# Chapter 2: Application layer

- 2.1 Principles of network applications
  - app architectures
  - app requirements
- 2.2 Web and HTTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

# So is it Peer to Peer or P2P???

**Peer to Peer**

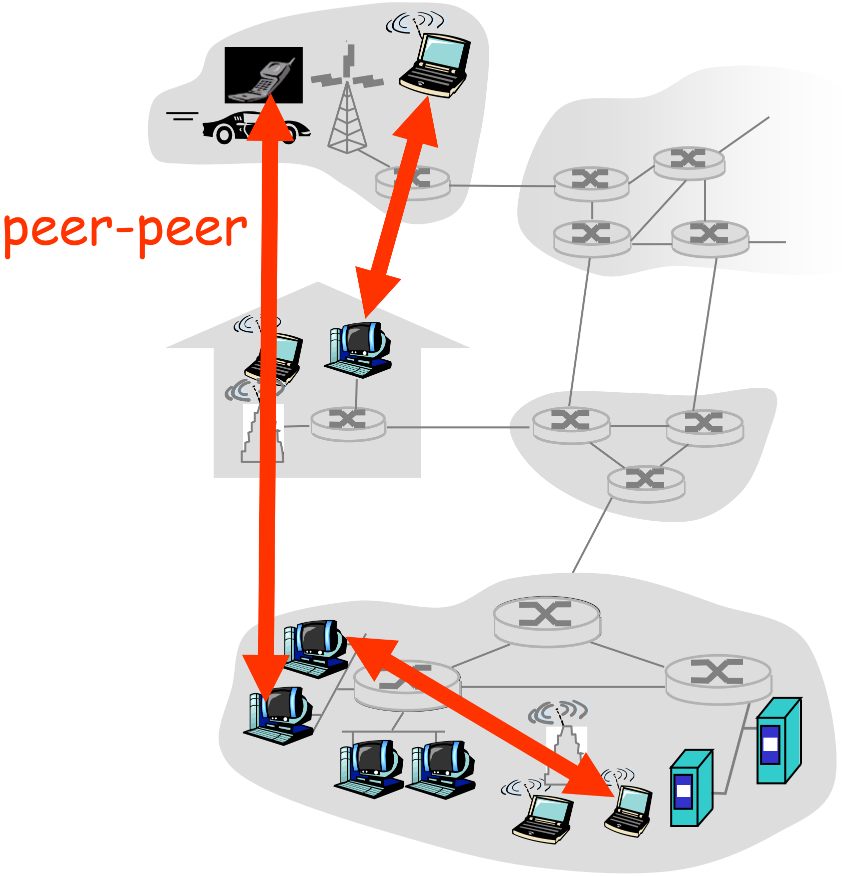
- Done typically on a LAN
- Uses software components built into the OS
- Protocol = TCP/IP
- Legally share devices and files

**P2P**

- Done typically on the internet
- Must install special 3rd party software (ares, trustyfiles, bearshare)
- Additional protocols (BitTorrent, Ants P2P, eDonkey)
- Pirated software, music and movies

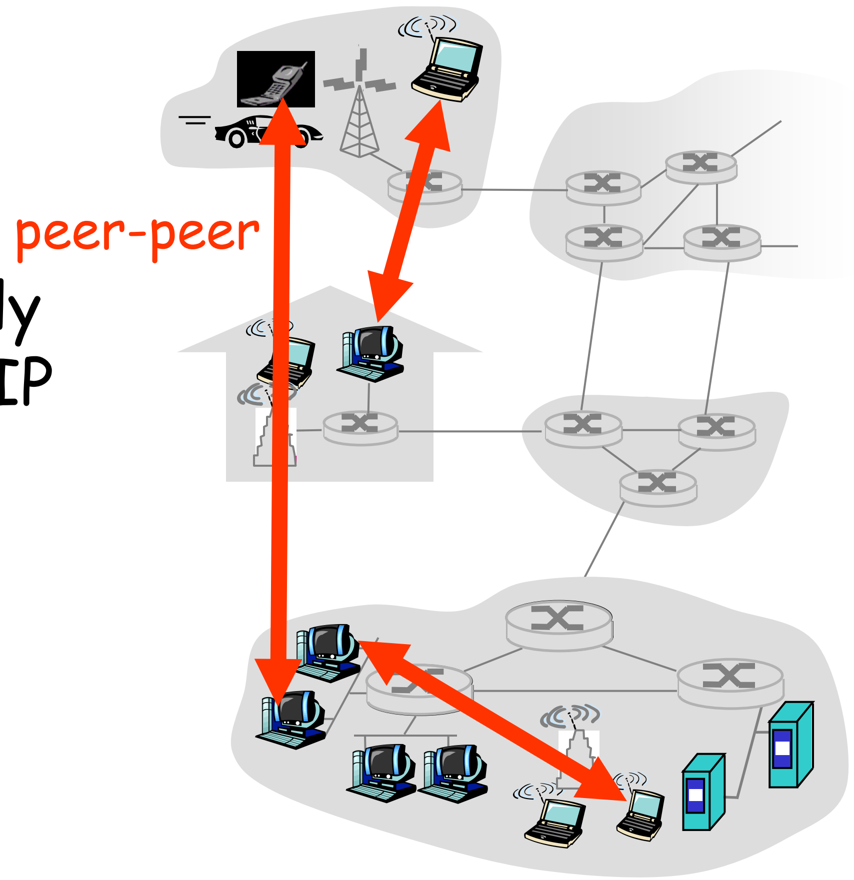
# Pure P2P architecture

- LAN model
- OS: Mac, Vista, XP, Linux
- Wireless or Wire
- Environments: Home, Office
- GUID: Unique Identification

