Application Layer: FTP, P2P and CDN

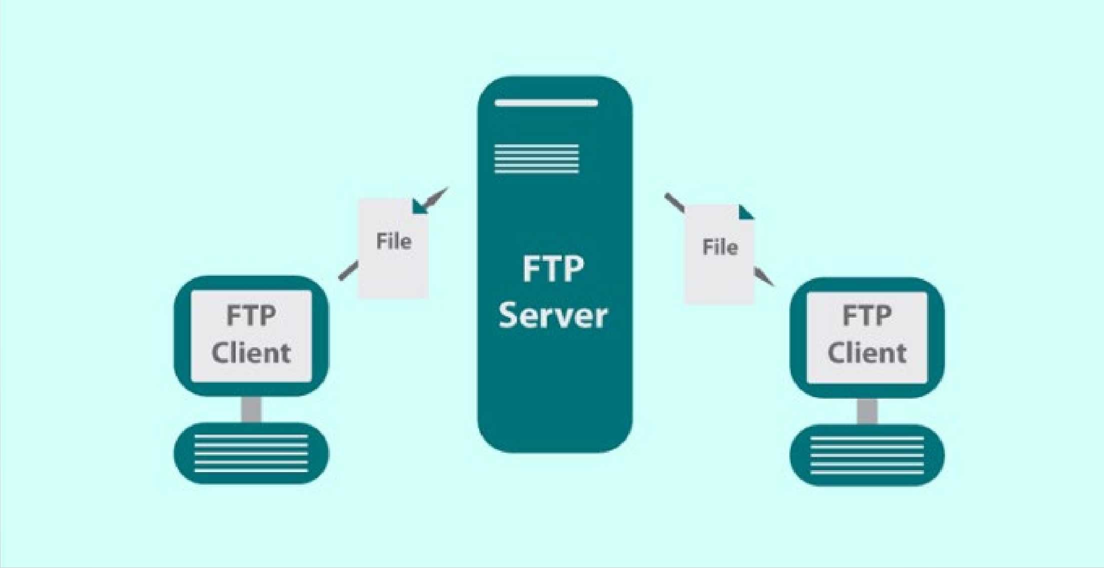
Lecture 5 | CSE421 – Computer Networks

Department of Computer Science and Engineering
School of Data & Science

Objectives

- File Distribution
- FTP
- Client-Server Architecture
- P2P Architecture
- CDN

FTP



FTP

- FTP stands for **File transfer protocol**.
- FTP is a standard internet protocol **provided by TCP/IP** used for transmitting the files from one host to another.
- It is **mainly used for transferring the web page files** from their creator to the computer that acts as a server for other computers on the internet.
- It is also used for **downloading the files to computer** from other servers.
- Objectives:
    - It provides the **sharing of files**.
    - It is used to **encourage** the use of remote computers.
    - It transfers the data more **reliably and efficiently**

Why FTP?

Two systems may have different:

- file conventions.
- ways to represent text and data.
- directory structures.

FTP protocol overcomes these problems by establishing two connections between hosts. One connection is used for data transfer, and another connection is used for the control connection.

