Computer Networks and Applications

COMP 3331/COMP 9331

Week 4

P2P + CDN

Reading Guide: Chapter 2, 2.5, 2.6, 2.7

Application Layer: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

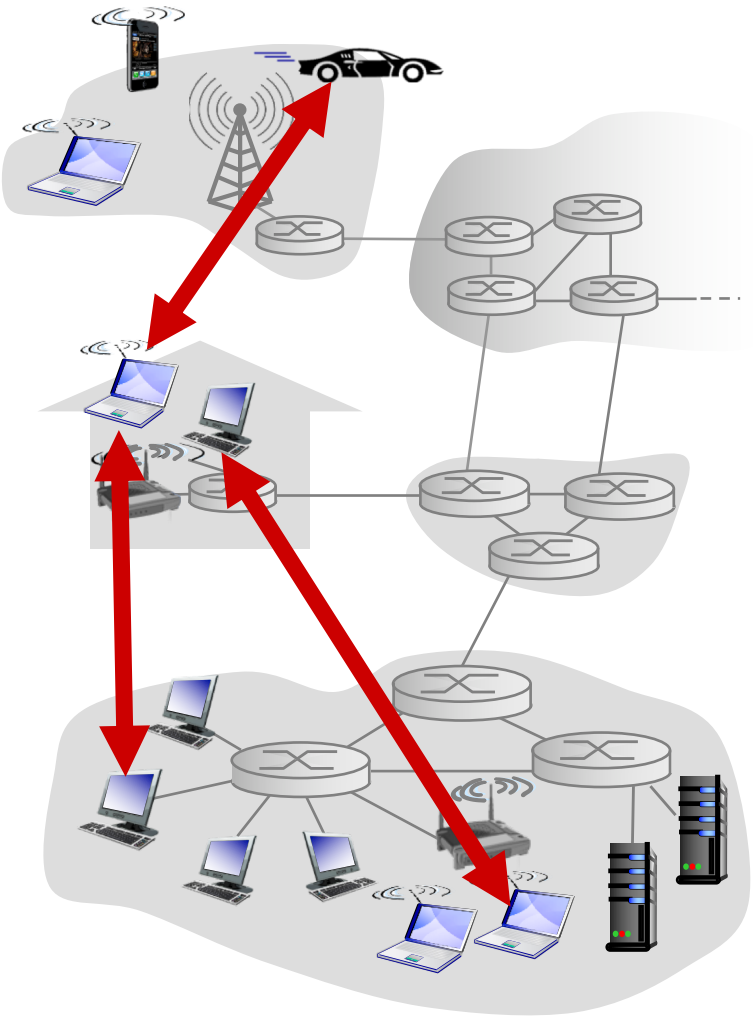
2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

Self study

Pure P2P architecture

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



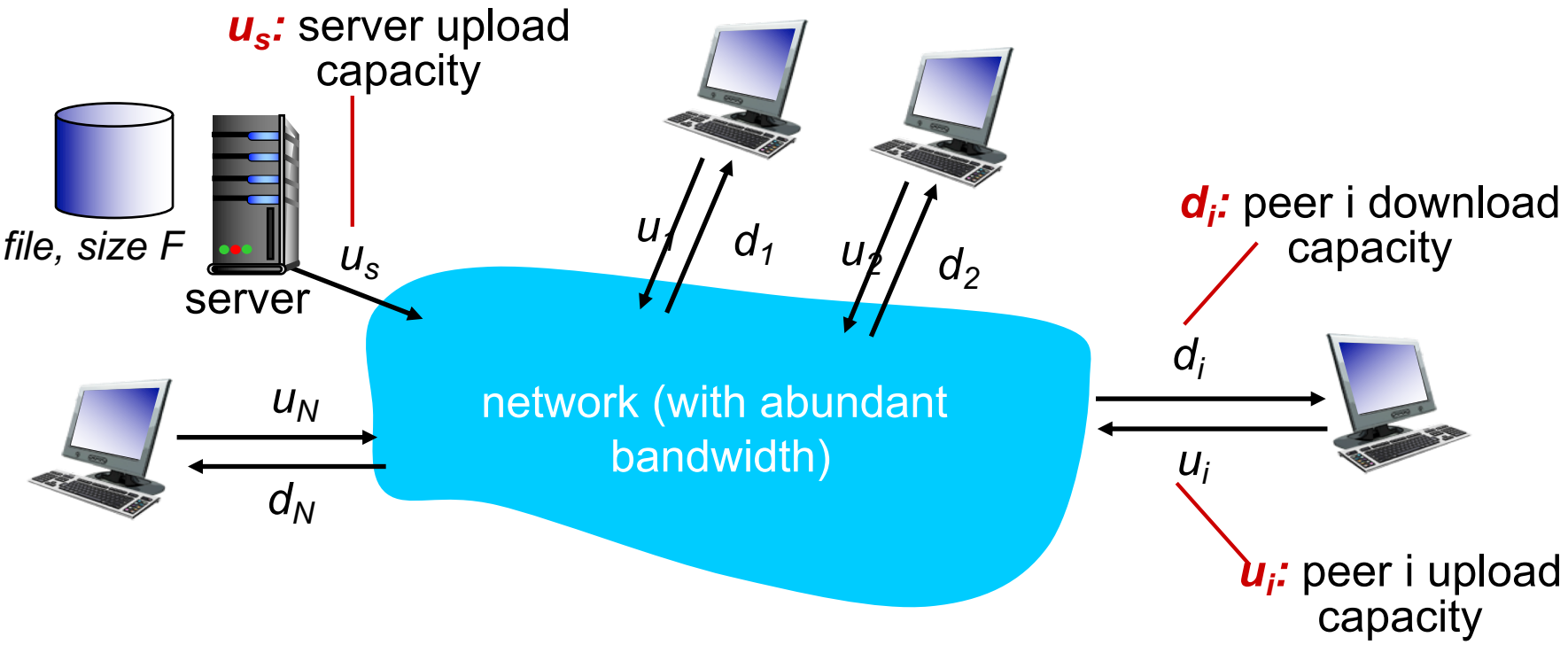
examples:

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

File distribution: client-server vs P2P

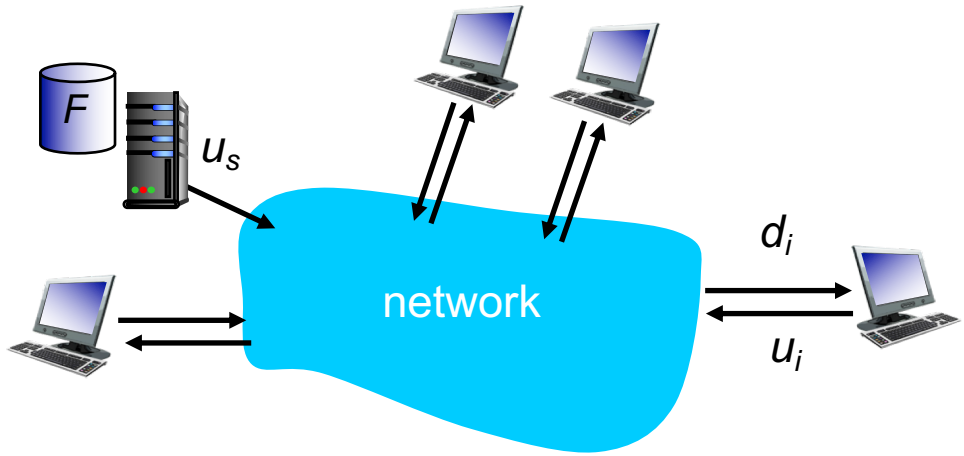
Question: how much time to distribute file (size *F*) from one server to *N* peers?

