

# Capítulo 2: Capa Aplicación - I

ELO322: Redes de Computadores

Agustín J. González

Este material está basado en:

- Material de apoyo al texto *Computer Networking: A Top Down Approach Featuring the Internet 3rd edition*. Jim Kurose, Keith Ross Addison-Wesley, 2004.
- Material del curso anterior ELO322 del Prof. Tomás Arredondo V.

# 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
- ❑ 2.6 P2P para archivos compartidos
- ❑ 2.7 Programación de sockets con TCP
- ❑ 2.8 Programación de sockets con UDP
- ❑ 2.9 Construcción de un servidor WEB

# 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 sockets

# 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 (VoIP)
- ❑ Conferencias de video en  
tiempo real
- ❑ Computación paralela  
masiva.

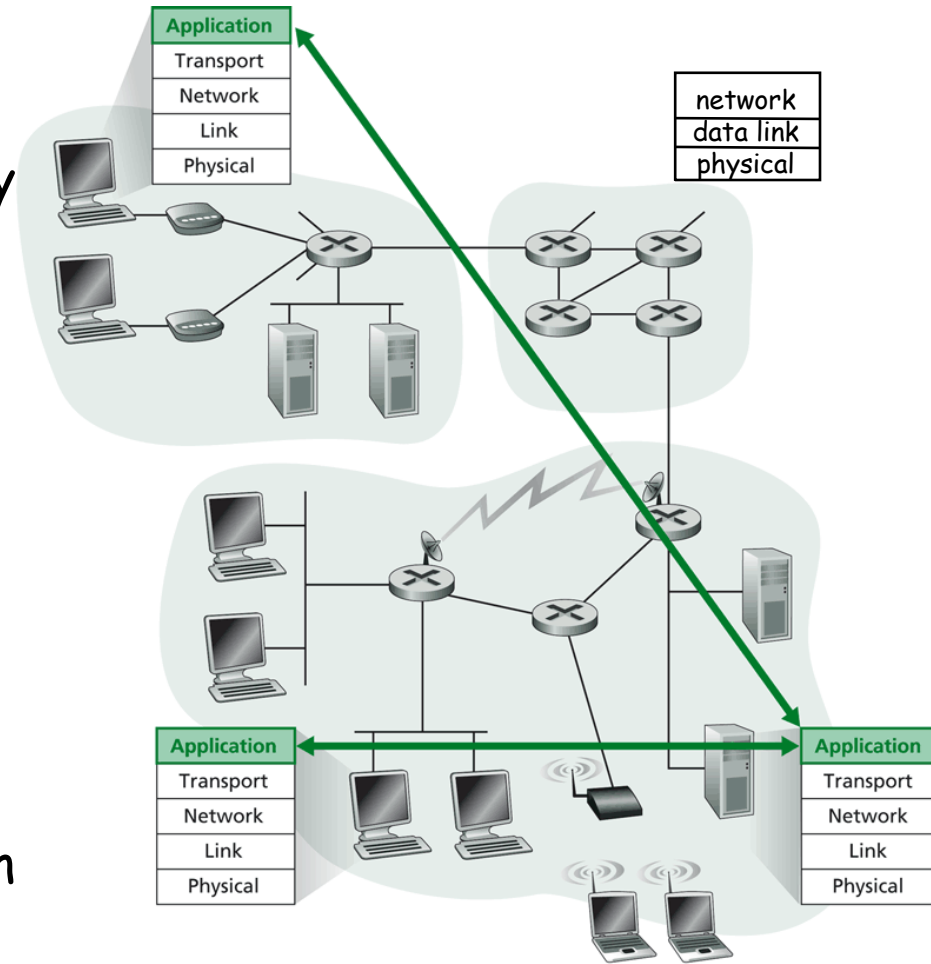
# Creación de una aplicación de red

## Aplicaciones de la red

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

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

- Dispositivos internos de la red (routers, switches) 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.