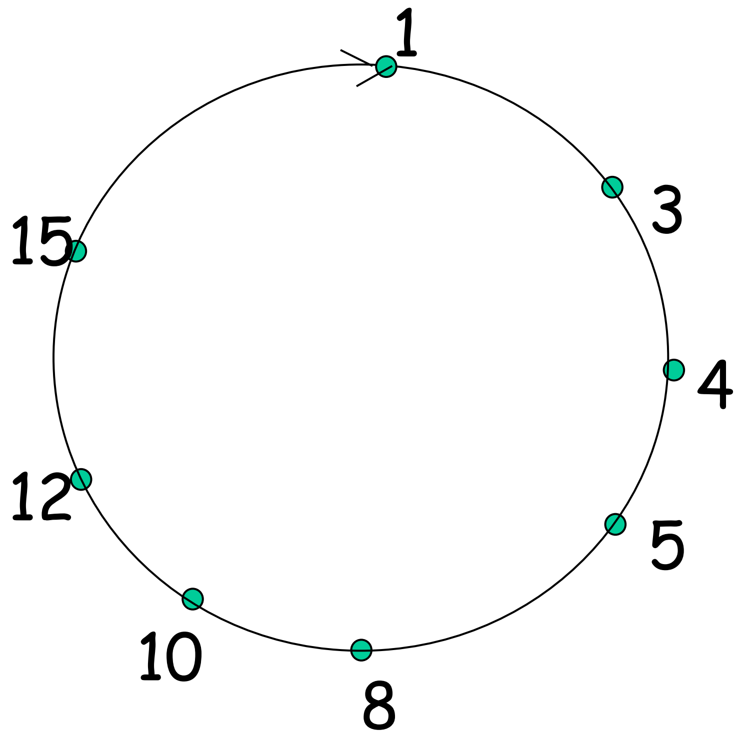


# Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses



# Peer Arrangement




- Only maintain reference to a subset of the peers (exponentially further and further away from them)

# Advantages of Peer-to-Peer

**Simple**

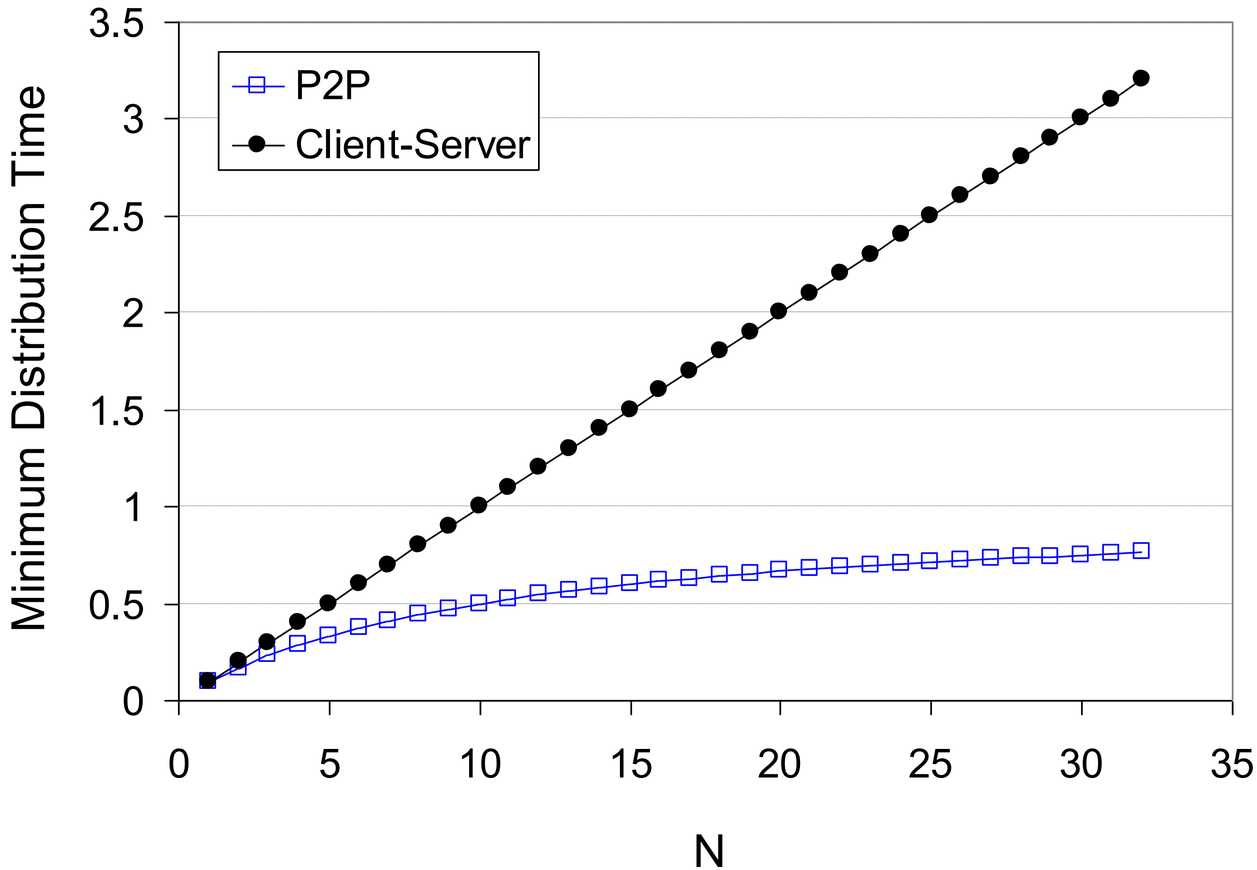
- For users to understand once setup
- Small companies do not have to hire IT staff
- Practical for small business and home offices (SOHOs)

**Low Cost**

- Media needed: hub or switch, NIC, cables
- Wireless technology
- Networking software comes with OS

# Server-client vs. P2P: example

Client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$



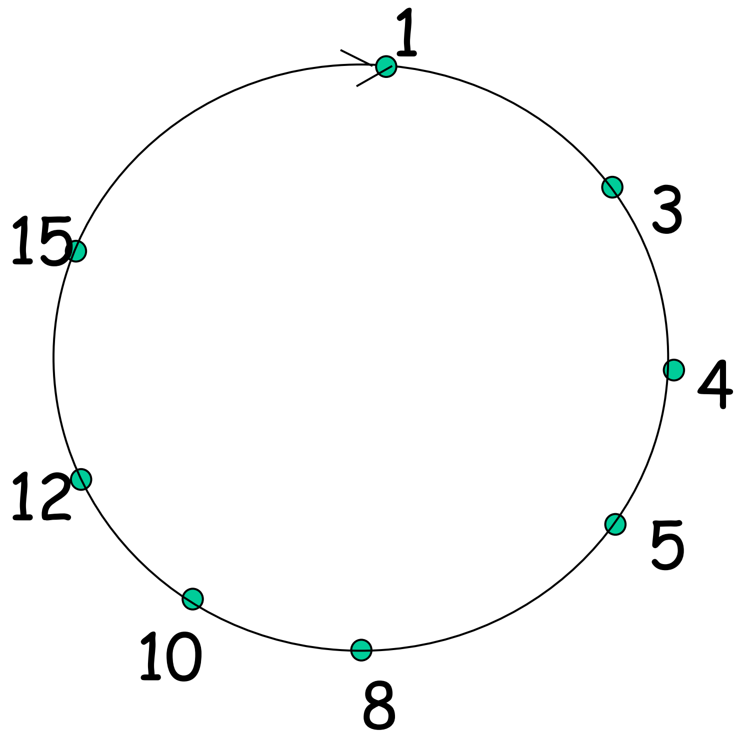
# Client/ Server Based Network?

