ECE 158B Data Networks II

Lecture 09. P2P; DHT; CDN

Prof. Xinyu Zhang

http://xyzhang.ucsd.edu

Department of Electrical and Computer Engineering

University of California San Diego

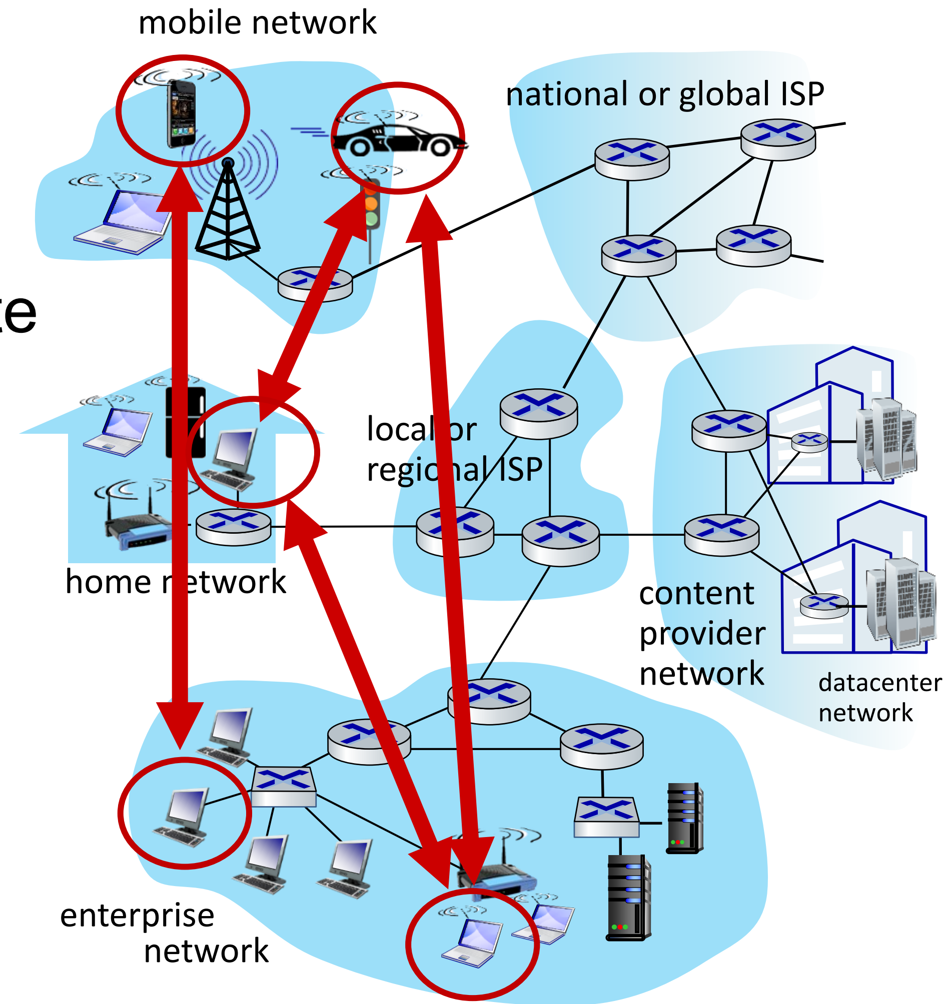
Today's agenda

- P2P architecture and applications
- DHT: Distributed Hash Table
- CDN

P2P architecture and applications

P2P architecture

- Characteristics
  - no always-on server
  - arbitrary end systems directly communicate
  - peers request service from other peers, provide service in return to other peers.
    Self scalability – new peers bring new service capacity, and new service demands
  - peers are intermittently connected and change IP addresses – complicating management

