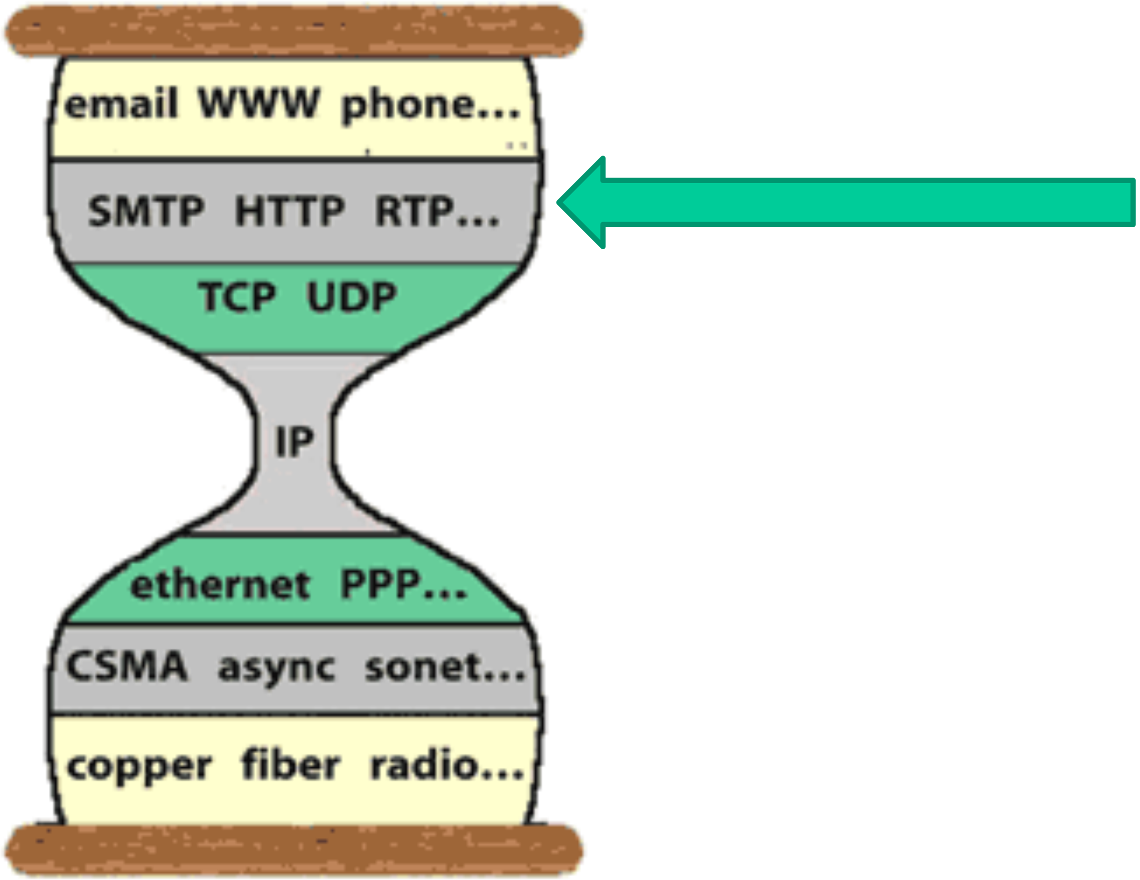
# A Little Bit More of Application Layer