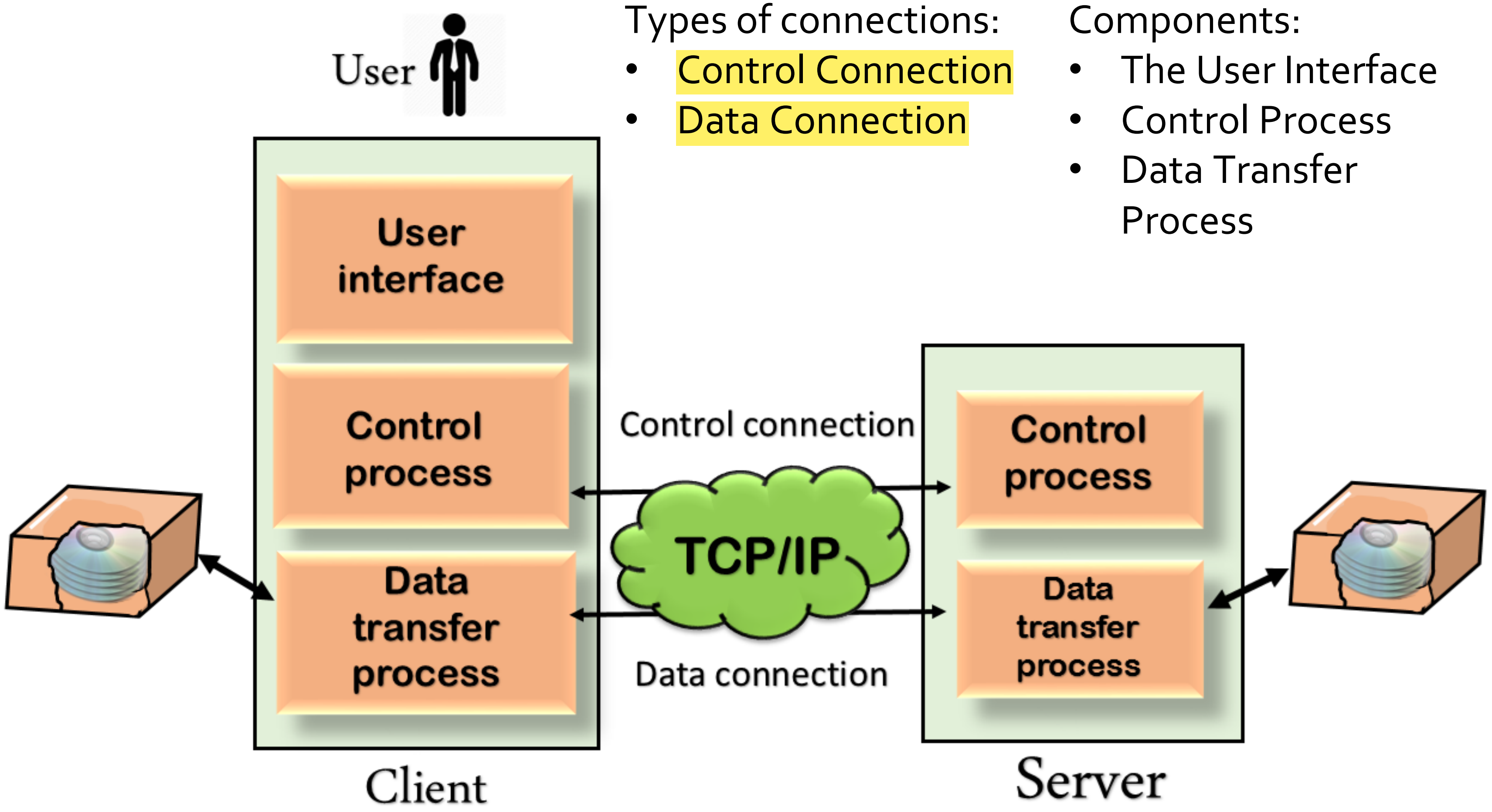
Mechanism of FTP

Types of connections:

- Control Connection
- Data Connection

Components:

- The User Interface
- Control Process
- Data Transfer Process



Types of FTP Connections

- **Control Connection:** The control connection uses very simple rules for communication. Through control connection, we can transfer a line of command or line of response at a time. The control connection is made between the control processes. The control connection remains connected during the entire interactive FTP session.
- **Data Connection:** The Data Connection uses very complex rules as data types may vary. The data connection is made between data transfer processes. The data connection opens when a command comes for transferring the files and closes when the file is transferred.

FTP Clients

- FTP client is a program that **implements a file transfer protocol** which allows you to transfer files between two hosts on the internet.
- It allows a user to **connect to a remote host** and upload or download the files.
- It has a **set of commands** that we can use to connect to a host, transfer the files between you and your host and close the connection.

FTP Ups and Downs

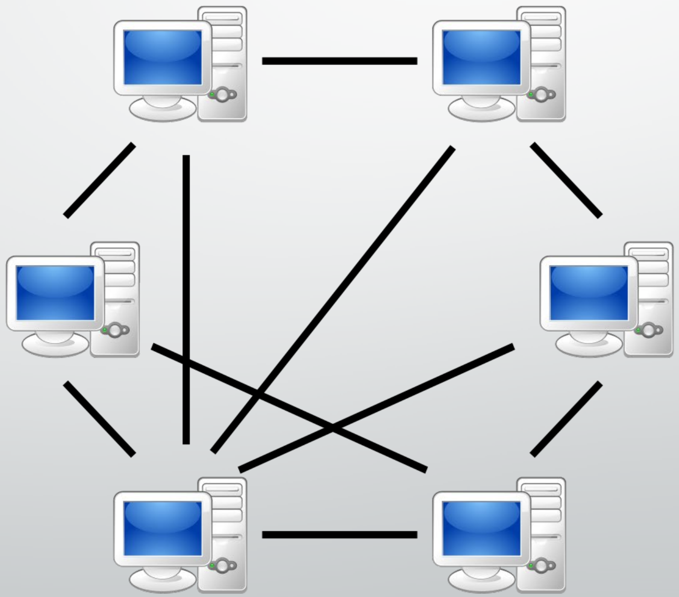
Advantages

- **Speed:** The FTP is one of the fastest way to transfer the files from one computer to another computer.
- **Efficient:** It is more efficient as we do not need to complete all the operations to get the entire file.
- **Security:** To access the FTP server, we need to login with the username and password.
- **Back & forth movement:** Suppose you are a manager of the company, you send some information to all the employees, and they all send information back on the same server.

Disadvantages

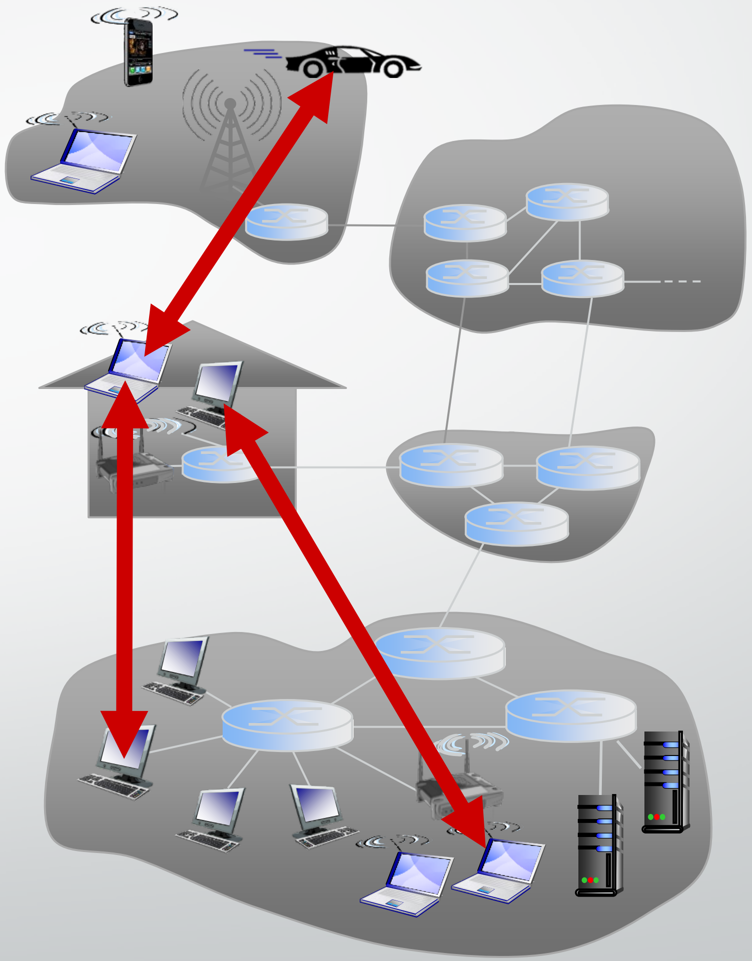
- The standard requirement of the industry is that all the **FTP transmissions should be encrypted**. However, not all the FTP providers are equal and **not all the providers offer encryption**.
- FTP has the **size limit of the file is 2GB** that can be sent. It also **doesn't allow** you to run **simultaneous transfers to multiple receivers**.
- **Passwords and file contents are sent in clear text** that allows unwanted eavesdropping.
- It is **not compatible** with every system.