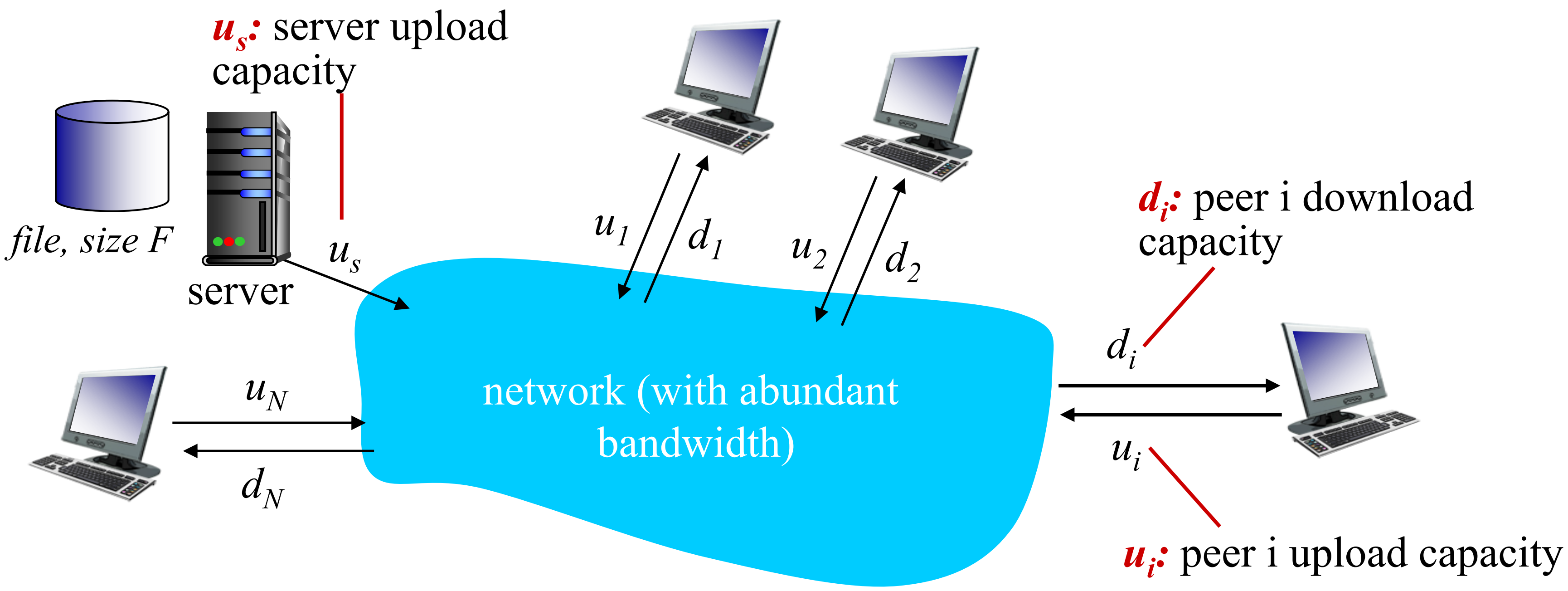


- Examples:
  - File distribution (BitTorrent), Streaming (PPLive), VoIP (Skype)

Client/server vs. P2P architecture

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

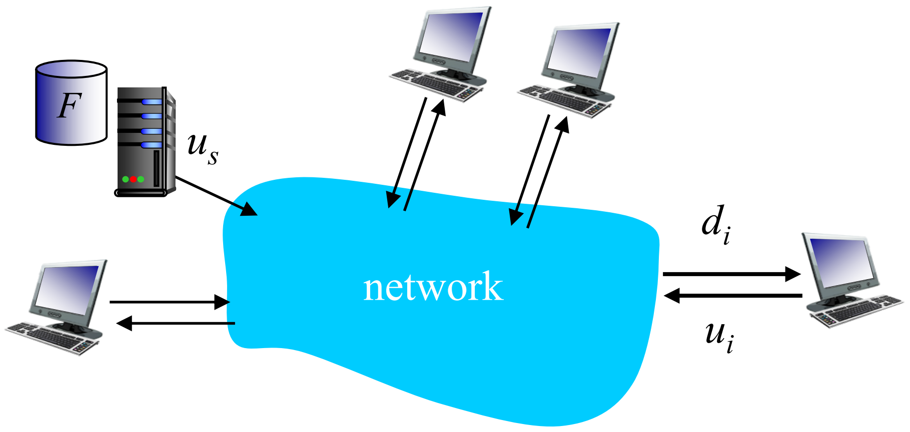
- peer upload/download capacity is limited resource



Client/server vs. P2P architecture

File distribution time using 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
  - ✓ max client download time: F/d_{min}

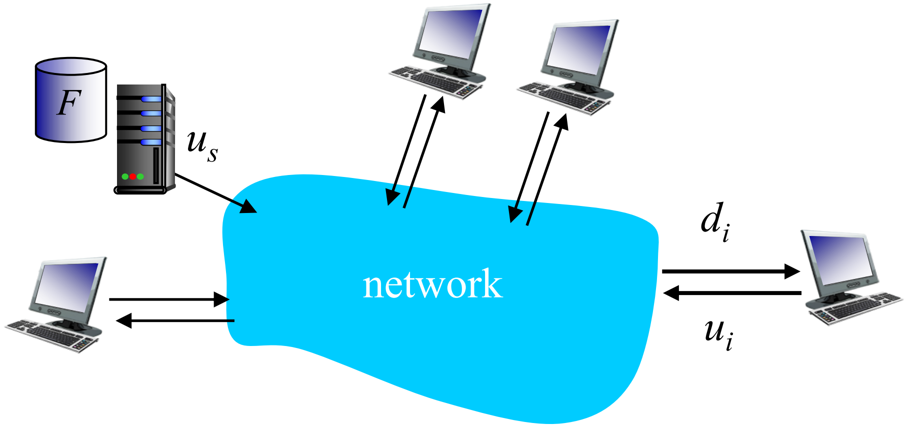
time to distribute F to N clients using client-server approach

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

increases linearly in N

Client/server vs. P2P architecture

File distribution time using P2P



- *server transmission:* must upload at least one copy
  - ✓ time to send one copy: F/u_s
- *client:* each client must download file copy
  - ✓ max client download time: F/d_{min}
- *clients:* as aggregate must download *NF* bits
  - max upload rate (limiting max download rate) is $u_s + \Sigma u_i$

