COMP 431
Internet Services & Protocols

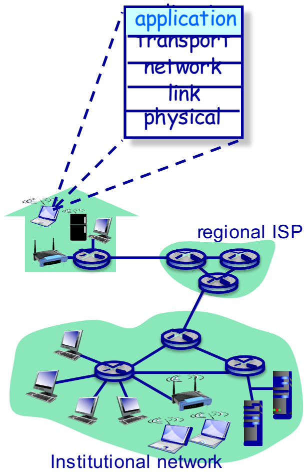
Application-Layer Protocols
Peer-to-Peer Systems, Media Streaming & Content Delivery Networks

Jasleen Kaur

February 13, 2020

Application-Layer Protocols
Outline

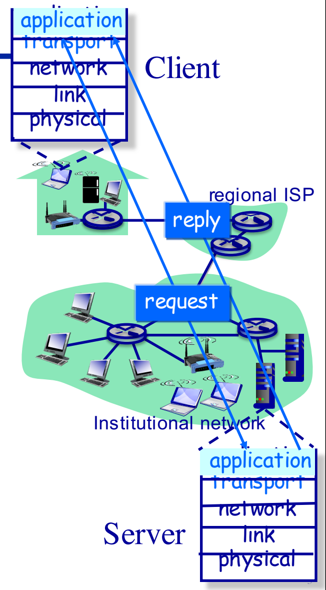
- Example client/server systems and their application-level protocols:
  - The World-Wide Web (HTTP)
  - Reliable file transfer (FTP)
  - E-mail (SMTP & POP)
  - Internet Domain Name System (DNS)
- Example p2p applications systems:
  - BitTorrent
- Other protocols and systems:
  - Streaming media — DASH
  - Content delivery networks (CDNs)



The Application Layer

The client-server paradigm

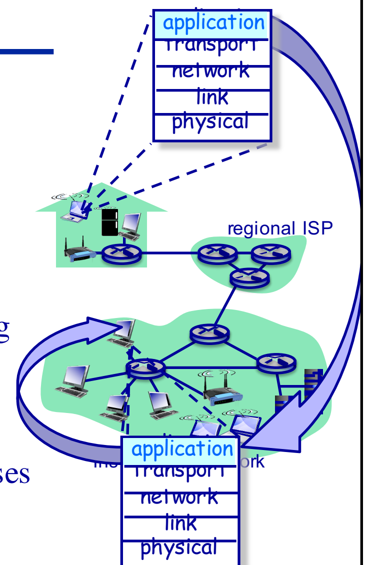
- Typical network application has two pieces: *client* and *server*
- Client — Initiates contact
  - » Requests service from server
  - » Does not communicate with other clients
- Server — Provided requested service to client
  - » "Always" running
  - » Typically has a static IP address
  - » A server may be a logical machine
    - » Implemented by one of thousands of physical servers in a data center



The Application Layer

The peer-to-peer-paradigm

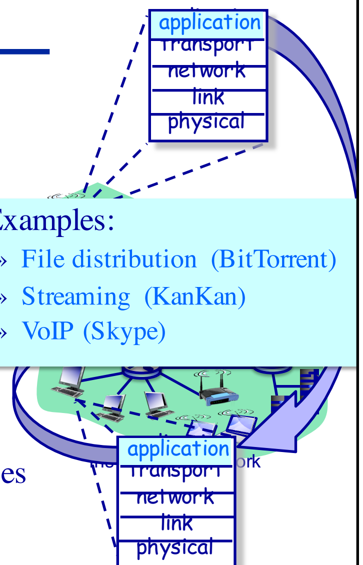
- No "always-on" server
- Arbitrary end systems directly communicate
  - » Peers request service from other peers, provide service in return to other peers
- Self scalability – new peers bring new service capacity, as well as new service demands
- Peers are intermittently connected and change IP addresses
- Complex management



The Application Layer

The peer-to-peer-paradigm

- 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, as well as new service demands
- Peers are intermittently connected and change IP addresses
- Complex management

