بسم الله الرحمن الرحیم

Information Technology Engineering

Mohammad Hossein Manshaei

manshaei@gmail.com

1393

p2p, DHT, …

PEER TO PEER NETWORKING

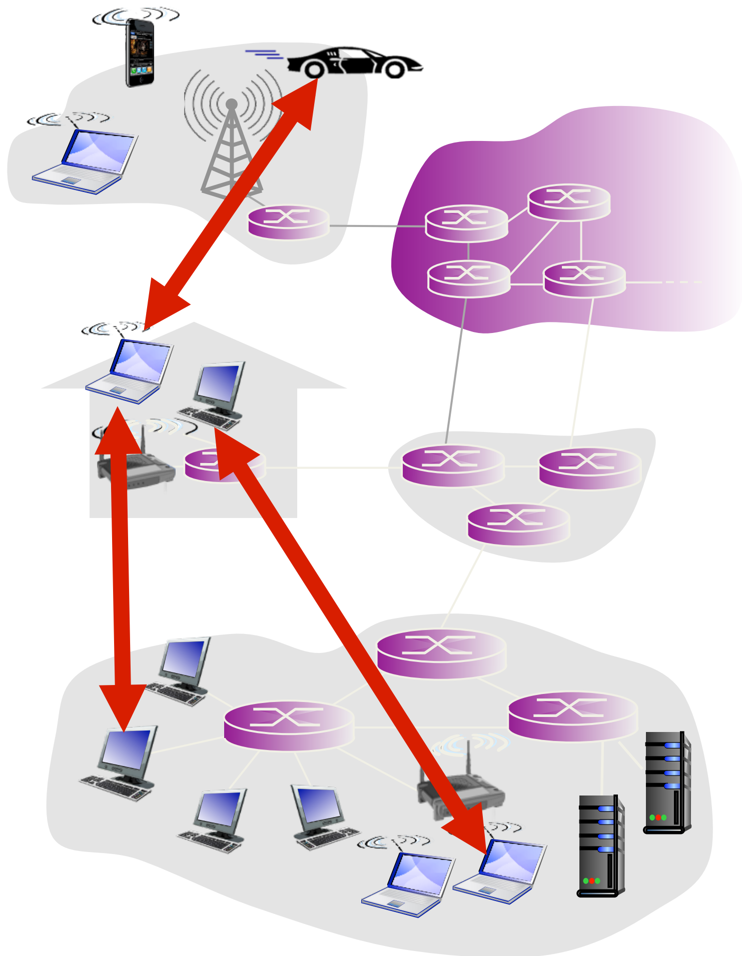
Slides derived from those available on the Web site of the book
"Computer Networking", by Kurose and Ross, PEARSON

Pure P2P architecture

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

examples:

