Capa Aplicación: P2P

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.

Capítulo 2: Capa Aplicación

- 2.1 Principios de la 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 sockets con TCP
- 2.8 Programación de sockets con UDP
- 2.9 Construcción de un servidor WEB

P2P file sharing (compartición de Archivos)

Ejemplo

- Alice ejecuta una aplicación cliente en su notebook
- Intermitentemente se conecta al Internet; recibe una nueva dirección IP en cada conexión
- Pide canción “Hey Jude”
- Aplicación muestra otros pares que tienen una copia de “Hey Jude”.
- Alice elige a uno de los pares, Pedro
- Archivo es copiado del PC de Pedro al notebook de Alice protocolo: HTTP
- Mientras que Alice lo baja, otros usuarios bajan música desde el notebook de Alice.
- El notebook de Alice es un cliente Web y también temporalmente un servidor Web.
- Todos los pares puedes ser servidores => altamente escalable!

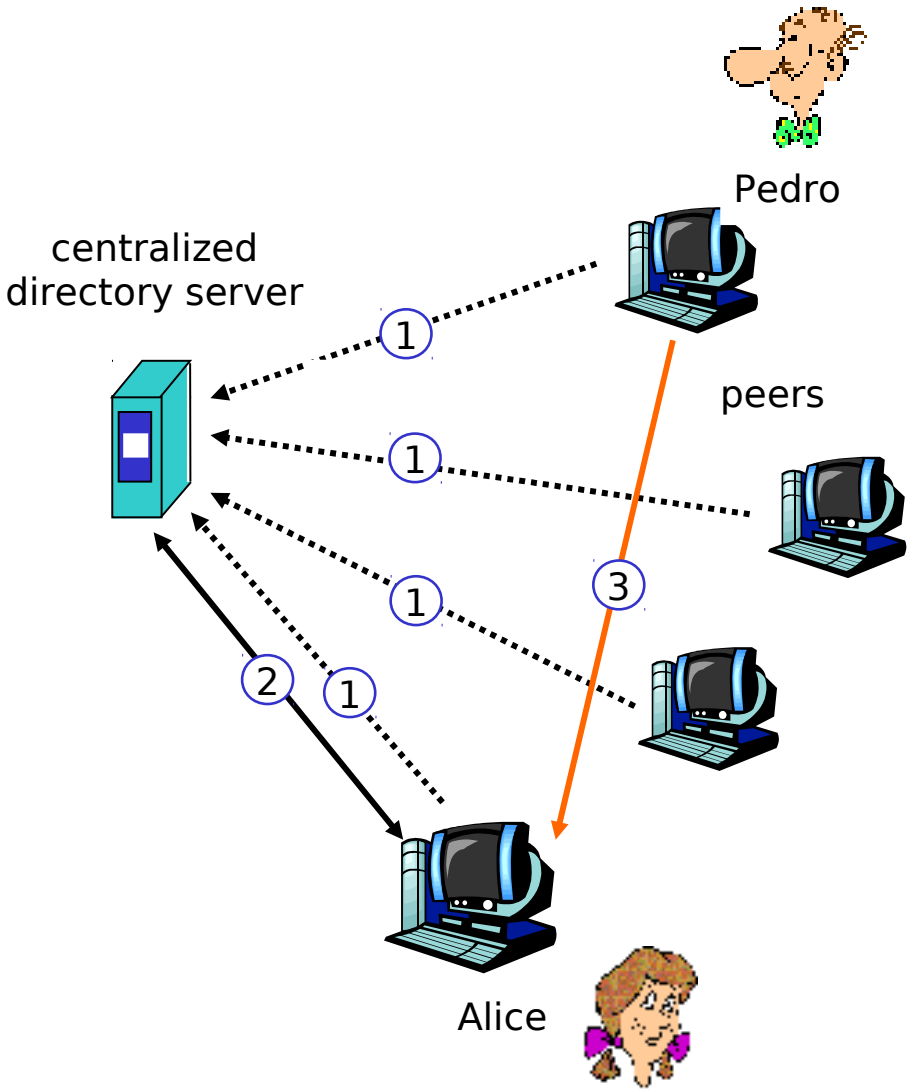
P2P: directorio centralizado

Diseño original de "Napster"