- peer upload/download capacity is limited resource



File distribution time: client-server

- *server transmission:* must 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
  - client download time: F/d_{min}

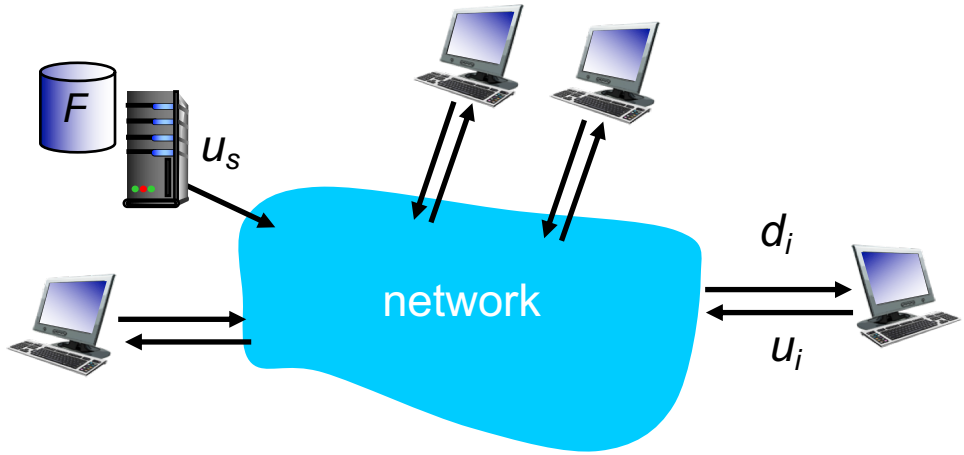
time to distribute F to N clients using client-server approach

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

increases linearly in N

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
  - 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

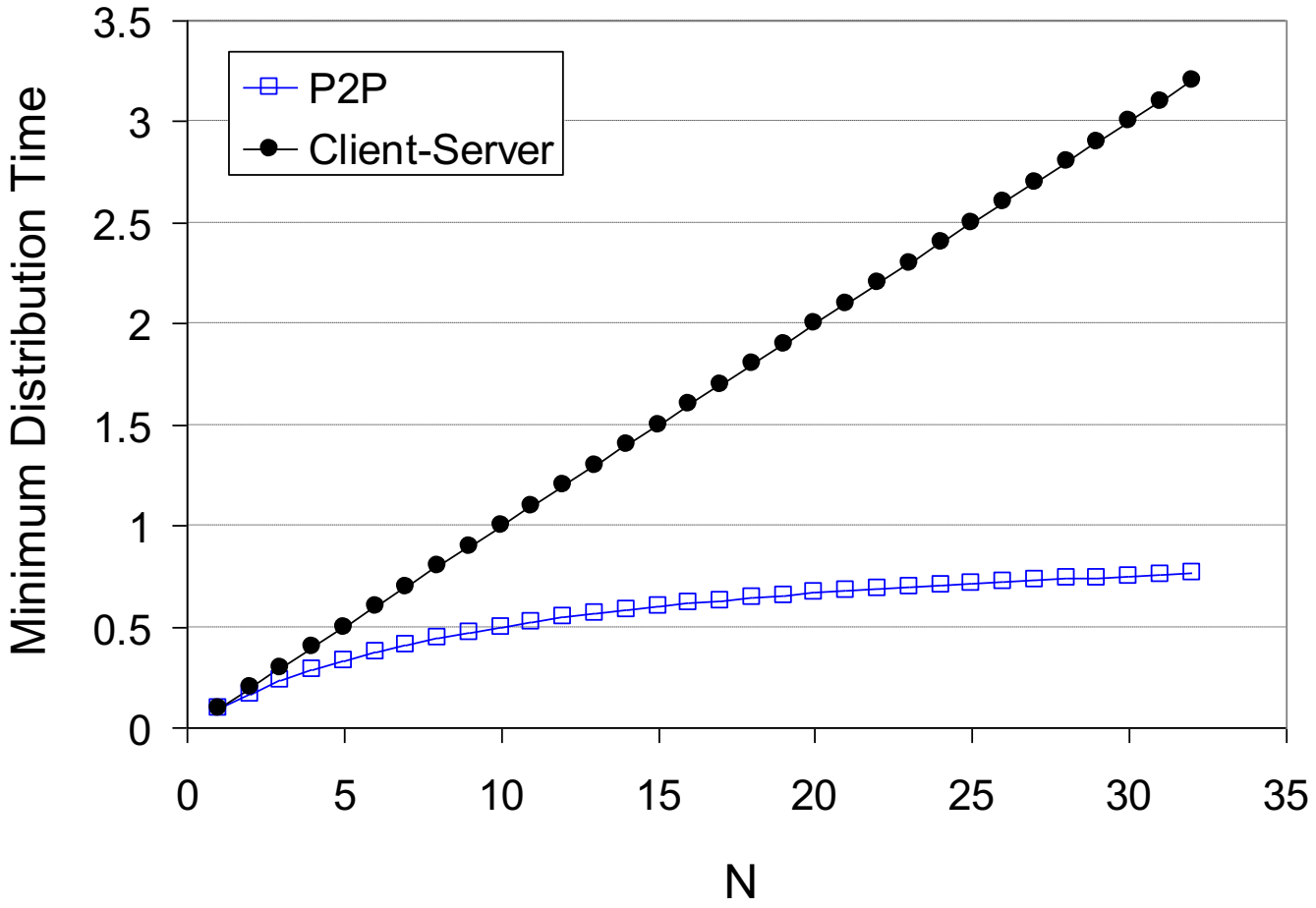
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \sum_{i=1}^{N} u_i)\}$$