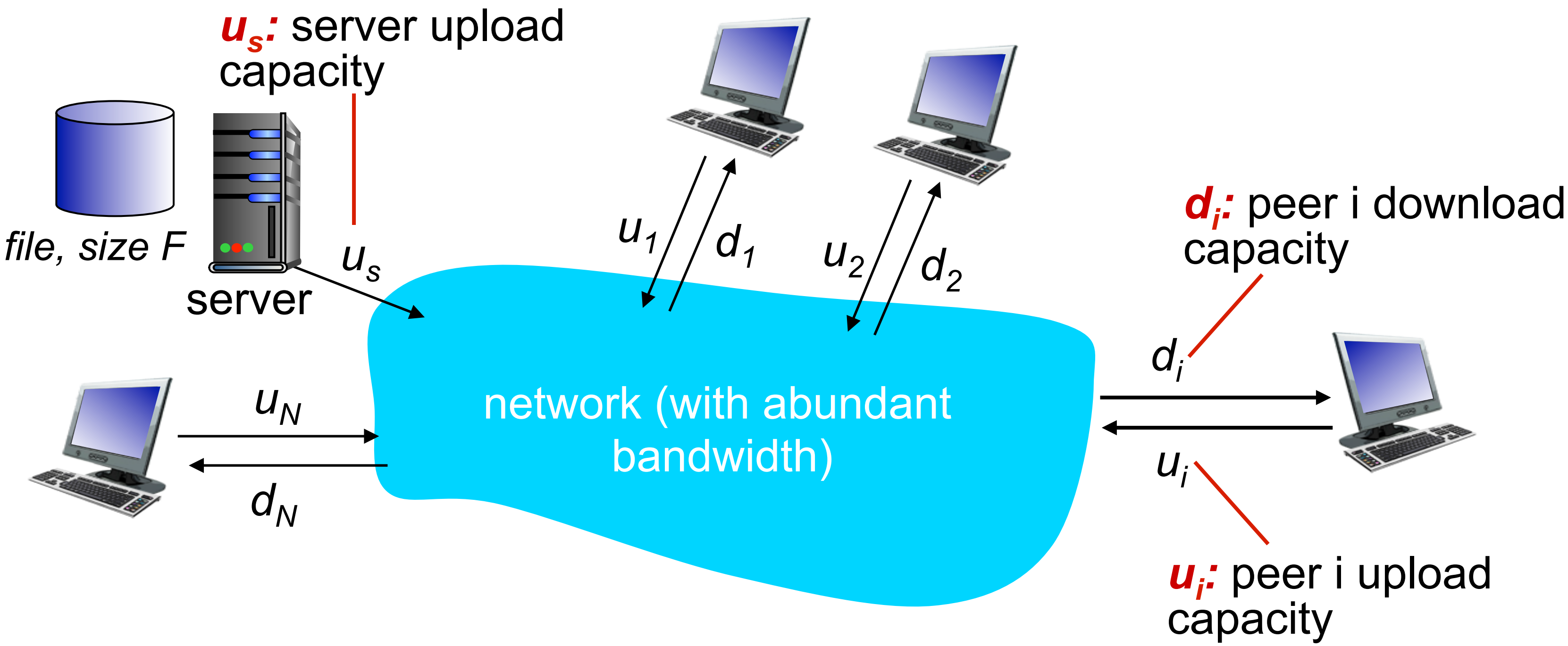
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



File distribution: client-server vs P2P

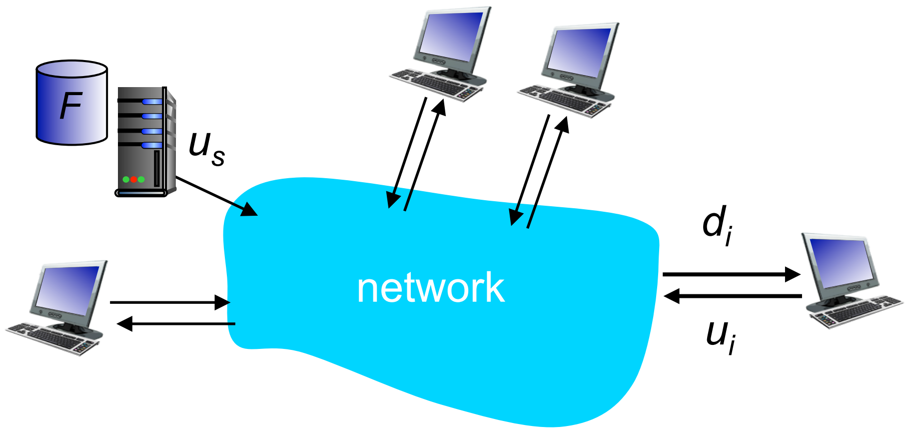
Question: how much time to distribute file (size *F*) from one server to *N* *peers?*