- Networking using special computers known as file servers, to process data for and facilitate communication between other computers on the network
  - File Server: manages shared resources, uses special system software designed to manage data and other resources
  - Client/Workstation: requests another computer on the network

# Advantages of Client/Server

- Security
- Centralized user accounts
- Access to multiple shared resources can be centrally granted
- Optimized to handle heavy processing loads and dedicated to handling requests from clients
- Can scale to thousands of computers
- Higher license fees
- Back ups data only required only at the server

# Peer to Peer Algorithms



# Components for Peer-to-Peer

- Hardware
  - NIC, modem, USB port
  - Cables or phone line, electrical outlet, wireless transmitter and receiver, USB cables and hub
  - Wireless home routers or switch (if just two computers you need only a crossover cable)
- Software
  - Windows XP/Vista
  - Linux/Mac OS with SMB services

# Software Pieces to Vista/XP

- **Client:** Software installed allowing your workstation to view across the network into a device/resource/object on the network, access data on other systems e.g. Client for Microsoft networks (PCs only), client NFS (interoperability)
- **Protocol:** the language of network communication, TCP/IP
- **Services:** additional network software to provide network monitoring, QoS, remote backup, server services, virtualization etc (File and Print Service sharing)
- **NIC Driver:** Software to interact with NIC

# Services

Advanced functions for a complex environment

- **Server services (file & print services):** found in all operating systems allows the sharing of files, folders, drivers, applications and printers on network
- **Backup agents:** allows a server based backup system to remotely backup the computer (*third party*)
- **Network Access Protection agent:** force clients to meet corporate policies related to required updates-patches, current anti-virus software (*third party*)
- **QoS:** Gives priority to TCP/IP packets determined by the network administrator, VoIP

# What can we share?

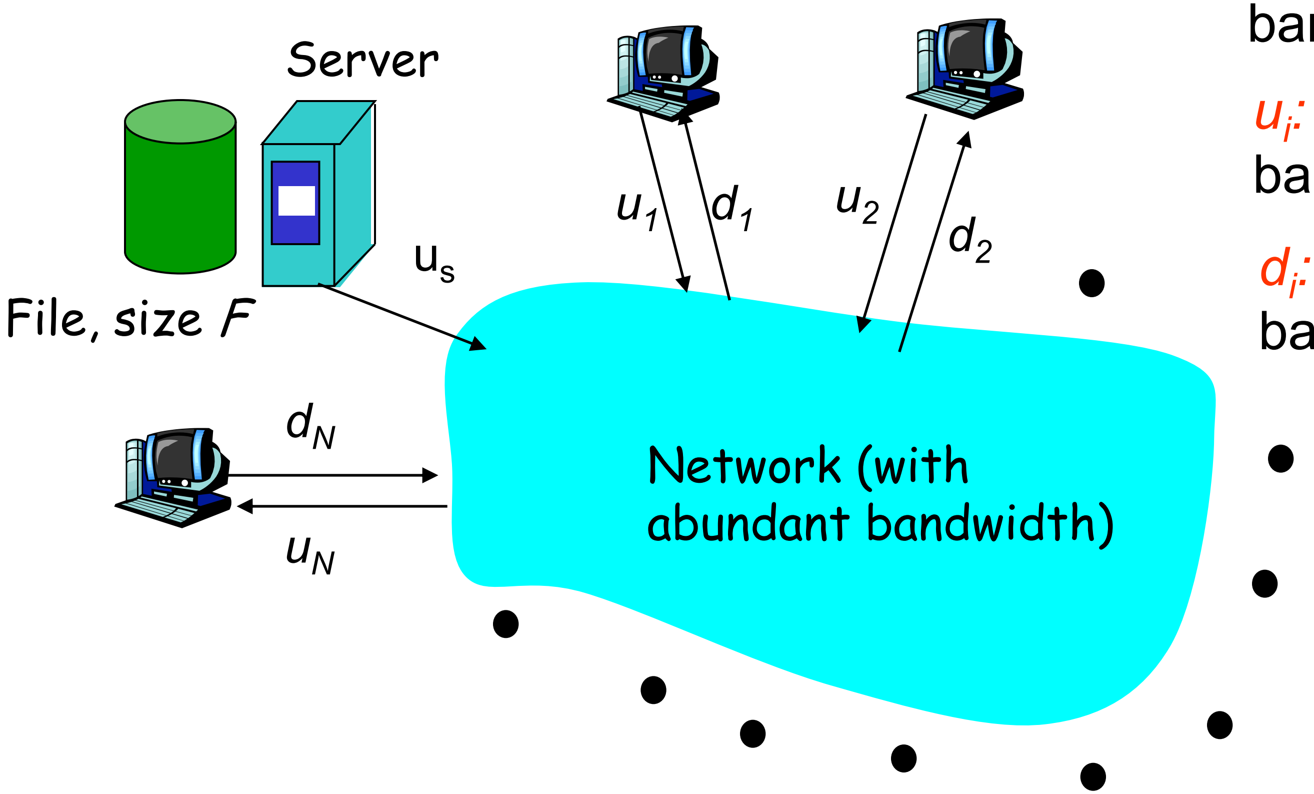
- Entire drivers: C:\, D:\ (bad security risk)
- Just a folder or directory
- Printers
- USB flash drive
- Software and applications

# Share Level Security

- Add the same accounts and passwords to all the computers on your network
- Each member can access the shared folder of the other computer, regardless of which computer they are using

# File Distribution: Server-Client vs P2P

*Question* : How much time to distribute file from one server to $N$ *peers*?

