Computer Networks
@CS.NYTU

Lecture 2: Applications

Instructor: Kate Ching-Ju Lin (林靖茹)

Slides modified from
"Computer Networking: A Top-Down Approach" 7th Edition

Outline

- Principles of network applications
- Web and HTTP
- E-Mail and SMTP
- DNS
- Peer-to-peer applications
- **Video streaming and CDN**

“Multimedia” Networking

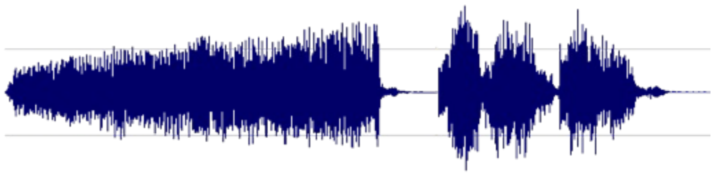
text

CS, NCTU
Multimedia
Networking
2017 Spring
ker ker

image

video

audio / music

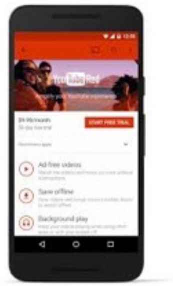


gaming

VR

Why Video Streaming Important?

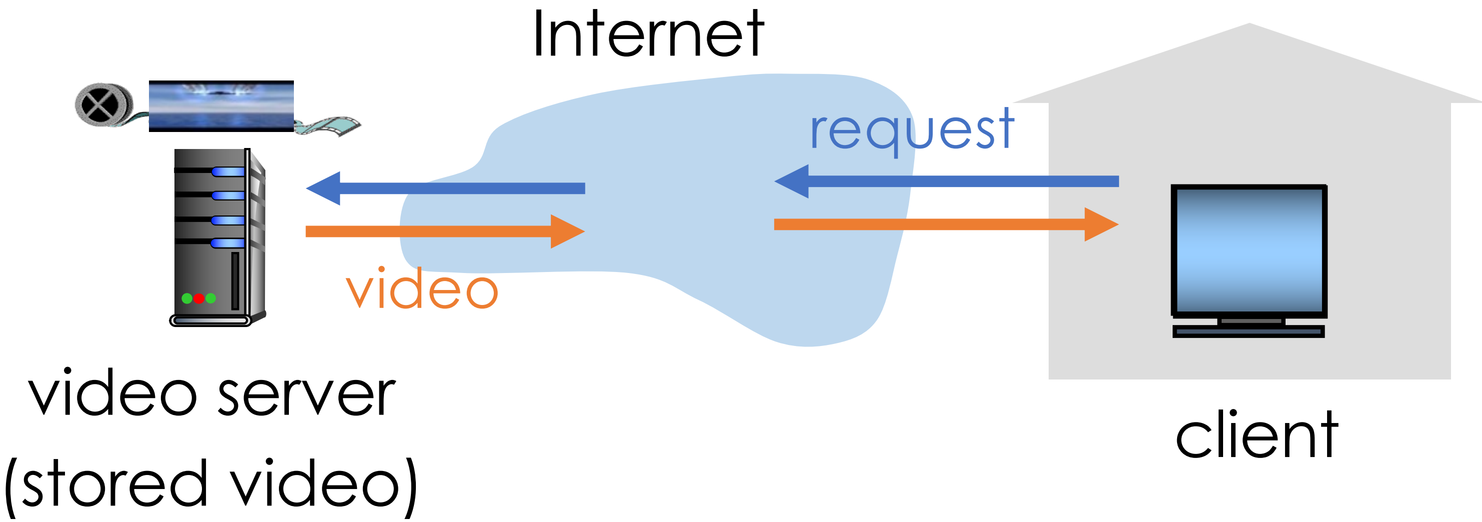
- **High demand**
  - Over 50% Internet traffic



- **High bandwidth requirement**
  - 30, 60, 120 frames per seconds
  - Video rate: 100kbps – 3Mbs or even >10Mbps (4K)
- **Complex compression algorithms**
  - **No need to receive all the bits**
  - Video quality is related to packet loss rate
    - But not linear proportional