increases linearly in *N* …

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

Client-server vs. P2P: example

client upload rate = u, F/u = 1 hour, $u_s = 10u$

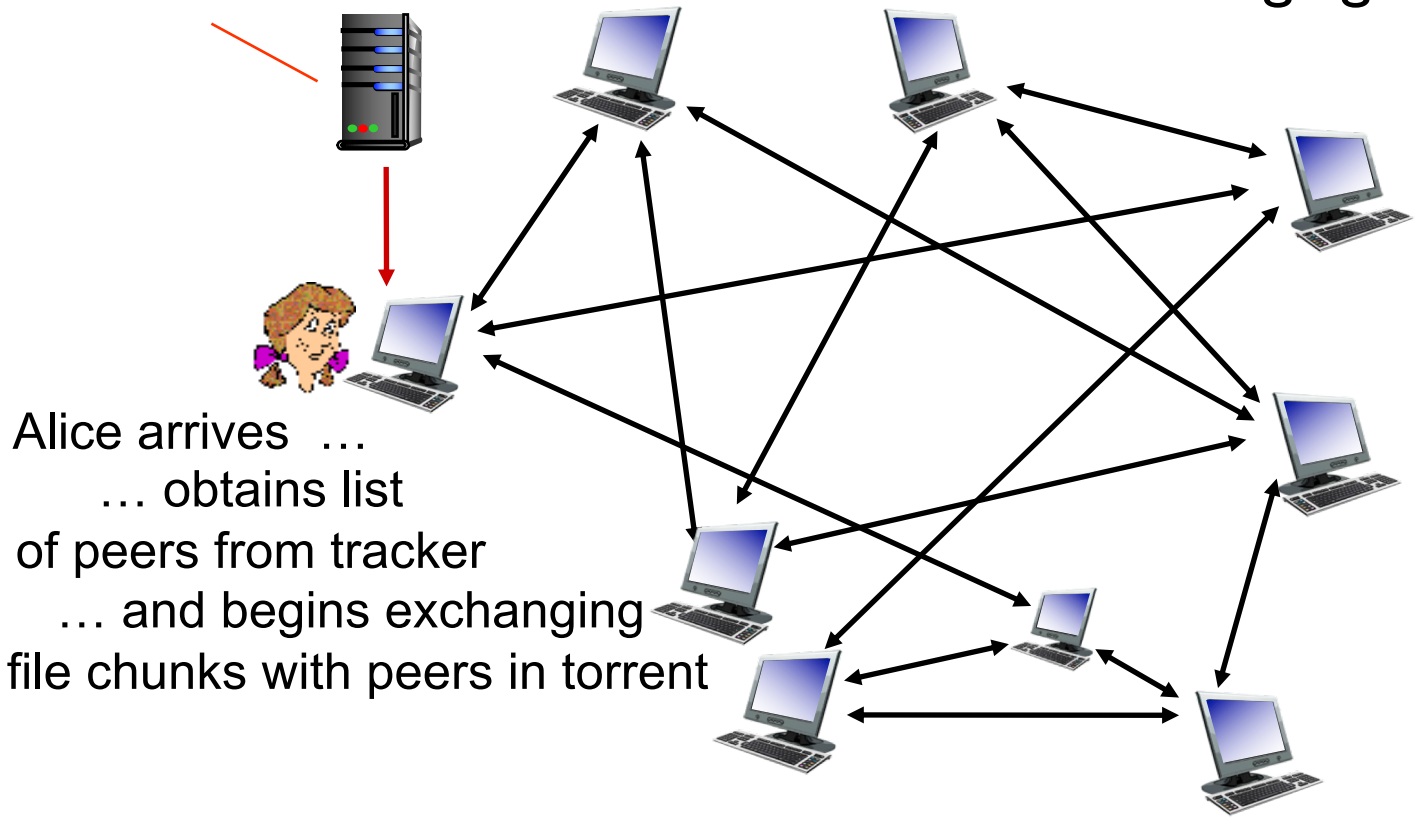


P2P file distribution: BitTorrent

- 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

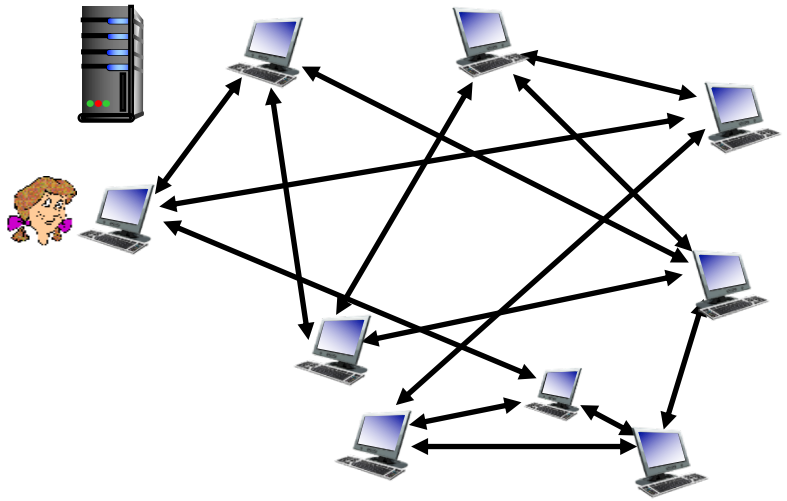
.torrent files

- Contains address of trackers for the file
  - Where can I find other peers?
- Contain a list of file chunks and their cryptographic hashes
  - This ensures that chunks are not modified

Title	Trackers
House of Cards Season 4	Tracker1-url
Walking Dead Season 6	Tracker2-url
Game of Thrones Season 8	Tracker2-url, Tracker3-url

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 ("neighbours")



- 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
- **Q:** Why rarest first?

sending chunks: tit-for-tat

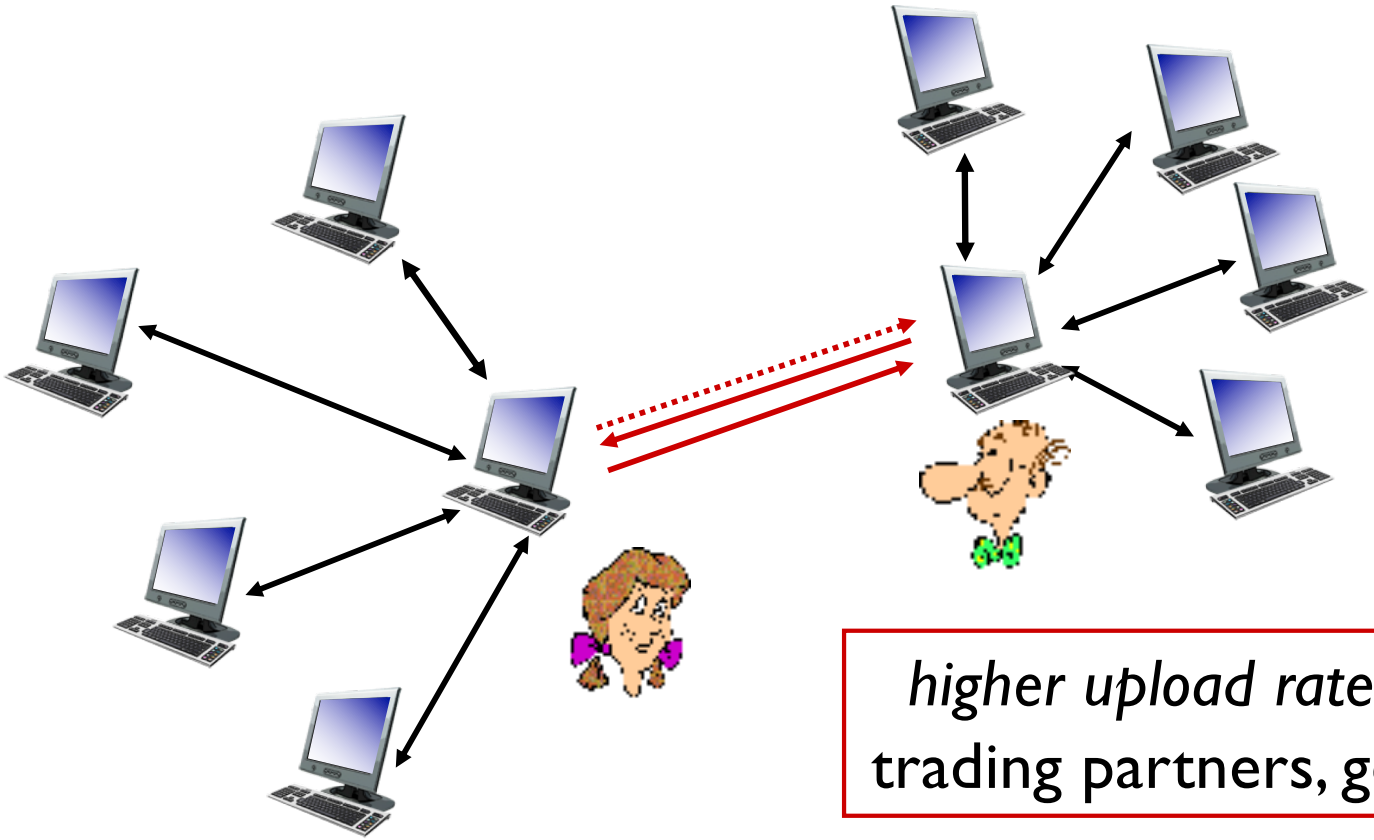
- Alice sends chunks to those four peers currently sending her chunks *at highest rate*
  - other peers are choked by Alice (do not receive chunks from her)
  - re-evaluate top 4 every10 secs
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke" this peer
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

Original Research Paper on BitTorrent linked to WebCMS

Quiz: Free-riding

- Suppose Todd joins a BitTorrent torrent, but he does not want to upload any data to any other peers. Todd claims that he can receive a complete copy of the file that is shared by the swarm. Is Todd's claim possible? Why or Why not?