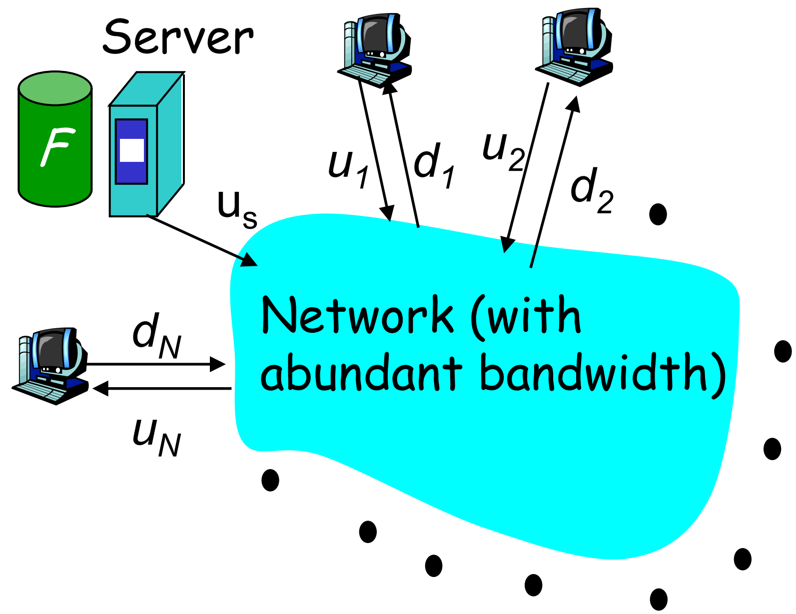



$u_s$: server upload bandwidth

$u_i$: peer i upload bandwidth

$d_i$: peer i download bandwidth

# File distribution time: server-client



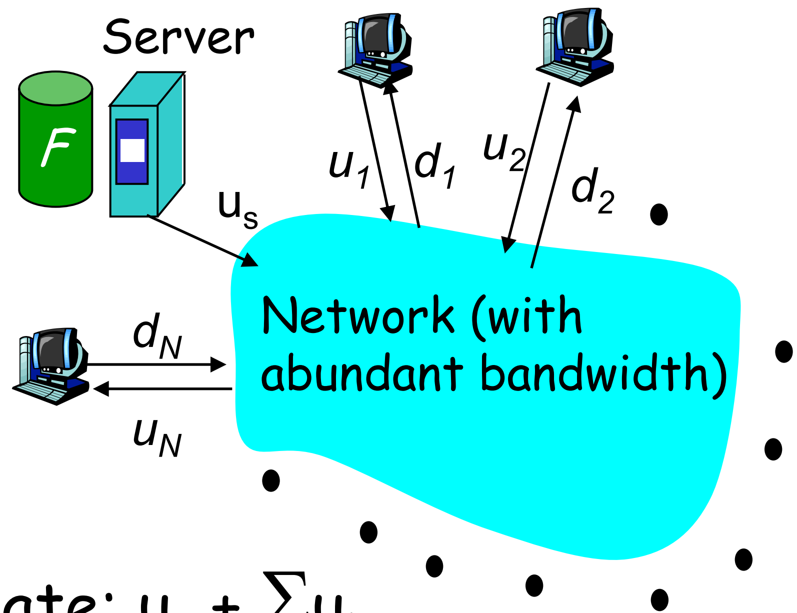

- server sequentially sends N copies:
  - $NF/u_s$ time
- client i takes $F/d_i$ time to download

Time to distribute $F$ to $N$ clients using client/server approach $= d_{cs} = \max_i \{ NF/u_s, F/\min(d_i) \}$

increases linearly in N (for large N)

# File distribution time: P2P

- server must send one copy: $F/u_s$ time
- client i takes $F/d_i$ time to download
- NF bits must be downloaded (aggregate)
    - fastest possible upload rate: $u_s + \Sigma u_i$