time to distribute F to N clients using P2P approach

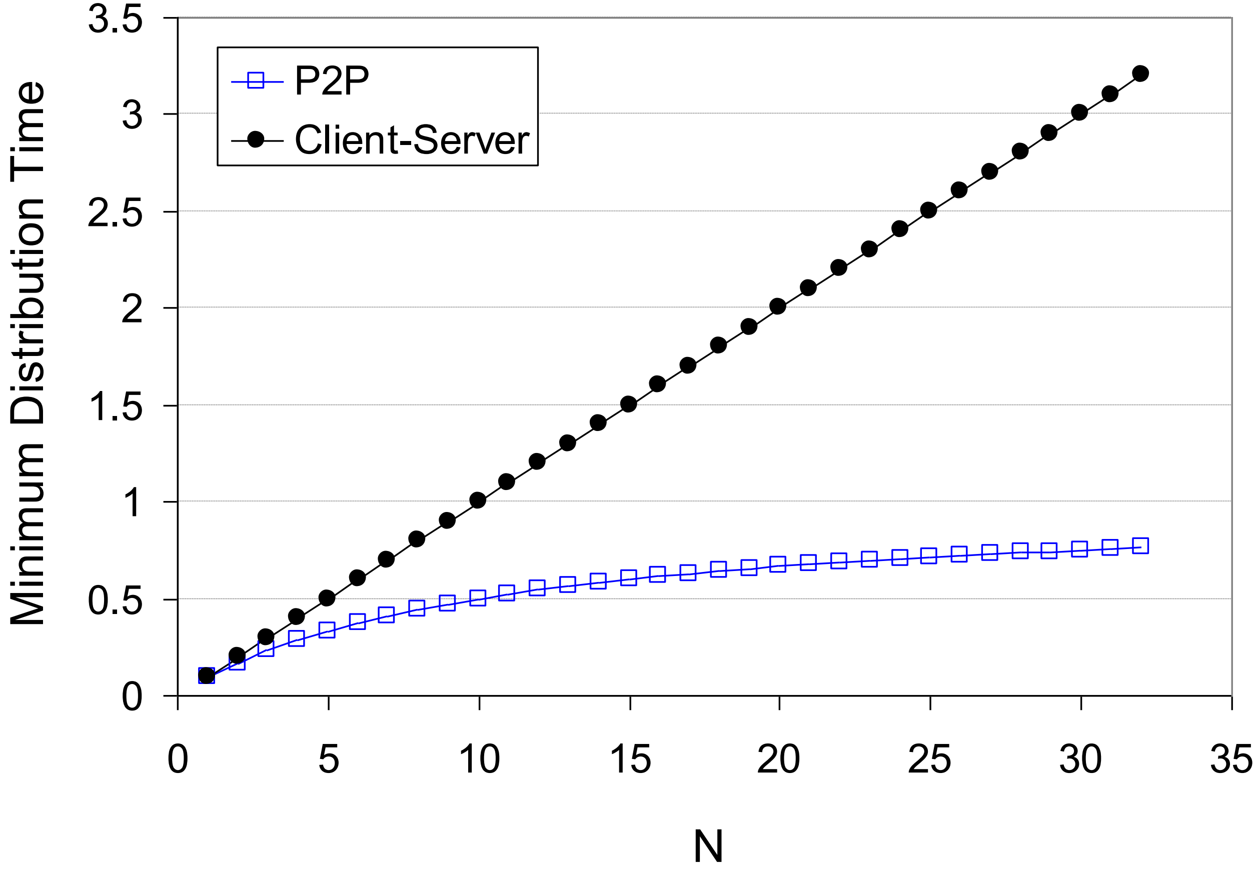
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