Data distribution methods

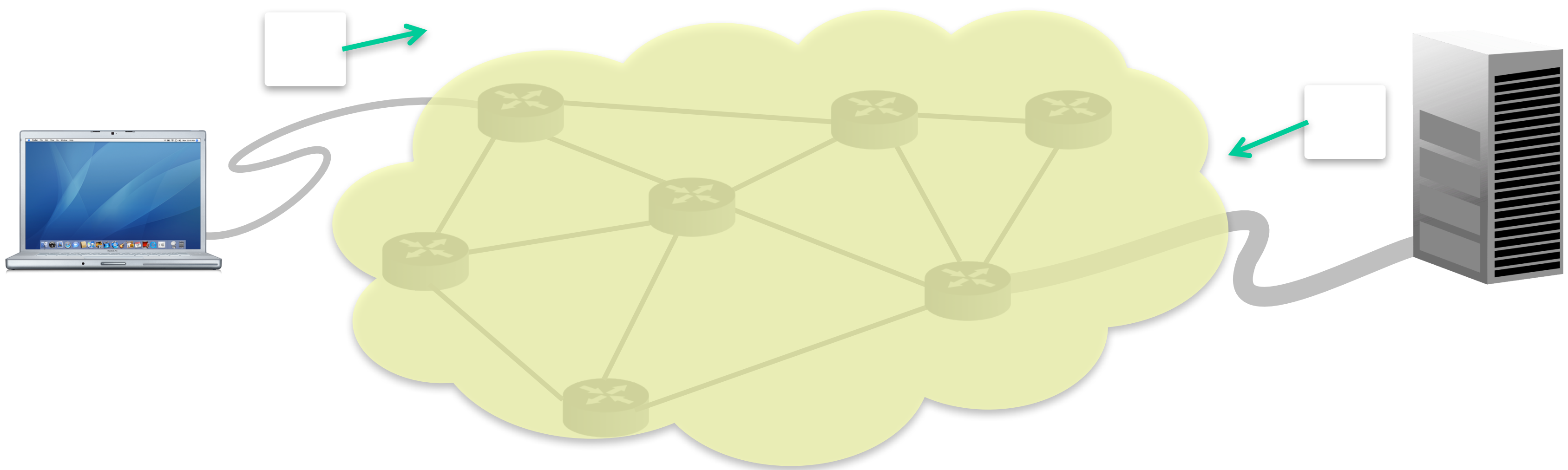
Content Distribution Networks

Peer-to-Peer

Internet Video

# Data Distribution

- Client-server model
  - HTTP, NFS, AFP, SCP, RSYNC, ...

