DATA COMMUNICATOIN NETWORKING

Instructor: Ouldooz Baghban Karimi

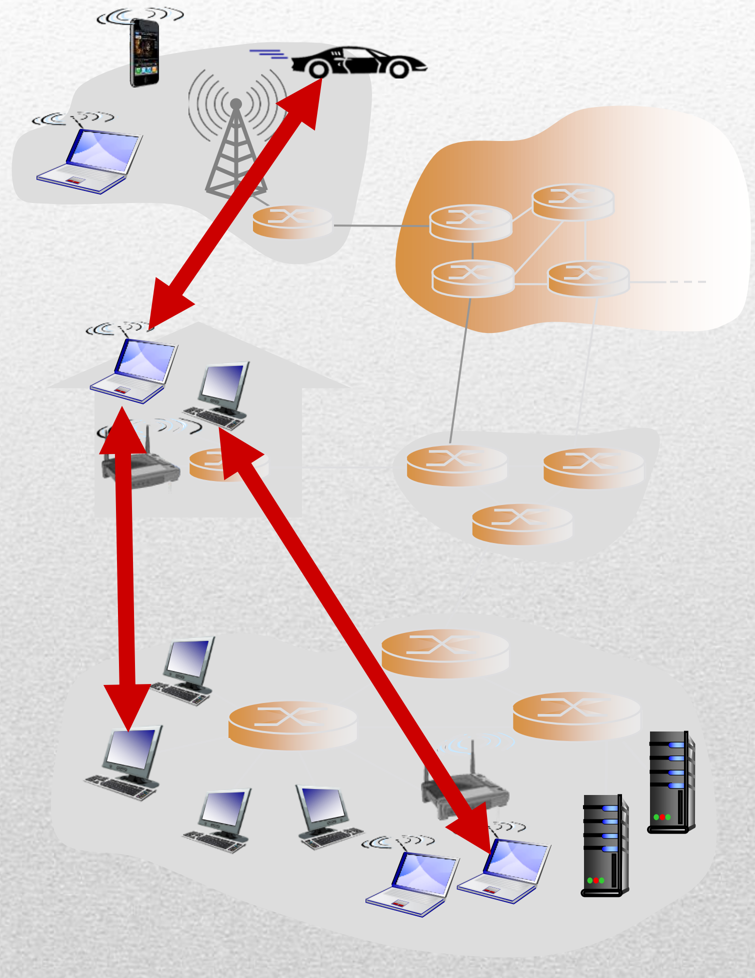
Course Book: Computer Networking, A Top-Down Approach
By: Kurose, Ross

Course Overview

- **Basics of Computer Networks**
  - Internet & Protocol Stack
  - Application Layer
  - Transport Layer
  - Network Layer
  - Data Link Layer
- **Advanced Topics**
  - Case Studies of Computer Networks
  - Internet Applications
  - Network Management
  - Network Security