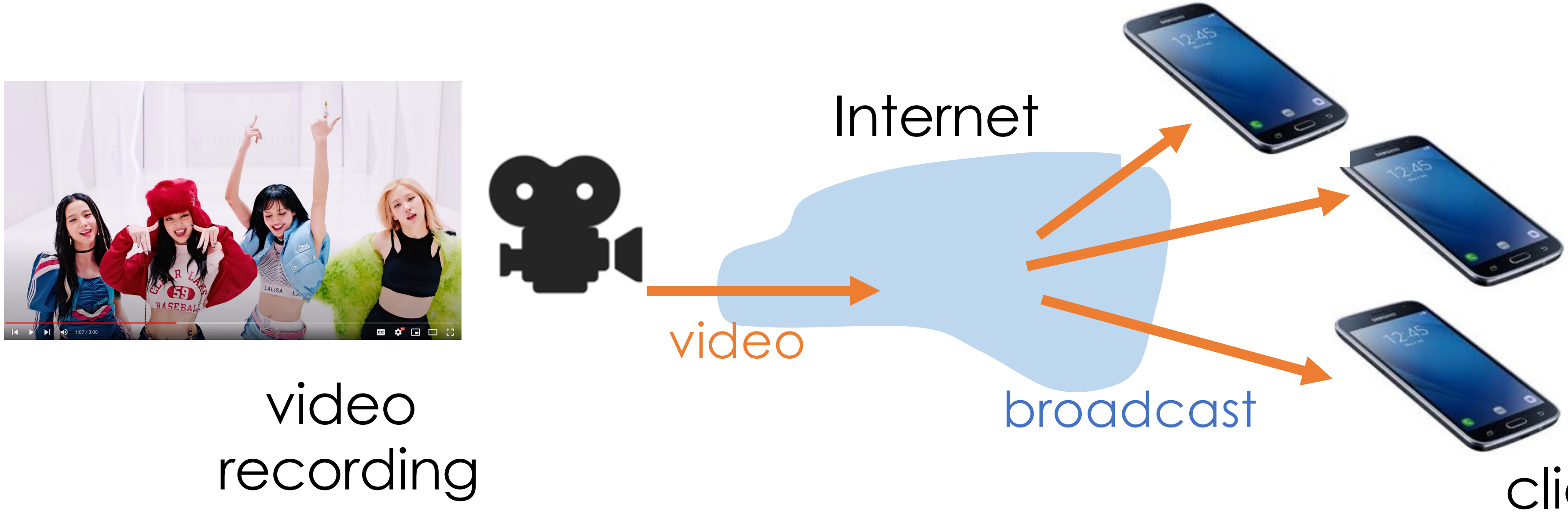
1. Video-On-Demand Streaming

- Request **on demand**
- Online downloading and playing



2. Real-Time Video Streaming

- Record and deliver live video immediately
- Usually broadcasting



3. HTTP Streaming

- ***DASH:*** *D*ynamic, *A*daptive *S*treaming over *H*TTP
- server:
  - Divide video file into multiple chunks
  - Each chunk stored, encoded at different rates
  - **Manifest file**: provides URLs for different chunks/rates
- client:
  - Periodically measures server-to-client bandwidth
  - Consulting manifest, requests one chunk at a time
    - Choose maximum coding rate sustainable given current bandwidth
    - Can choose different coding rates at different points in time (depending on available bandwidth at time)

Why HTTP Streaming

- Traditional streaming delivered over RTP/UDP
- However, in today's Internet, content objects are stored **Content Delivery Networks (CDN)**
  - Many CDNs do not support RTP streaming
  - RTP often does not allow traffic through firewall
  - Each RTP stream is a separate session → not scalable
- Benefits of HTTP streaming
  - Firewall friendly
  - No need to maintain the session state on the server
  - Has been standardized as "ISO/IEC 23009-1, also known as MPEG-DASH" in Apr. 2012

Why CDNs?

- **Challenge:** how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
- **Option 1:** single, large "mega-server"
  - Reliability: single point of failure
  - Not enough bandwidth: point of network congestion
  - Far from users: long path to distant clients
  - multiple copies of video sent over outgoing link

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

Why CDNs?

- **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 geographically distributed sites (CDN)
  - enter deep: push CDN servers deep into many access networks
    - ✓ close to users
    - ✓ used by Akamai, 1700 locations
  - bring home: smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - ✓ used by Limelight