- 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}

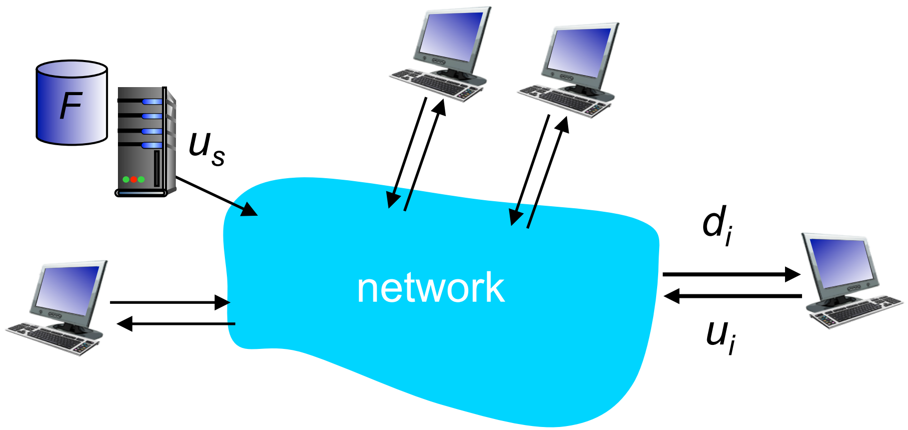
time to distribute F to N clients using client-server approach

$$D_{c-s} \geq max\{NF/u_{s,},F/d_{min}\}$$

increases linearly in N

File distribution time: P2P

- *server transmission:* must 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 (limting max download rate) is $u_s + \Sigma u_i$

time to distribute F to N clients using P2P approach

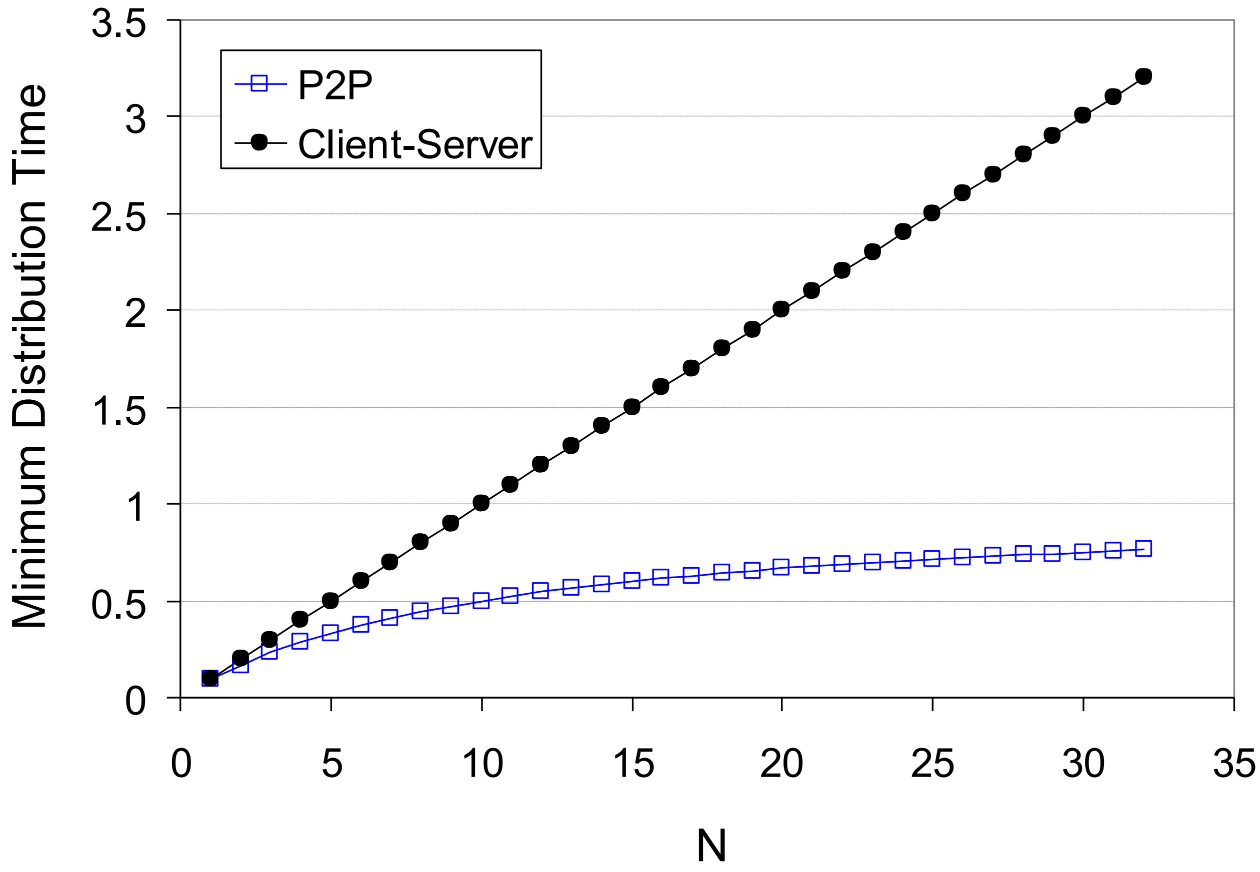
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