P2P Architectures

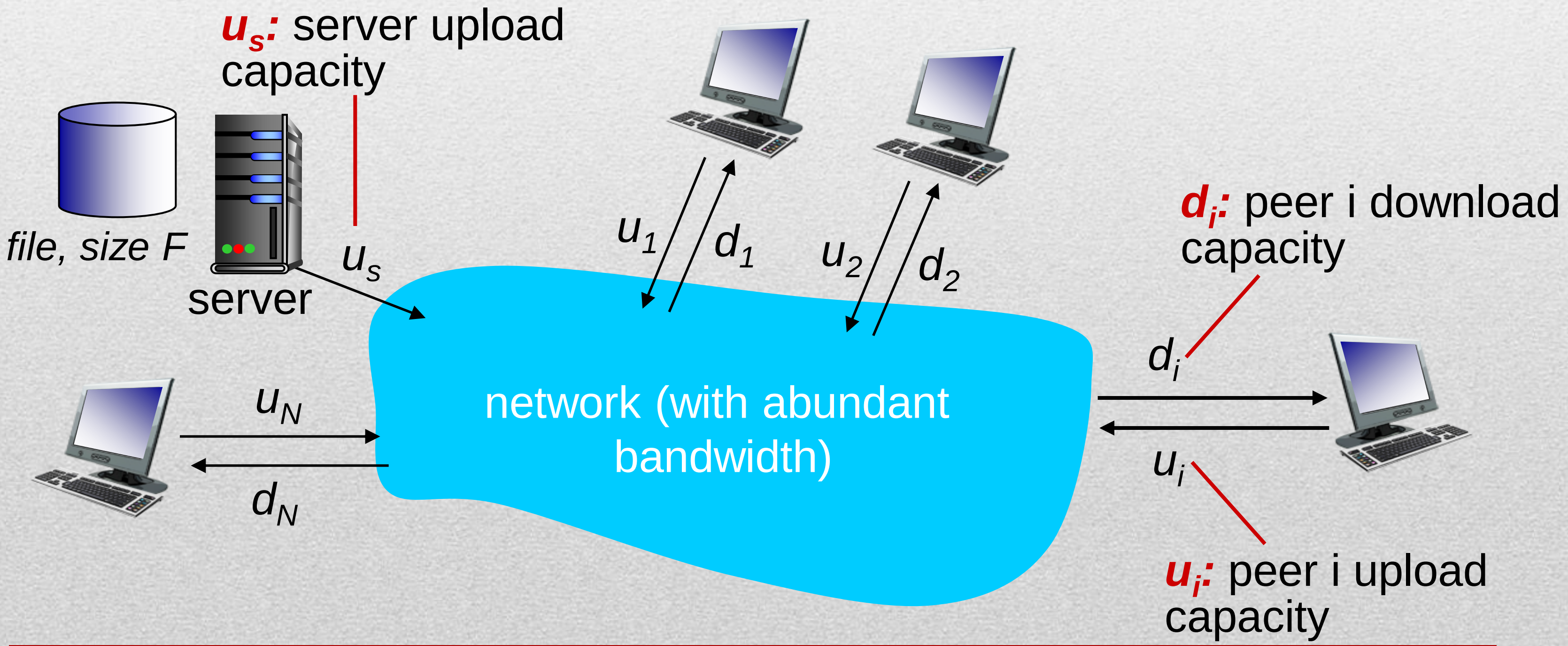
- No always-on server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change IP addresses
- Examples
  - File distribution (BitTorrent)
  - Streaming (KanKan)
  - VoIP (Skype)



File Distribution: Client/Server vs. P2P

Question: how much time to distribute file (size *F*) from one server to *N* peers?

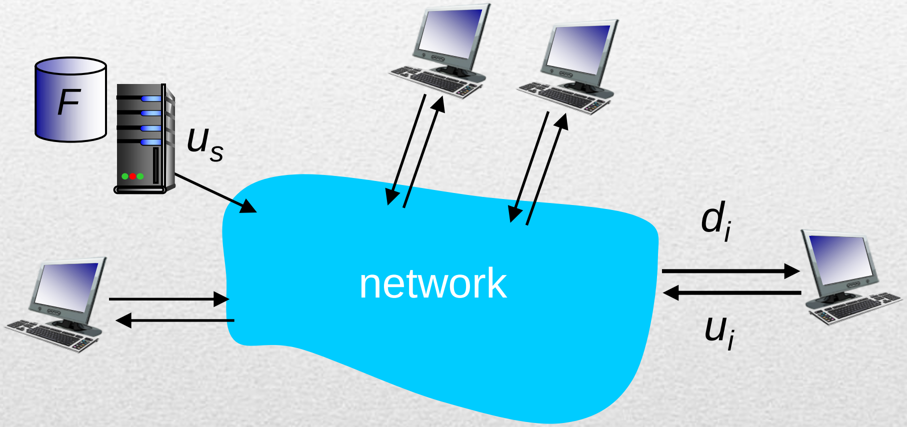
Peer upload/download capacity is limited resource



File Distribution Time: Client/Server

- *Server transmission:* must sequentially send (upload) *N* file copies
  - Time to send one copy: F/u_s
  - Time to send N copies: NF/u_s
- *Client:* each client must download file copy
  - d_{min} = min client download rate
  - min client download time:

  F/d_{min}

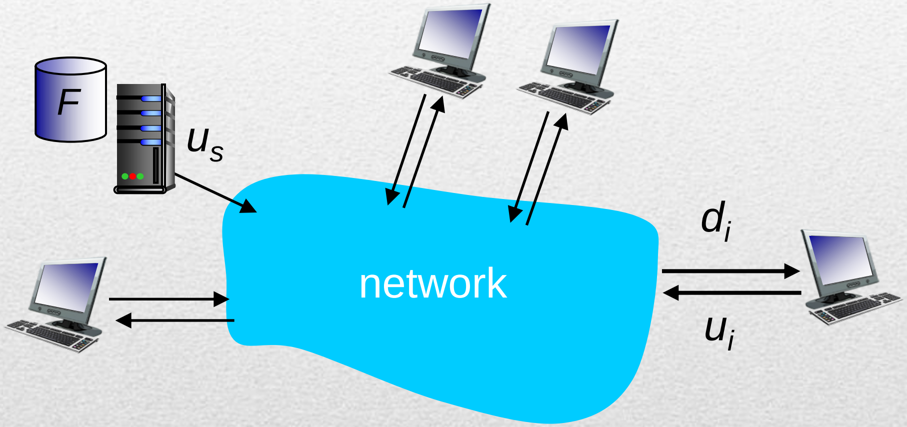


time to distribute F to N clients using client-server approach

$$D_{cs} > \max\{ NF/u_s , F/d_{min} \}$$

File Distribution Time: P2P

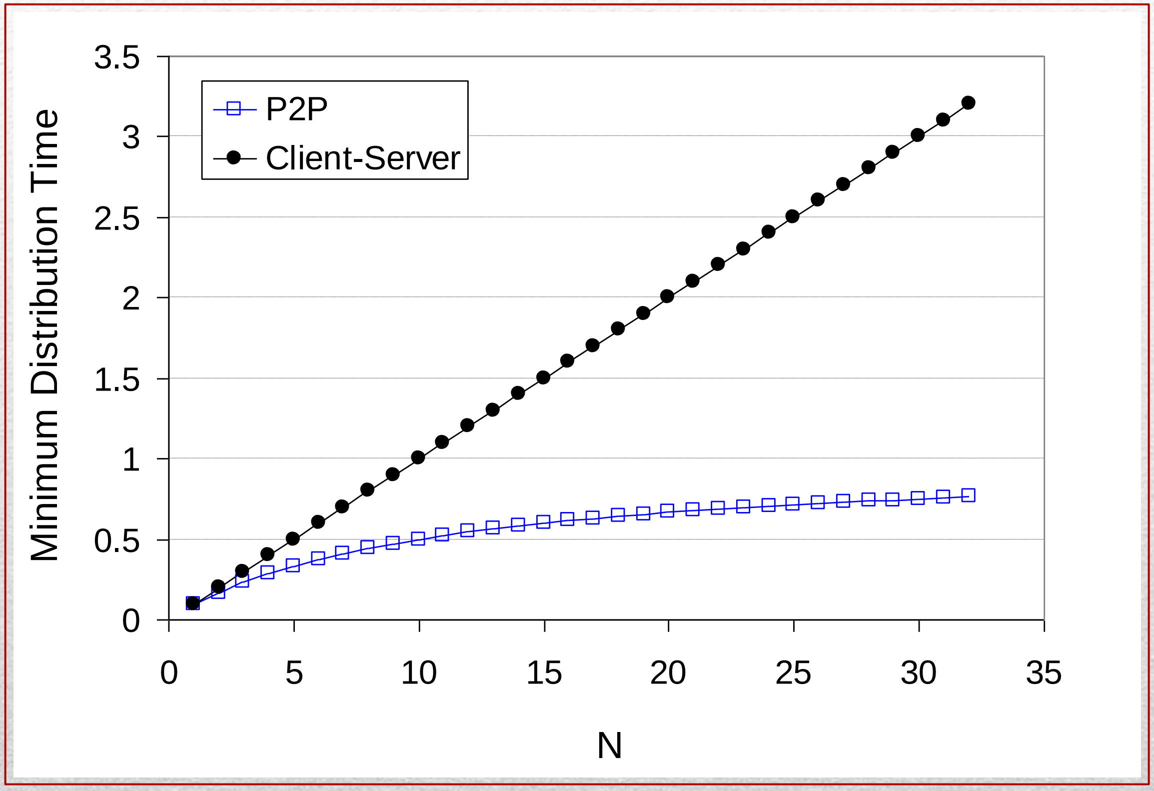
- *Server transmission:* upload at least one copy
  - Time to send one copy: F/u_s
- *Client:* each client must download file copy
  - Min client download time: F/d_{min}
- *Clients:* as aggregate must download *NF* bits
  - Max upload rate (limiting max download rate) is $u_s + Su_i$