P2P Applications



Pure P2P Architecture

- No always-on server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change IP addresses
- Files are shared in chunks rather than a whole single file
- A successful file transfer is possible if all clients collectively have all the chunks of a file

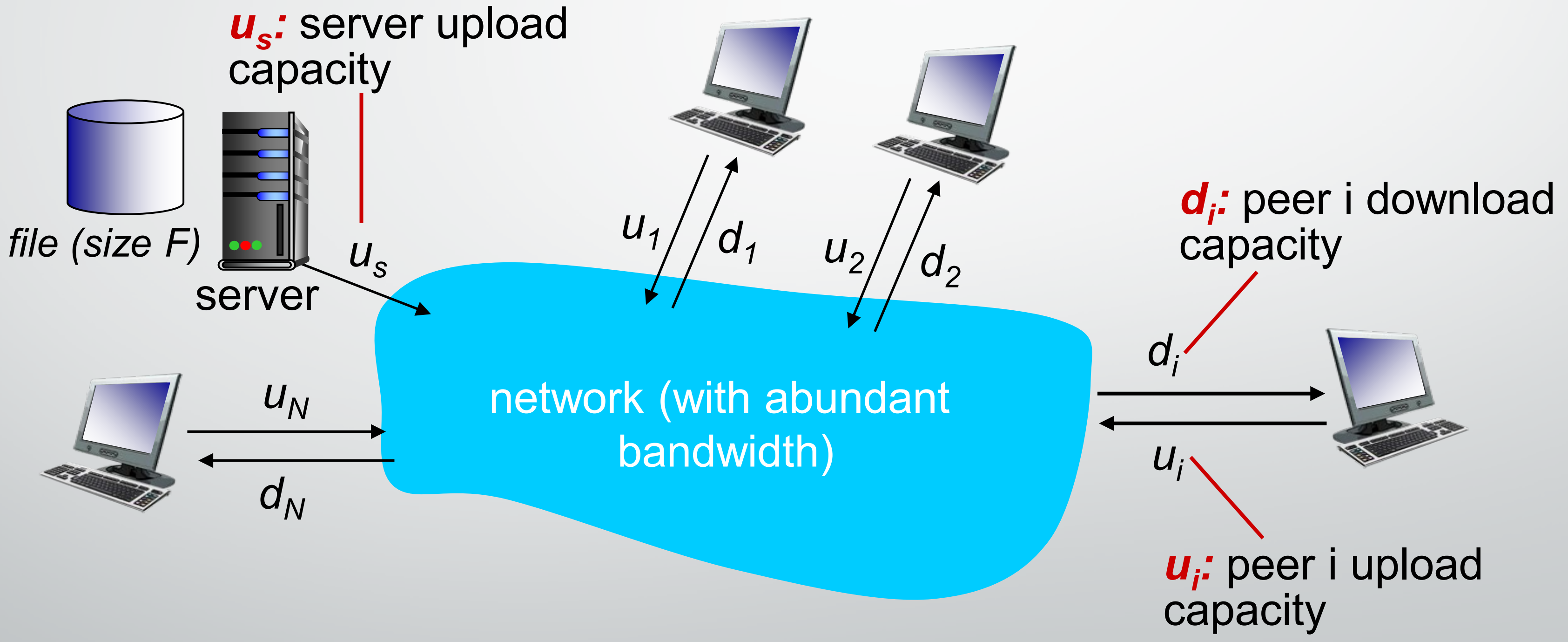


Examples:

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

File Distribution

- How much time is required to distribute a file (of size F) from one server to 'N' number of peers?
    - Peer (client) upload/download capacity is limited resource



Terms of equations

- Given:
  - File size = **f**
  - Number of clients = **n**
  - Server upload speed = $\mathbf{u_s}$
  - Download speed of peer 'i' = $\mathbf{d_i}$
  - Upload speed of peer 'i' = $\mathbf{u_i}$
  - Time to distribute files using client-server approach = $\mathbf{T_{c\text{-}s}}$
  - Time to distribute files using peer-to-peer approach = $\mathbf{T_{P2P}}$
  - The peer with the slowest download speed = $\mathbf{d_{min}}$
  - Summation of upload speed of all peers = $\mathbf{\Sigma u_i}$

File Distribution: Client-Server

Server:

- Time to send one copy of file from the server = **f/u_s**
- Hence, time to send this one file to n number of clients = **$n * f/u_s = \boxed{nf/u_s}$**

Client:

- Downloading time of the slowest client $\boxed{= \mathbf{f/d_{min}}}$

Time to distribute file 'f' to 'n' clients using client-server approach

- **$T_{c\text{-}s} \geq \max\{nf/u_s, f/d_{min}\}$**
  - So, why the max value of the above two?
    - It's because, if the server needs 10minutes to upload a file, a client can never download it before 10 minutes.
    - If the slowest client needs 15 minutes to download the file, even though server needs 10 minutes to upload, there's no point. The transfer won't finish before 15 minutes.
  - Lastly, **$T_{c\text{-}s}$** will take a equal or greater value because, this is the minimum time possible (does not consider any delays). In real world, speed is always not at its maximum, speed varies.