1) Cuando un terminal inicia napster, él informa a un servidor central:
   - dirección IP
   - música que tiene

2) Alice pregunta por "Hey Jude", se entera lo tiene Pedro

3) Alice pide luego el archivo a Pedro directamente



P2P: problemas con directorio centralizado

- Punto individual de falla
- Cuello de botella a la capacidad (performance)
- Problemas legales con música (Copyright infringement)

La transferencia de archivos es descentralizada pero la localización de contenido (archivos) es altamente centralizado

Inundación de preguntas (Query flooding): Gnutella

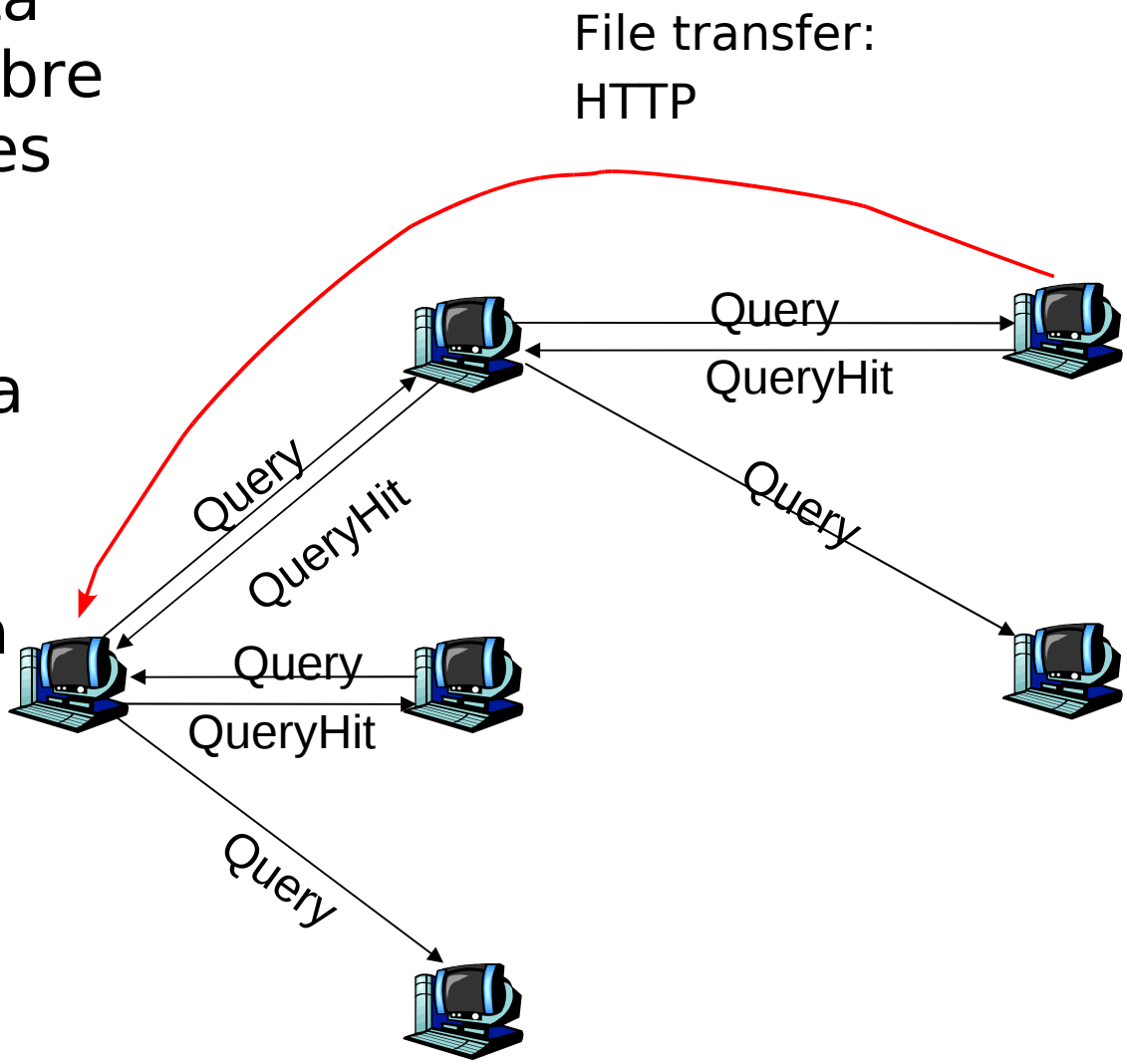
- Completamente distribuido
  - sin servidor central
- Protocolo de dominio público
- Muchos clientes Gnutella implementan el protocolo