time to distribute F to N clients using P2P approach

$$D_{P2P} > \max\{F/u_s\ ,\ F/d_{min}\ ,\ NF/(u_s + \Sigma u_i)\}$$

Client-Server vs. P2P Example



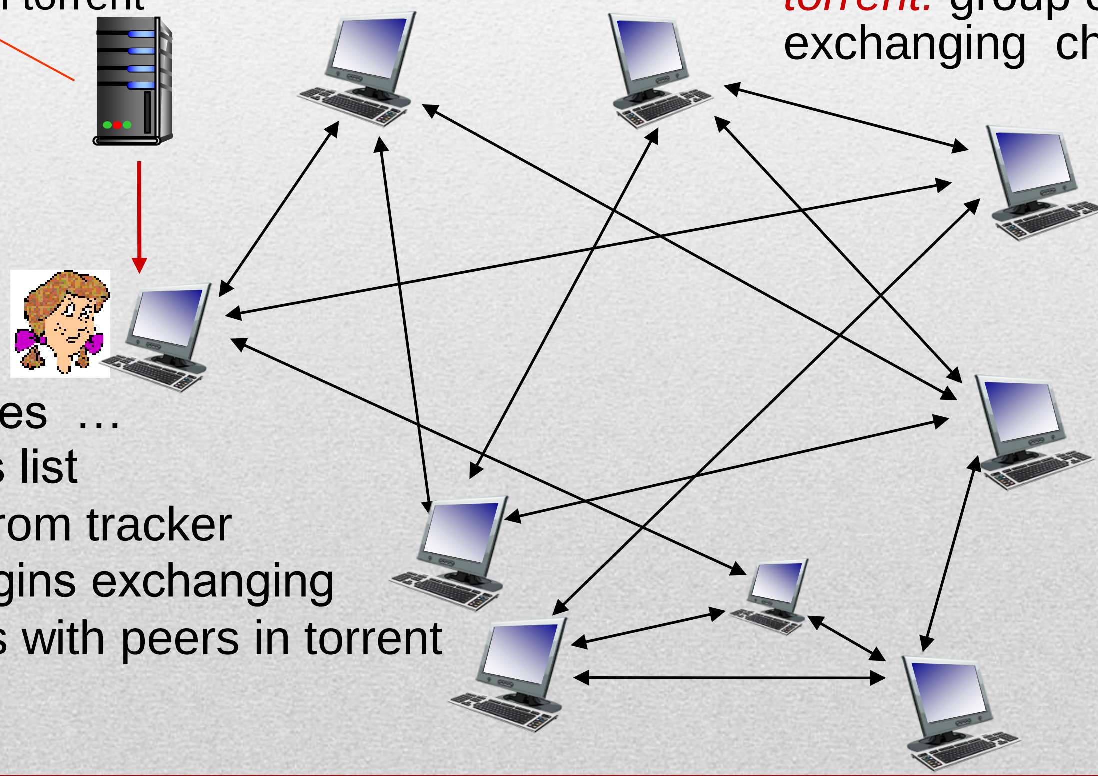
client upload rate = u, F/u = 1 hour, $u_s = 10u$, $d_{min} \geq u_s$

BitTorrent

File divided into 256Kb chunks
Peers in torrent send/receive file chunks

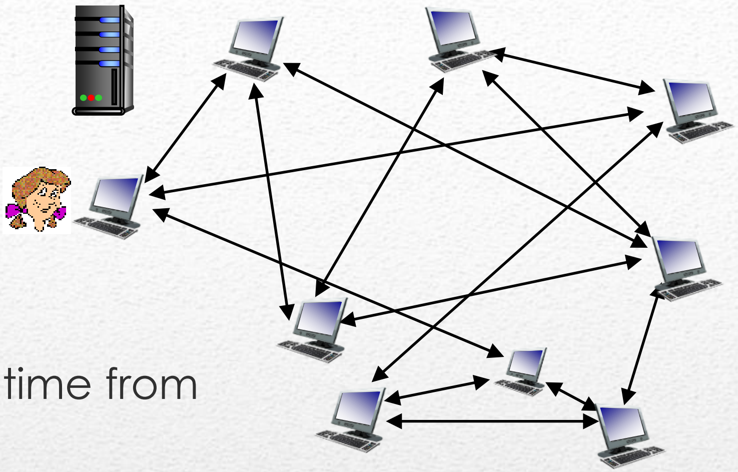
tracker: tracks peers participating in torrent