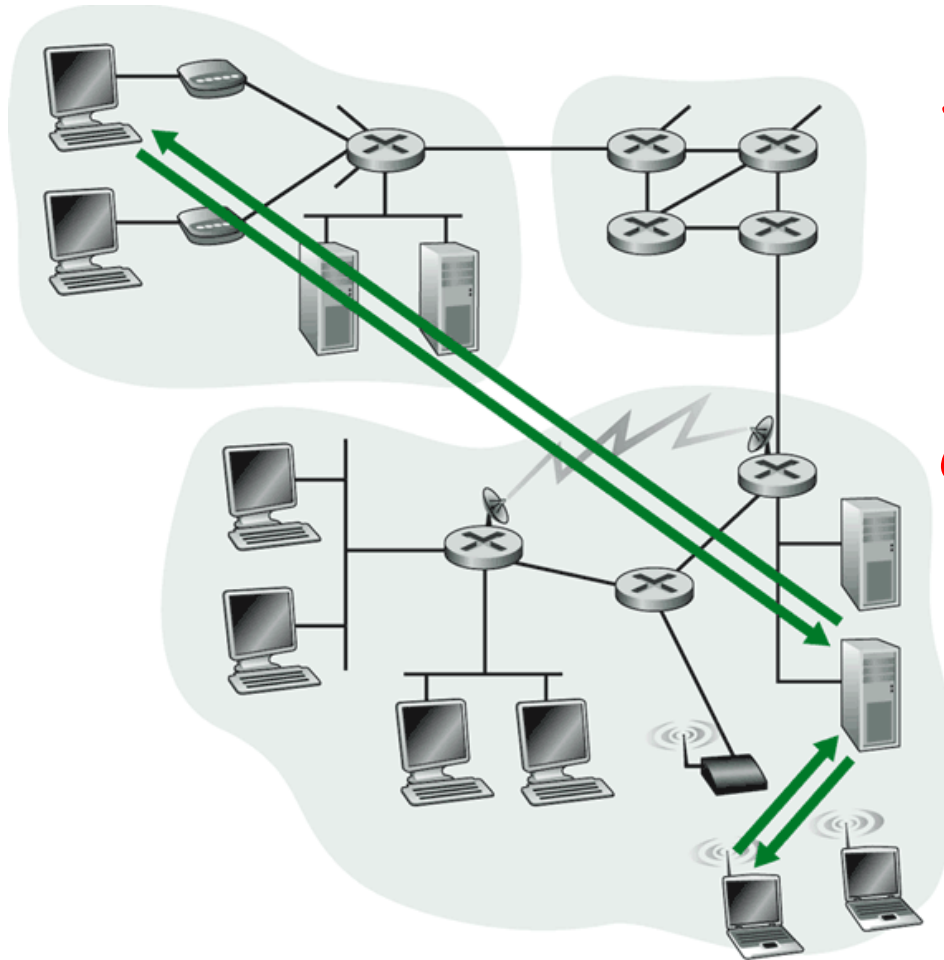
# Capítulo 2: Capa Aplicación

- ❑ 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
- ❑ 2.6 P2P Compartición de archivos
- ❑ 2.7 Programación de socket con TCP
- ❑ 2.8 Programación de socket con UDP
- ❑ 2.9 Construcción de un servidor WEB

