Capítulo 2: Capa Aplicación

Capítulo 2: Capa Aplicación

- 2.1 Principios de las aplicaciones de red
- 2.2 Web y HTTP
- 2.3 FTP
- 2.4 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.5 DNS

Capítulo 2: Capa Aplicación

Objetivos:

- Aspectos conceptuales y de implementación de los protocolos de aplicación.
  - Modelo de servicio de la capa transporte.
  - Paradigma cliente-servidor.
  - Paradigma peer-to-peer (par-a-par).
- Aprendizaje de protocolos examinando protocolos de aplicación populares
  - HTTP
  - FTP
  - SMTP / POP3 / IMAP
  - DNS
- Programación de aplicaciones de red
  - API de socket

Principios de los protocolos de Capa Aplicación

- El software de una aplicación está distribuido entre dos o más sistemas.
- Ese “pedacito” de software es un proceso.
- Los procesos se comunican a través de mensajes.

- Aplicación de red <> Protocolo de capa de aplicación: el protocolo de la capa de aplicación es sólo una parte de la aplicación de red.

Algunas aplicaciones de red

- E-mail
- Web
- Mensajería instantánea
- Login remoto
- Compartición de archivos P2P
- Juegos de red multi-usuarios
- Reproducción de clips de video almacenados
- Telefonía Internet
- Conferencias de video en tiempo real
- Computación paralela masiva.

Protocolos de capa aplicación definen

- Tipos de mensajes intercambiados, e.g., mensajes de requerimiento y respuesta
- Sintaxis de los tipos de mensajes: qué campos del mensajes y cómo éstos son delimitados.
- Semántica de los campos, es decir el significado de la información en los campos
- Reglas para cuándo y cómo los procesos envían y responden a mensajes

Protocolos de dominio público:

- Definidos en RFCs
- Permite inter-operatividad
- eg, HTTP, SMTP

Protocolos propietarios:

- eg, KaZaA

Creación de una aplicación de red

Escribe un programa que

- Corra en diferentes sistemas y
- Se comunique por la red.
- e.g., Web: Programa del servidor Web se comunica con el programa del navegador

No se refiere al software escrito para los dispositivos en la red interna

- Dispositivos internos no funcionan en la capa aplicación
- Este diseño permite desarrollos rápidos