… but so does this, as each peer brings service capacity

Client-server vs. P2P: example

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

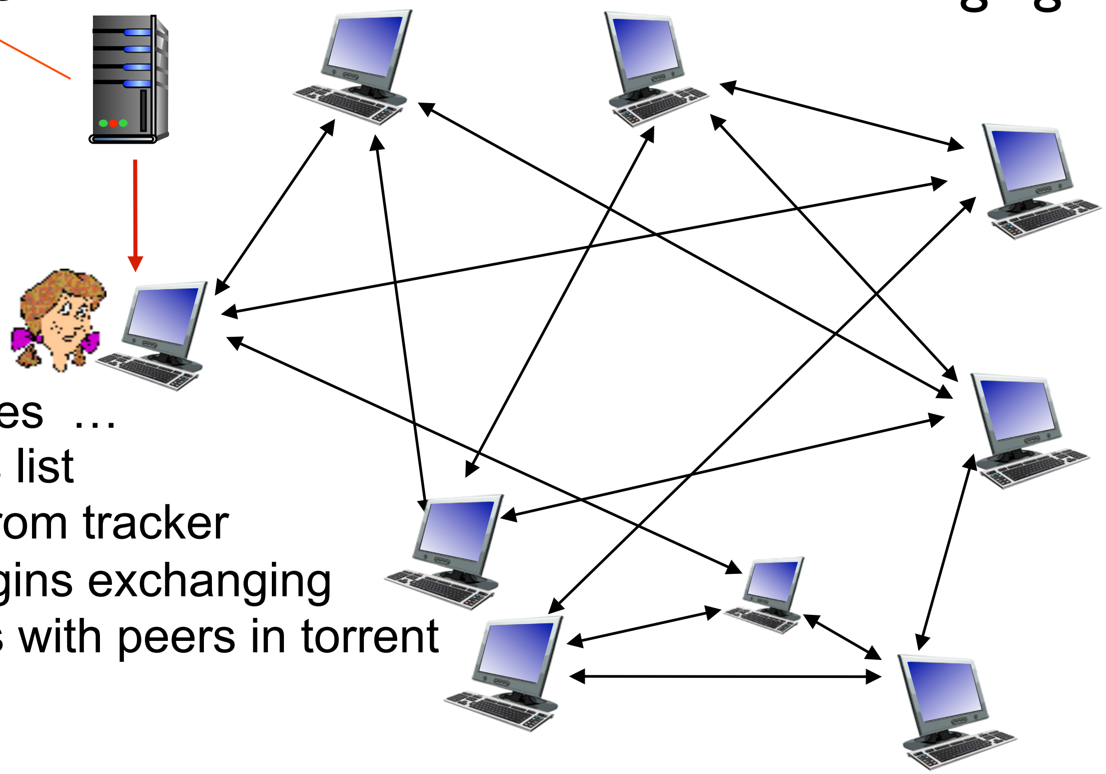


P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

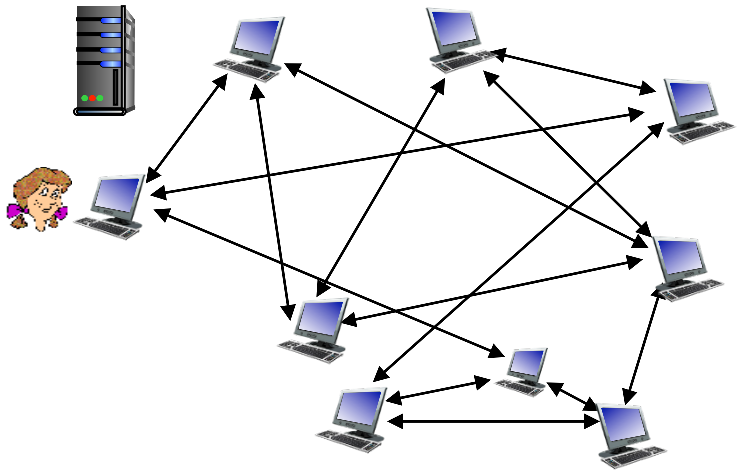
torrent: group of peers exchanging chunks of a file



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