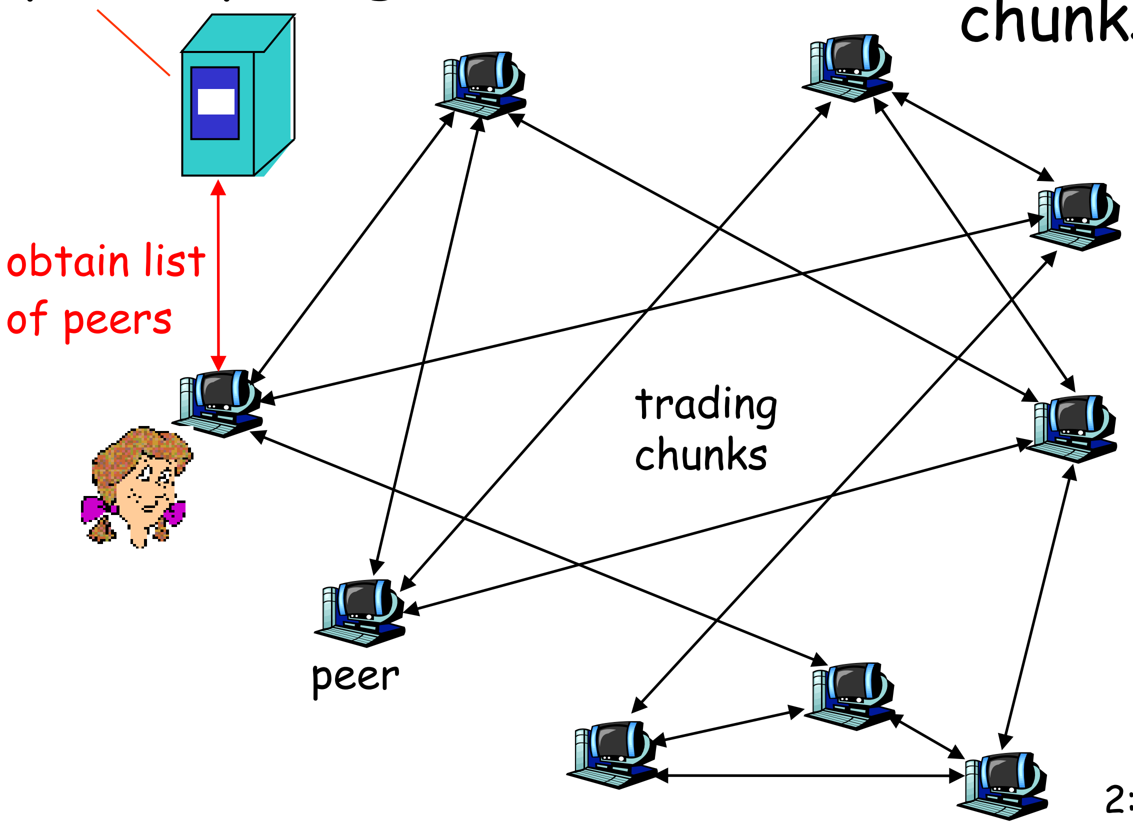

$$d_{P2P} = \max \{ F/u_s,\ F/\min_i(d_i),\ NF/(u_s + \Sigma u_i) \}$$

# File distribution: BitTorrent

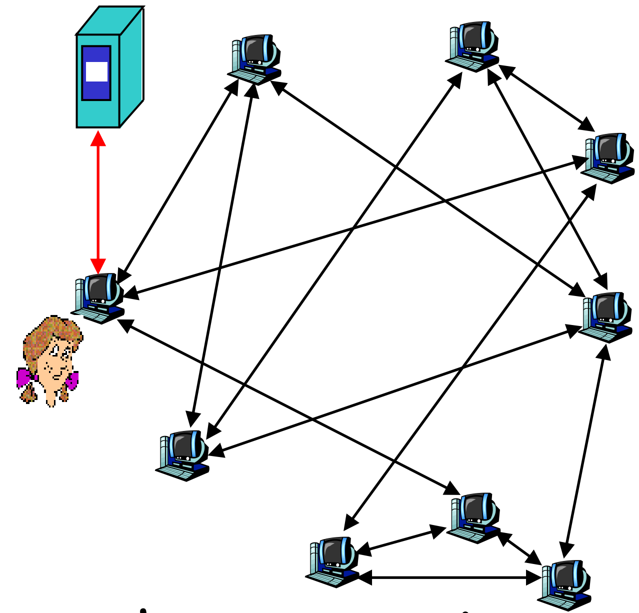
- P2P file distribution

*tracker:* tracks peers participating in torrent

*torrent:* group of peers exchanging chunks of a file



# BitTorrent (1)



- file divided into 256KB *chunks*.
- peer joining torrent:
  - has no chunks, but will accumulate them over time
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")
- while downloading, peer uploads chunks to other peers.
- peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain

# BitTorrent (2)

## Pulling Chunks

- at any given time, different peers have different subsets of file chunks
- periodically, a peer (Alice) asks each neighbor for list of chunks that they have.
- Alice sends requests for her missing chunks
  - rarest first

## Sending Chunks: tit-for-tat

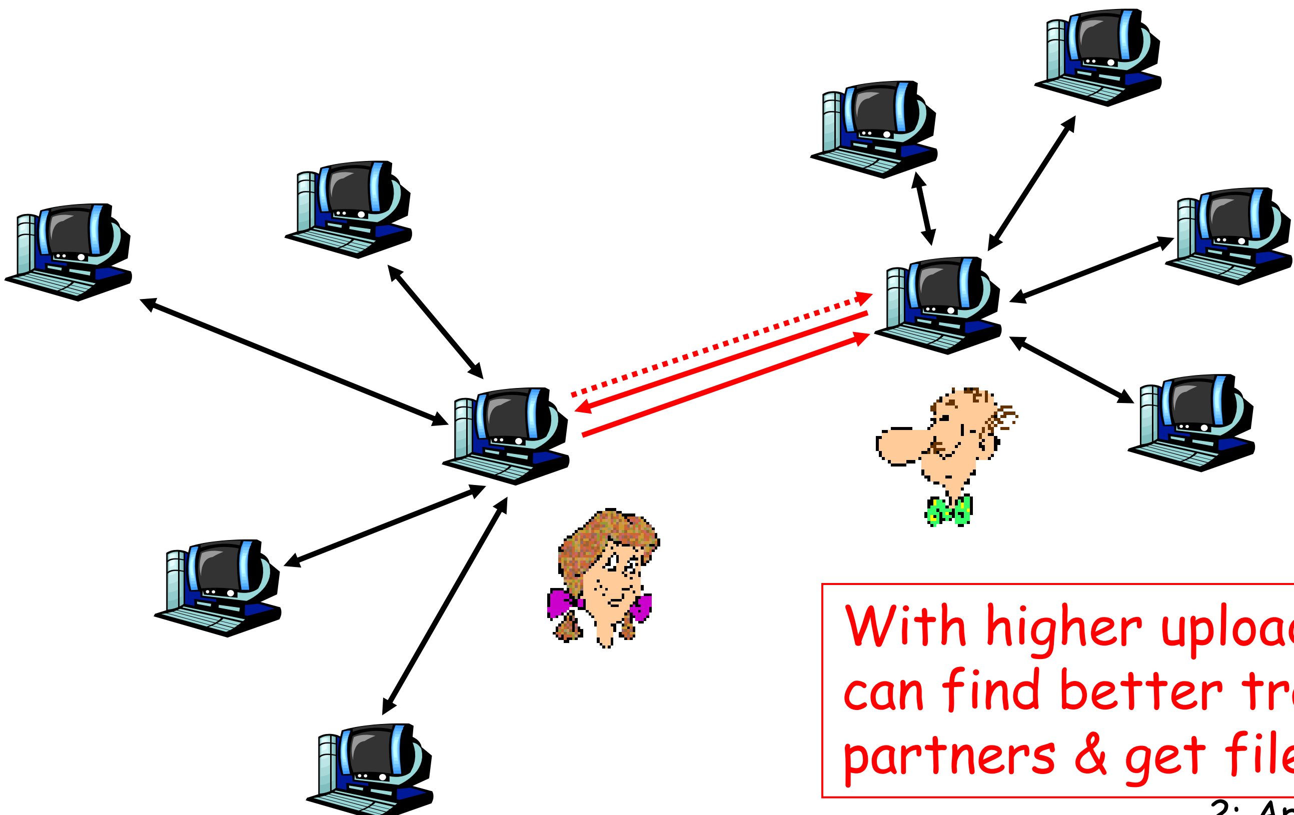
- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - newly chosen peer may join top 4
  - "optimistically unchoke"

