CSC 4350 – Computer Networks

Lecture 9 – Peer-to-Peer Networks, Video Streaming

September 19, 2024

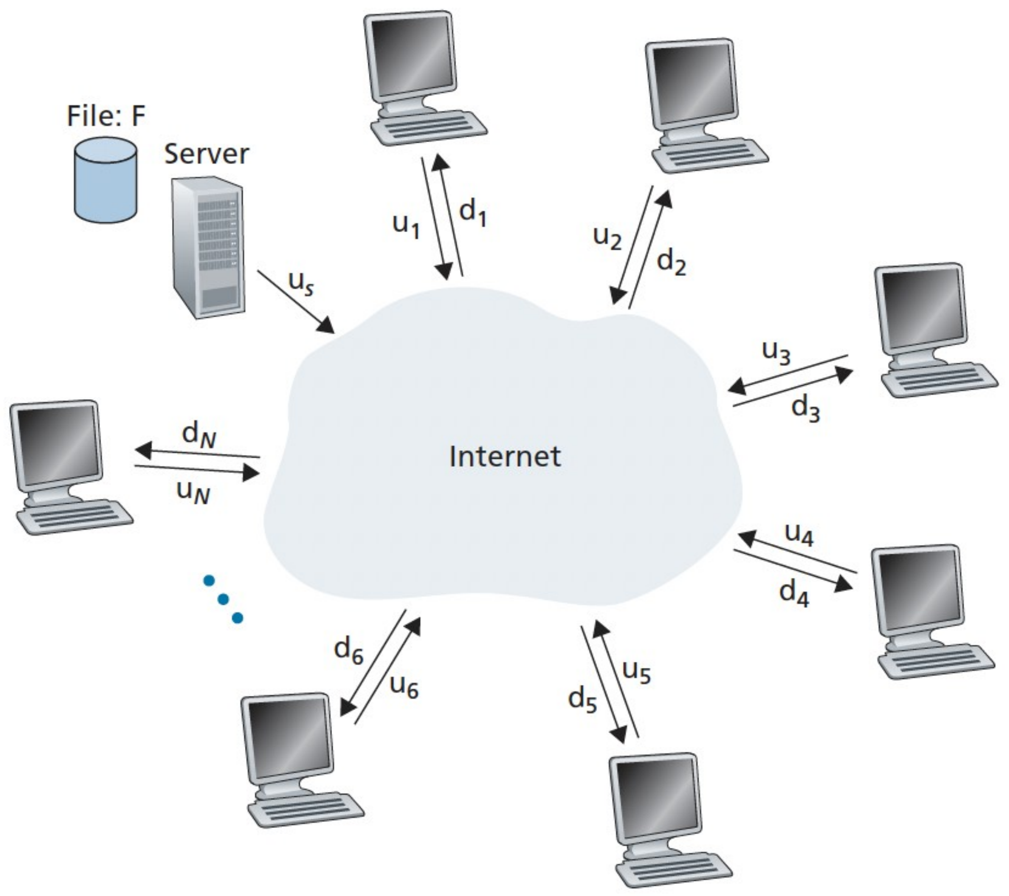
Note

- Material in this lecture is heavily borrowed from Kurose & Ross' "Computer Networking: A Top Down Approach, 8th Edition"

Peer-To-Peer File Distribution

- Recall: there is minimal/no reliance on always-on servers
- Pairs of intermittently connected hosts (peers) communicate with each other
  - PCs, laptops, smartphones
- Scalability
- Bit Torrent

Figure 2.22 – File Distribution Problem



Scalability of P2P Architectures

- Server and peers are connected to the Internet with access links
- Denote upload rate of server's access link by u_s, upload rate of the ith peer's access link by u_i, and download rate of the ith peer's access link by d_i
- Size of the file to be distributed in bits (F)
- Number of peers that want to obtain a copy of the file by N
- Distribution time – time it takes to get a copy of the file to all N peers
  - Assumption: Internet core has abundant bandwidth
    - All bottlenecks are access networks
  - Assumption: server and clients are not participating in any other network applications so that all of their upload and download access bandwidth can be fully devoted to distributing the file