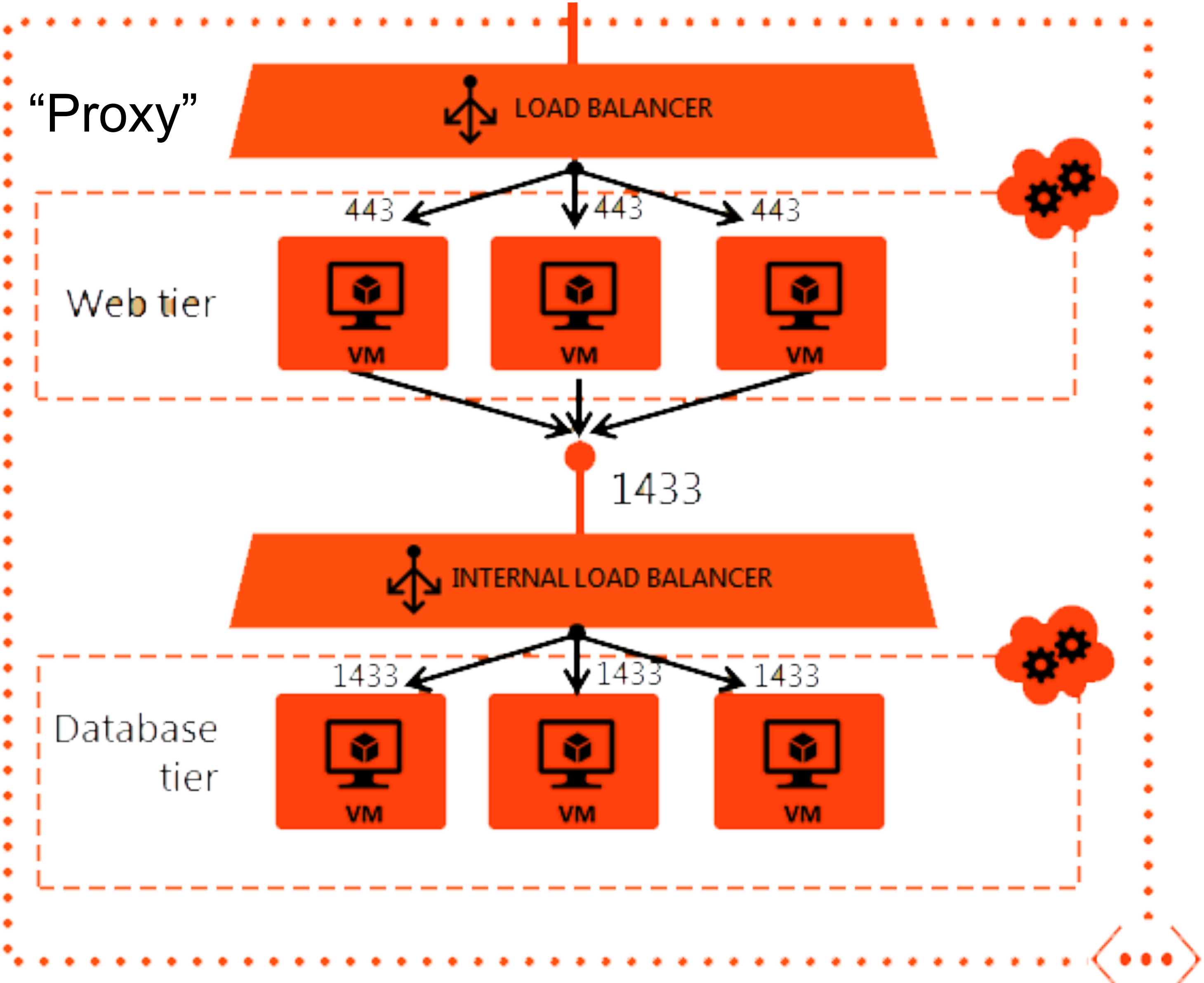


Main concern: how to scale?

# Scaling Data Distribution

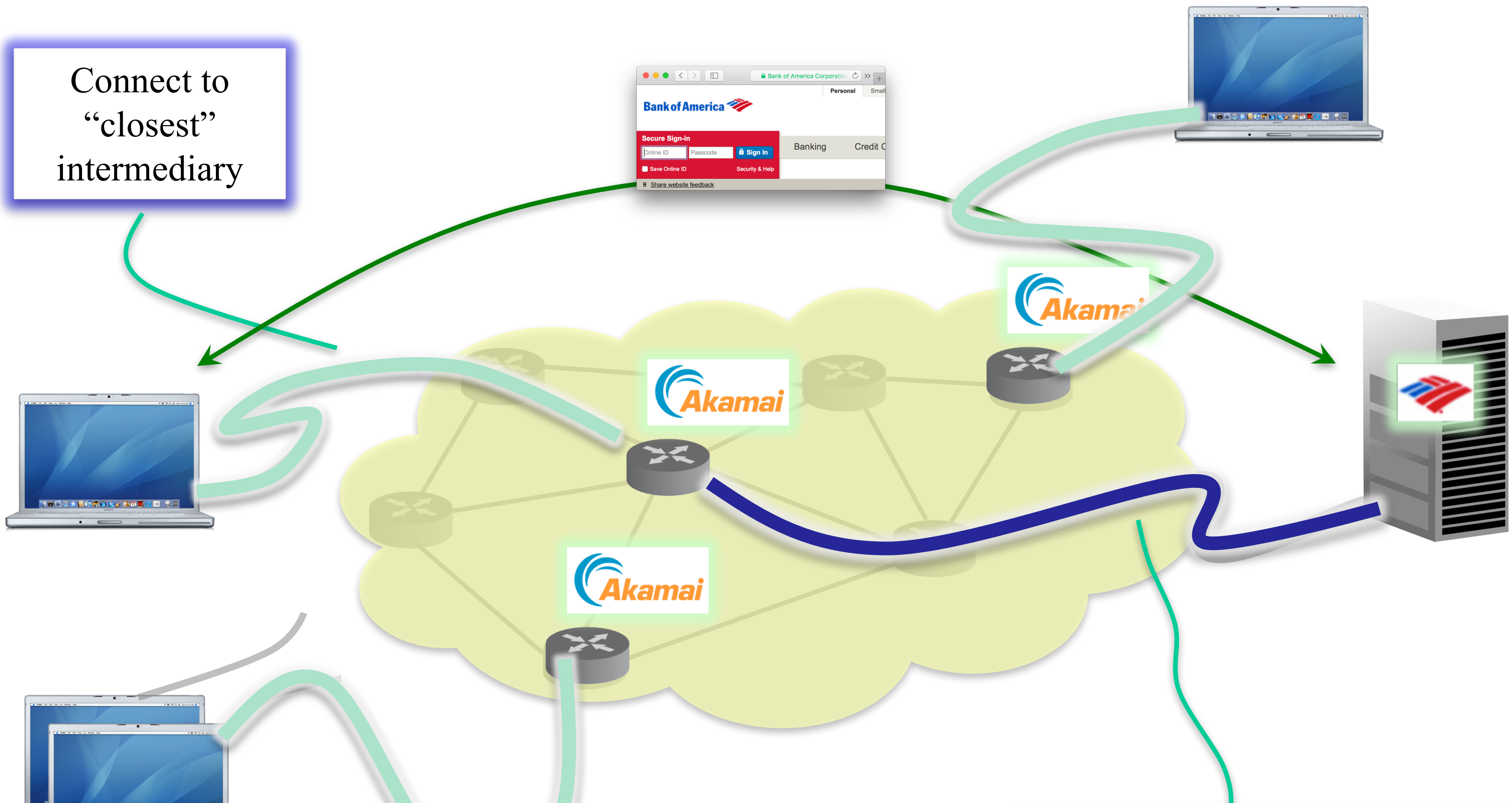
- Remember DNS from previous lecture
- Does it scale?
- How?
  - Replication of authoritative servers
  - Aggregation and suppressing of similar requests by caching resolvers