File Distribution: P2P

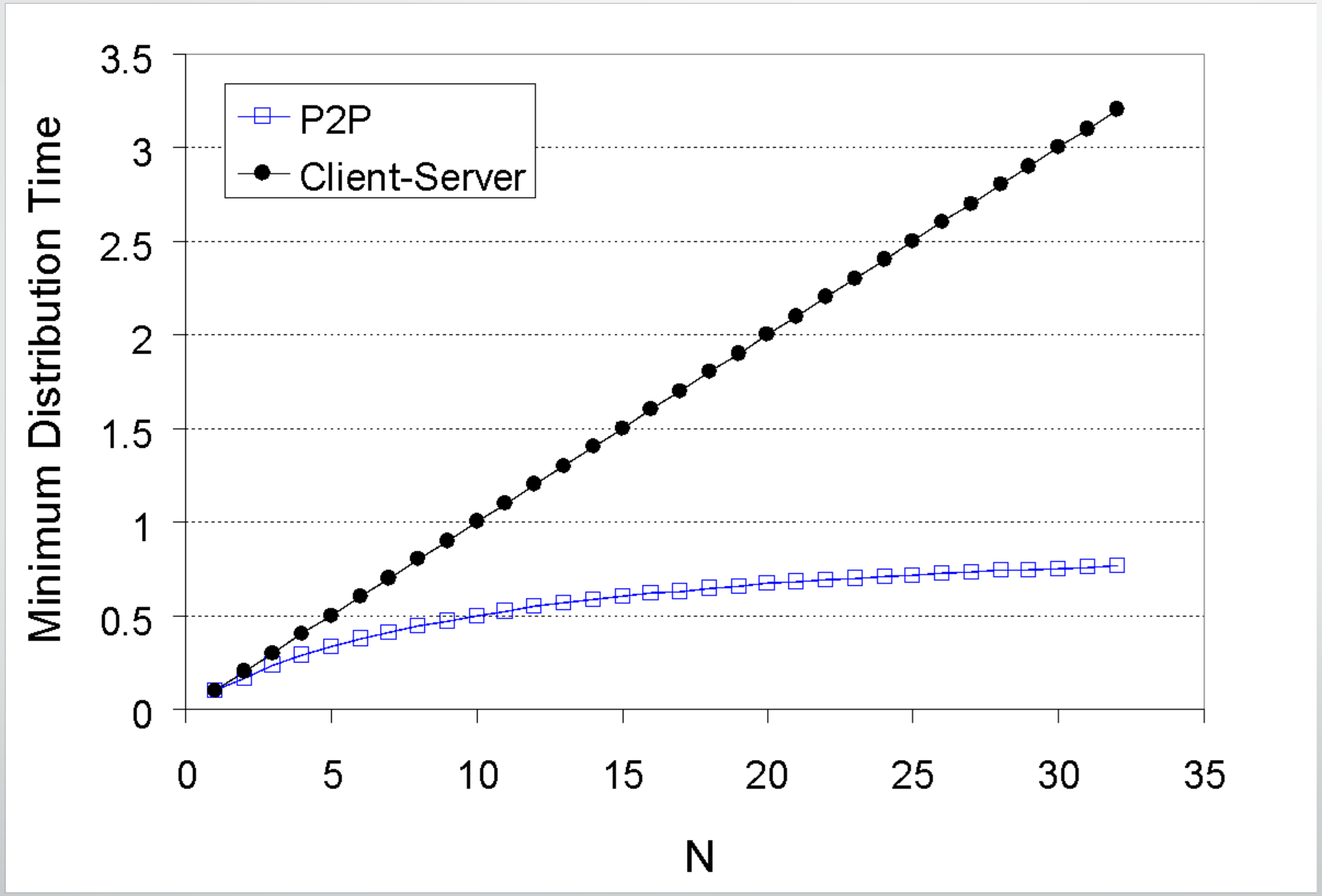
Server transmission: must upload at least one copy

- Time to upload file f from the server = $\boxed{\mathbf{f/u_s}}$
- Downloading time of the slowest client = $\boxed{\mathbf{f/d_{min}}}$
- Total downloaded file size by n clients = **n * f** = **nf**
- The more clients participate in the file sharing.. the more upload speed
- Total upload speed of n clients $= \mathbf{u_1 + u_2 + u_3 + ... + u_n = \Sigma u_n}$
- Max upload rate (limiting max download rate) = $\mathbf{u_s + \Sigma u_n}$
- Time to download the files n times by using the upload speed of all clients $\boxed{= \mathbf{nf/(u_s + \Sigma u_n)}}$

Time to distribute file ‘f’ to ‘n’ clients using peer-to-peer approach

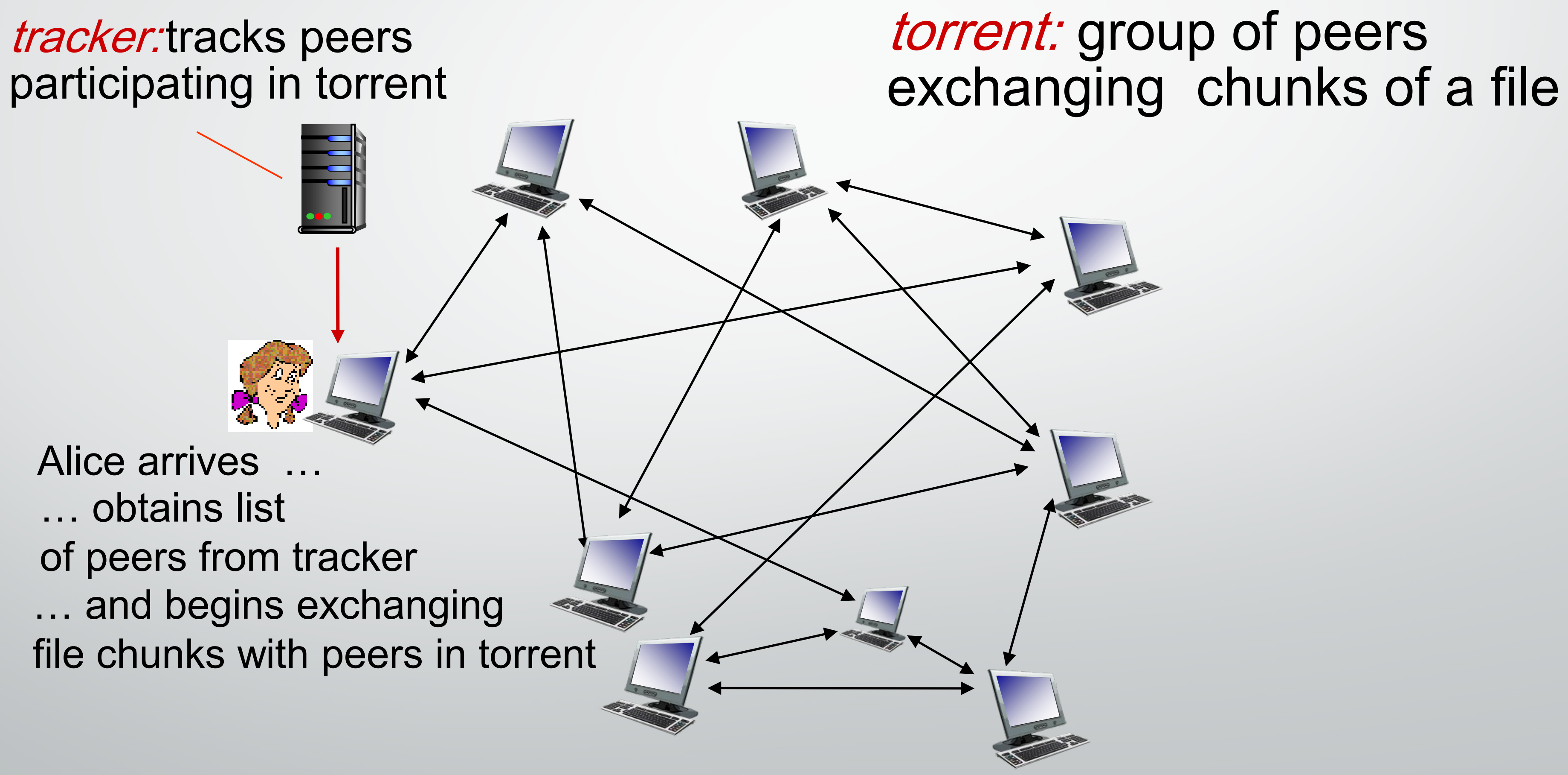
- $\mathbf{T_{c\text{-}s} \geq max\ \{f/u_s,\ f/d_{min},\ nf/(u_s + \Sigma u_n)\}}$