Scalability – Client/Server

- Distribution time for client-server architecture – observations
  - Server must transmit one copy of the file to each of the N peers
    - Server must transmit NF bits
    - Since the server's upload rate is u_s, time to distribute the file must be at least NF/u_s
  - Let d_{min} denote the download rate of the peer with the lowest download download rate, $d_{min} = \{d_1, d_2, \ldots, d_N\}$
  - The peer with the lowest download rate cannot obtain all F bits of the file in less than F/d_{min} seconds
    - Minimum distribution time: F/d_{min}

Scalability – P2P

- At the beginning of the distribution, only the server has the file
  - To get the file into the community, server must send each bit of the file at least once into its access link
  - Minimum distribution time is at least F/U_s
- As with the client-server architecture, the peer with the lowest download rate cannot obtain all F bits of the file in less than F/d_{min} seconds
  - Minimum distribution time is at least F/d_{min}
- Observe that the total upload capacity of the system as a whole is equal to the upload rate of the server plus the upload rates of each of the individual peers
  - $u_{total} = u_s + u_1 + \ldots + u_N$
  - System must deliver F bits to each of the N peers, delivering a total of NF bits
    - Cannot be done at a faster rate than U_{total}
    - Min distribution time is at least $NF/(u_s + u_1 + \ldots + u_N)$