Getting rid of the server/tracker

- Distribute the tracker information using a Distributed Hash Table (DHT)

- A DHT is a lookup structure
  - Maps keys to an arbitrary value
  - Works a lot like, well …. hash table

Content available in 6th Edition of the textbook Section 2.6.2

Hash table - review

- (key,value) pairs
- Centralised hash table – all (key,value) pairs on 1 node
- Distributed hash tables – each node has a "section" of (key,value) pairs

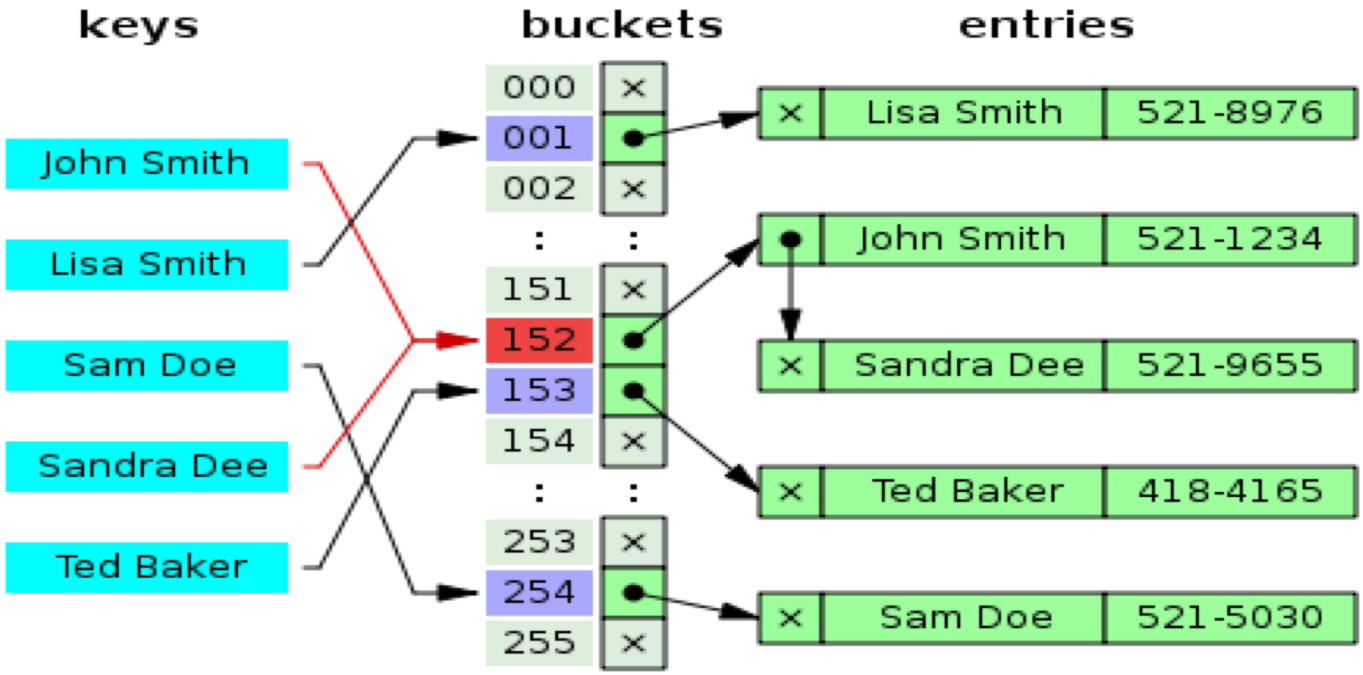


Figure src: http://en.wikipedia.org/wiki/Hash_table

Distributed Hash Table (DHT)

- DHT: a *distributed P2P database*
- database has (key, value) pairs; examples:
  - key: TFN number; value: human name
  - key: file name; value: BT tracker(s)
- 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?

- 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

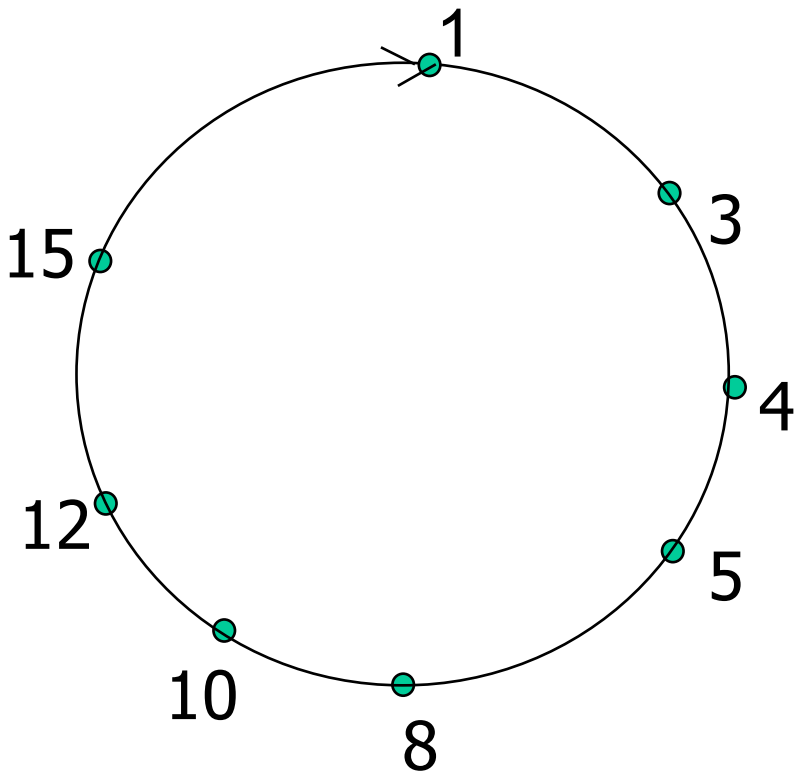
DHT identifiers: Consistent Hashing

- assign integer identifier to each peer in range $[0,2^n-1]$ for some *n*-bit hash function
  - E.g., node ID is hash of its IP address
- require each key to be an integer in same range
- to get integer key, hash original key
  - e.g., key = hash("House of Cards Season 4")
  - 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.
- common convention: closest is the *immediate successor* of the key.
- e.g., *n*=4; all peers & key identifiers are in the range [0-15], peers: 1,3,4,5,8,10,12,14;
  - key = 13, then successor peer = 14
  - key = 15, then successor peer = 1