Client-Server vs P2P



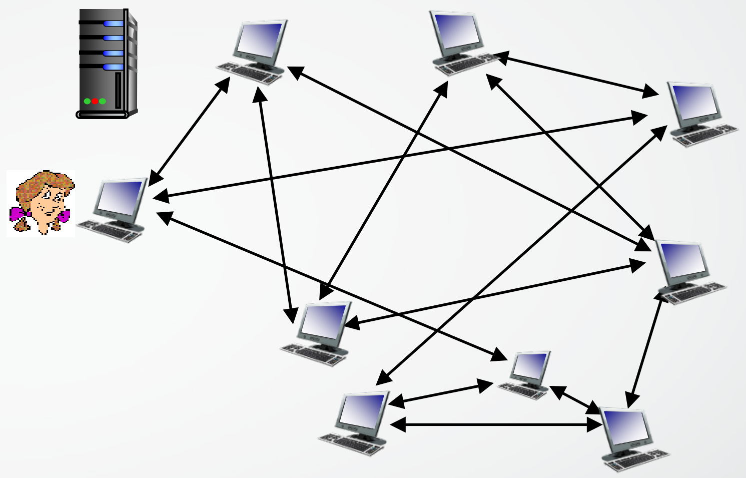
P2P File Distribution: BitTorrent

- File divided into 256Kb (it can be any size!) chunks
- Peers in torrent send/receive file chunks



P2P File Distribution: BitTorrent

- A new peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers and connects to a subset of peers ("neighbors") found on the tracker list.



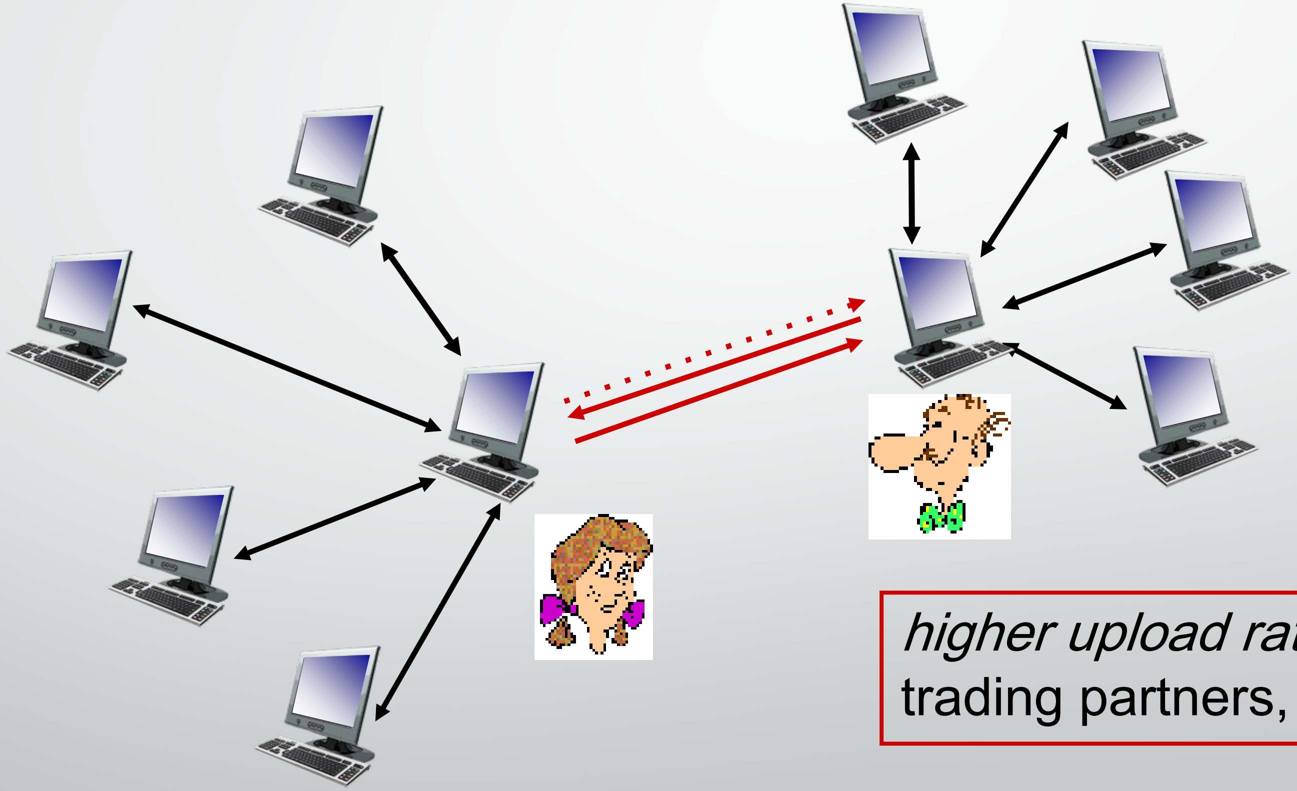
- While downloading, a peer also uploads chunks to other peers
- A peer may change its connected peers with whom it exchanges chunks
- **Churn:** connected peers may come and go
- Once a peer has the entire file, it may (selfishly; **a leacher**) leave or (altruistically; **a seeder**) remain in the torrent (sharing its chunks with others)