increases linearly in *N* …

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

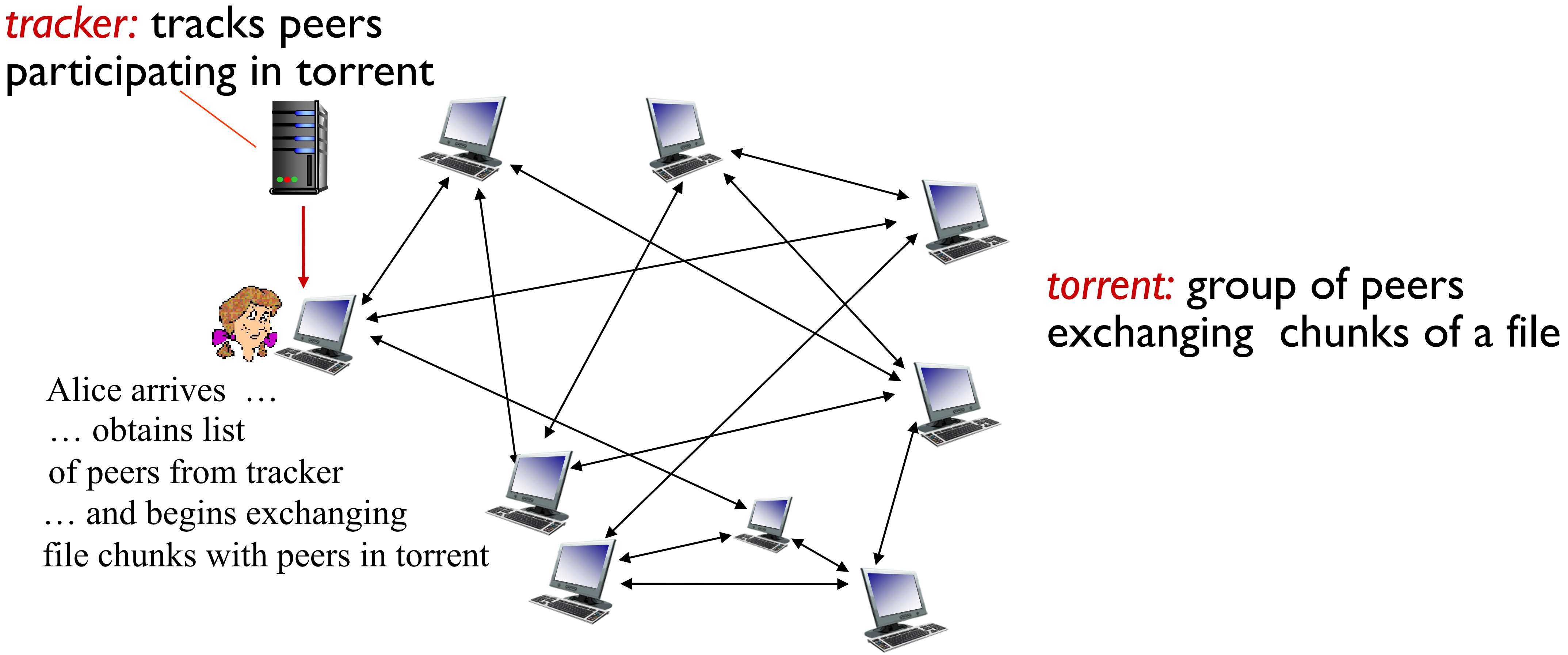
Client/server vs. P2P architecture

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



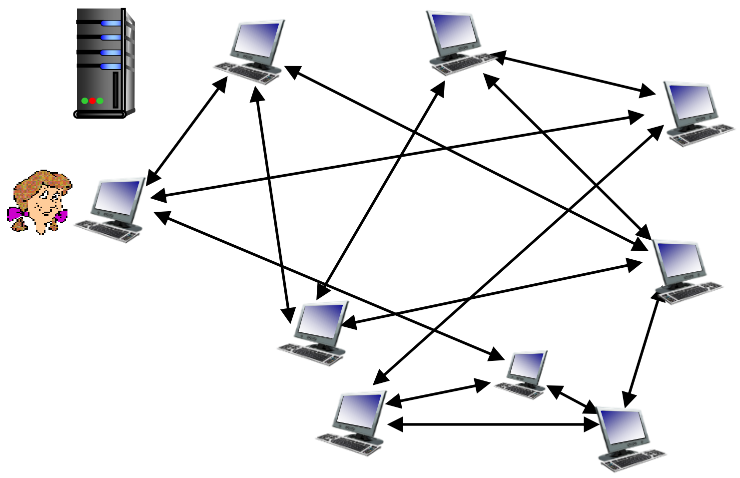
P2P file distribution protocol: BitTorrent

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



BitTorrent: initialization, and dynamics

- New 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
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent
  - Peers may come and go (referred to as "churn")

BitTorrent: requesting and sending file chunks

Chunk selection: rarest first

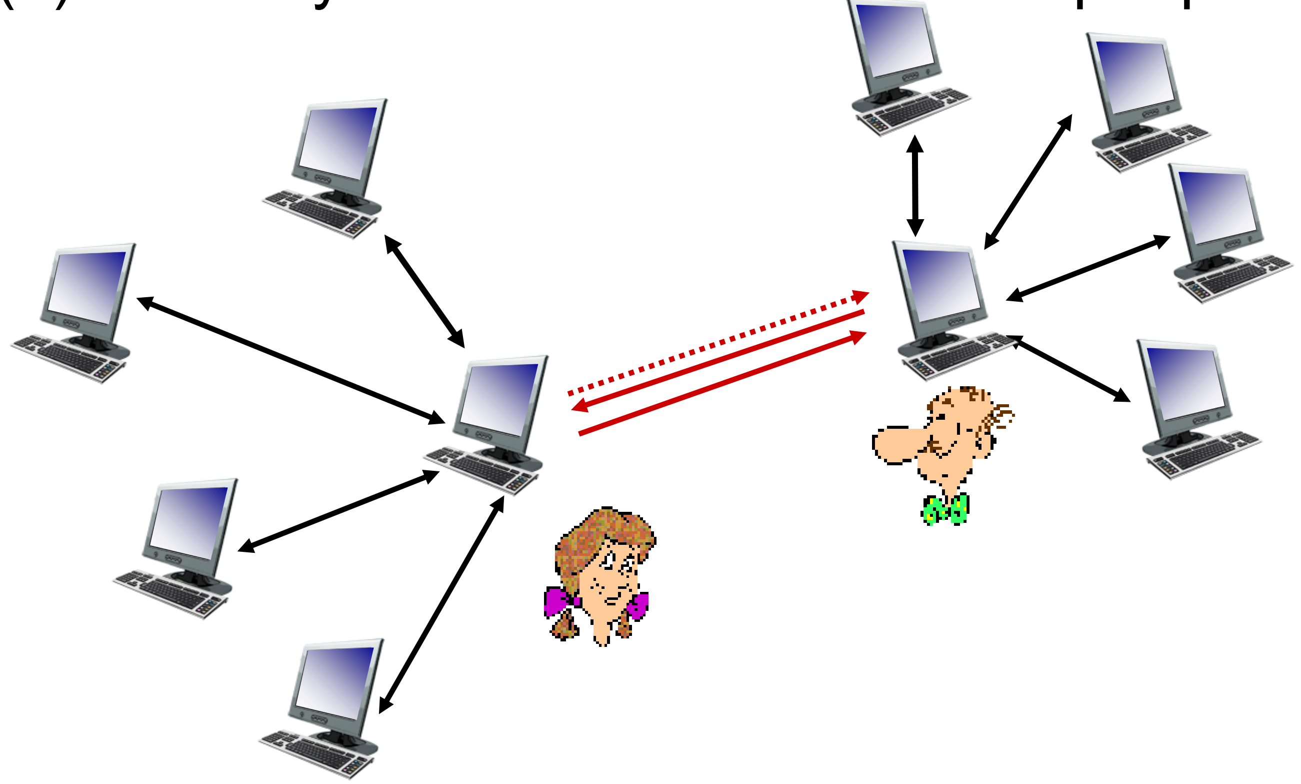
- at any given time, different peers have different subsets of file chunks
- periodically, Alice asks each peer for list of chunks that they have
- Alice requests missing chunks from peers, rarest first

Peer selection: tit-for-tat with optimistic unchoke

- Alice sends chunks to those 4 peers currently sending her chunks *at highest rate*
  - ✓ other peers are choked by Alice (do not receive chunks from her)
  - ✓ re-evaluate top 4 every 10 seconds
- Exploring new peers: randomly select another peer every 30 seconds, starts sending chunks
  - ✓ "optimistically unchoke" this peer
  - ✓ newly chosen peer may join top 4

BitTorrent peer selection example

(1) Alice "optimistically unchokes" Bob
(2) Alice becomes one of Bob's top 4 providers; Bob reciprocates
(3) Bob may become one of Alice's top 4 providers as well



Can tit-for-tat completely remove free-riders?

DHT: another P2P application

- DHT: a distributed P2P database
- Database has (key, value) pairs; examples:
  - key: SSN; value: human name
  - key: movie title; value: IP address
- DHT: 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

DHT: another P2P application

- Motivation of building a distributed database
  - Realizing content addressable network
  - Scalability: No peer is powerful enough to keep a million-entry database
- Use cases: real-world P2P file-sharing or P2P multimedia systems
  - e.g., BitTorrent can use a DHT to create a distributed tracker.
    - ✓ Key: the torrent identifier
      Value: IP addresses of all the peers currently participating in the torrent

How to assign (key, value) entries to peers?