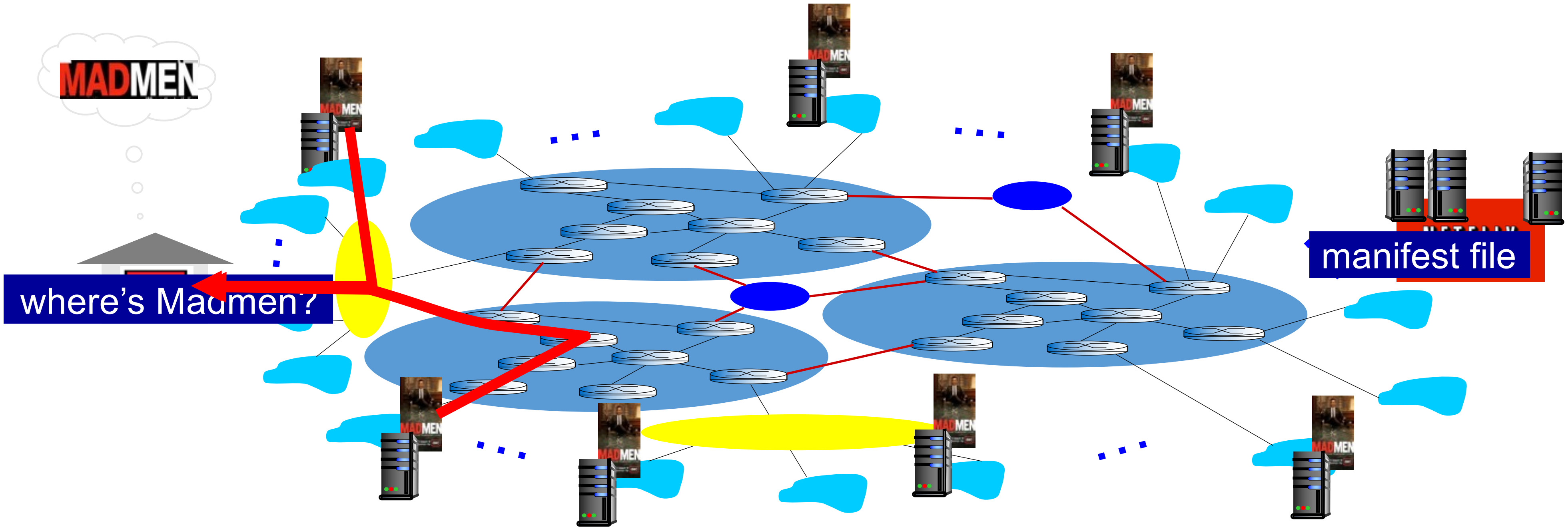
Content Distribution Networks

- **CDN:** an application overlay (e.g., Akamai)
- Design Space
  - **Caching** (data-driven, passive)
    - explicit
    - transparent (hijacking connections)
  - **Replication** (pro-active)
    - server farms
    - geographically dispersed (CDN)

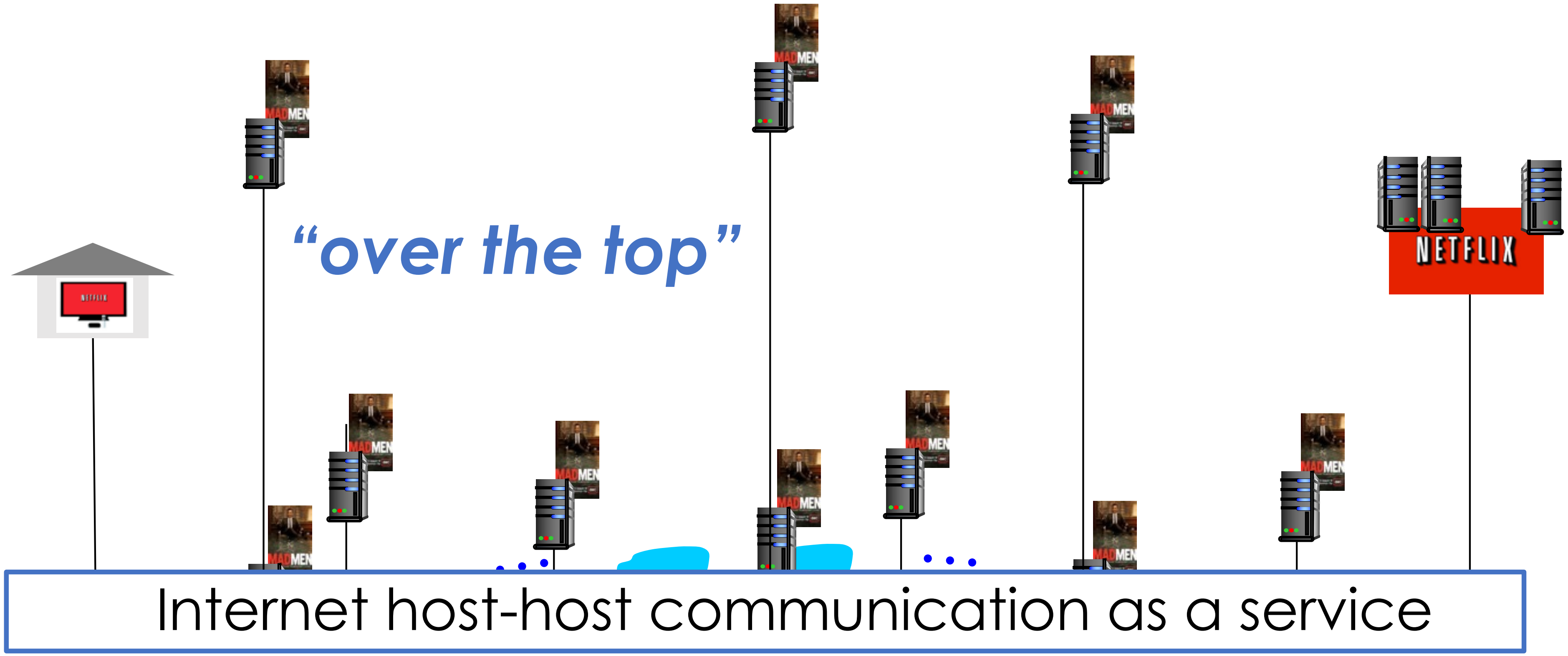
- Three Main CDN Providers (in North America, Europe):
  - Akamai, Limelight, Level 3 CDN