Red sobrepuesta: grafo

- Hay enlace entre pares X e Y si hay una conexión TCP
- Todos los pares activos y sus enlaces forman la red sobrepuesta (overlay net)
- Cada enlace no es un enlace físico sino conceptual
- Un par típicamente va a estar conectado a < 10 vecinos en su red sobrepuesta

Gnutella: protocolo

- Mensaje de pregunta (Query) mandado sobre conexiones existentes TCP
- Pares reenvían mensaje de pregunta (Query message)
- Resultado positivo (QueryHit) se manda por ruta reversa



Escalable:

inundación de mensajes limitada

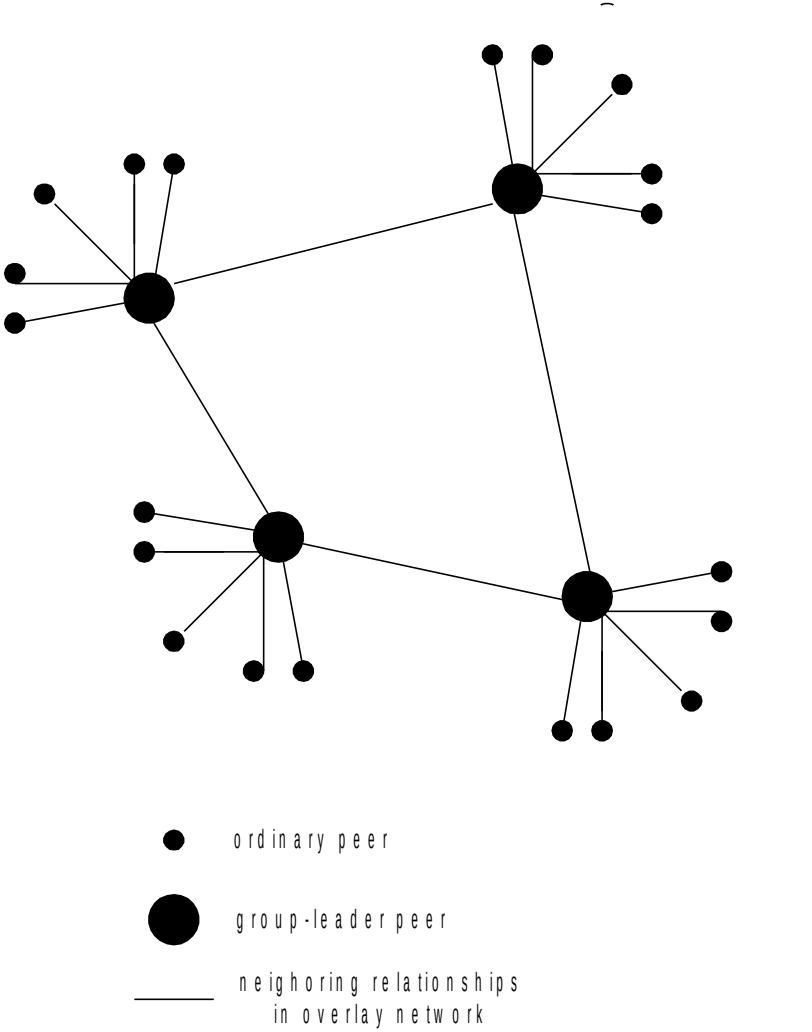
Gnutella: Conectarse a Pares

1. Nodo X debe encontrar otro par en la red Gnutella: usa lista de pares candidatos
2. X trata secuencialmente de conectarse vía TCP con pares en su lista hasta hacer una conexión con Y
3. X manda mensaje Ping a Y; Y reenvía mensaje Ping
4. Todos los pares que reciben el mensaje Ping responden con mensaje Pong

- X recibe muchos mensajes Pong. Ahora él puede establecer conexiones TCP adicionales.

Explotando heterogeneidad: KaZaA

- Protocolo no público
- Cada nodo es un líder de grupo o asignado a un líder de grupo
  - Conexión TCP entre nodo y líder de grupo
  - Conexiones TCP entre pares de lideres de grupo
- Líder de grupo sabe los contenidos (archivos) de todos sus hijos



KaZaA: Búsquedas

- Cada archivo tiene un hash y un descriptor (incluye el nombre del archivo y descripción en texto del objeto)
- Cliente manda una pregunta usando palabras claves a su líder de grupo (él busca en el descriptor)
- Líder de grupo responde con aciertos:
  - Para cada acierto: metadatos, hash, direccion IP
- Si un líder de grupo reenvía la búsqueda a otros lideres de grupo, esos lideres contestan con aciertos (usando ruta inversa red sobrepuesta)
- Cliente selecciona archivos para bajar
  - Mensajes HTTP usando hash como identificador son mandados a pares que contienen archivo deseado

Trucos KaZaA

- Limitación para subidas (uploads) (y downloads?) simultaneas
- Encolamiento de peticiones