# BitTorrent: Tit-for-tat

(1) Alice "optimistically unchokes" Bob

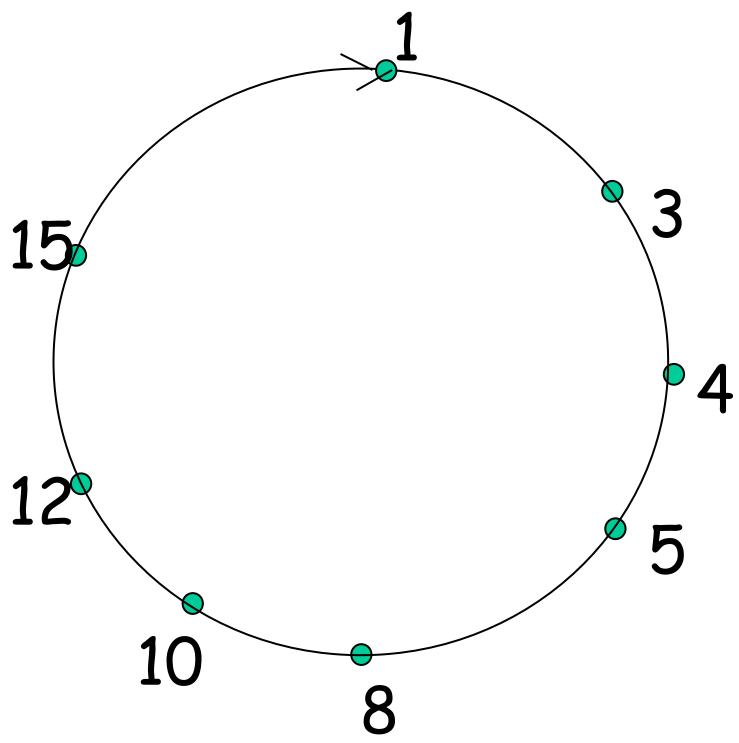
(2) Alice becomes one of Bob's top-four providers; Bob reciprocates

(3) Bob becomes one of Alice's top-four providers



With higher upload rate, can find better trading partners & get file faster!

# Circular DHT (1)




- Each peer *only* aware of immediate successor and predecessor.
- "Overlay network"

# Distributed Hash Table (DHT)

- DHT = distributed P2P database
- Database has (key, value) pairs;
  - key: ss number; value: human name
  - key: content type; value: IP address
- Peers query DB with key
  - DB returns values that match the key
- Peers can also insert (key, value) peers

# DHT Identifiers

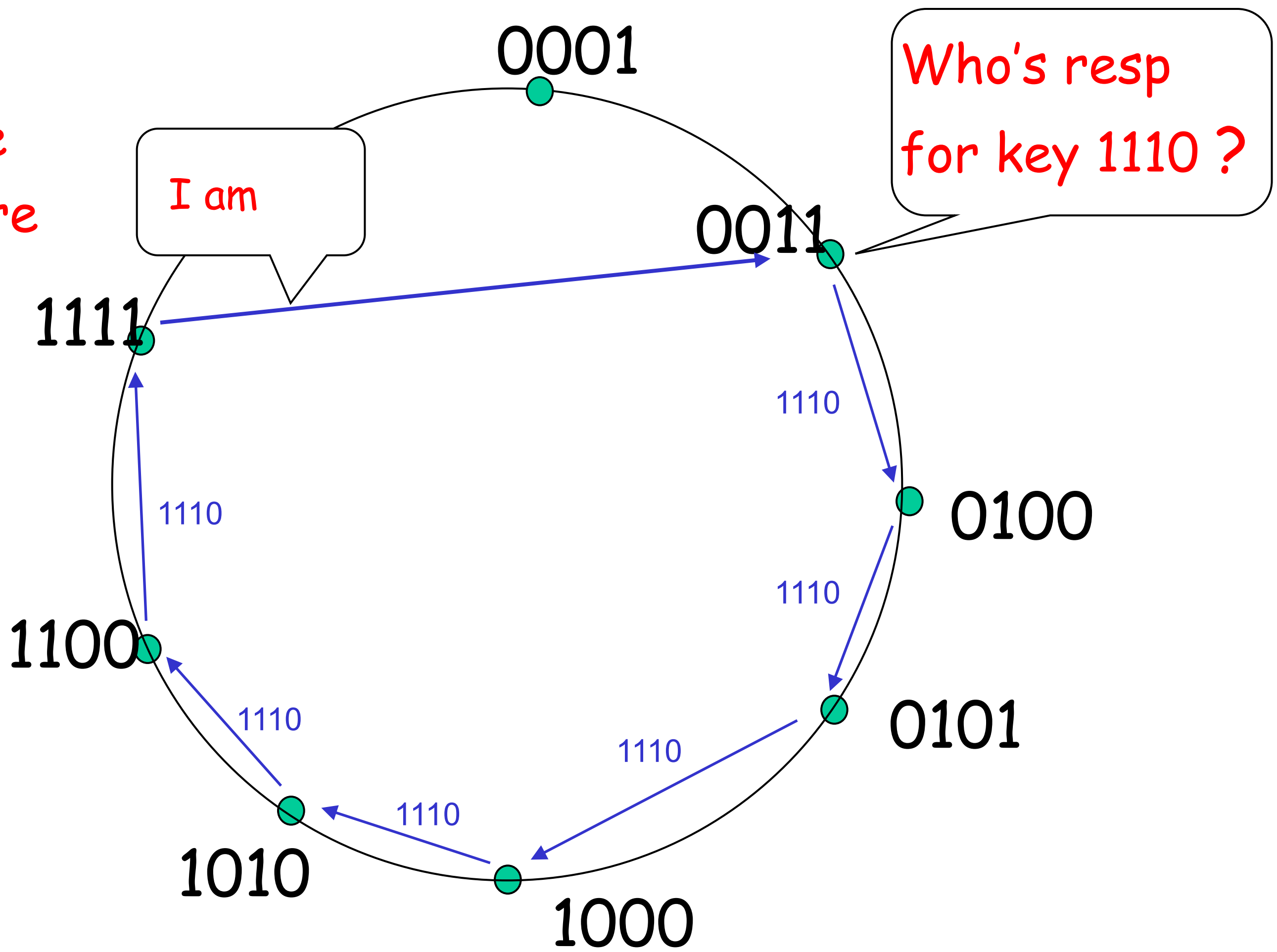
- Assign integer identifier to each peer in range $[0,2^n-1]$.
  - Each identifier can be represented by n bits.
- Require each key to be an integer in same range.
- To get integer keys, hash original key.
  - eg, key = h("Led Zeppelin IV")
  - This is why they call it a distributed "hash" table