- The central issue of DHT!
- Basic idea:
  - Convert each key to an integer
  - Assign integer “ID” to each peer
  - Put (key,value) pair in the peer whose integer ID is closest to the key

How to assign (key, value) to peers?

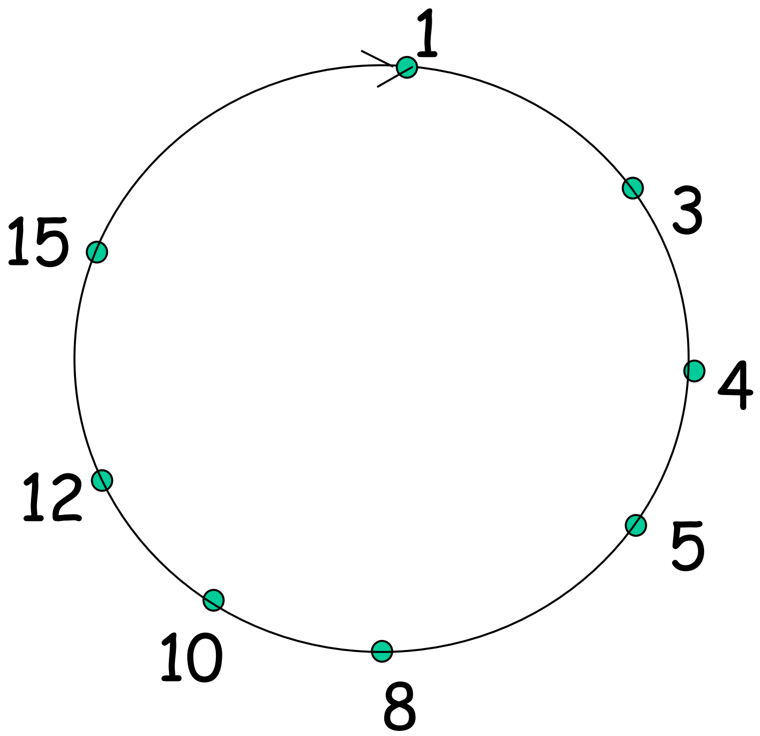
- assign integer identifier to each peer in range $[0, 2^n - 1]$ for some n
  - each identifier represented by n bits.
- require each key to be an integer in same range
  - to get integer key, hash original key
    - ✓ e.g., key = hash("Led Zeppelin IV")
  - This is why it is referred to as a distributed "hash" table (DHT)

A hash function is a many-to-one function for which two different inputs can have the same output (same integer), but the likelihood of having the same output is extremely small.
Think of it as a function that compresses information into a few digits.

How to assign (key, value) to peers?

- rule: assign key to the peer that has the closest ID.
  - Closest: equals the key, or 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 (modulo operation)

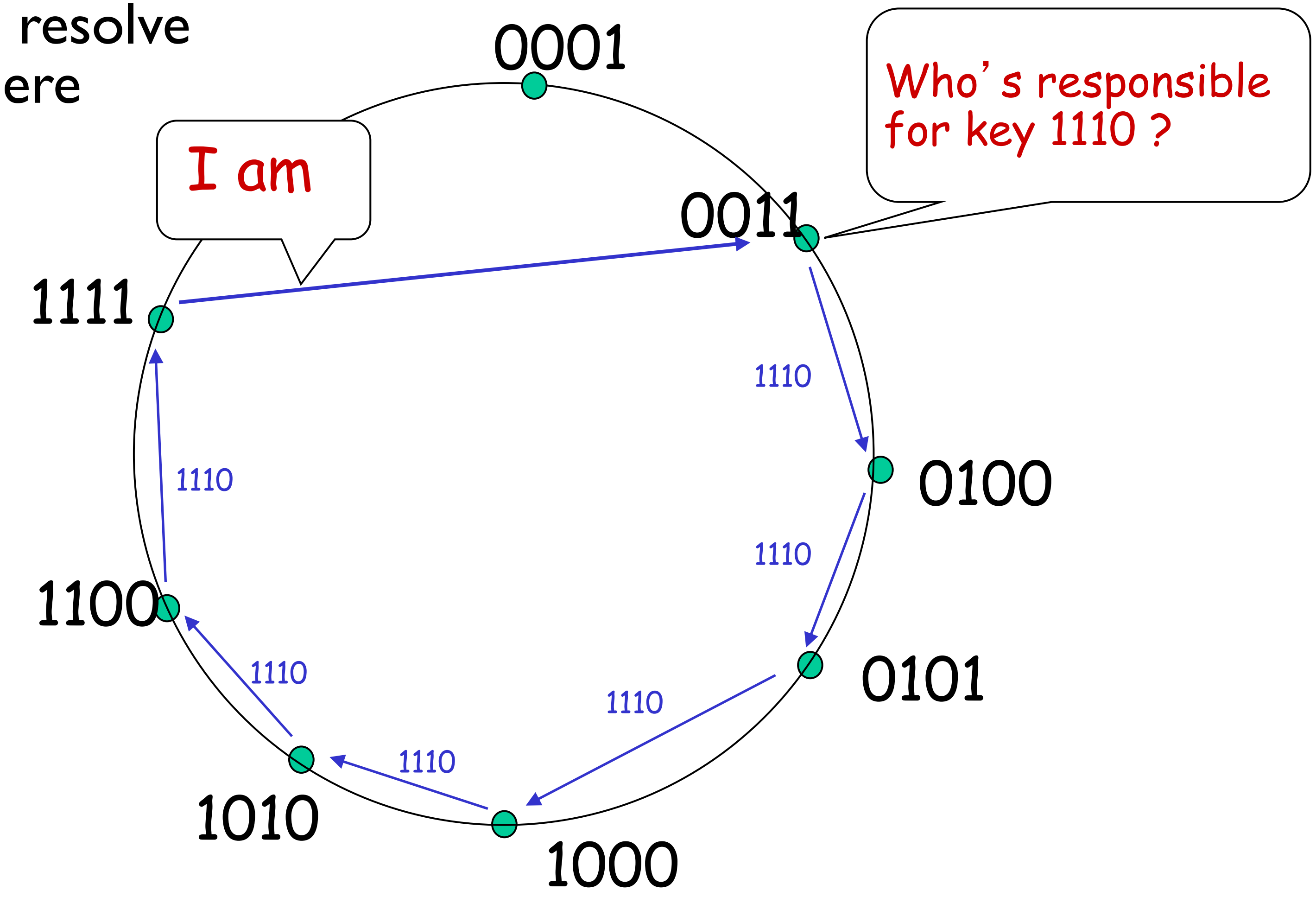
(key, value) assignment in a circular DHT