# 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
- Granja de servidores por escalamiento

## Cliente:

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

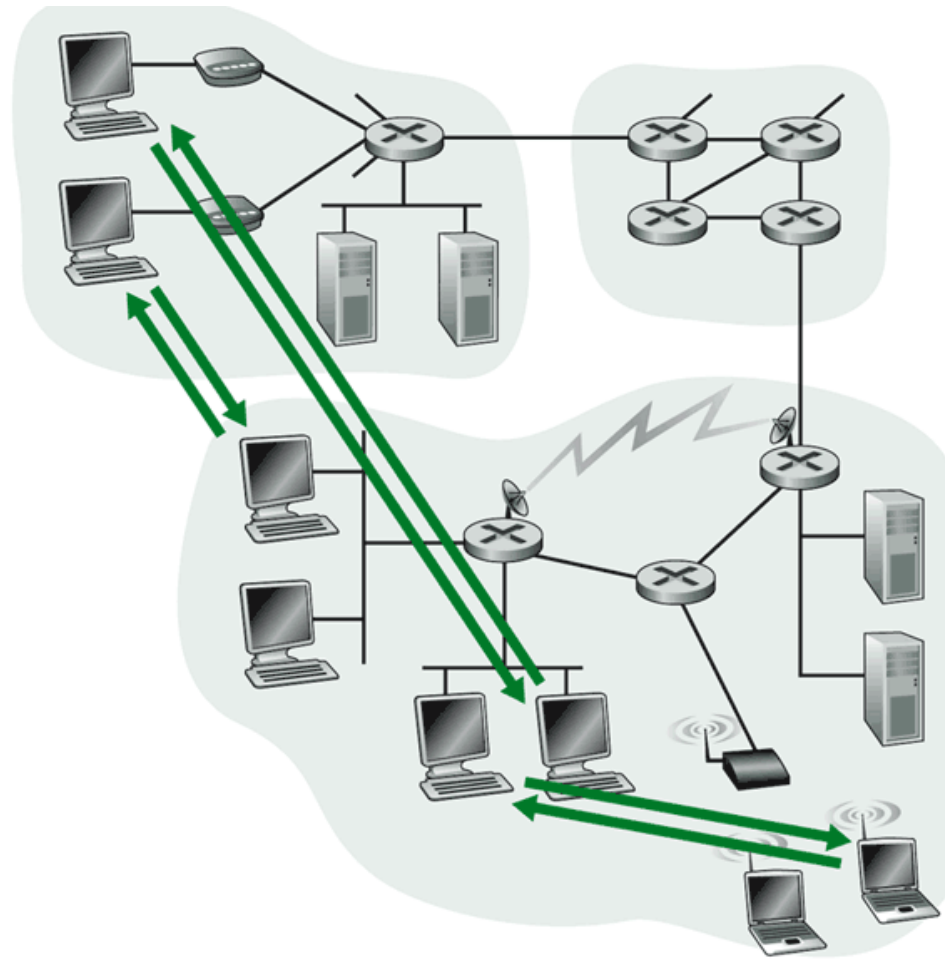


# Arquitectura P2P Pura

- ❑ No hay servidor siempre on
- ❑ Sistemas terminales arbitrarios se comunican directamente
- ❑ Pares se conectan intermitentemente y cambian 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 un computador.

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

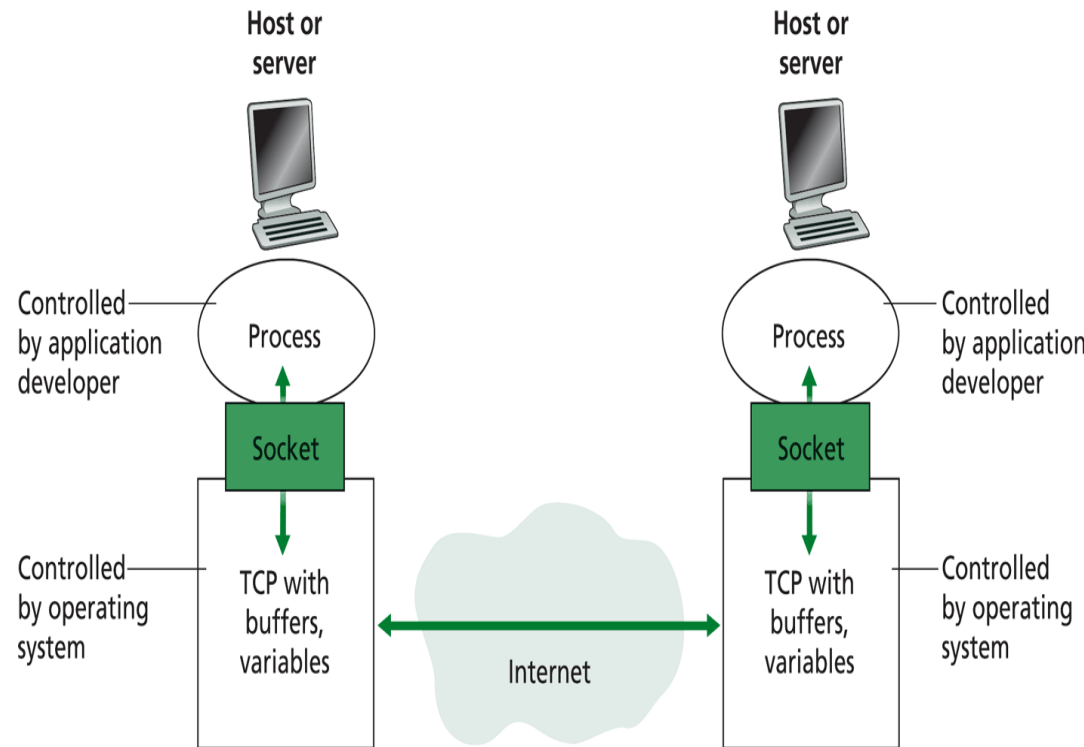
**Proceso Cliente:** proceso que inicia la comunicación

