la confidencialidad, como pueden ser mecanismos para garantizar la integridad de datos y mecanismos de autenticación en el punto terminal.