P2P File Distribution: BitTorrent

(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 !

P2P Example

Say, a torrent has 100 pieces/chunks.

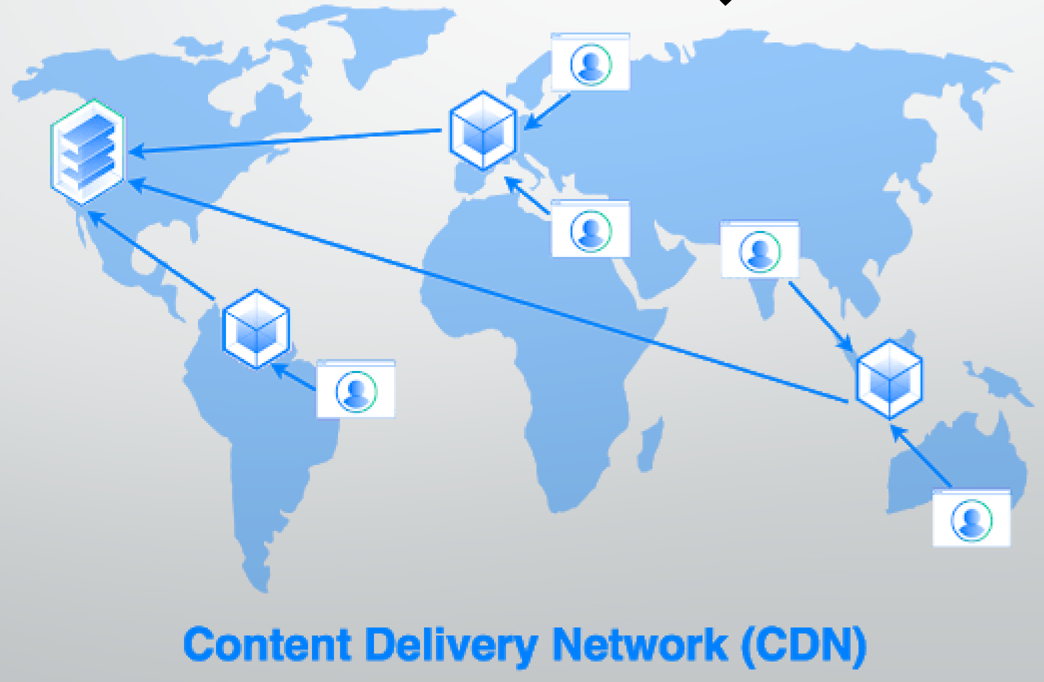
Example 1:

- Client 1 has chunks 1 to 30 – Peer
- Client 2 has chunks 25 to 60 – Peer
- Client 3 has chunks 1 to 100 – Seeder
- Client 4 joins the torrent… Will this client be able to download the whole torrent?

Example 2:

- Client 1 has chunks 1 to 30 – Peer
- Client 2 has chunks 25 to 60 – Peer
- Client 3 has chunks 60 to 99 – Seeder
- Client 4 joins the torrent… Will this client be able to download the whole torrent?

Content Distribution Networks (CDN)



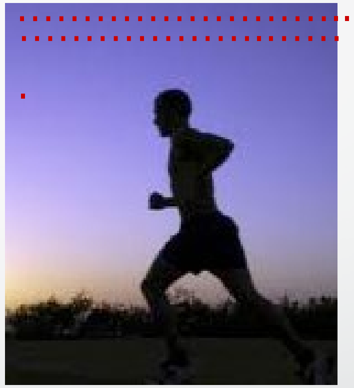
Video Streaming and CDNs: Context

- **Video traffic:** major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- **Challenges:**
  - **Scale:** how to reach ~1B users? single mega-video server won't work (why?)
  - **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 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*

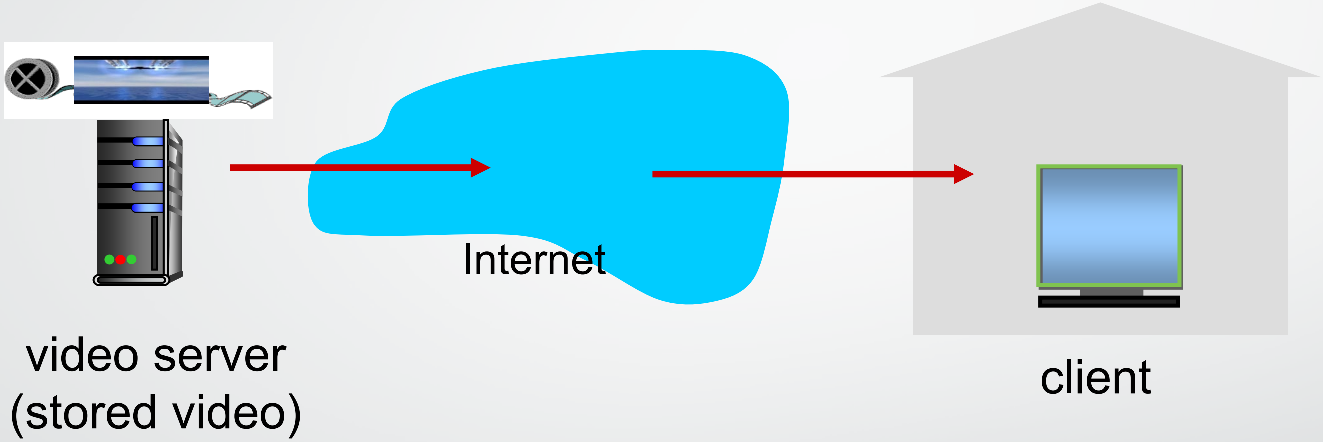
temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