- Prioridades basadas en incentivos a mejores usuarios (los que suben más archivos a la red)
- Bajada de datos para un archivo en paralelo (puede usar múltiples conexiones HTTP a diferentes pares para el mismo archivo)

Capítulo 2: Capa Aplicación

- 2.1 Principios de la 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 (Lo saltaremos)
- 2.7 Programación de sockets con TCP
- 2.8 Programación de sockets con UDP
- 2.9 Construcción de un servidor WEB

Construyendo un servidor Web simple

- Maneja una petición HTTP
- Acepta la petición
- Analiza cabecera (parses header)
- Obtiene archivo pedido de su sistema de archivos (file system)
- Crea mensaje HTTP de respuesta:
  - líneas cabecera + archivo
- Manda respuesta al cliente

- Después de crear este servidor, tu le puedes pedir un archivo usando un browser (eg Mozilla, Netscape o IE explorer)
- Este es un ejercicio principalmente de programación.
- Ver texto guía para más detalles

Resumen de Capa aplicación

Hemos cubierto varias aplicaciones de red

- Arquitectura de la aplicaciones
  - cliente-servidor
  - P2P
  - híbridos
- Servicios requeridos por aplicaciones:
  - confiabilidad, ancho de banda, retardo
- Modelo de servicio de transporte en Internet
  - Confiable y orientada a la conexión: TCP
  - No confiable, datagramas: UDP
- Protocolos específicos:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
- Programación de sockets
- Un servidor web simple (ver texto)

Resumen de Capa aplicación

Lo más importante aprendido sobre *protocolos*

- Intercambio de mensajes típicos requerimiento/respuesta:
  - cliente requiere info o servicio
  - servidor responde con datos, código de estatus
- Formato de mensajes:
  - encabezado: campos dando info sobre datos
  - datos: info siendo comunicada
- Mensajes de control vs. datos
  - in-band, out-of-band
- Centralizado vs. descentralizado
- Sin estado vs. con estado
- Transferencia confiable vs. Transferencia no confiable
- “la complejidad es puesta en los bordes de la red (las aplicaciones)” Distinto a sistema telefónico clásico.