P2P file distribution: BitTorrent

- Peer joining torrent:
  - has 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, sending file chunks

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

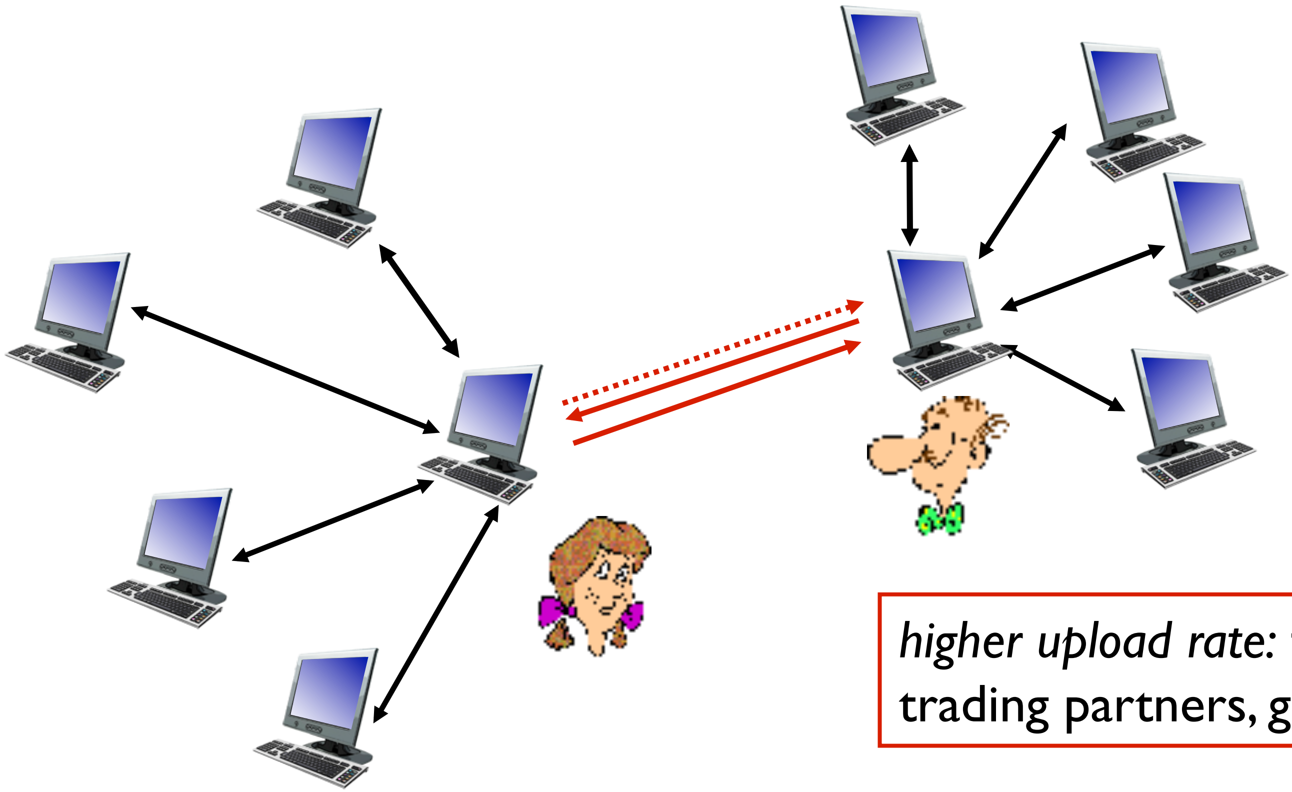
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- Every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
  - 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 !