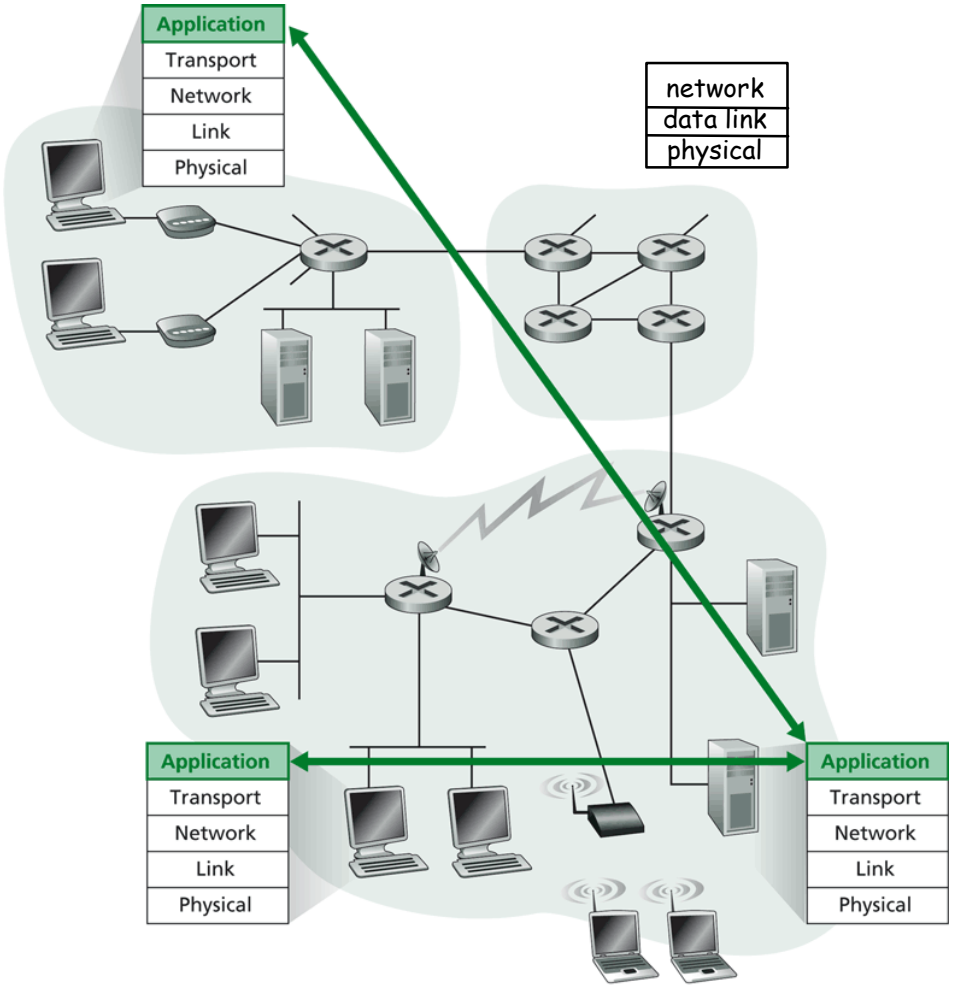
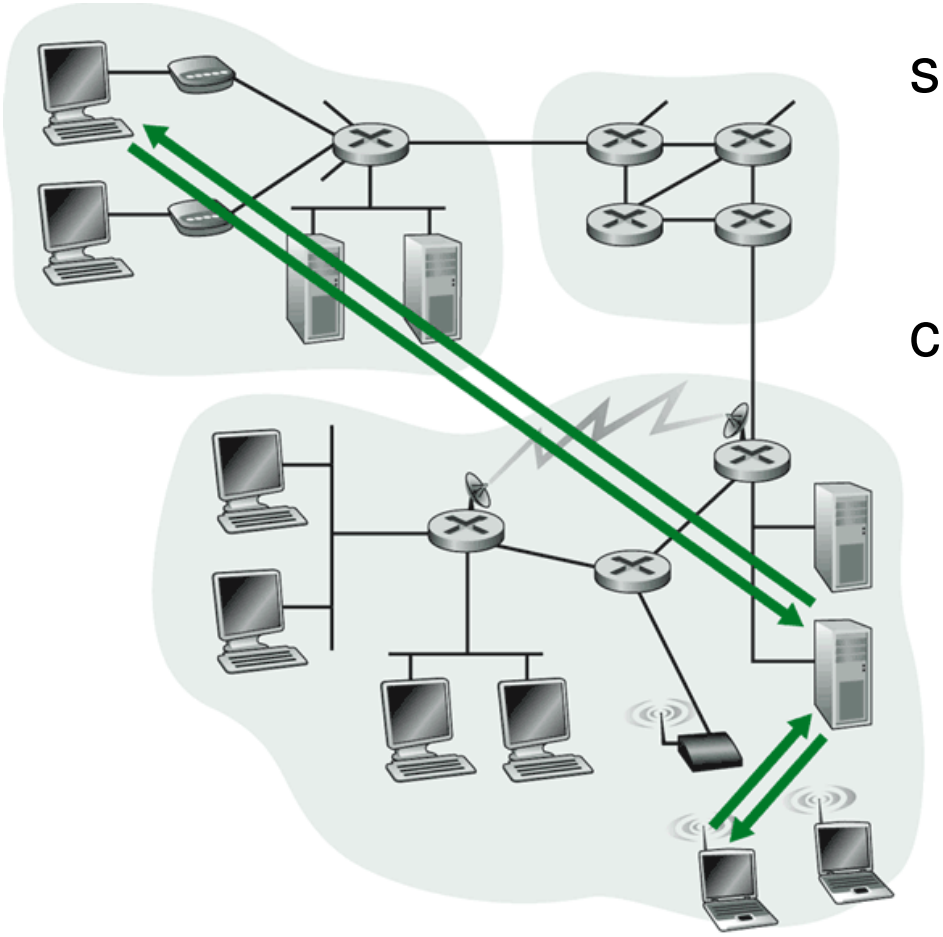