# Scaling Data Distribution: Load Balancers



# Scaling Data Distribution: Content Delivery Networks (CDNs)

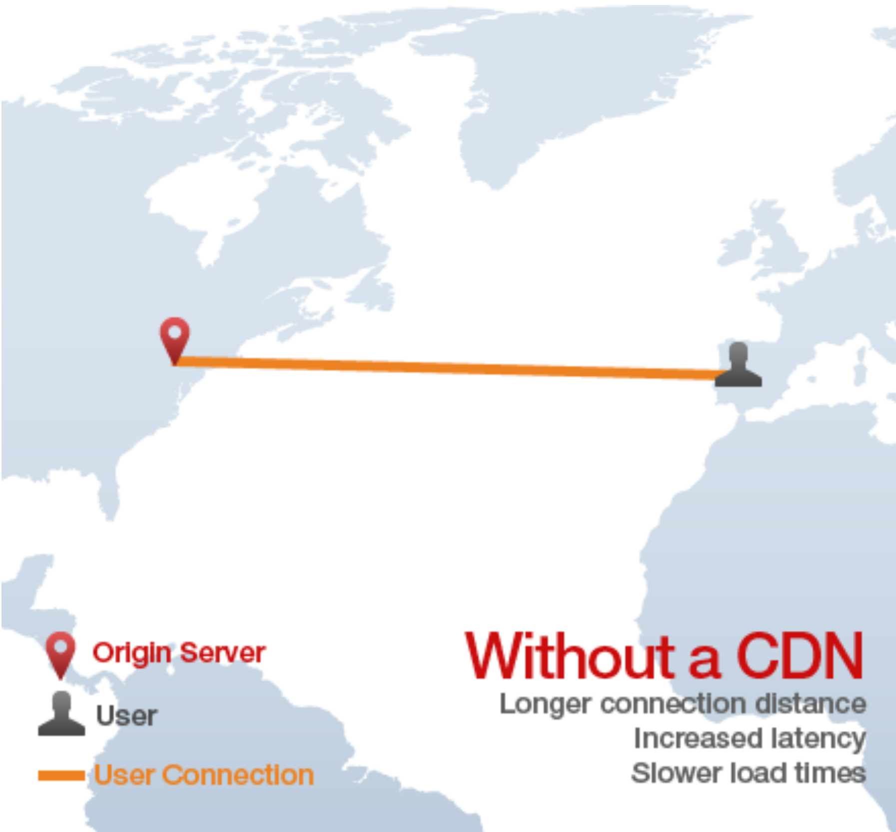
Connect to “closest” intermediary

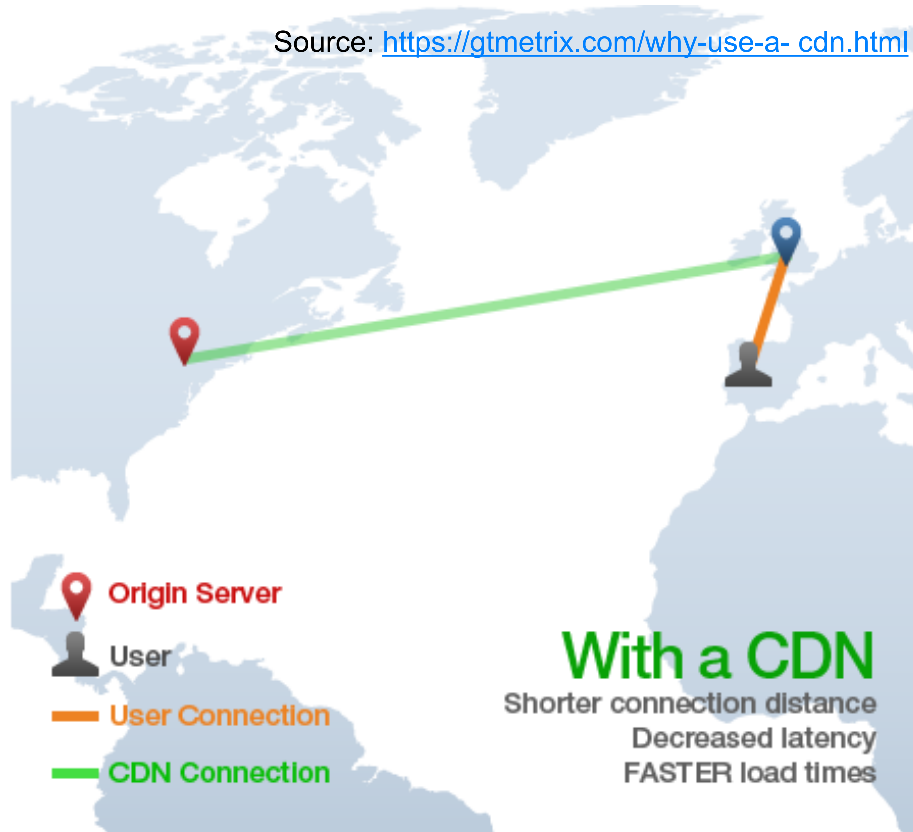



If needed, intermediary will forward request to the real back-end

# With And Without CDNs

Source: https://gtmetrix.com/why-use-a- cdn.html



# CDN Vendors

- Akamai
- Alcatel Lucent (carrier platform)
- Allot Communications (traffic management)
- Amazon
- ARA Networks (traffic management)
- Aryaka
- Blue Coat (transparent caching)
- Broadpeak (carrier platform)
- BTI Systems (traffic management)
- CDNetworks
- Cedexis (traffic management)
- CDN77
- ChinaCache
- Cisco (carrier platform)
- Conversant (carrier platform)
- Comcast
- Conviva (analytics)
- DeepField (analytics)
- Edgeware (carrier platform)
- Ericsson (carrier platform)
- Fastly
- Fortinet (traffic management)
- Hibernia Networks
- Highwinds
- Huawei (carrier platform/transparent caching)
- Instart Logic
- Internap
- Jetstream (licensed CDN)
- Juniper (transparent caching)
- LeaseWeb
- Level 3
- Limelight Networks
- MaxCDN
- Microsoft (Windows Azure)
- MileWEB
- Mirror Image
- OnApp (traffic management)
- PeerApp (transparent caching)
- Qwilt (transparent caching)
- Radware
- Revsw (mobile CDN)
- Solbox (licensed CDN)
- Swiftserve (licensed CDN)
- Tata Communications
- Verizon EdgeCast
- Vidscale (carrier platform)
- Yottaa