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)

Streaming Stored Video

- Simple scenario



Streaming: 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: DASH

- **"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)

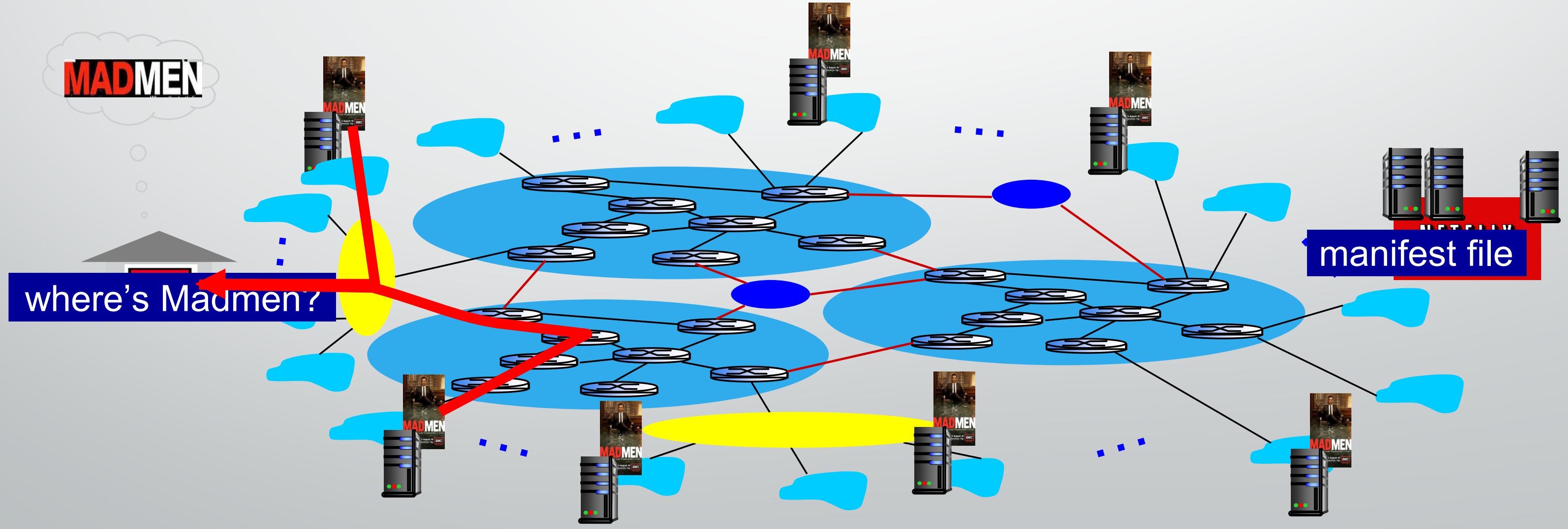
Content Distribution Networks (CDNs)

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

- **Option 1:** single and large "mega-server"
  - Single point of failure
  - Long path to distant clients
  - Single point of network congestion
  - This solution doesn't scale
  - Multiple copies of video sent over outgoing link
- **Option 2:** store multiple copies of videos at multiple 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 (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