Key Idea of CDN

- 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



Framework of CDN

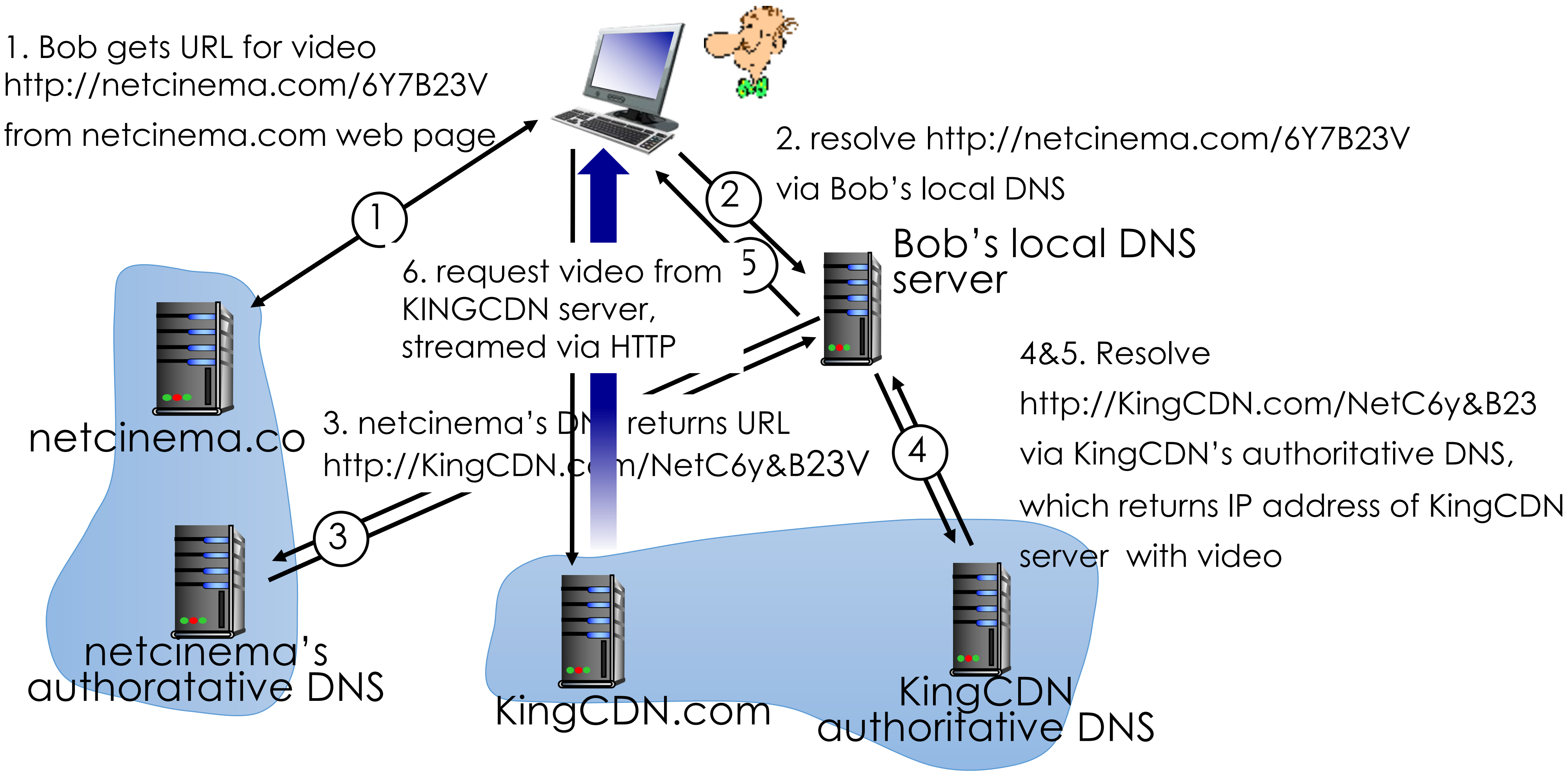


OTT challenges: coping with a congested Internet

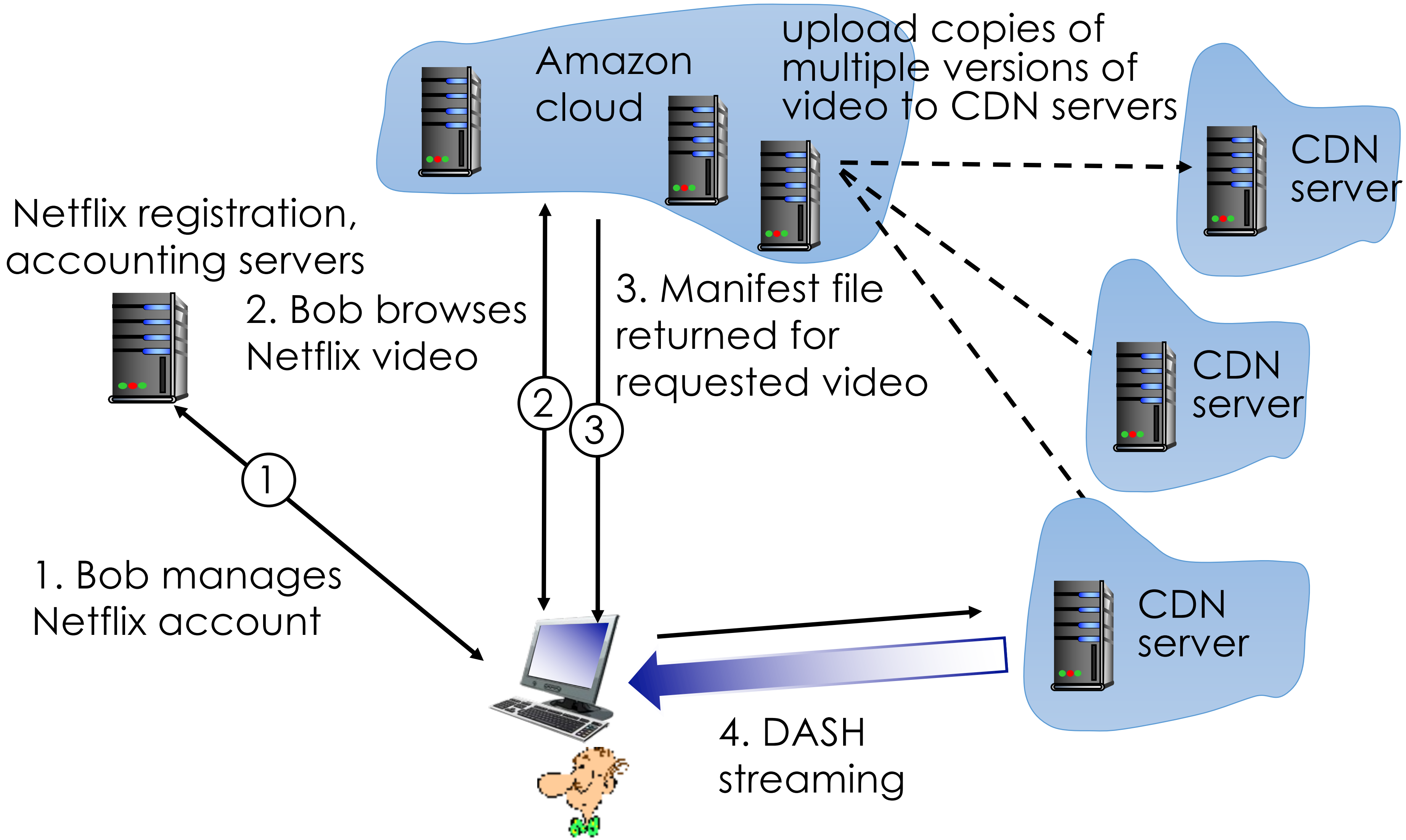
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

CDN Content Access

- Bob (client) requests video http://netcinema.com/6Y7B23V
- video stored in CDN at http://KingCDN.com/NetC6y&B23V



Case Study: Netflix



Cluster Selection Strategies

- **Geographically closest**
  - Geographical distance ≠ path length
  - Local DNS location ≠ user location
  - Do not explicitly consider bandwidth and delay (varying with the congestion level)