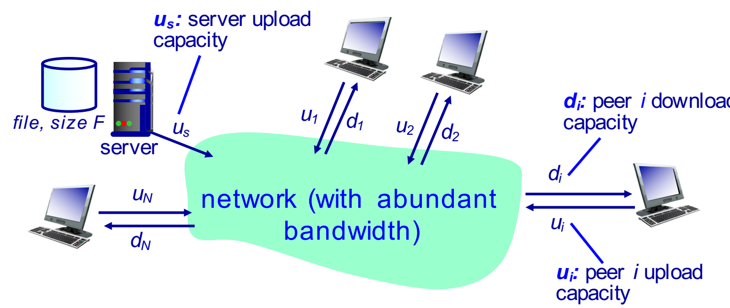


- Examples:
  - » File distribution (BitTorrent)
  - » Streaming (KanKan)
  - » VoIP (Skype)

Peer-to-Peer Systems

Peer-to-peer *v.* client/server file distribution

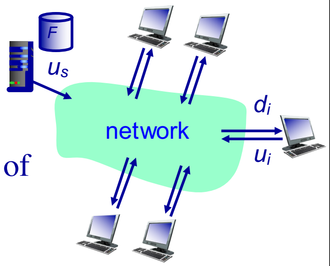
- Question: How much time is required to distribute a file from one server to *N* peers?
  - » In the case when peer upload/download capacity is the limiting resource



Peer-to-Peer Systems

Client/server file distribution time

- Server must sequentially send (upload) N file copies:
  - » Time to send one copy: F/u_s
  - » Time to send N copies: NF/u_s
- Each client must download a copy of the file
  - » d_{min} = minimum client download rate
  - » Minimum client download time = F/d_{min}



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

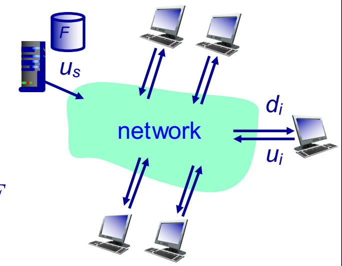
$$D_{cs} \geq max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$$

Increases linearly in N

Peer-to-Peer Systems

Peer-to-Peer file distribution time

- Server must also upload at least one copy
  - » Time to send one copy: F/u_s
- Each client must download a copy of the file
  - » Minimum client download time = F/d_{min}
- In aggregate, clients must download NF bits
  - » Max upload rate (thus, the limiting max download rate) is $u_s + \Sigma u_i$



time to distribute F to N clients using P2P approach

$$D_{P2P} \geq max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \Sigma u_i}\right\}$$

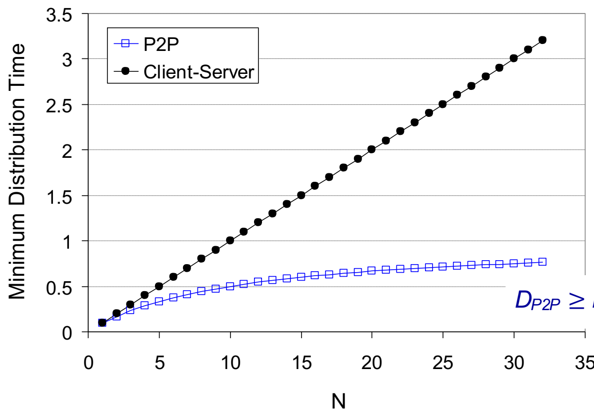
increases linearly in *N* …

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

Peer-to-Peer Systems

Peer-to-peer *v.* client/server example

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



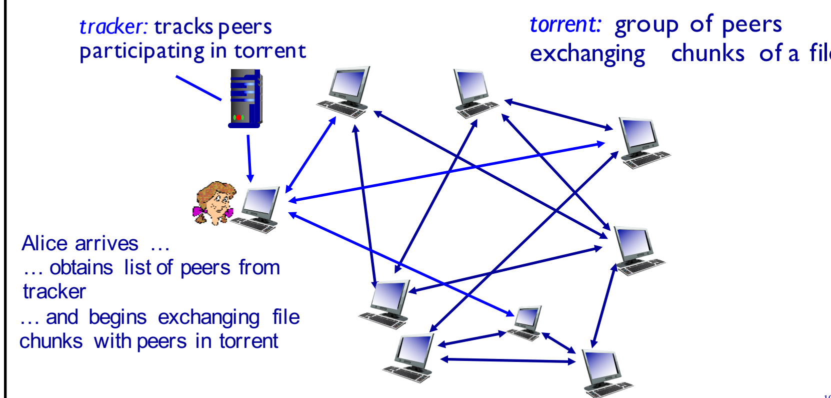
$D_{cs} \geq max\left\{\frac{NF}{u_s}, \frac{F}{d_{min}}\right\}$