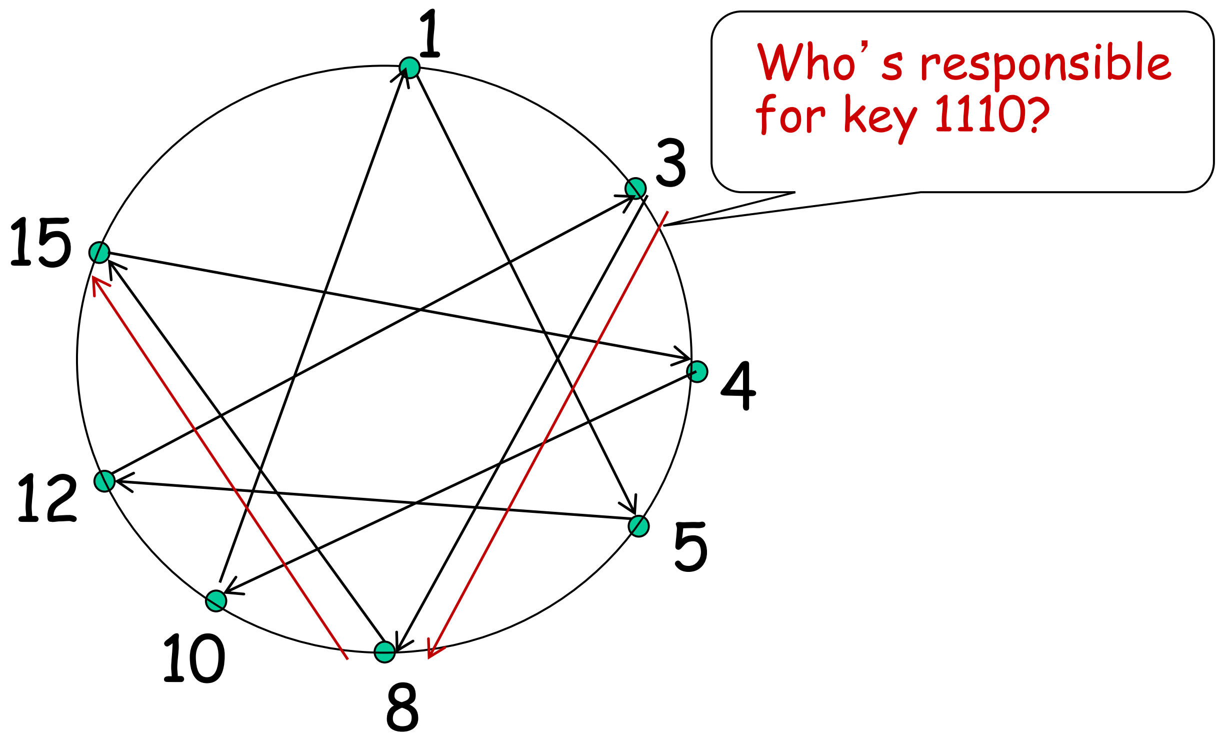
- Each peer *only* aware of immediate successor and predecessor.
- An application layer "overlay network"

(key, value) assignment in a circular DHT

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



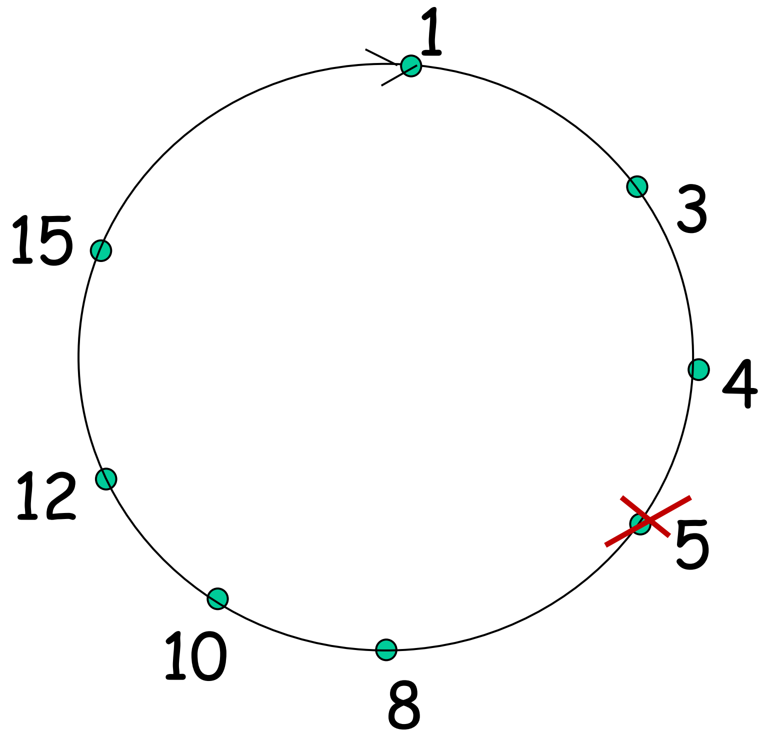
(key, value) assignment in a circular DHT, with shortcuts



- each peer keeps track of IP addresses of predecessor, successor, short cuts.
- reduced from 6 to 2 messages.
- Proven result: possible to design shortcuts so *O(log N)* neighbors, *O(log N)* messages in query

Handling peer churns in circular DHT

- How to handle peer churns?
  - 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, and it only knows peer 1?