# How to assign keys to peers?

- Central issue:
  - Assigning (key, value) pairs to peers.
- Rule: assign key to the peer that has the closest ID.
- Convention in lecture: closest is the immediate successor of the key.
- Ex: n=4; peers: 1,3,4,5,8,10,12,14;
  - key = 13, then successor peer = 14
  - key = 15, then successor peer = 1

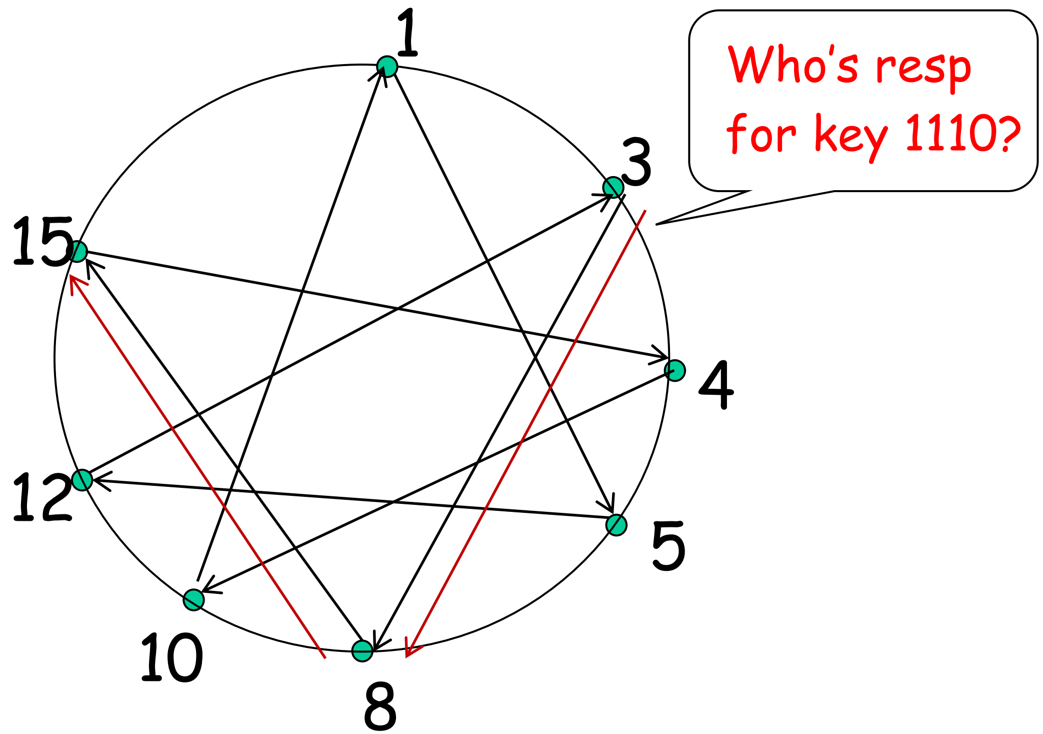
# Circle DHT (2)

O(N) messages on avg to resolve query, when there are N peers



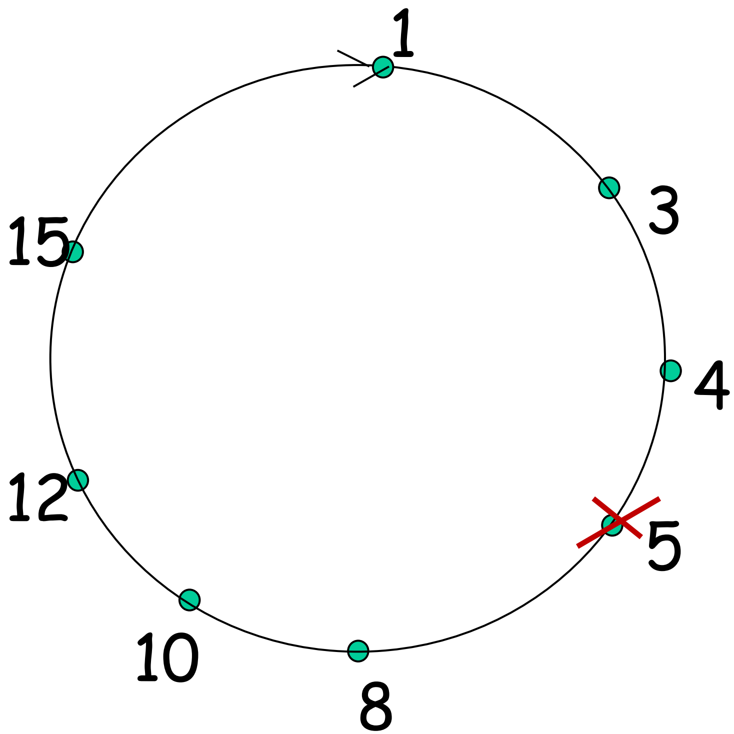

Define closest as closest successor

# Circular DHT with Shortcuts




- Each peer keeps track of IP addresses of predecessor, successor, short cuts.
- Reduced from 6 to 2 messages.
- Possible to design shortcuts so O(log N) neighbors, O(log N) messages in query