$D_{P2P} \geq max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum u_i}\right\}$

Peer-to-Peer File Distribution

BitTorrent

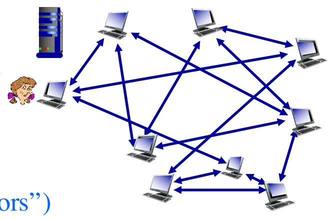
- Files are divided into 256Kb chunks
- Peers in the "torrent" send and receive file chunks



Peer-to-Peer File Distribution

BitTorrent

- When a peer joins a torrent:
  - » It has no chunks, but will accumulate them over time from other peers
  - » It registers with the tracker to get a list of peers
  - » Connects to subset of peers ("neighbors")



- While downloading, peer uploads chunks to other peers
- A peer may change peers with whom it exchanges chunks
  - » ("Churn" — peers may come and go)
- Once the peer has the entire file, it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent

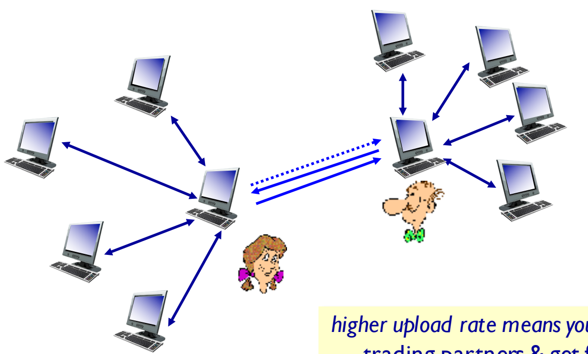
Requesting & sending file chunks

- Requesting chunks:
  - » At any given time, different peers have different subsets of file chunks
  - » Periodically, Alice asks each peer for the list of chunks that they have
  - » Alice requests missing chunks from peers, starting with the "rarest" chunk first
- Sending chunks: "tit-for-tat"
  - » Alice sends chunks to the four peers currently sending her chunks at the highest rate
    - ❖ Other peers are "choked" by Alice (do not receive chunks from her)
    - ❖ Re-evaluate top 4 every 10 secs
  - » Every 30 secs the peer 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 means you find better trading partners & get file faster!

Application-Layer Protocols

Video streaming and content delivery networks

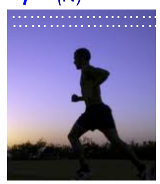
- Video streaming is the major consumer of Internet bandwidth
  - » Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - » ~2B YouTube users, ~160M Netflix users
- How do you design a service to reach ~2B users?
  - » A single mega-video server won't work
- How do you deal with end-system heterogeneity?
  - » Different users have different capabilities (*e.g* wired versus mobile; bandwidth rich versus bandwidth poor)
- Solution: A distributed, application-level delivery infrastructure

Video Streaming

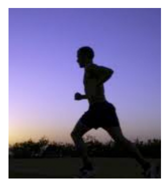
Multimedia basics

- Video is a sequence of images displayed at constant rate
  - » *e.g.*, 24 images/sec
- Each image is a (2D) array of pixels
  - » Each pixel represented by some number of bits
- Coding: use redundancy within and between images to decrease the number of 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*

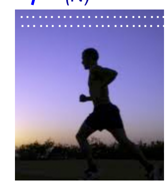
temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

Video Streaming

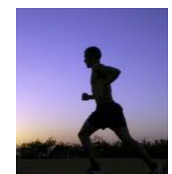
Multimedia basics

- CBR (constant bit rate): Video is encoded at a rate fixed
- VBR (variable bit rate): Video encoding rate changes as amount of spatial, temporal redundancy changes
- Examples:
  - » MPEG 1 (CD-ROM) 1.5 Mbps
  - » MPEG 2 (DVD) 3-6 Mbps
  - » MPEG 4 (often used for Internet streaming) < 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*