Content distribution networks (CDN)

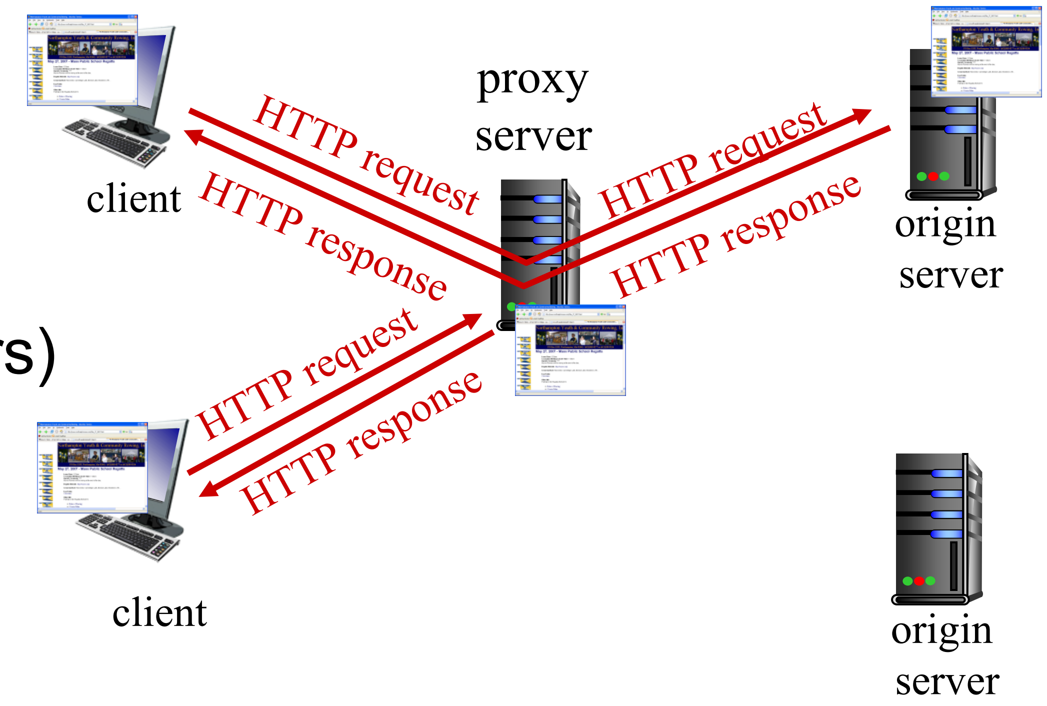
Why CDN?

- Challenge: how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- Potential solution: single, large “mega-server”?
  - single point of failure
  - point of network congestion
  - long (and possibly congested) path to distant clients

Centralized solution doesn’t scale well!

Why CDN?

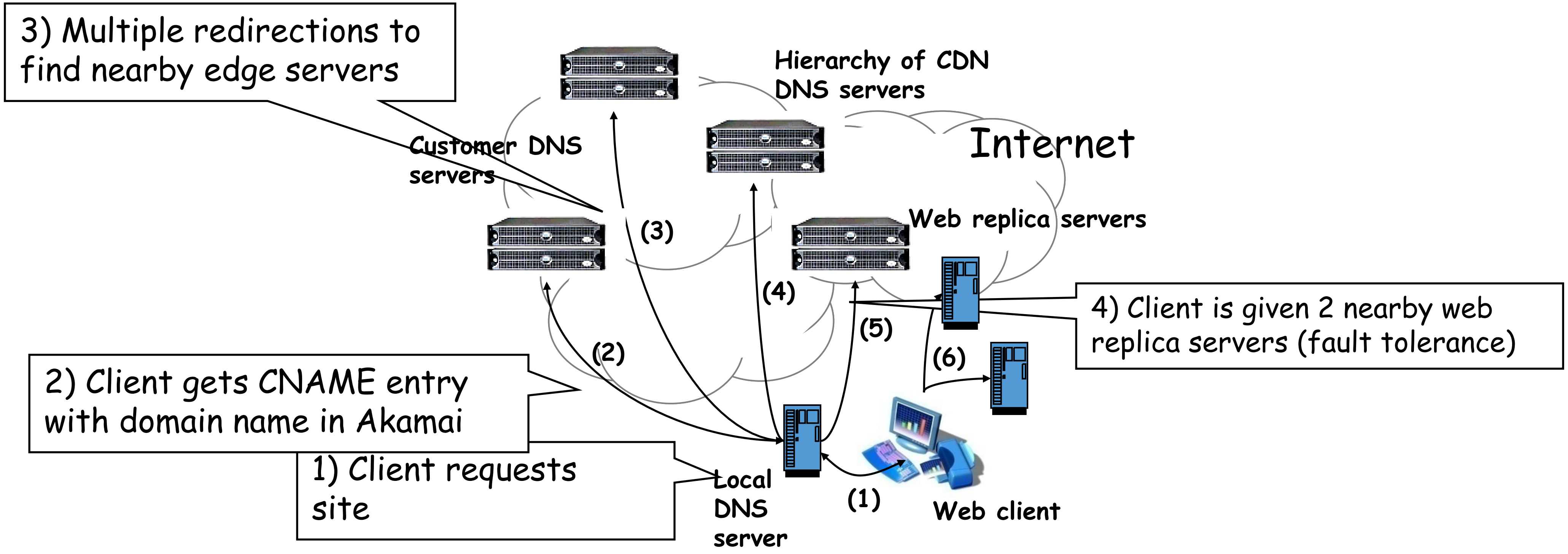
- Recap: web cache
- Can web cache satisfy needs of all content publishers? e.g., YouTube
  - Freshness of content
  - Performance scalability (multiple servers)
  - Flexibility
  - Direct control
- CDN is thus designed, e.g.,
  - Akamai, YouTube CDN