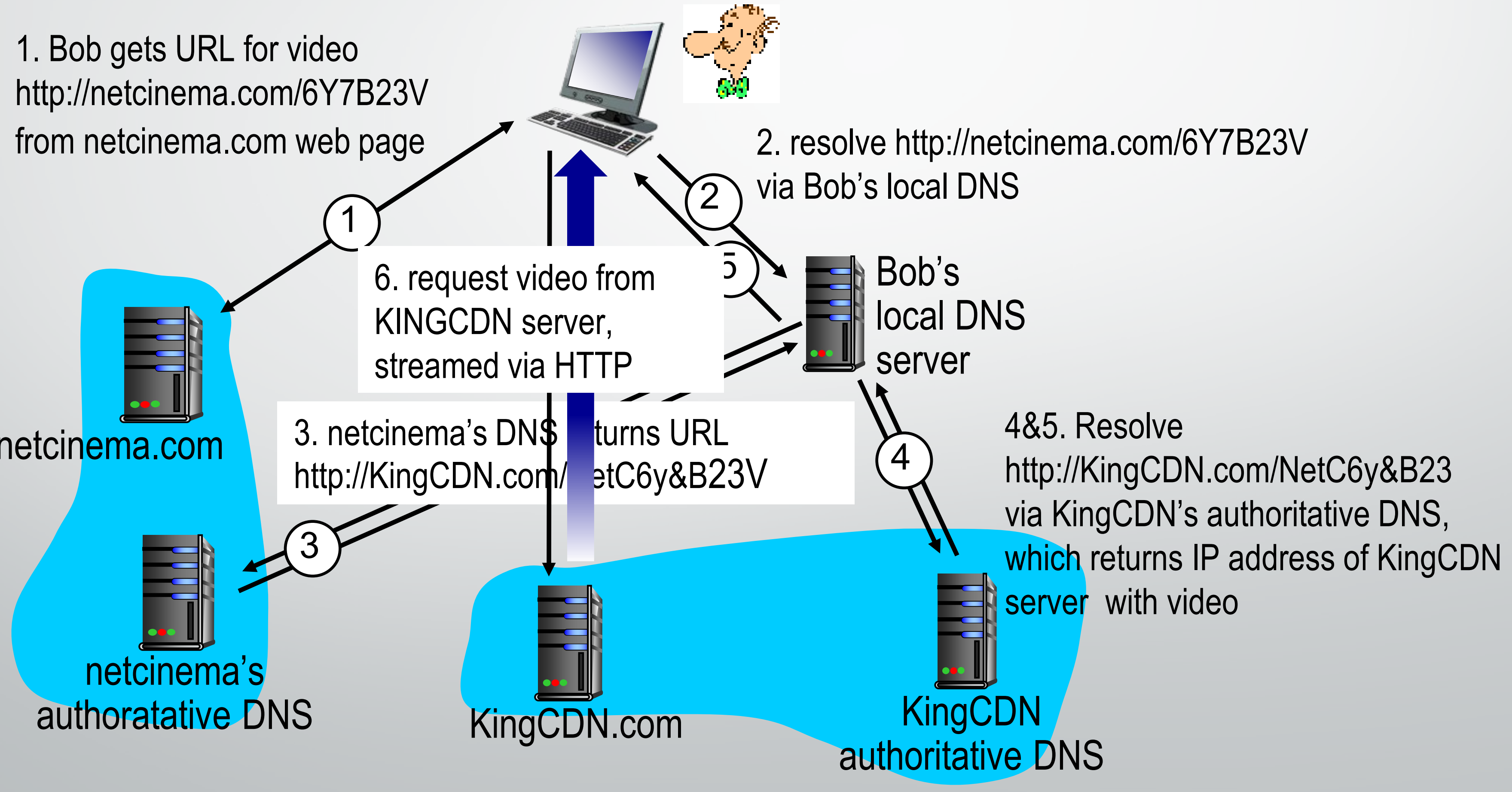


Content Distribution Networks (CDNs)

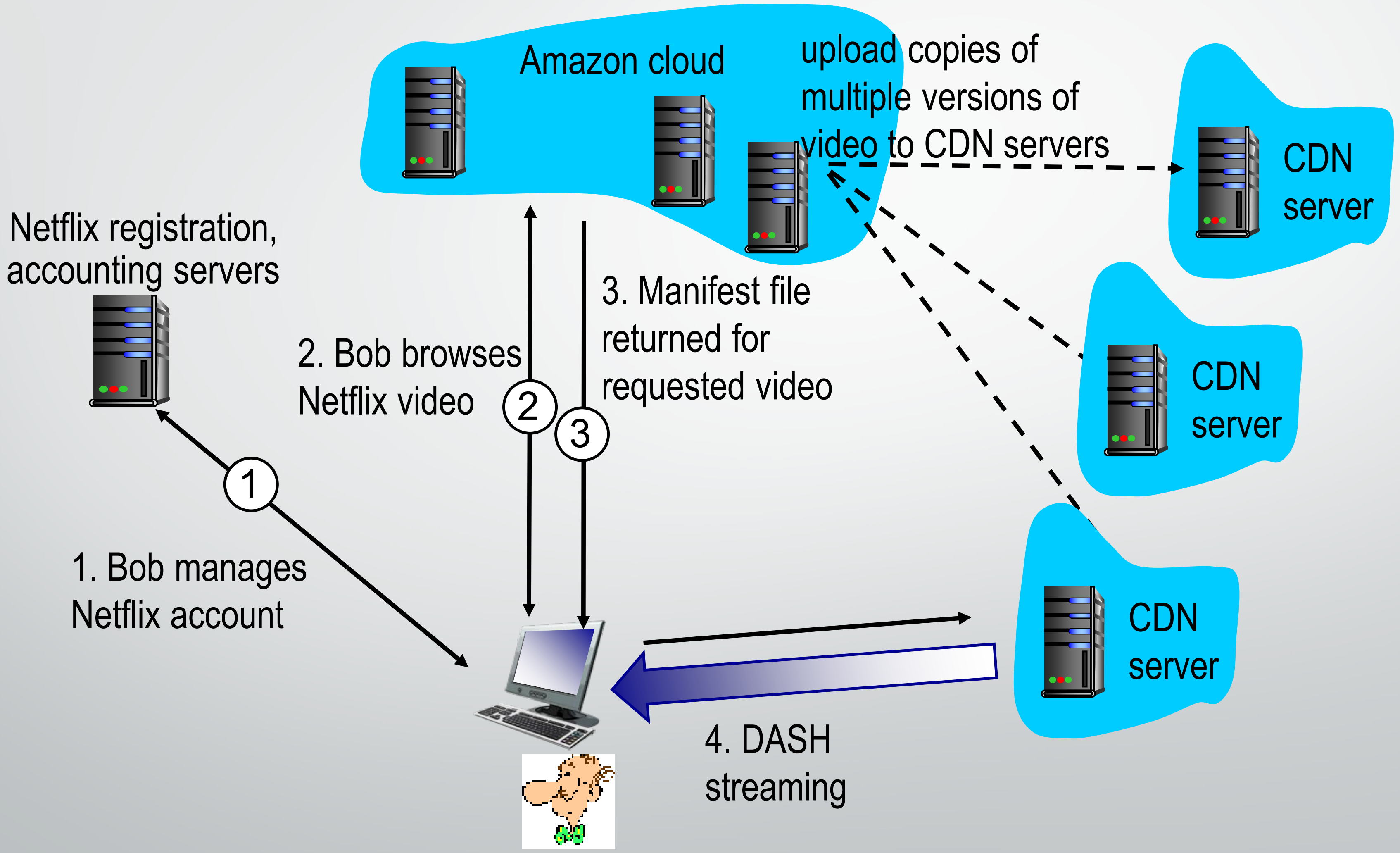
- **Over the top challenges:** coping with a congested Internet
  - From which CDN node to retrieve content for a user?
  - What's the viewer behavior in presence of a congestion?
  - What content to place in which CDN node?

CDN Content Access: A Closer Look

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



Case Study: Netflix



The End

- **References**

  - **[1]** Brownlee, M. [MKBHD]. (2019, October 12). This Is What Happens When You Re-Upload a YouTube Video 1000 Times! . Retrieved from https://www.youtube.com/watch?v=JR4KHfqw-oE
  - **[2]** Kurose, J. F., & Ross, K. W. (2017). Computer networking: A top-down approach.