Figure 2.23 – Distribution Time for P2P and Client-Server Architectures

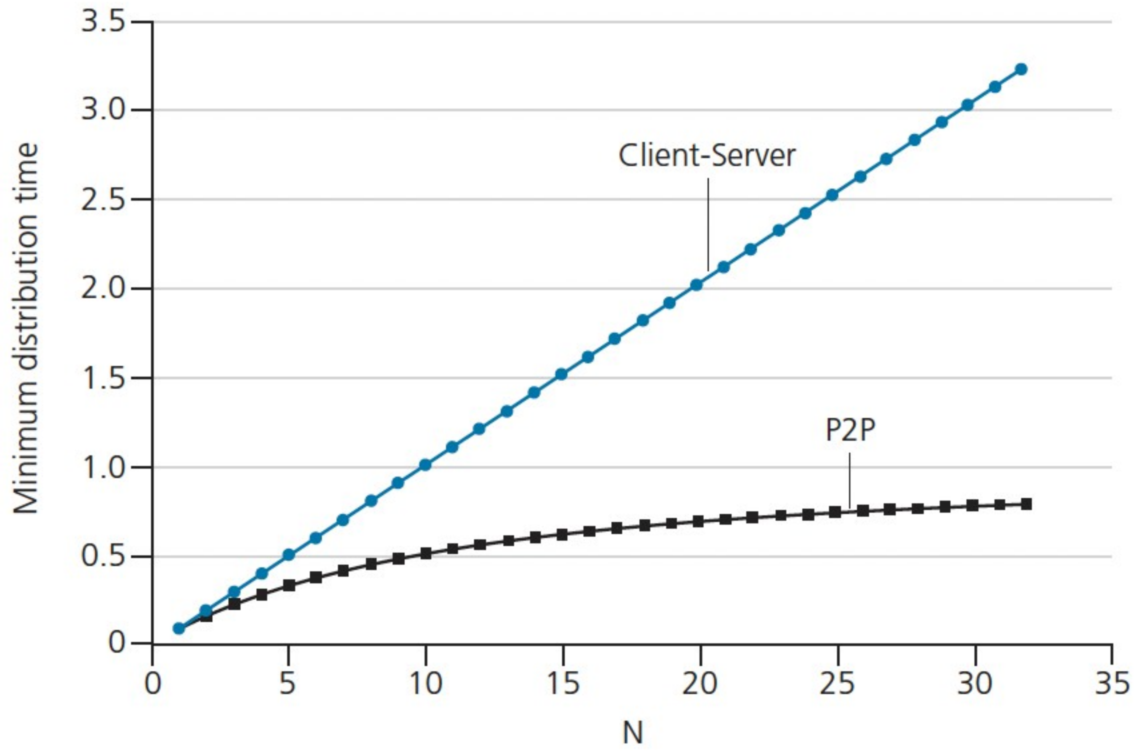


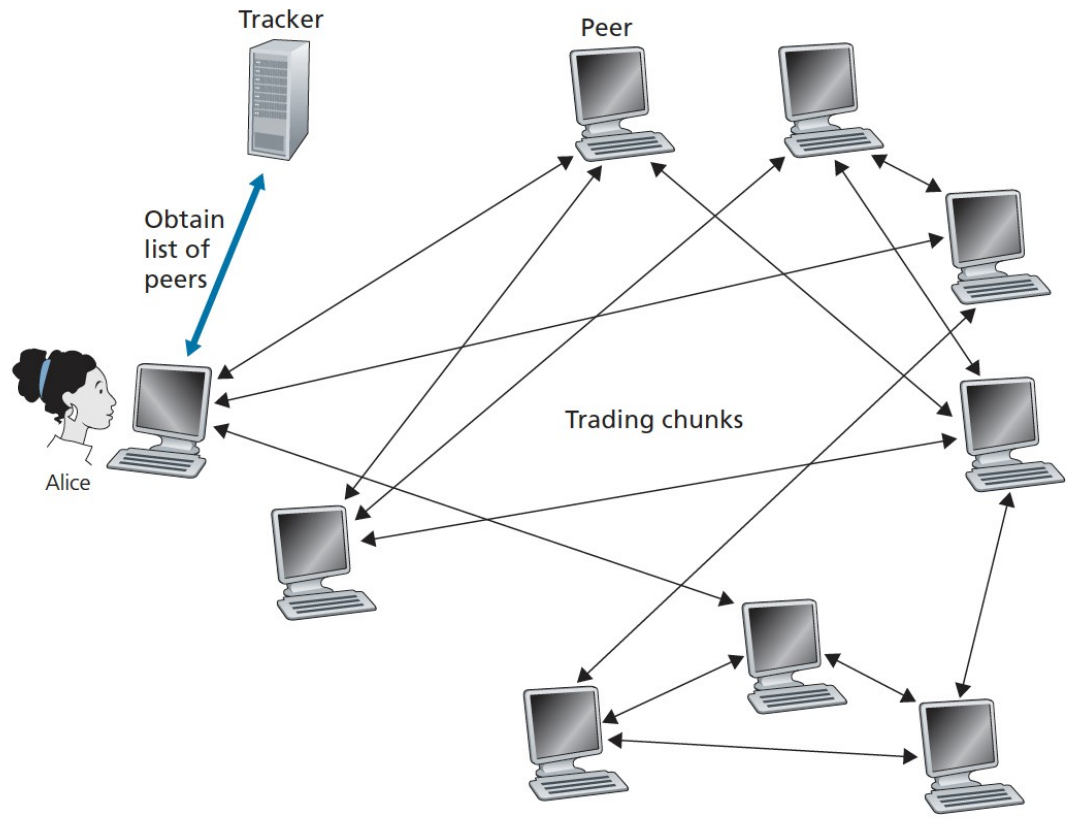
Figure 2.23 - Explained

- Compares minimum distribution time for the client-server and P2P architectures, assuming that all peers have the same upload rate u
- F/u = 1 hour, $u_s = 10u$, $d_{min} >= u_s$
- Peer can transmit an entire file in one hour
- Server transmission rate is 10 times the peer upload rate
- Peer download rates are set large enough so as not to have an effect
- Figure 2.23
  - Client-server – distribution time increases linearly and without bound as the number of peers increases
  - P2P – minimal distribution time is not always less than that of the client-server architecture
    - It is also less than one hour for any number of peers N
    - So, P2P can be self-scaling