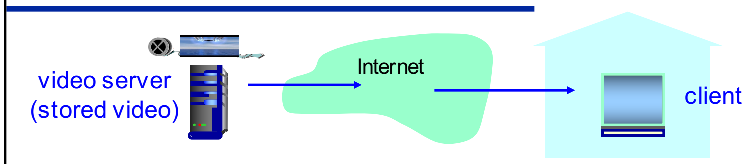


frame *i+1*

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

Video Streaming

Application-layer protocols

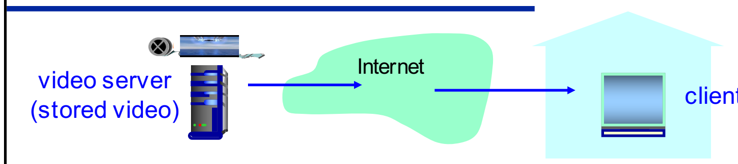


- DASH: Dynamic, Adaptive Streaming over HTTP
- Server functions:
  - » Divides video file into multiple chunks
  - » Each chunk stored, encoded at different rates
  - » Provides a "manifest file" with URLs for different chunks
- Client:
  - » Periodically measures server-to-client bandwidth,
  - » Requests one chunk at a time chooses maximum coding rate sustainable given current bandwidth

All the client needs to stream a video is a browser

Video Streaming

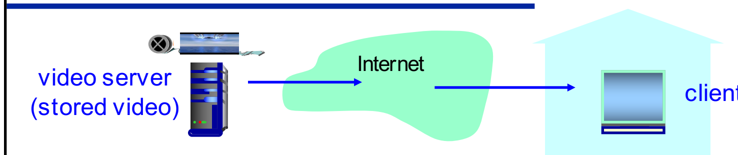
Application-layer protocols



- DASH: Dynamic, Adaptive Streaming over HTTP

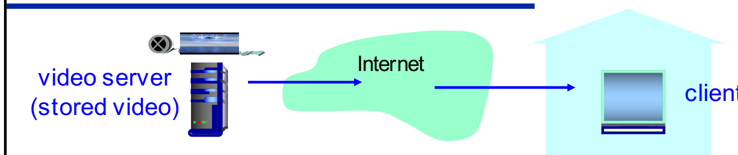
- "Intelligence" at the client determines:
  - » When to request chunks (so that buffer starvation, or overflow does not occur)
  - » What encoding rate to request (higher quality when more bandwidth available)
  - » Where to request chunk (can request from Web server that is "close" to client or has high available bandwidth)

Video Streaming
Content distribution networks



- The challenge is 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
  - » Single point of network congestion
  - » Long delivery path to distant clients (high latency), reliability issues, …
  - » Multiple copies of the video sent over outgoing link
  - » THIS "SOLUTION" DOESN'T SCALE!!

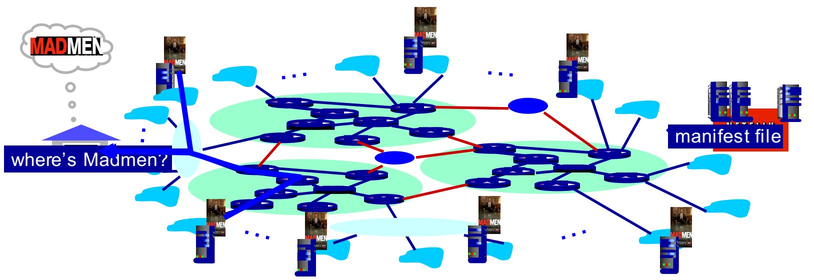
Video Streaming
Content distribution networks