Distributed Hash Table (DHT)

- 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

Q: how to assign keys 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

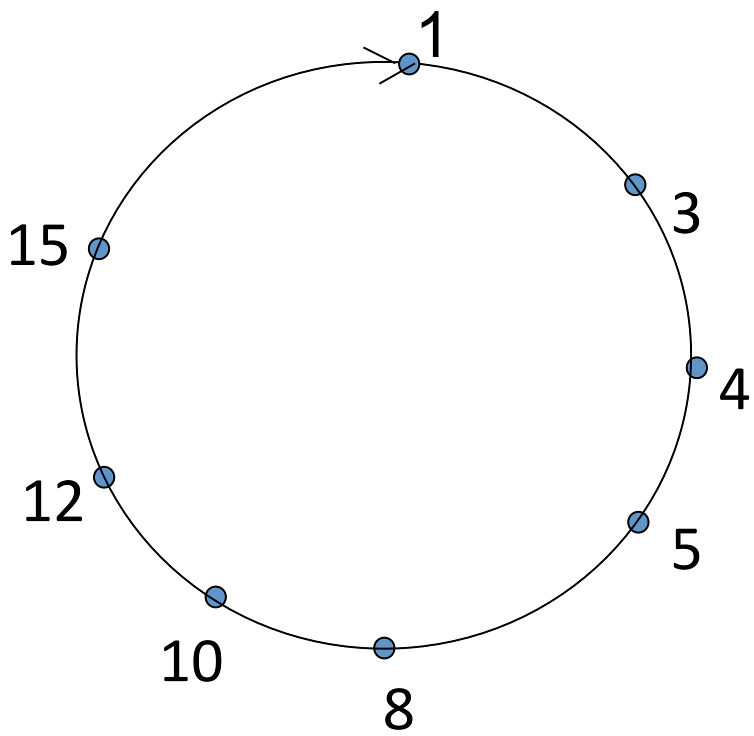
- Assign integer identifier to each peer in range $[0,2^n-1]$ for some *n*.
  - each identifier represented by *n* bits.