Figure 2.1 ♦ Communication for a network application takes place between end systems at the application layer.

Arquitecturas de Aplicación

- Cliente-servidor
- Peer-to-peer (P2P)
- Híbridos de cliente-servidor y P2P

Arquitectura Cliente-servidor



a. Client-server application

servidor:

- Computador siempre on
- Dirección IP permanente

cliente:

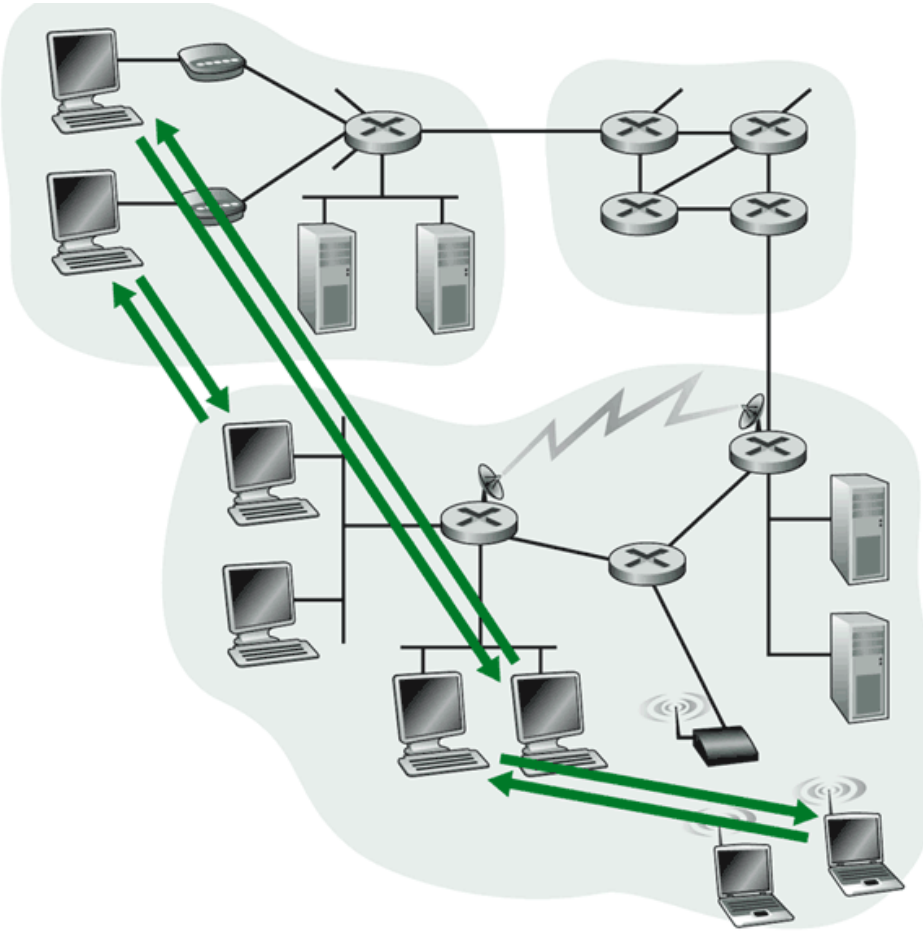
- Se comunica con un servidor
- Puede ser conectado intermitentemente
- Puede tener direcciones IP dinámicas
- No se comunican directamente entre sí (dos clientes puros)