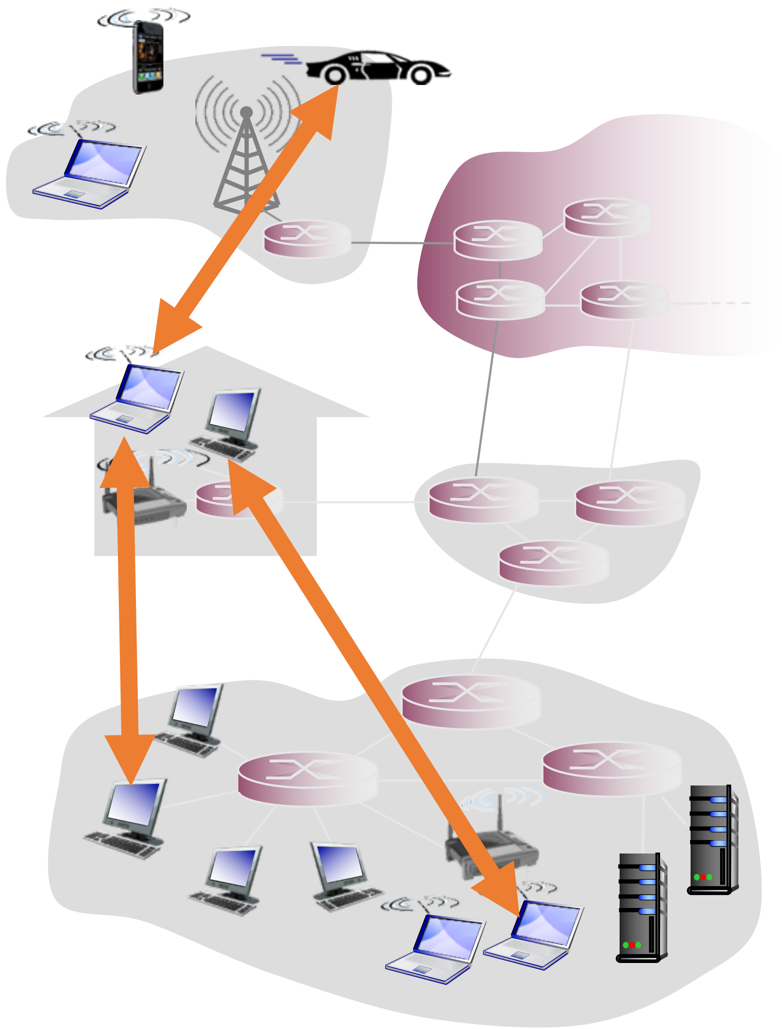
- **Fastest response time**
  - Periodically probe from CDN clusters to DNS servers
  - Ingest huge probing traffic
  - Some DNS servers configured to "not reply"

Outline

- Principles of network applications
- Web and HTTP
- E-Mail and SMTP
- DNS
- **Peer-to-peer applications**
- Video streaming and CDN

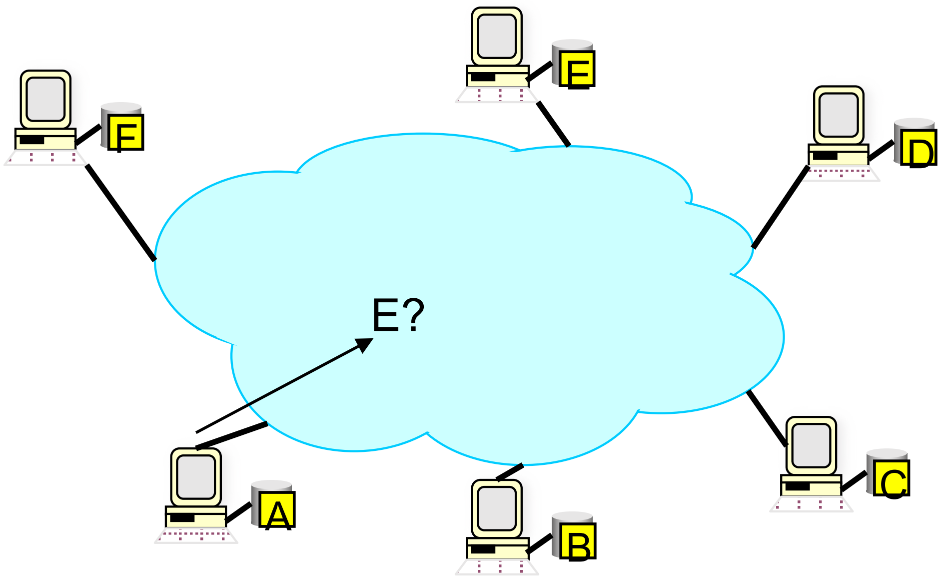
Pure P2P Architecture

- *No* always-on server
  - Might have some super nodes, mainly used for management
- Arbitrary end systems (peers) directly communicate
  - Self-scalable
- Peers are intermittently connected and change IP addresses dynamically
- Examples:
  - File distribution (BitTorrent)
  - Streaming (PPstream)
  - VoIP (Skype)



P2P: Challenges

- How to locate your peer & find what you want?
- Need some kind of "directory" or "look-up" service



- centralized
- distributed,
  using a hierarchal structure
- distributed,
  using a flat structure
- distributed,
  without structure ("flooding")
- distributed,
  using "hybrid"
  structured/unstructured