- 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*

Assign 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

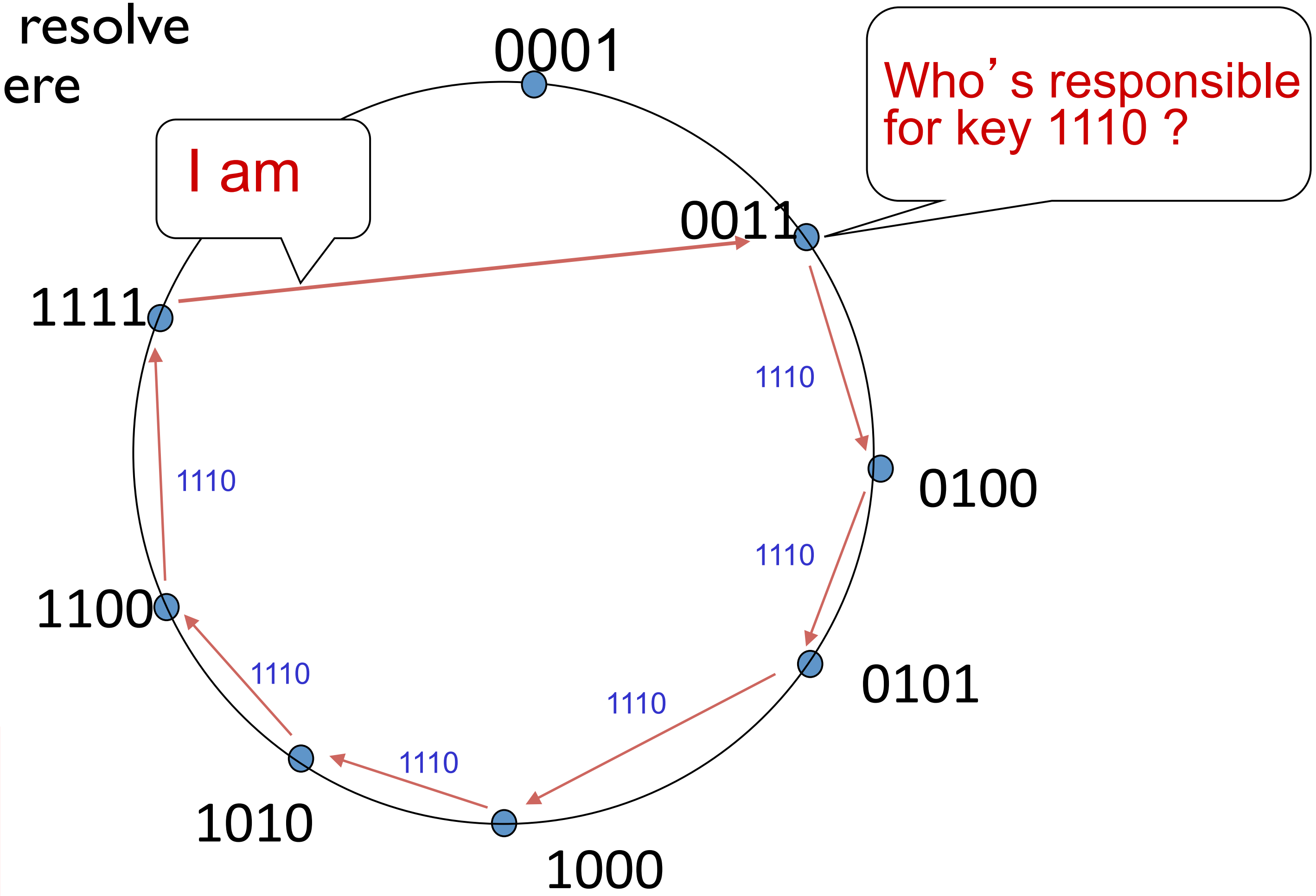
Circular DHT (1)



- each peer *only* aware of immediate successor and predecessor.
- "overlay network"