BitTorrent

- Collection of all peers participating in the distribution of a particular file is called a torrent
- Peers in a torrent download equal-size chunks of the file from one another
  - Typical size: 256 kbytes
- When a peer first joins the torrent, it has no chunks
  - Over time it accumulates more chunks
  - While it downloads, it also uploads chunks to other peers
- Once a peer has acquired the entire file, it may leave the torrent or remain and continue to upload chunks to other peers
- Any peer may leave the torrent at any time with only a subset of chunks and later rejoin the torrent

Figure 2.24 – File Distribution with BitTorrent



More on BitTorrent

- Each torrent has an infrastructure node called a tracker
  - When a peer joins the torrent, it registers itself with the tracker and periodically informs the tracker that it is still in the torrent
  - Any given torrent may have fewer than 10 or more than 1000 peers participating at any instance of time
- Figure 2.24
  - When a new peer joins the torrent, the tracker randomly selects a subset of peers from the set of participating peers and sends the IP addresses of those peers to the user machine
  - Possessing the list – attempt to establish concurrent TCP connections with all of the peers on the list
  - Success in establishing a connection – “neighboring peer”
    - Number of neighboring peers fluctuates over time

More on BitTorrent

- Each peer will have a subset of chunks from the file, with different peers having different subsets
- Ask each neighboring peer for the list of chunks they have
- In deciding which chunks to request, use a technical called rarest first
  - Determine from among the chunks remaining, those that are rarest among her neighbors
  - Request the rarest first; get redistributed more quickly
- Respond to requests – trading algorithm
  - Priority given to the neighbors currently supplying data at the highest rate
  - Recalculated every 10 seconds
  - Reciprocates sending chunks to these same peers
  - Peers are said to be unchoked
- Every 30-ish seconds, a new trading partner will be chosen at random to initiate trading
  - If 2 peers are satisfied with the trading, they will put each other on their top lists and continue trading with each other until one of the peers finds a better partner

Video Streaming and Content Distribution Networks

- Internet Video
- HTTP Streaming and DASH
- Content Distribution Networks

Internet Video

- Underlying medium – prerecorded video
  - Placed on servers and users send requests to the servers to view the videos on demand
- Video – sequence of images, typically displayed at a constant rate, like 24-30 images per second
- Uncompressed, digitally encoded image consists of an array of pixels, with each pixel encoded into a number of bits for luminance and color
- Can be compressed – trade-off of video quality to bit rate
- Most important aspect – high bit rate
  - Compressed internet video typically ranges from 100 kbps for low-quality video to over 4Mbps for streaming high-definition movies
  - 4K streaming – bitrate of more than 10Mbps
  - Translates to huge amounts of traffic and storage, esp. high-end video
    - A single 2Mbps video with a duration of 67 minutes will consume 1 gigabyte of storage and traffic
  - Utilize compression rates to help with size, congestion

HTTP Streaming and DASH

- When a user wants to see a video, the client establishes a TCP connection with the server and issues an HTTP GET request
- Server sends the video file within an HTTP response message as quickly as the underlying protocols and traffic will allow
- Client side – Bytes are collected in a client application buffer
- Once a number of bytes exceeds the threshold, the streaming video application grabs video frames from the client buffer, decompresses the frames, and displays them on screen
- Major shortcoming: all clients receive the same encoding of the video, despite large variations in bandwidth available
- Development of Dynamic Adaptive Streaming over HTTP (DASH)

DASH

- Video is encoded into several different versions, each with a different bit rate and a different quality level
- Client dynamically requests chunks of video segments of a few seconds in length
- When the amount of available bandwidth is high, the client selects chunks from the high-rate version; when bandwidth is low, it will select from a low-rate version
- Client selects different chunks one at a time with HTTP GET request messages
- DASH allows clients with different access rates to stream in video at different encoding rates
- Each version is stored in the HTTP server, each with a different URL
- HTTP server also has a manifest file – provides a URL for each version along with its bit rate

DASH and the Client

- The client first requests the manifest file and learns about different versions
- Client selects one chunk at a time by specifying a URL and a byte range in an HTTP GET request message for each chunk
- While downloading chunks, the client also measures received bandwidth and runs a rate determination algorithm to select the chunk to request next
- If client has a lot of video buffered and if measured bandwidth is high, it will choose a chunk from a high-bitrate version and vice-versa