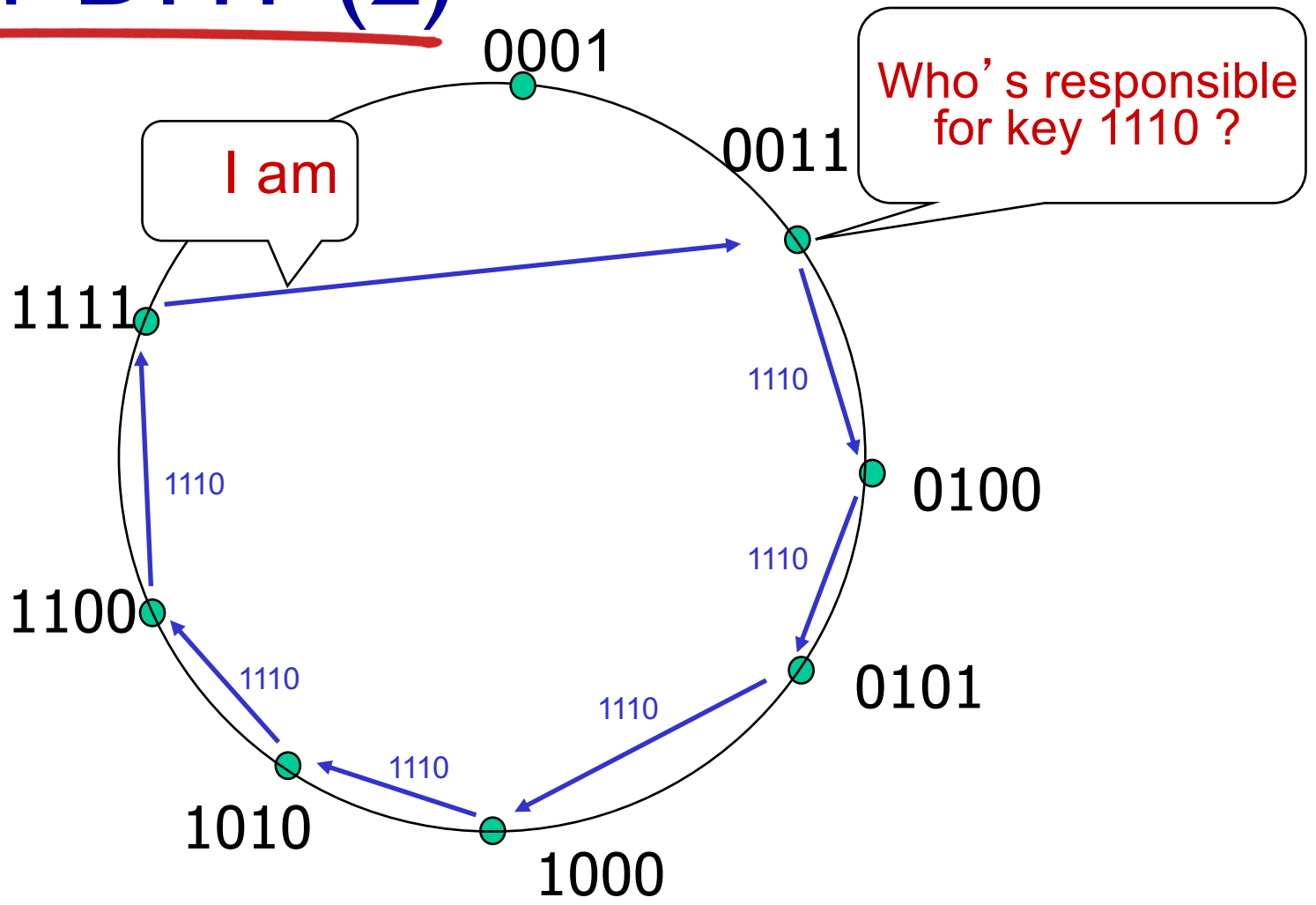
Circular DHT (1)



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

Circular DHT (2)

Define closest as closest successor



Worst case all peers probed, N messages, on average N/2
Mesh overlay (each peer tracks all other N-1 peers) only one message is sent per query

Application Layer: outline

2.1 principles of network applications

2.2 Web and HTTP

2.3 electronic mail

- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

2.7 socket programming with UDP and TCP

Self study

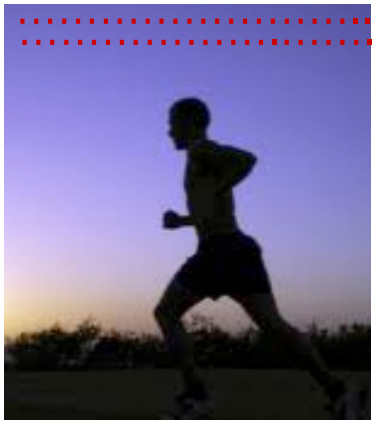
Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1.8B YouTube users, ~140M Netflix users
- challenge: scale - how to reach ~2B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution:* distributed, application-level infrastructure

Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)



frame *i*

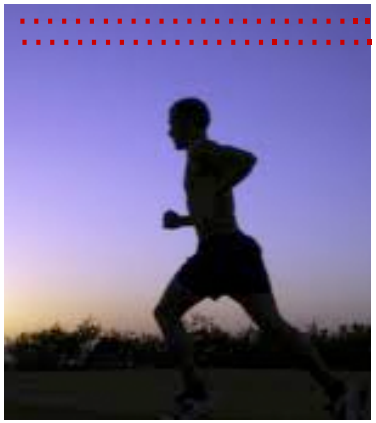
frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Multimedia: video

- CBR: (constant bit rate): video encoding rate fixed
- VBR: (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes
- examples:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)



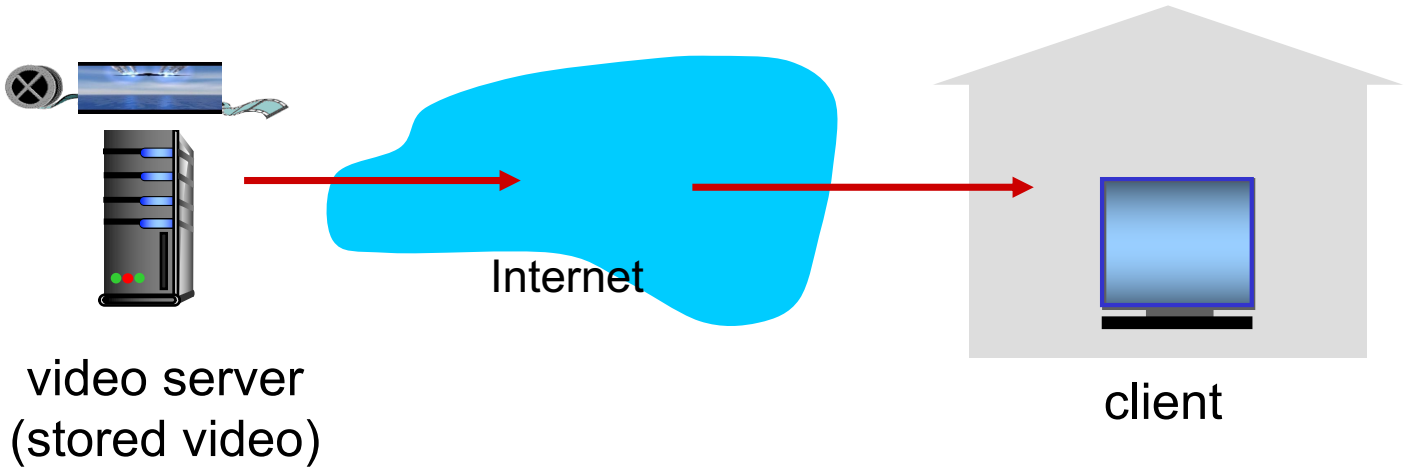
frame *i*

frame *i+1*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Streaming stored video:

simple scenario:

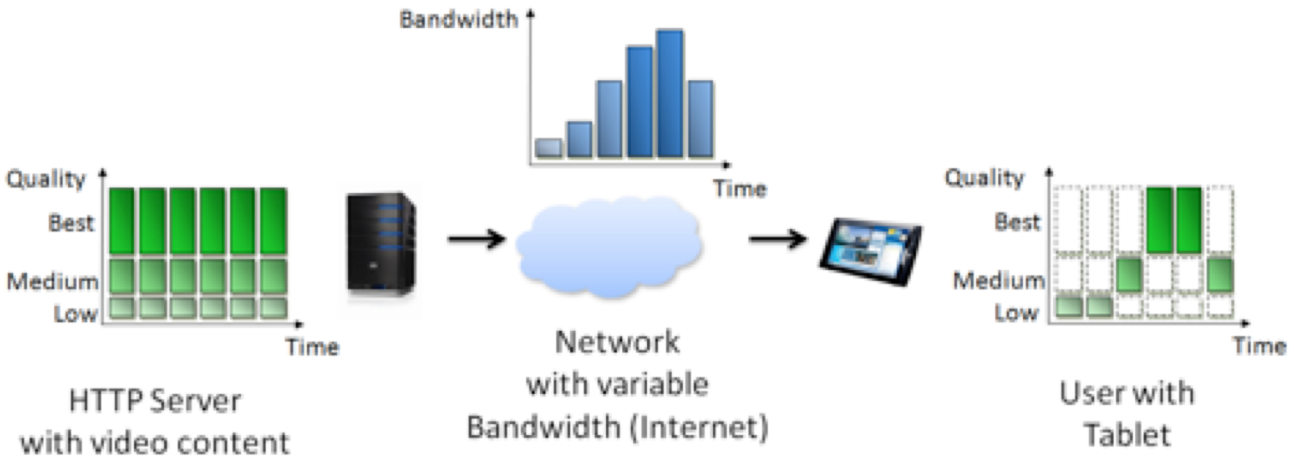


Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

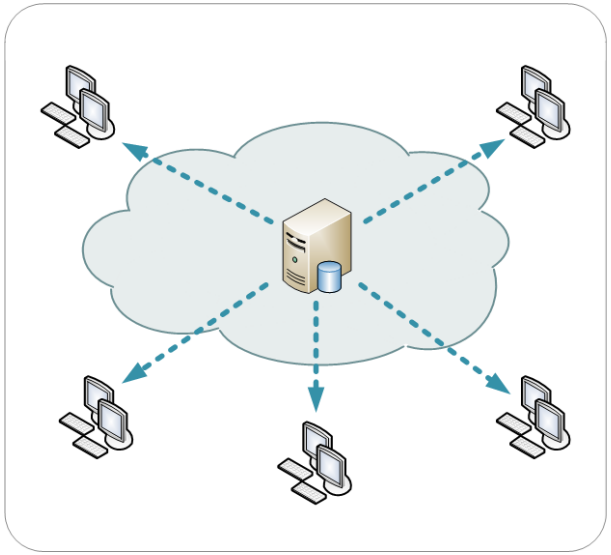
- *DASH: Dynamic, Adaptive Streaming over HTTP*
- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more bandwidth available)
  - *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)



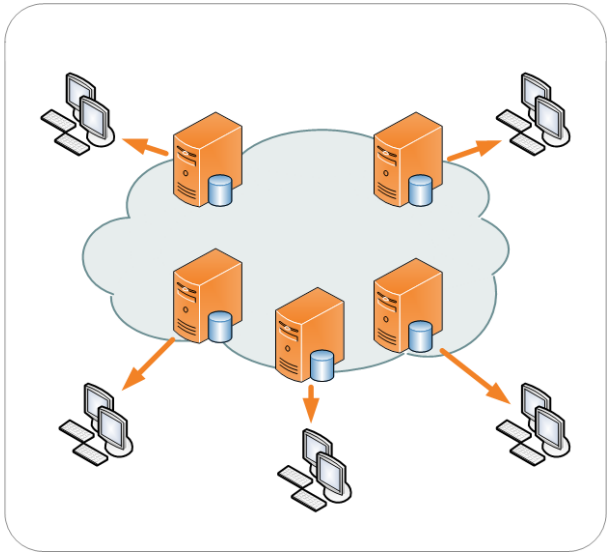
Content distribution networks

- Caching and replication as a service (amortise cost of infrastructure)
- Goal: bring content close to the user
- Large-scale distributed storage infrastructure (usually) administered by one entity
  - *e.g.*, Akamai has servers in 20,000+ locations
- Combination of (pull) caching and (push) replication
  - **Pull:** Direct result of clients' requests
  - **Push:** Expectation of high access rate

Single server



CDN



An example

```
bash-3.2$ dig www.mit.edu

; <<>> DiG 9.8.3-P1 <<>> www.mit.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 27387
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 9, ADDITIONAL: 9

;; QUESTION SECTION:
;www.mit.edu.                   IN      A

;; ANSWER SECTION:
www.mit.edu.            1800    IN      CNAME   www.mit.edu.edgekey.net.
www.mit.edu.edgekey.net. 60     IN      CNAME   e9566.dscb.akamaiedge.net.
e9566.dscb.akamaiedge.net. 20   IN      A       23.77.150.125

;; AUTHORITY SECTION:
dscb.akamaiedge.net.    681     IN      NS      n4dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      n5dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      a0dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      n6dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      n1dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      n3dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      n0dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      n7dscb.akamaiedge.net.
dscb.akamaiedge.net.    681     IN      NS      n2dscb.akamaiedge.net.

;; ADDITIONAL SECTION:
a0dscb.akamaiedge.net.  7144    IN      AAAA    2600:1480:e800::c0
n0dscb.akamaiedge.net.  3048    IN      A       88.221.81.193
n1dscb.akamaiedge.net.  2752    IN      A       88.221.81.194
n2dscb.akamaiedge.net.  1380    IN      A       104.72.70.167
n3dscb.akamaiedge.net.  3048    IN      A       88.221.81.195
n4dscb.akamaiedge.net.  2810    IN      A       104.71.131.100
n5dscb.akamaiedge.net.  1326    IN      A       104.72.70.166
n6dscb.akamaiedge.net.  49      IN      A       104.72.70.174
n7dscb.akamaiedge.net.  2554    IN      A       104.72.70.175

;; Query time: 246 msec
;; SERVER: 129.94.172.11#53(129.94.172.11)
;; WHEN: Thu Mar  9 18:04:37 2017
;; MSG SIZE  rcvd: 463
```

Many well-known sites are hosted by CDNs. A simple way to check using dig is shown here.