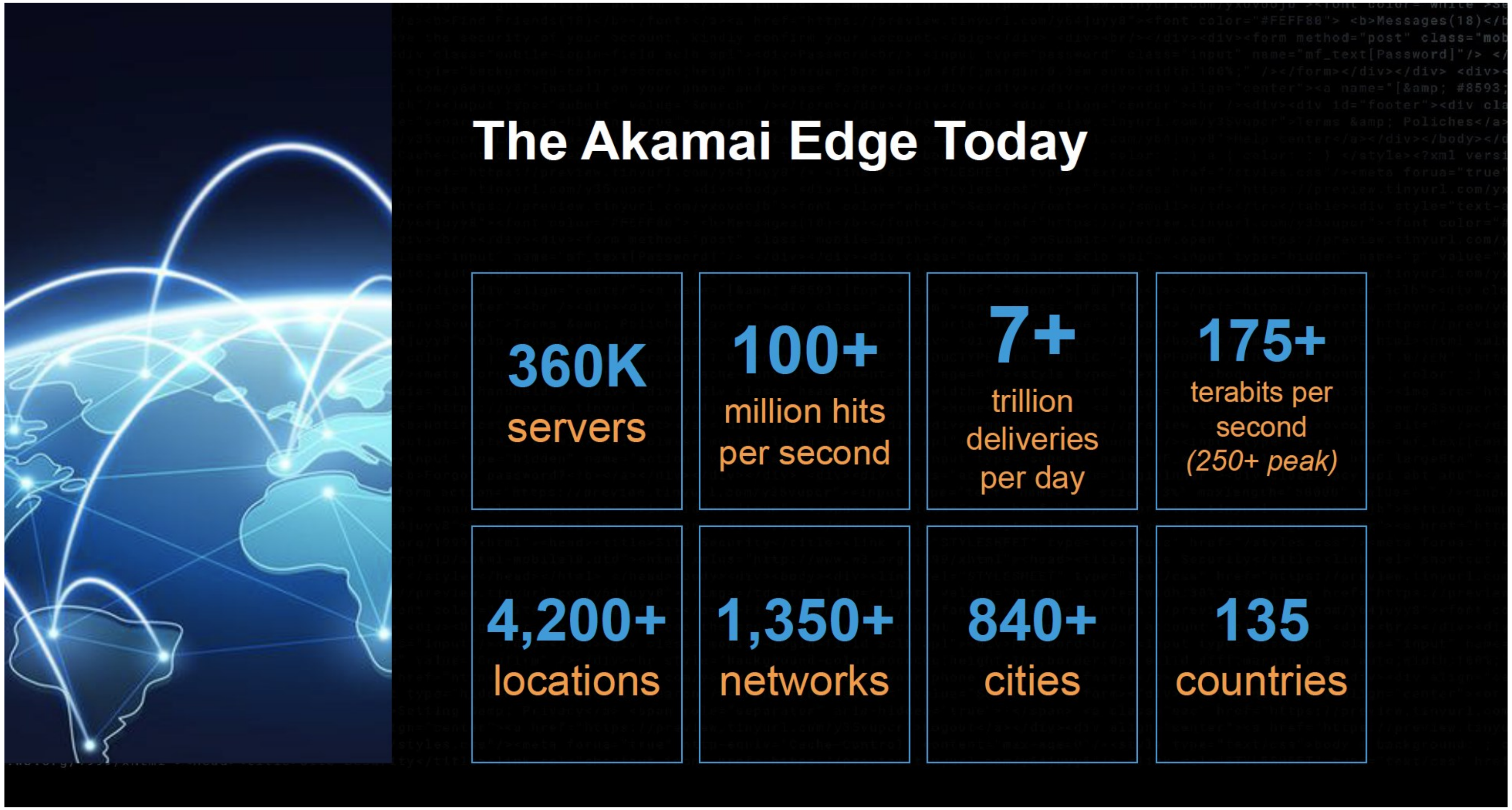
Basics of Akamai CDN

- Content publisher (e.g., CNN, NYTimes)
  - provides base HTML documents
  - runs origin server(s)
- Akamai
  - Runs edge servers for hosting content;
    deep deployment into 1000+ networks
  - Direct control
- customized DNS redirection servers to select edge servers based on
  - closeness to client browser
  - server load

Akamai load direction flow



Akamai today



Source: https://networkingchannel.eu/living-on-the-edge-for-a-quarter-century-an-akamai-retrospective-downloads/

Review questions

- ✓ P2P: quantitative understanding of the advantage of P2P file distribution in comparison to client/server; Important operations of BitTorrent, e.g., chunk request, chunk selection, peer selection, optimistic unchoking, tit-for-tat
- ✓ DHT: motivation and use cases; assignment of (key, value) to a circular DHT; handling peer churns
- ✓ Web caching vs. CDN (Why is CDN needed if we already have Web cache?)

References

- ✓ Chapter 2.6 (Peer-to-Peer Applications) of the book "Computer networking : a top-down approach"
- ✓ Chapter 7.2.4 (Content Distribution Networks) of the book "Computer networking : a top-down approach"