# Peer Churn

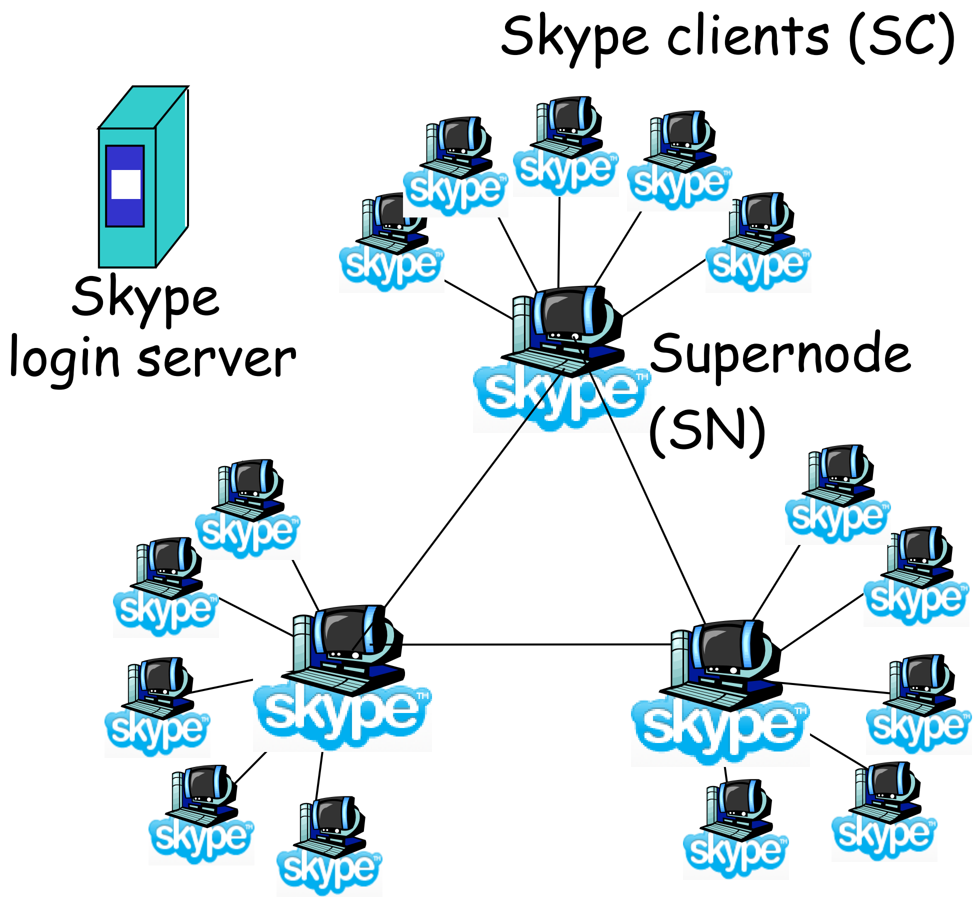



- To handle peer churn, require each peer to know the IP address of its two successors.
- Each peer periodically pings its two successors to see if they are still alive.

- Peer 5 abruptly leaves
- Peer 4 detects; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8's immediate successor its second successor.
- What if peer 13 wants to join?

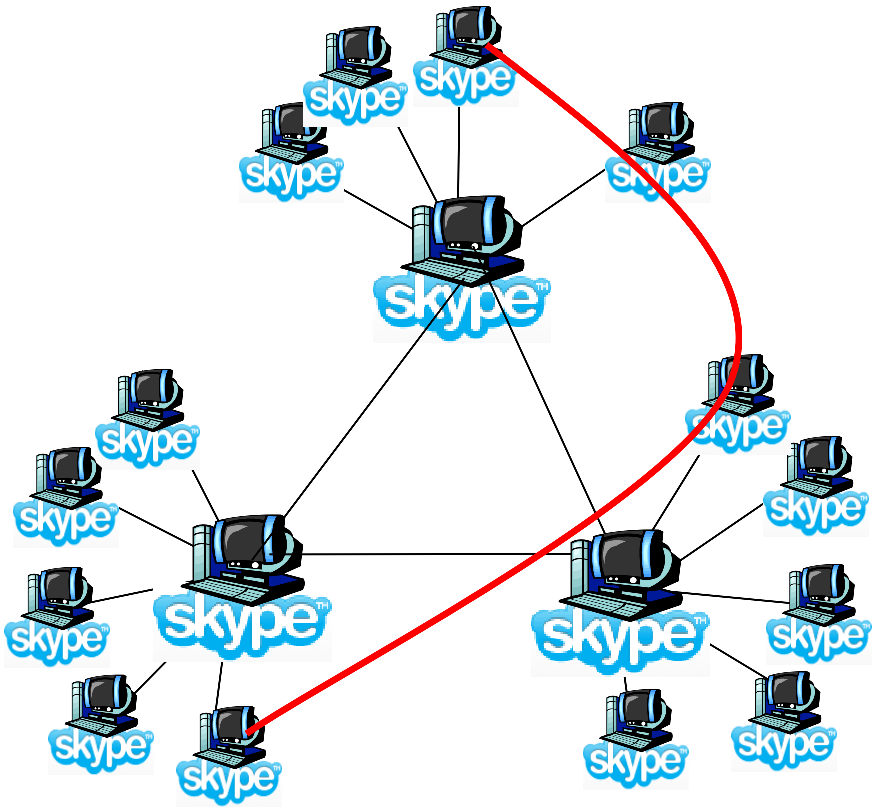
# P2P Case study: Skype

- inherently P2P: pairs of users communicate.
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay with SNs
- Index maps usernames to IP addresses; distributed over SNs



# Peers as relays

- Problem when both Alice and Bob are behind "NATs".
  - NAT prevents an outside peer from initiating a call to insider peer
- Solution:
  - Using Alice's and Bob's SNs, Relay is chosen
  - Each peer initiates session with relay.
  - Peers can now communicate through NATs via relay



# Chapter 2: Summary

our study of network apps now complete!

- application architectures
  - client-server
  - P2P
  - hybrid
- application service requirements:
  - reliability, bandwidth, delay
- Internet transport service model
  - connection-oriented, reliable: TCP
  - unreliable, datagrams: UDP
- specific protocols:
  - HTTP
  - FTP
  - SMTP, POP, IMAP
  - DNS
  - P2P: BitTorrent, Skype
- socket programming

# Chapter 2: Summary

## Most importantly: learned about *protocols*

- typical request/reply message exchange:
  - client requests info or service
  - server responds with data, status code
- message formats:
  - headers: fields giving info about data
  - data: info being communicated

*Important themes:*

- control vs. data msgs
  - in-band, out-of-band
- centralized vs. decentralized
- stateless vs. stateful
- reliable vs. unreliable msg transfer
- "complexity at network edge"