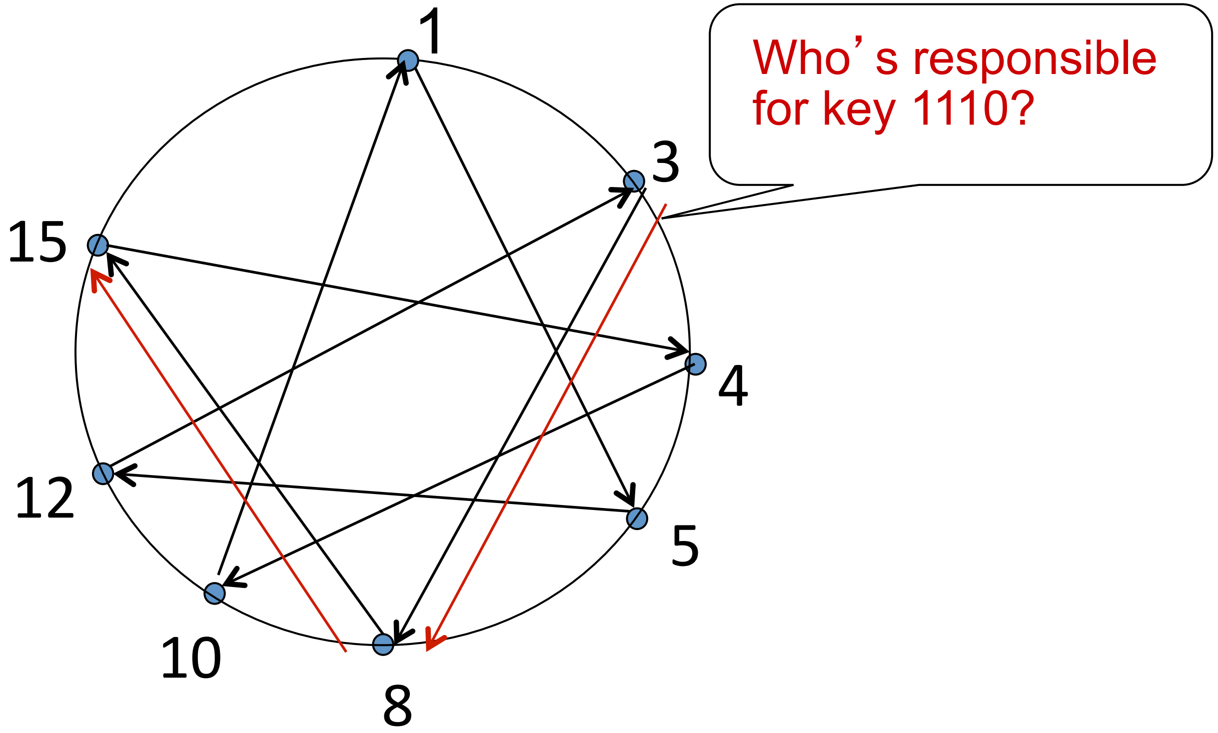
Circular DHT (1)

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



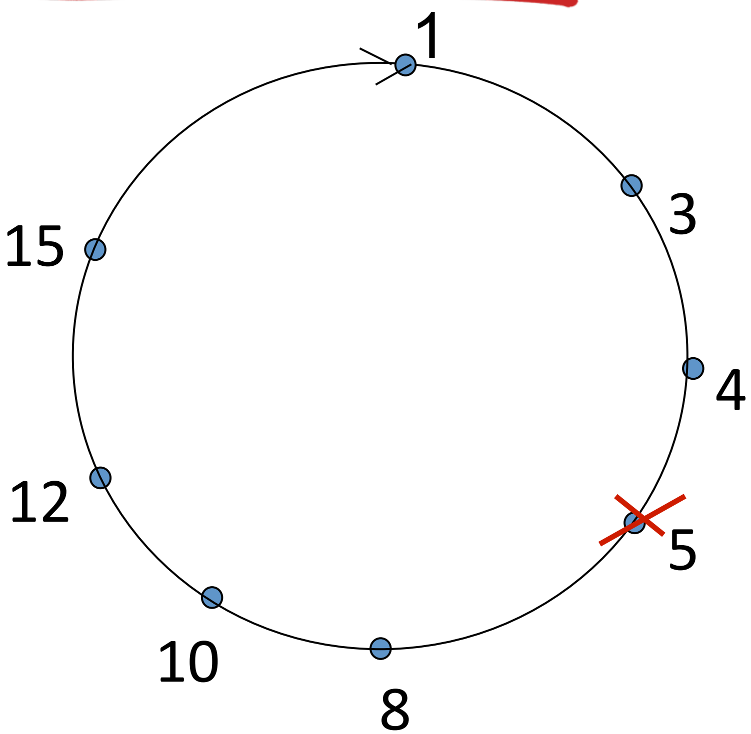
Define closest
as closest
successor

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?