torrent: group of peers exchanging chunks of a file



Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

BitTorrent



- **New Peer**
  - No chunks, but will accumulate them over time from other peers
  - Registers with tracker to get list of peers, connects to subset of peers (“neighbors”)
- While downloading, peer uploads chunks to other peers
- Peer may change peers with whom it exchanges chunks
- *Churn:* peers may come and go
- Once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent

- **Requesting chunks**
  - At any given time, different peers have different subsets of file chunks
  - Periodically, Alice asks each peer for list of chunks that they have
  - Alice requests missing chunks from peers, rarest first

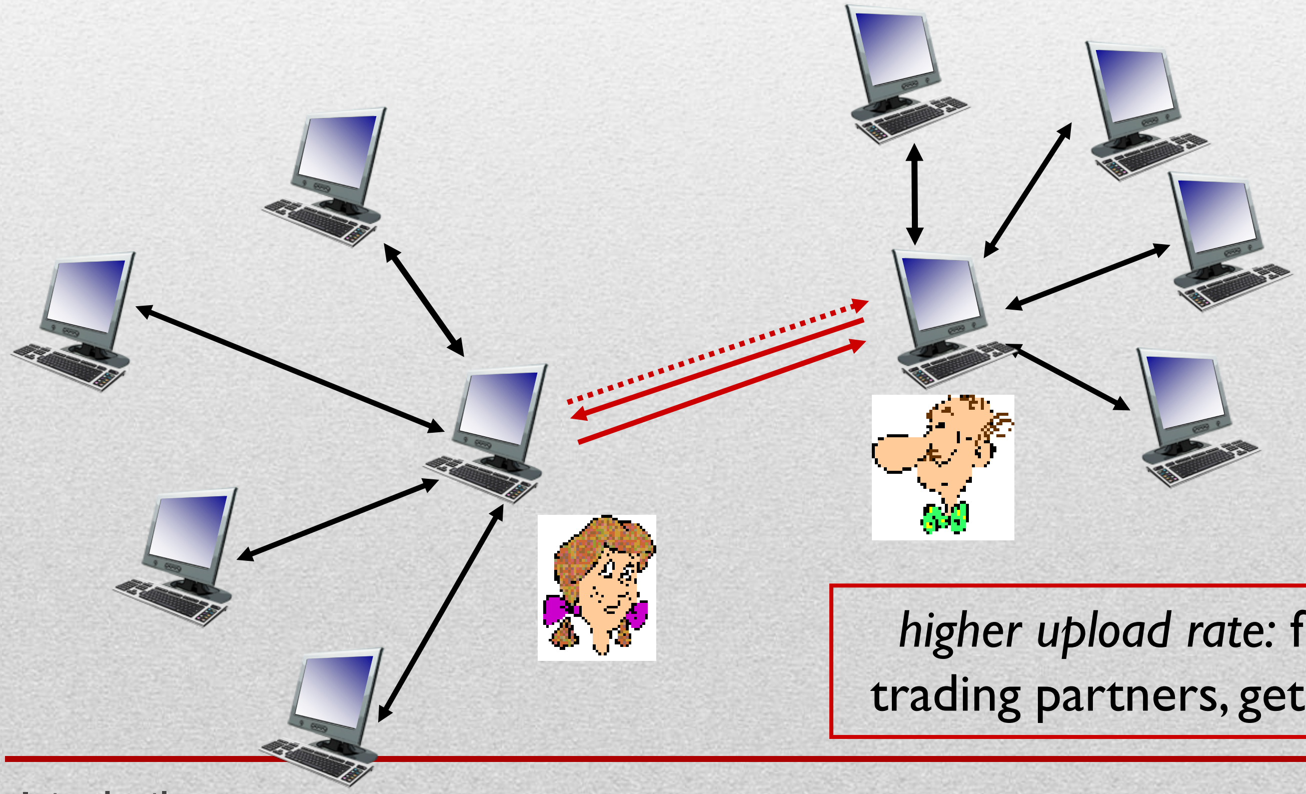
- **Sending chunks: tit-for-tat**
  - Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - Other peers are choked by Alice (do not receive chunks from her)
    - re-evaluate top 4 every10 secs
  - Every 30 secs: randomly select another peer, starts sending chunks
    - "optimistically unchoke" this peer
    - Newly chosen peer may join top 4

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



higher upload rate: find better trading partners, get file faster !