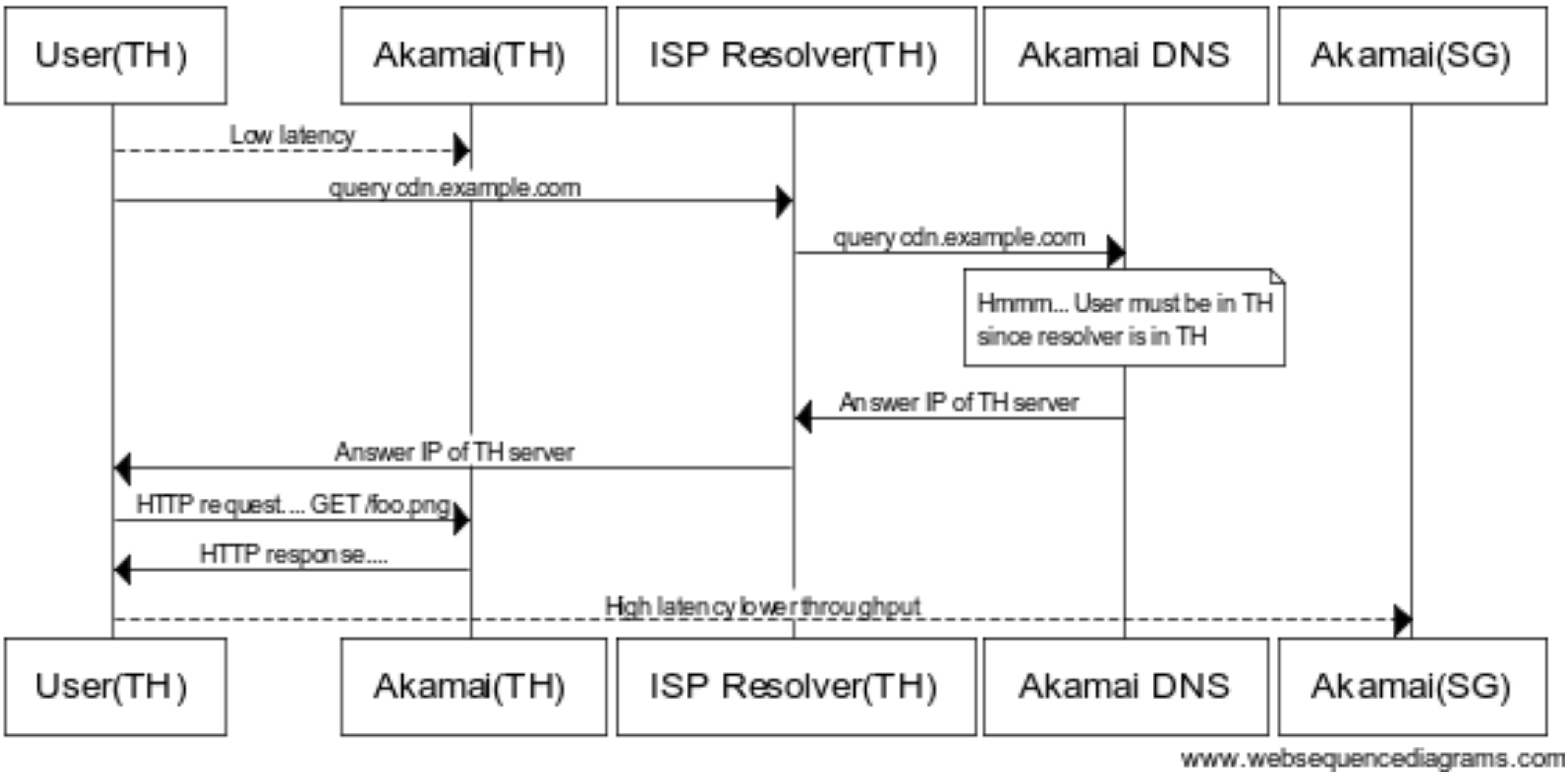
http://blog.streamingmedia.com/2014/07/cdnvendors.html