Content Distribution Networks – Rationale

- Many of the Internet video companies are distributing on-demand multi-Mbps streams to millions of users on a daily basis
  - YouTube is definitely one of many (mentioned in the text)
- Streaming traffic to locations all over the world while providing continuous playout and high interactivity is clearly a challenging task
- Single massive center?
  - If the client is far from the data center, packets will cross (potentially) many communication links and pass through many ISPs, with some located on different continents
  - Popular video will likely be sent many times over the same links; waste of bandwidth; video company, itself, will be paying the provider ISP for sending the same bytes over and again
  - Single point of failure – if the server goes down, how can I get content?
- A potential solution: Content Distribution Networks (CDNs)

CDNs

- Manages servers in multiple geographically distributed locations
- Stores copies of the videos/other web content in its servers
- Attempts to direct each user request to a CDN location that will provide the best user experience
- Private CDN – owned by the content provider itself
  - Google's CDN distributes YouTube videos
- Third-Party CDN – distributes content on behalf of multiple content providers

CDNs – Server Placement Philosophies

- Enter deep into the access networks of ISPs
  - Deploy server clusters in ISPs all over the world
  - Goal: get close to end users, hopefully improving user-perceived delay and throughput by decreasing the number of links/routers/hops between end user and CDN server delivering content
  - Highly-distributed, maintaining and managing the clusters becomes a challenge
- Bring the ISPs home by building large clusters at a smaller number of sites
  - Instead of getting inside the access ISPs, these CDNs place clusters in Internet Exchange Points (IXPs)
  - Lower maintenance/management overhead
  - Problems: higher delay and lower throughput to end users

CDNs

- Once clusters in place, the CDN replicates content across clusters
- Many not want to place a copy of every video in each clusters, due to rare viewings
  - Utilize a pull strategies
    - If a client requests a video from a cluster not storing the video, then the cluster retrieves thee video and stores a copy locally
    - When a cluster’s storage becomes full, it removes videos that are not frequently requested

CDN Operation

- When a browser in a user’s host instructed to retrieve a specific video, the CDN must intercept the request so it can
  - Determine a suitable CDN server cluster for that client at that time
  - Redirect the client’s request to a server in that cluster
- Most CDNs take advantage of DNS to intercept and redirect requests
- Example: Content provider NetCinema employs the third-party CDN company KingCDN to distribute video to customers
  - On NetCinema web pages, each of its videos is assigned for a URL that includes the string “video” and unique identifier for the video itself
  - For instance, ”The Dark Knight” might be assigned http://video.netcinema.com/6Y7B23V