DHT: Distributed Hash Table

- DHT: A *distributed P2P database*
- Database has (key, value) pairs; examples:
  - Key: ss number; value: human name
  - Key: movie title; value: IP address
- Distribute the (key, value) pairs over the (millions of peers)
- A peer queries DHT with key
  - DHT returns values that match the key
- Peers can also insert (key, value) pairs

How to assign key to peers?

- **Central issue**
  - Assigning (key, value) pairs to peers.
- **Basic idea**
  - Convert each key to an integer
  - Assign integer to each peer
  - Put (key,value) pair in the peer that is closest to the key

DHT Identifiers

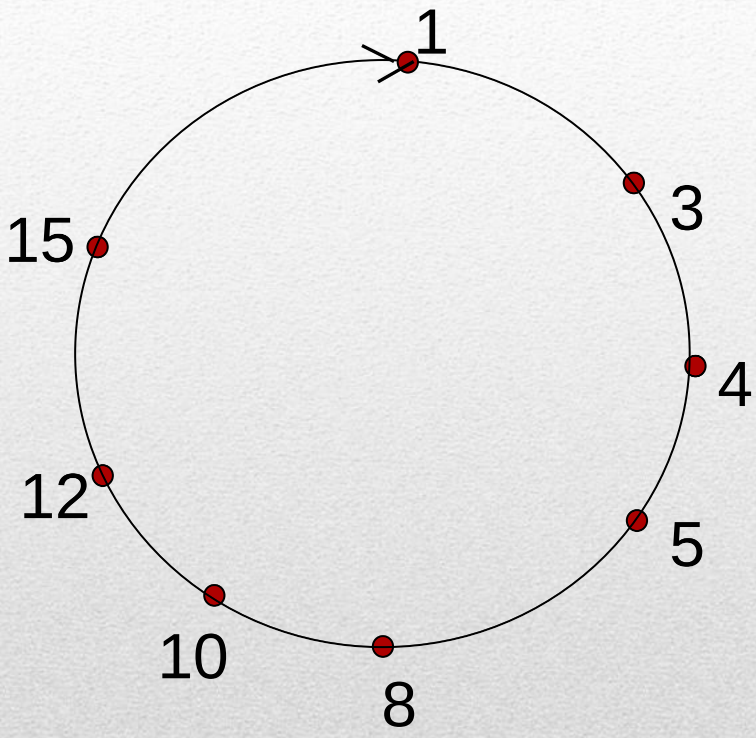
- Assign integer identifier to each peer in range $[0,2^n-1]$ for some *n*.
  - Each identifier represented by *n* bits.

- Require each key to be an integer in same range
- To get integer key, hash original key
  - *e.g.,* key = hash("Led Zeppelin IV")
  - This is why its is referred to as a *Distributed "Hash" Table*

Assigning Keys to Peers

- Rule: assign key to the peer that has the *closest* ID.
- Convention in lecture: closest is the *immediate successor* of the key.
- e.g., n=4; peers: 1,3,4,5,8,10,12,14;
  - Key = 13, then successor peer = 14
  - Key = 15, then successor peer = 1