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

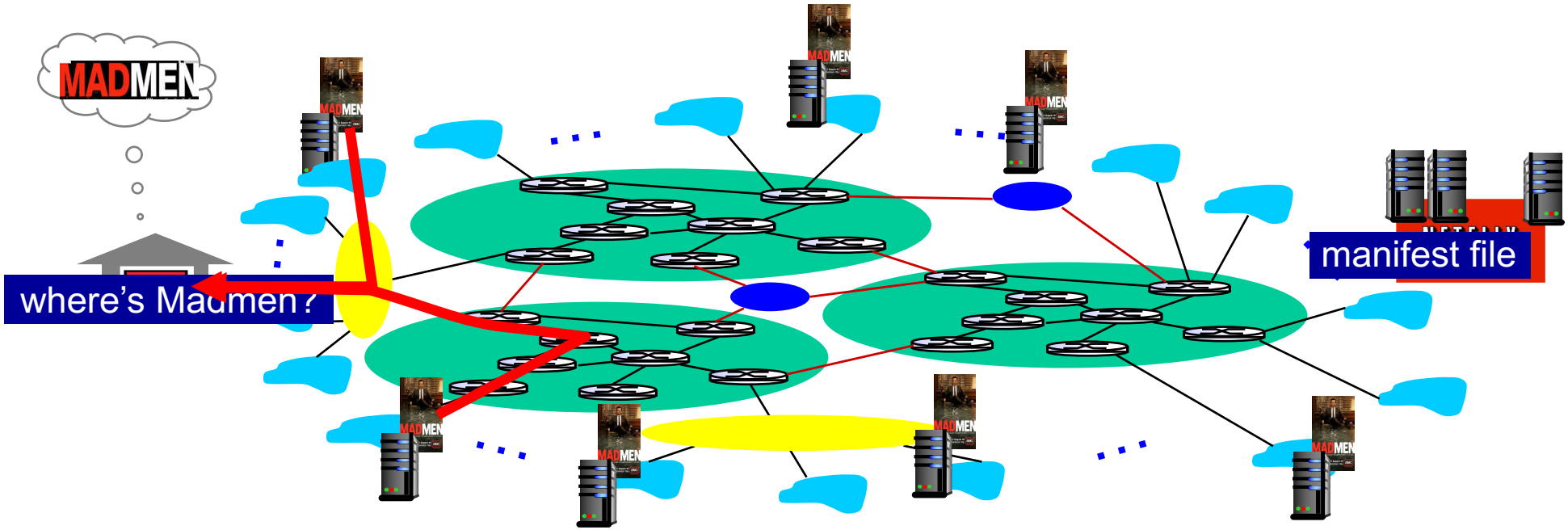
Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- *option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, thousands of locations
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Content Distribution Networks (CDNs)

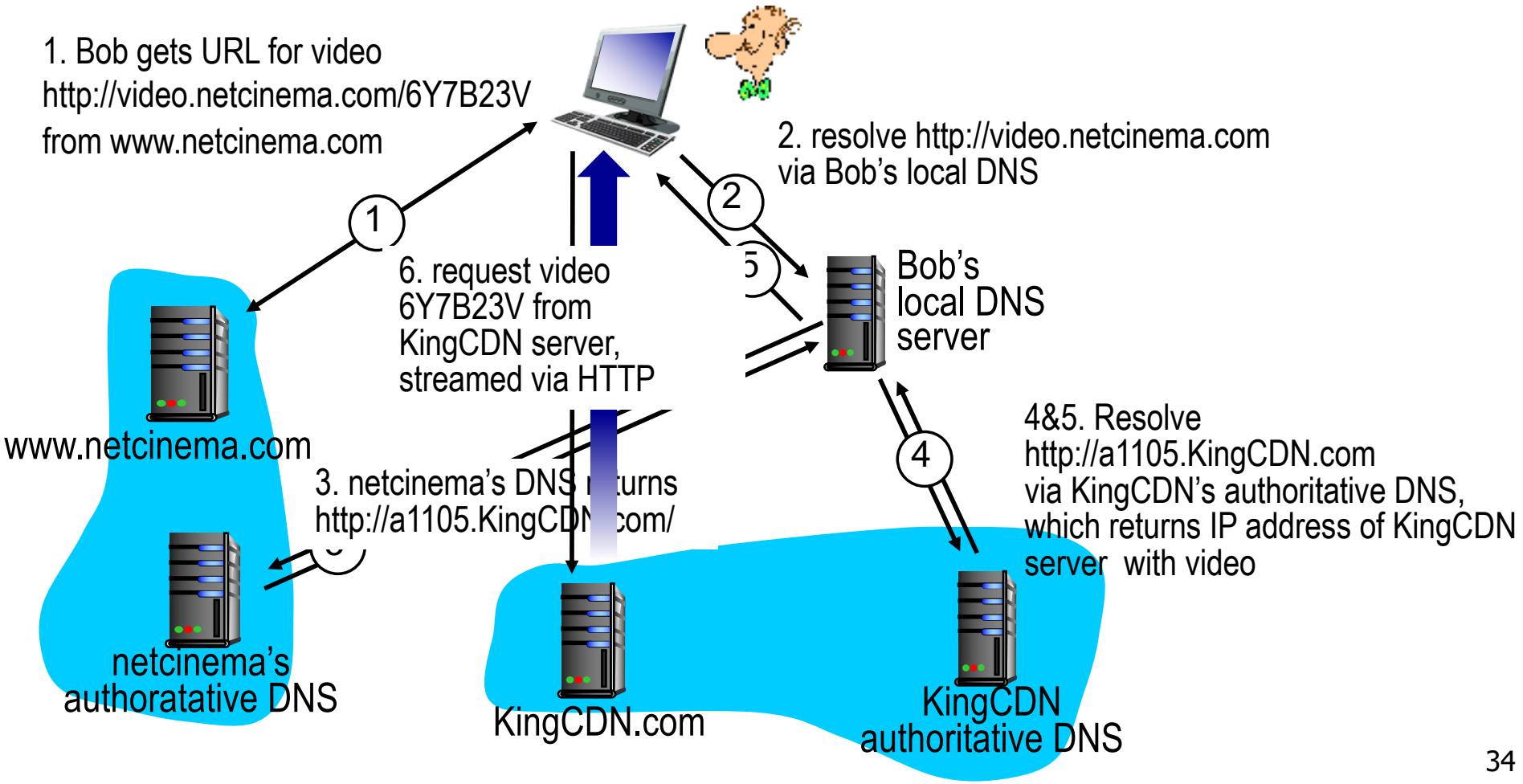
- CDN: stores copies of content at CDN nodes
  - e.g. Netflix stores copies of MadMen
- subscriber requests content from CDN
  - directed to nearby copy, retrieves content
  - may choose different copy if network path congested