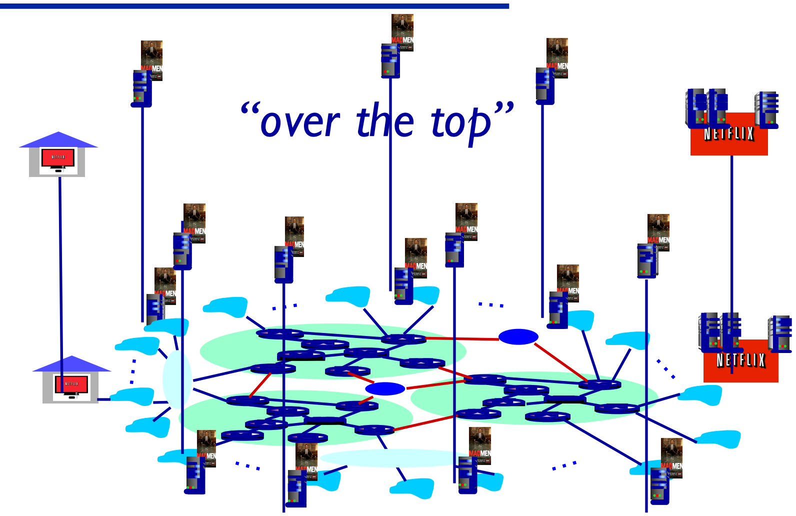
- The challenge is how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- Option 2: store/serve multiple copies of videos at multiple geographically distributed sites (CDN)
  - » "Enter deep" design: push CDN servers deep into access networks
    - ❖ Cache close to users (in the "last mile")
    - ❖ Used by Akamai with servers at over 1,700 locations
  - » "Bring home" design: smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - ❖ Used by Limelight

Content Distribution Networks
Distribution example

- CDN: stores copies of content at CDN nodes
  - » *e.g.*, Netflix stores copies of Mad Men
- Subscriber requests content from CDN (via the Web)
  - » Client is directed to nearby server with the content
  - » Client can choose a different copy if the network path is congested

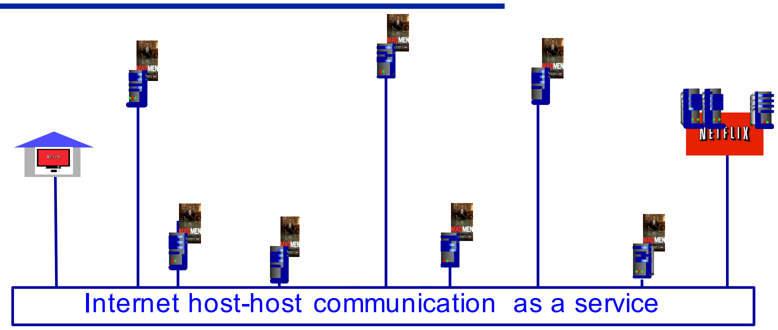


Content Distribution Networks
Challenges



Content Distribution Networks

Challenges

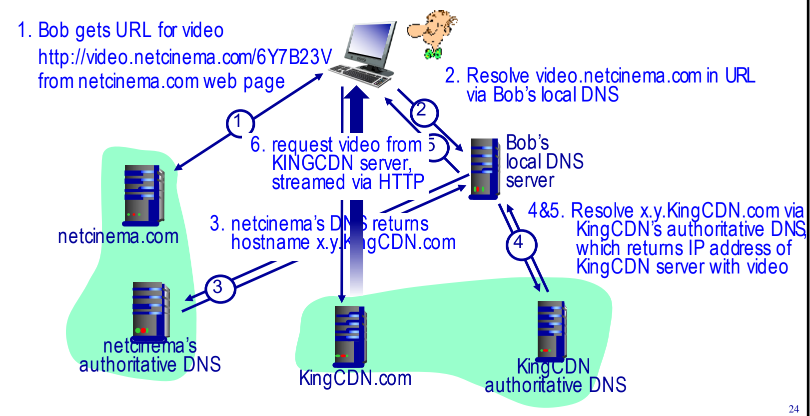


- OTT challenges: coping with a congested Internet
  - » From which CDN node should we retrieve content?
  - » Viewer behavior in presence of congestion?
  - » What content to cache in which CDN node?

Content Distribution Networks

Distribution example

- Bob requests the video *http://netcinema.com/6Y7B23V*
  - » The video stored in "King CDN" at *http://KingCDN.com/NetC6y&B23V*



Content Distribution Networks

Case study: Netflix