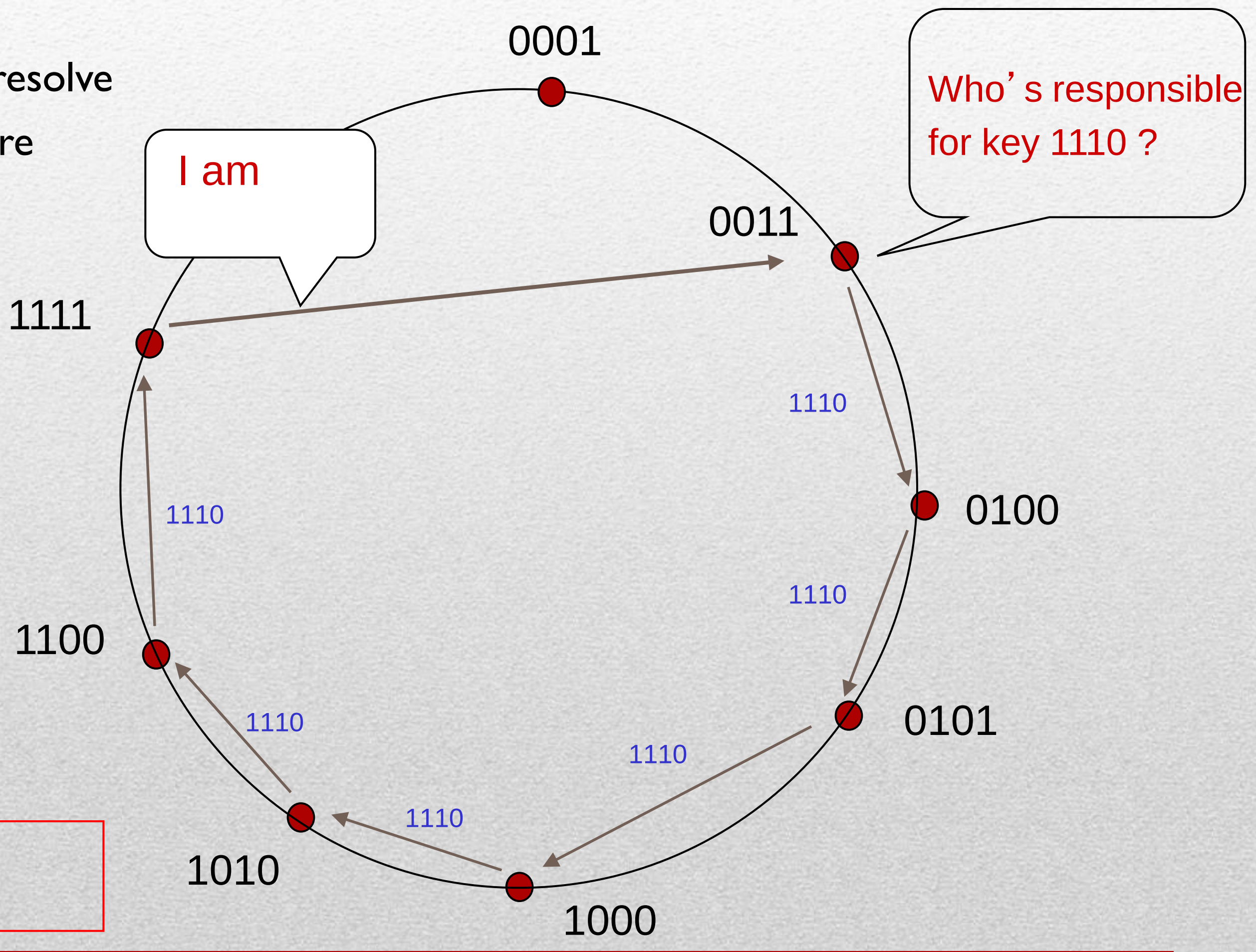
Circular DHT



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

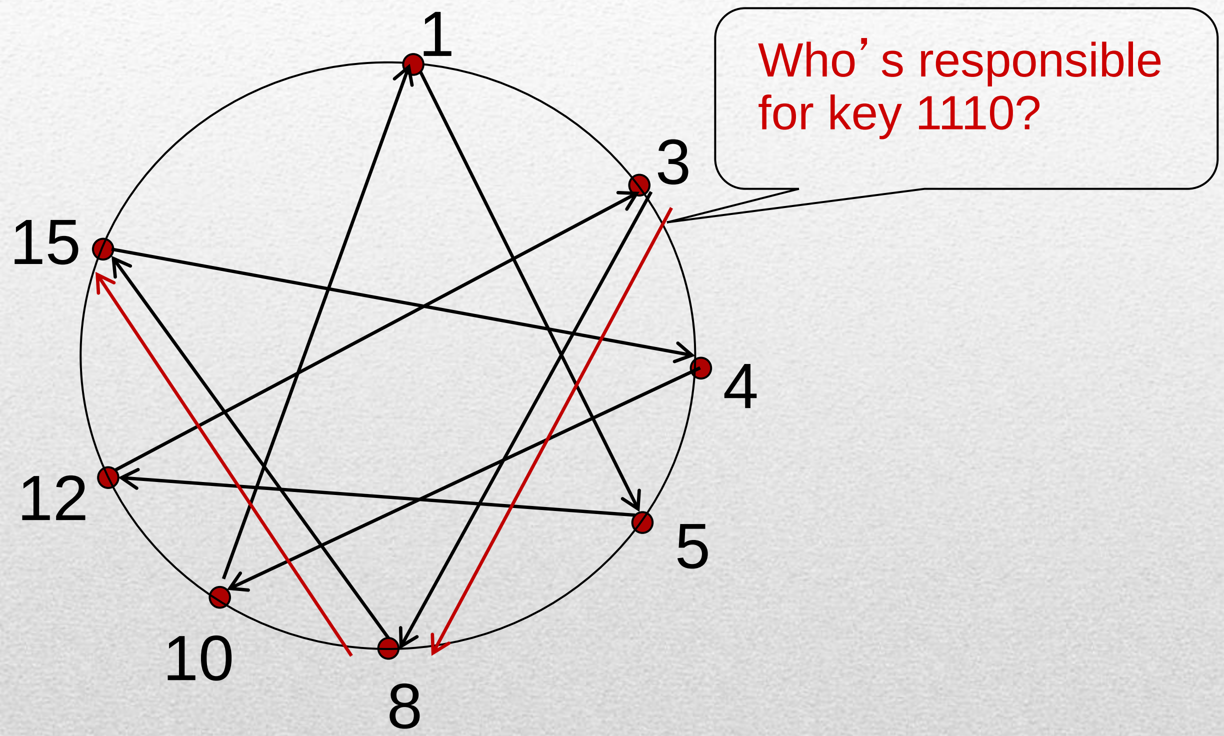
Circular DHT

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



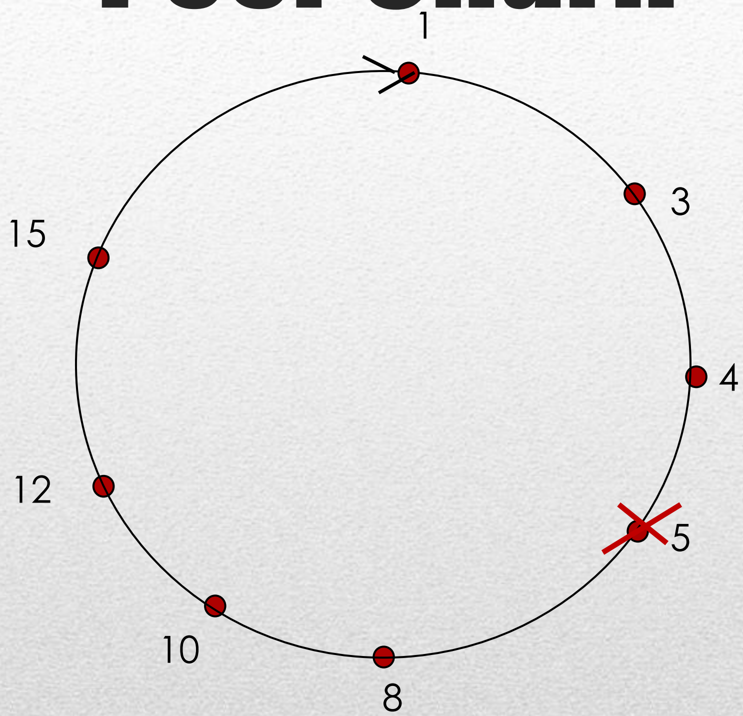
Define closest as
closest successor

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



Handling peer churn

- Peers may come and go (churn)
- Each peer knows address of its two successors
- Each peer periodically pings its two successors to check aliveness
- If immediate successor leaves, choose next successor as new immediate successor

Example: peer 5 abruptly leaves

- Peer 4 detects peer 5 departure; 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?

Socket Programming

Two socket types for two transport services:

- UDP: unreliable datagram
- TCP: reliable, byte stream-oriented

Application Example

- Client reads a line of characters (data) from its keyboard and sends the data to the server.
- The server receives the data and converts characters to uppercase.
- The server sends the modified data to the client.
- The client receives the modified data and displays the line on its screen.