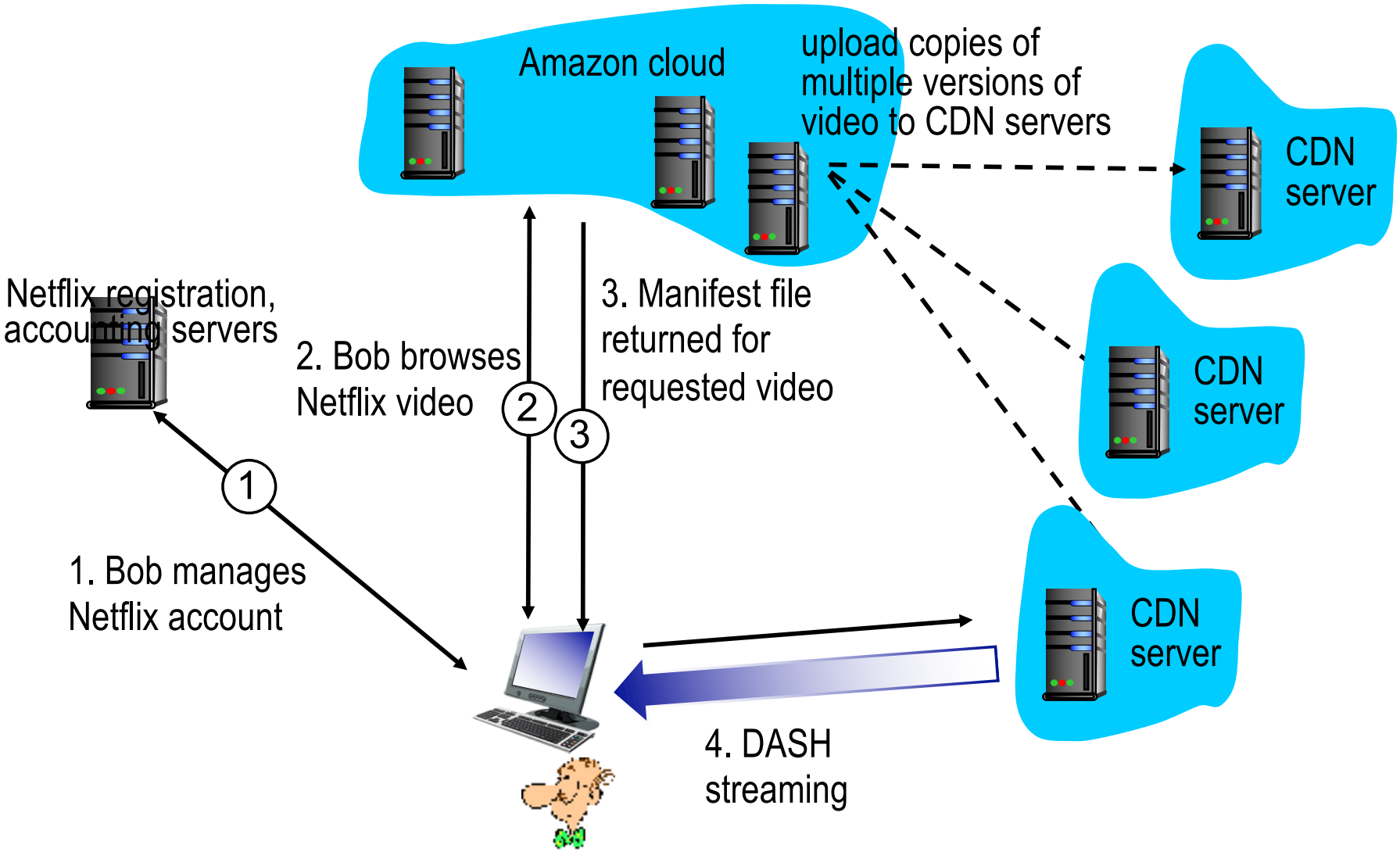
CDN content access: a closer look

Bob (client) requests video http://video.netcinema.com/6Y7B23V

- video stored in CDN at managed by KingCDN.com

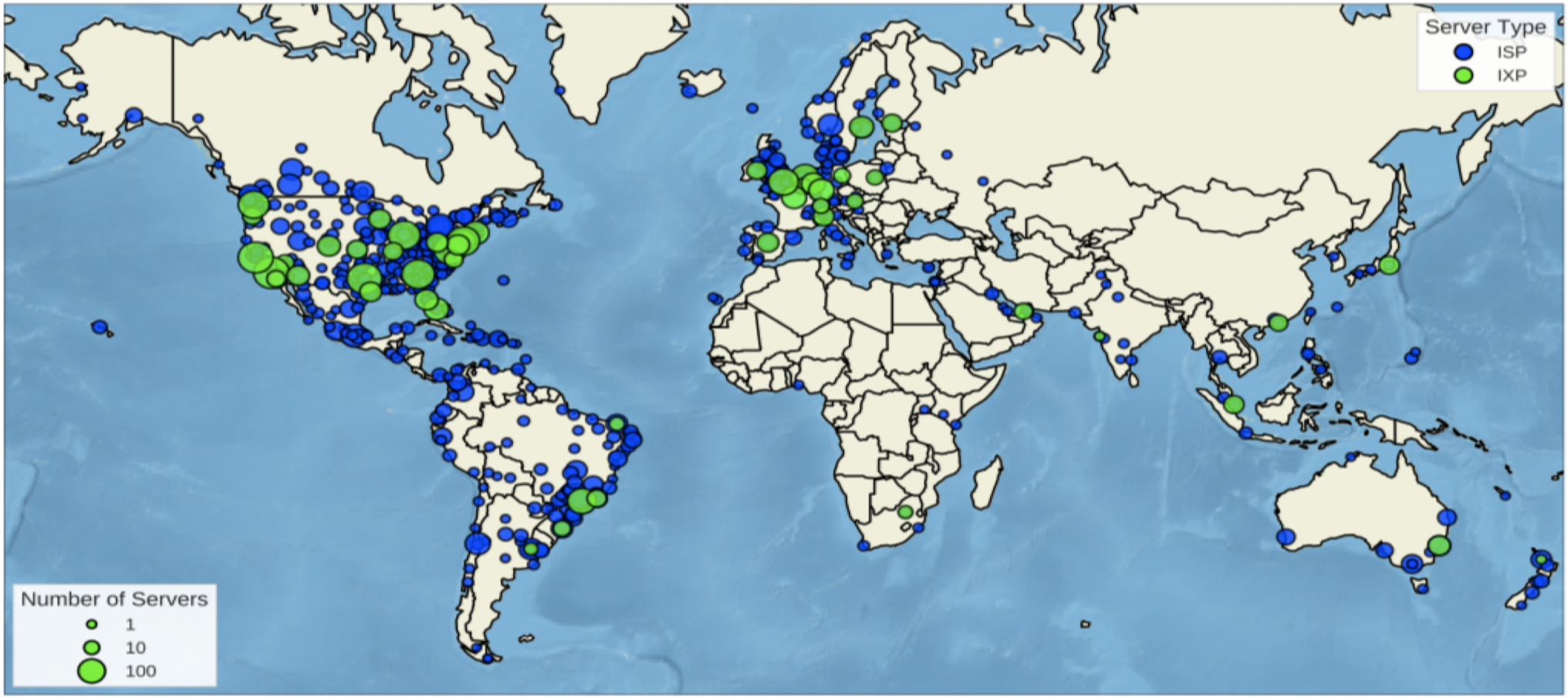


Case study: Netflix



Uses Push caching (during offpeak)
Preference to "deep inside" followed by "bring home"

NetFlix servers (snap shot from Jan 2018)



Researchers from Queen Mary University of London (QMUL) traced server names that are sent to a user's computer every time they play content on Netflix to find the location of the 8492 servers (4152 ISP, 4340 IXP).They have been found to be scattered across 578 locations around the world.

Quiz: CDN

- The role of the CDN provider's authoritative DNS name server in a content distribution network, simply described, is:
  a) to provide an alias address for each browser access to the "origin server" of a CDN website
  b) to map the query for each CDN object to the CDN server closest to the requestor (browser)
  c) to provide a mechanism for CDN "origin servers" to provide paths for clients (browsers)
  d) none of the above, CDN networks do not use DNS