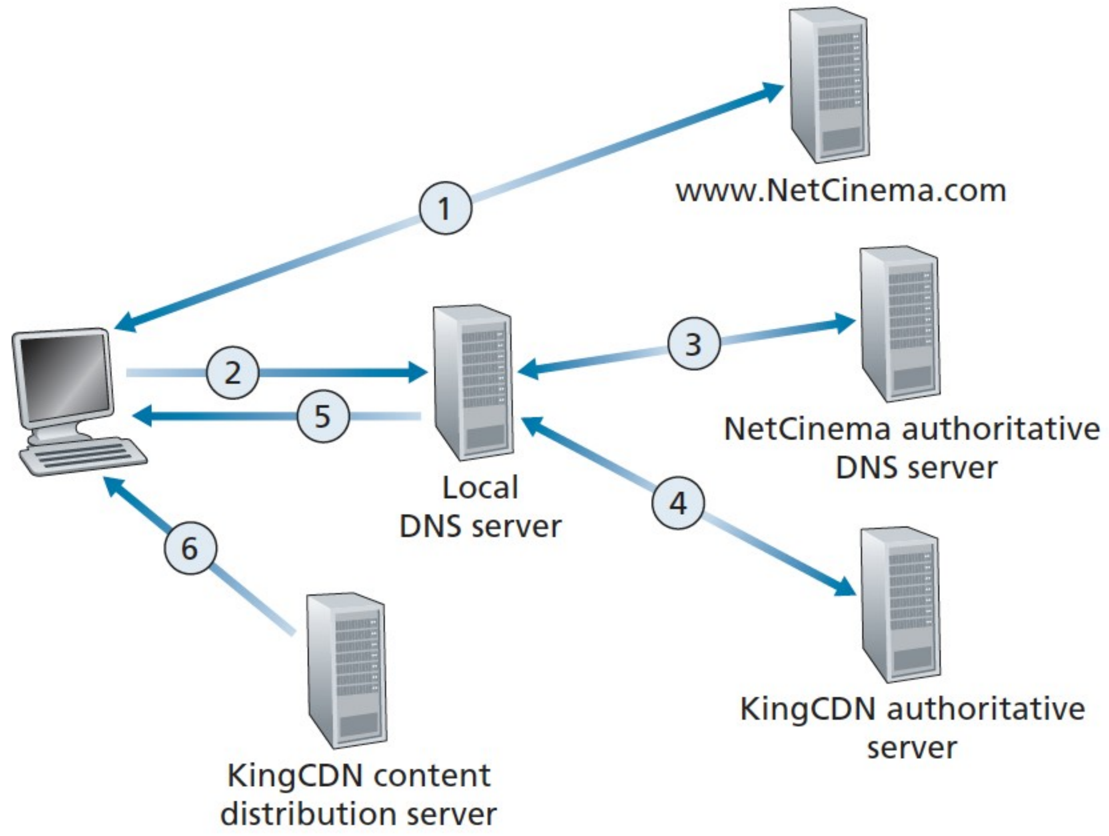
Steps (Also Illustrated on Figure 2.25)

- User visits the web page at NetCinema
- When the user clicks on the link http://video.netcinema.com/6YB23V, the user's host sends a DNS query for video.netcinema.com
- The user's local DNS Server (LDNS) relays the DNS query to an authoritative DNS server for NetCinema
  - Observes the string "video" in the hostname video.netcinema.com
  - To hand over the query to KingCDN, instead of returning an IP address, the NetCinema authoritative DNS server returns to the LDNS a hostname in the KingCDN's domain, like maybe a1105.kingcdn.com

Steps (Also Illustrated on Figure 2.25)

- The DNS query enters into KingCDN’s private DNS infrastructure
  - User’s LDNS then sends a second query, now for a1105.kingcdn.com and KingCDN’s DNS system eventually returns the IP addresses of a KingCDN content server to the LDNS
  - Within the KingCDN’s DNS system – CDN server from which the client will receive its content is specified
- The LDNS forwards the IP address of the content-serving CDN node to the user’s host
- Once the client receives the IP address for a KingCDN content server, it establishes a direct TCP connection with the server at that IP address
  - Issues an HTTP GET request for the video
  - If DASH used – server will first send to the client a manifest file with a list of URLs, one for each version of the video
  - Client will dynamically select chunks from the different versions

Figure 2.25 – DNS Redirects a User’s Request to a CDN Server



www.NetCinema.com
1
2
5
3
NetCinema authoritative DNS server
Local DNS server
4
6
KingCDN authoritative server
KingCDN content distribution server

Cluster Selection Strategies

- The core of any CDN deployment
  - Previous slides: CDN learns the IP address of the client's LDNS server via the client's DNS lookup
  - After learning this IP address, the CDN needs to select an appropriate cluster based on the IP address
  - CDNs generally employ proprietary cluster selection strategies
- Assign client to the cluster that is geographically closest (simple strategy)
  - Using commercial geo-location databases, each LDNS IP address is mapped to a geographic location
  - When DNS request is received from a particular LDNS, CDN chooses the geographically closest cluster
  - Works well for majority of clients
  - May not work well for some: may not be the closest cluster in terms of the length/number of hops of the network path
  - Ignores variation in delay and available bandwidth over time, always assigning the same cluster to a particular client
  - Maybe combat using real-time measurements of delay and loss performance between clusters and clients?
    - Not many LDNSs are configured to respond to such probes/queries