Arquitectura P2P Pura

- Servidor no siempre on
- Sistemas terminales arbitrarios se comunican directamente
- Pares se conectan intermitentemente y cambias sus direcciones IP
- ejemplo: Gnutella

Altamente escalable

Pero difícil de administrar



b. Peer-to-peer application

Híbridos de cliente-servidor y P2P

Napster

- Transferencia de archivos P2P
- Búsqueda de archivos centralizada:
  - Pares registran contenidos en servidor central
  - Pares consultan algún servidor central para localizar el contenido

Mensajería Instantánea

- Diálogo es entre los usuarios es P2P
- Detección/localización de presencia es centralizada:
  - Usuario registra su dirección IP en un servidor central cuando ingresa al sistema
  - Usuarios contactan servidor central para encontrar las direcciones IP de sus amigos.

Procesos que se comunican

Proceso: programa que corre en una máquina.

- Dentro de la máquina dos procesos se comunican usando comunicación entre proceso (definida por OS).
- Procesos en diferentes hosts se comunican vía intercambio de mensajes

Proceso Cliente: proceso que inicia la comunicación (crea y envía mensajes sobre la red)

Proceso servidor: proceso que espera por ser contactado (recibe los mensajes y, posiblemente responde enviando mensajes)

- Nota: Aplicaciones con arquitectura P2P tienen procesos clientes y procesos servidores

Sockets

- Los procesos envían/reciben mensajes a/desde sus socket
- socket son análogos a puertas
  - Proceso transmisor saca mensajes por la puerta
  - Proceso transmisor confía en la infraestructura de transporte al otro lado de la puerta la cual lleva los mensajes al socket en el proceso receptor

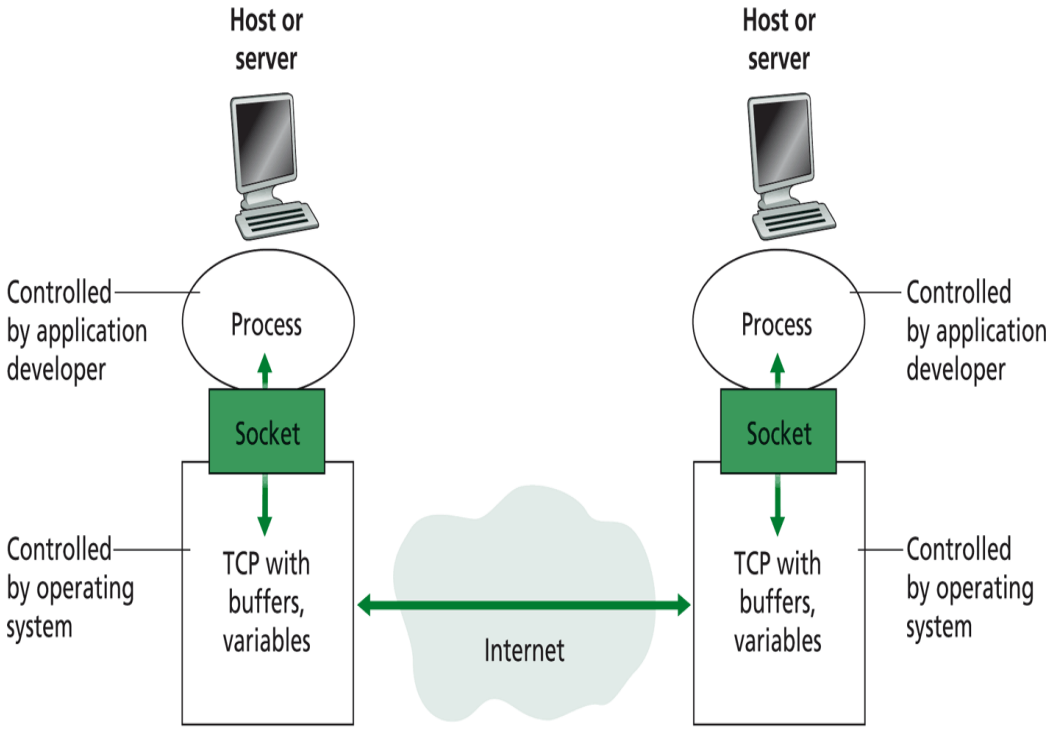


Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

- API: (1) debemos elegir el protocolo de transporte; (2) podemos definir algunos parámetros (volveremos más adelante)