P2P: Challenges

- **Technical**
  - Scale: up to hundred of thousands or millions of machines
  - Dynamics: machines can come and go any time
- **Social, economic and legal**
  - Incentive Issues: free-loader problem
  - Vast majority of users are free-riders
  - Most share no files and answer no queries
  - A few individuals contributing to the "public good"
  - Copyrighted content and pivacy
  - Trust & security issues

BitTorrent: Popular P2P App

- Designed for large file (e.g., video) distribution
- Focused on efficient *fetching*, not search
  - Distribute same file to many peers
  - Single publisher, many downloaders
- Divide large file into many pieces (chunks)
  - Replicate different pieces on different peers
  - A peer with a complete piece can trade with others
  - Peer can (hopefully) assemble the entire file
- Allows simultaneous downloading
  - <u>Retrieve different pieces from different peers simultaneously</u>
- Usually need to prevent "free loading"

BitTorrent Components

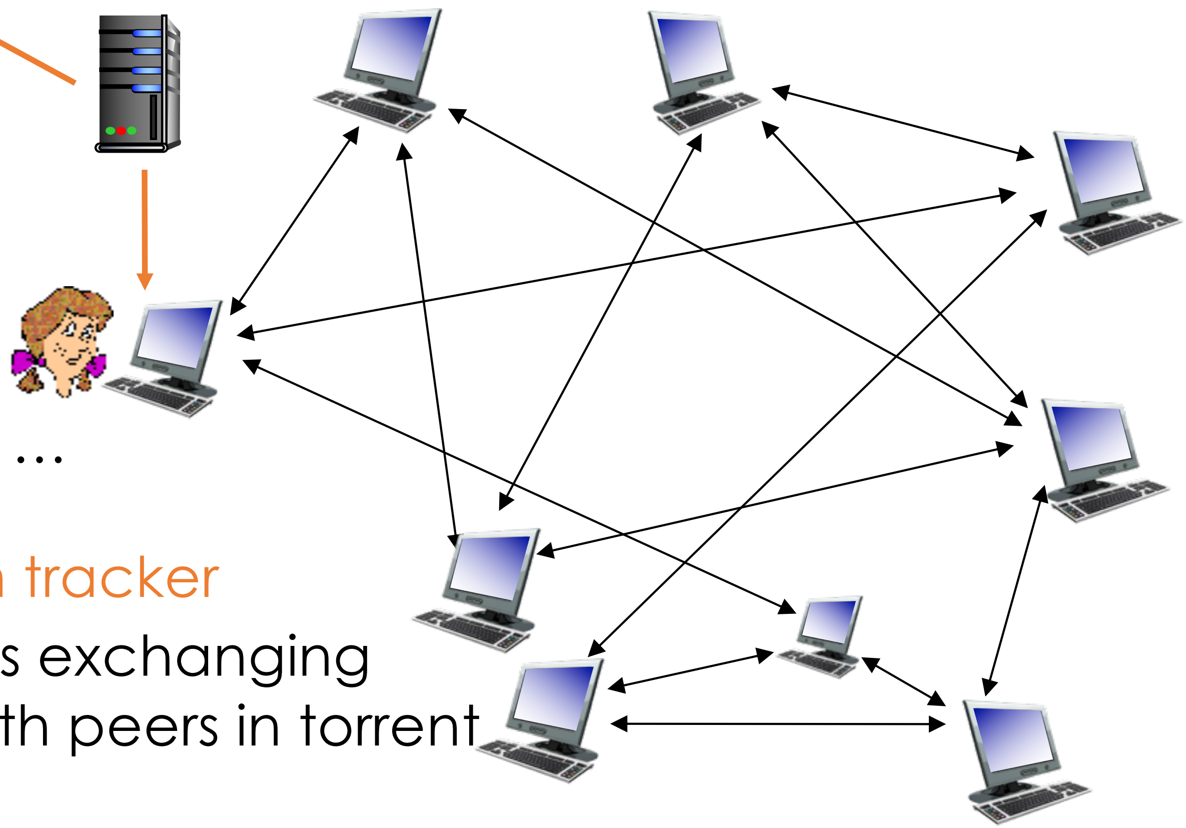
- **Seed**
  - Peer with entire file
  - Fragmented in pieces
- **Leacher**
  - Peer with an incomplete copy of the file
- **Torrent file**
  - Passive component
  - Store summaries of the pieces to allow peers to verify their integrity
- **Tracker**
  - Allows peers to find each other
  - Returns a list of random peers