**Proceso servidor:** proceso que espera a ser contactado

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

# Sockets

- Un proceso envía/recibe mensajes a/desde su **socket**
- socket es análogo a una puerta (punto de comunicación entre dos partes)
  - Proceso transmisor envía 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
- API: (1) debemos elegir el protocolo de transporte; (2) podemos definir algunos parámetros (**volveremos más adelante**)



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

# Direccionamiento de procesos

- ❑ Para que un proceso reciba un mensaje, éste debe tener un identificador
- ❑ Un terminal/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 (= computador).
- ❑ El identificador incluye la dirección IP y un **número de puerto (port)** asociado con el proceso en el host.
- ❑ Ejemplo de números de puerto (port number):
  - Servidor HTTP: 80
  - Servidor de Mail: 25

# Protocolos de capa aplicación definen

- ❑ Tipos de mensajes intercambiados, e.g., mensajes de requerimiento y respuesta
- ❑ Sintaxis de los tipos de mensajes: los campos en los mensajes & cómo éstos son delimitados.
- ❑ Semántica de los campos, i.e., 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,
- ❑ ¿skype?

# ¿Qué servicios de la capa transporte necesita una aplicación?

## Confiabilidad en la entrega (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 en internet, juegos interactivos) requieren bajo retardo para ser "efectivas"

## Ancho banda (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 servicios de transporte de aplicaciones comunes

<u>Aplicación</u>	<u>Pérdidas</u>	<u>Bandwidth</u>	<u>Sensible a Tiempo</u>
file transfer	no	elastic	no
e-mail	no	elastic	no
Web documents	no	elastic	no
real-time audio/video	tolerante	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	tolerante	Igual al de arriba	yes, few secs
interactive games	tolerante	few kbps up	yes, 100's msec
instant messaging	no	elastic	yes and no



# Servicios de los protocolos de transporte en Internet

## Servicio TCP:

- ❑ *Es Orientado a la conexión* establecer conexión (setup) requerido entre procesos cliente y servidor antes de transferencia
- ❑ *Ofrece Transporte confiable* entre proceso Transmisor (Tx) y Receptor (Rx)
- ❑ *Tiene Control de flujo:* Tx no sobrecargará al Rx
- ❑ *Tiene 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: establecimiento conexión, confiabilidad, control de flujo, control de congestión, garantías de retardo o ancho de banda

Q: ¿Por qué existe UDP?

# Aplicaciones Internet: aplicación, protocolo de transporte

<b>Aplicación</b>	<b>Protocolo capa aplicación</b>	<b>Protocolo de transporte que lo sustenta</b>
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (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
- ❑ 2.6 P2P Compartición de archivos
- ❑ 2.7 Programación de socket con TCP
- ❑ 2.8 Programación de socket con UDP
- ❑ 2.9 Construcción de un servidor WEB