Direccionamiento de procesos

- Para que un proceso reciba un mensaje, éste debe tener un identificador
- Un host tiene una dirección IP única de 32 bits.
- Q: ¿Es suficiente la dirección IP para identificar un proceso en un host?
- Respuesta: No, muchos procesos pueden estar corriendo en el mismo host.
- El identificador incluye la dirección IP y un número de puerta asociado con el proceso en el host.
- Ejemplo de números de puertas:
  - Servidor HTTP: 80
  - Servidor de Mail: 25

Capa de aplicación: Agentes de usuario

Agente de usuario

- Es una interfaz entre el usuario y la aplicación de red.
- *Por ejemplo: en la WEB, el agente de usuario es el navegador, el cual permite al usuario visualizar las páginas WEB e interactuar con los elementos de la misma. El navegador es un proceso que envía/recibe mensajes por medio de un socket y además brinda la interfaz al usuario.*

¿Qué servicios de transporte necesita una aplicación?

Pérdida de Datos

- Algunas aplicaciones (e.g., audio) pueden tolerar pérdida
- otras (e.g., transferencia de archivos, telnet) requieren transferencia 100% confiable

Retardo

- Algunas Aplicaciones (e.g., Telefonía Internet, juegos interactivos) requieren bajo retardo para ser “efectivas”

Bandwidth

- Algunas aplicaciones (e.g., multimedia) requieren cantidad mínima de ancho de banda para ser “efectivas”
- otras (“aplicaciones elásticas”) hacen uso del bandwidth que obtengan

Requerimientos de servicio de transporte de aplicaciones comunes

Aplicación	Pérdidas	Bandwidth	Sensible a Time
file transfer	no	Flexible	no
e-mail	no	Flexible	no
Web documents	no	Flexible	no
real-time audio/video	tolerante	audio: 5kbps-1Mbps video:10kbps-5Mbps	si, 100's msec
stored audio/video	tolerante	Igual al de arriba	si, pocos secs
interactive games	tolerante	Pocos Kbps	si, 100's msec
instant messaging	no	flexible	Si y no

Servicios de los protocolos de transporte en Internet

Servicio TCP:

- *Orientado a la conexión* acuerdo requerido entre procesos cliente y servidor antes de transferencia
- *Transporte confiable* entre proceso Tx y Rx
- *Control de flujo:* Tx no sobrecargará al Rx
- *Control de congestión:* frena al Tx cuando la red está sobrecargada
- *No provee:* garantías de retardo ni ancho de banda mínimos

Servicio UDP:

- Transferencia de datos no confiable entre proceso Tx y Rx.
- No provee: acuerdo entre los procesos, confiabilidad, control de flujo, control de congestión, garantías de retardo o ancho de banda

Q: ¿Por qué molestarse? ¿Por qué existe UDP?

Aplicaciones Internet: aplicación, protocolo de transporte

Aplicación	Protocolo capa aplicación	Protocolo de transporte que lo sustenta
e-mail	SMTP [RFC 2821] Telnet [RFC 854]	TCP
remote terminal access	HTTP [RFC 2616]	TCP
Web	FTP [RFC 959]	TCP
file transfer	proprietary	TCP
streaming multimedia	(e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Dialpad)	typically UDP

Capítulo 2: Capa Aplicación

- 2.1 Principios de las aplicaciones de red
- 2.2 Web y HTTP
- 2.3 FTP
- 2.4 Correo Electrónico
  - SMTP, POP3, IMAP
- 2.5 DNS