BitTorrent Protocol

- 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



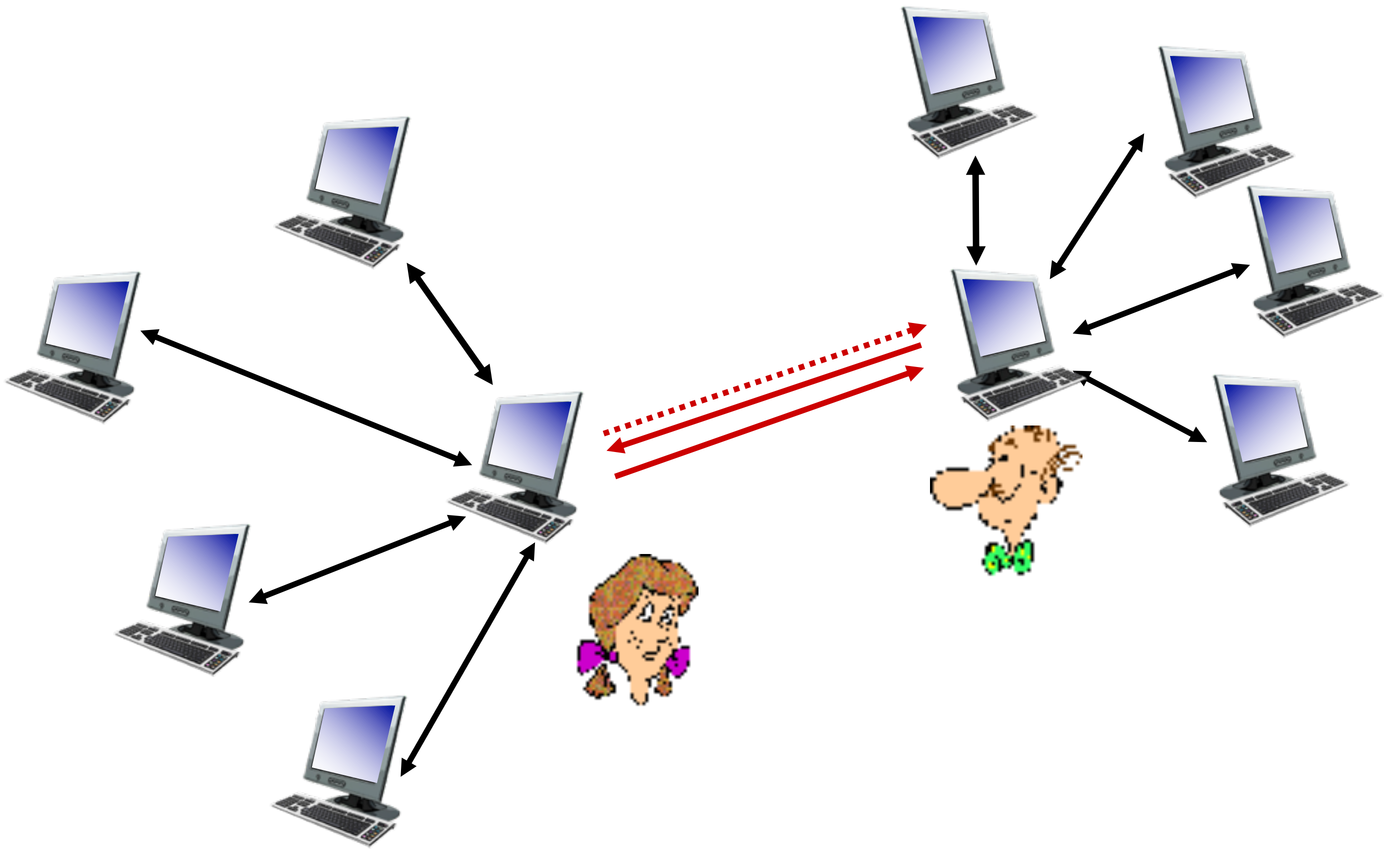
Alice arrives ...
... obtains list of peers from tracker
... and begins exchanging file chunks with peers in torrent

BitTorrent Protocol

- Peer joining torrent:
  - Has no chunks, but will accumulate them over time from other peers
  - Register with tracker to get a list of peers (“neighbors”)
  - Exchange chunks with neighbors
- Peer may change neighbors over time
  - Track periodically suggests some random peers
  - Keep those random peers as neighbors if they are better (in terms of bandwidth or content)
- **Churn**: peers may come and go
- Once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

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



BitTorrent: Scheduling

Two choices should be made

- **Which chunk to download?**
  - rarest first: download the chunk with the least copies first
  - Why? More precious and might become disappear in the network
  - Tracker should provide such information

- **Where to download?**
  - Highest rate first: from a neighbor who can provide a higher downloading rate
  - Update the neighbor list if the random peer suggested by the tracker supports a higher rate

BT: 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 peers currently sending her chunks at highest rate
  - other peers do not receive chunks from Alice
  - 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