# How to Connect to “Closest” CDN Node

- Abuse DNS
  - send DNS query for the name
  - DNS server
    - extracts IP address from the query
    - maps IP to “closest” network or geo coordinate
    - returns set of IP addresses that seem to be closer to you

- This is what DNS Scavenger hunt challenge is about

# How CDNs work




Source: http://www.cdnplanet.com/blog/which-cdns-support-edns-client-subnet/

# A Caveat

- DNS ”redirection” works great if users use ISP-provided caching resolvers
  - Source IP of the caching resolver determines which answer authoritative DNS resolver returns

- But there is a problem with “public” DNS servers
  - Google’s 8.8.8.8, 8.8.4.4
  - What answer to return to caching resolver 8.8.8.8?
  - CDN’s DNS server will see request from google, not from client

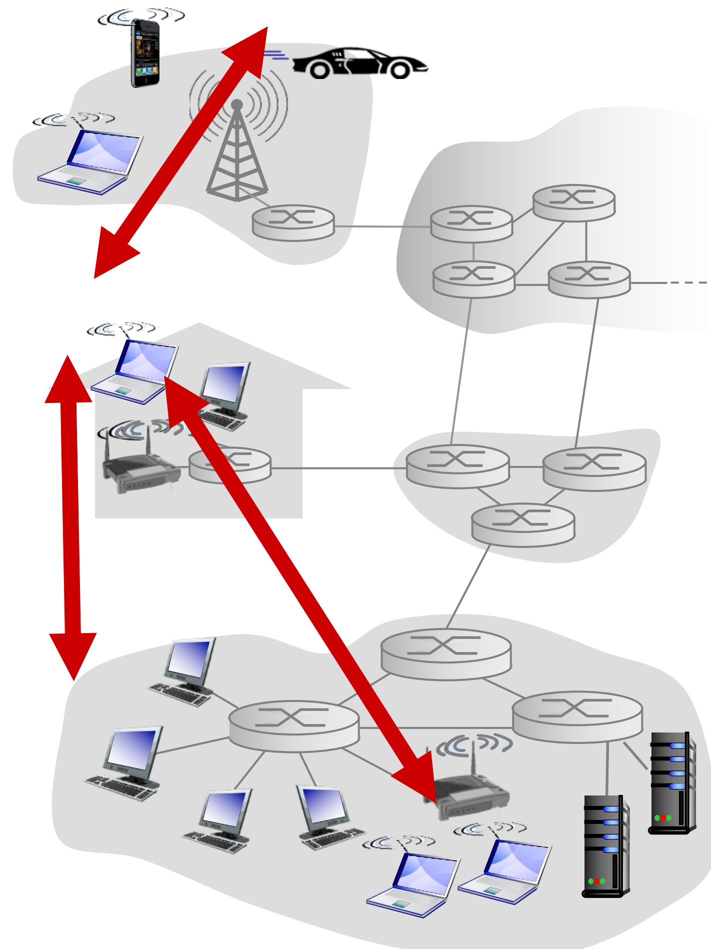
- Solution
  - DNS protocol was extended to include ”edns-client-subnet” option, which is set by caching resolver to tell authoritative name server IP of a client
  - How does this work with caching?

# Other Ways to Scale Data Distribution

- Peer-to-peer networks
  - Unstructured p2p networks
    - BitTorrent
    - Gnutella, Gossip, and Kazaa
  - Structured p2p networks (will not cover, but recommend reading in the textbook)
    - Chord, CAN, Tapestry, Pastry, Kademlia

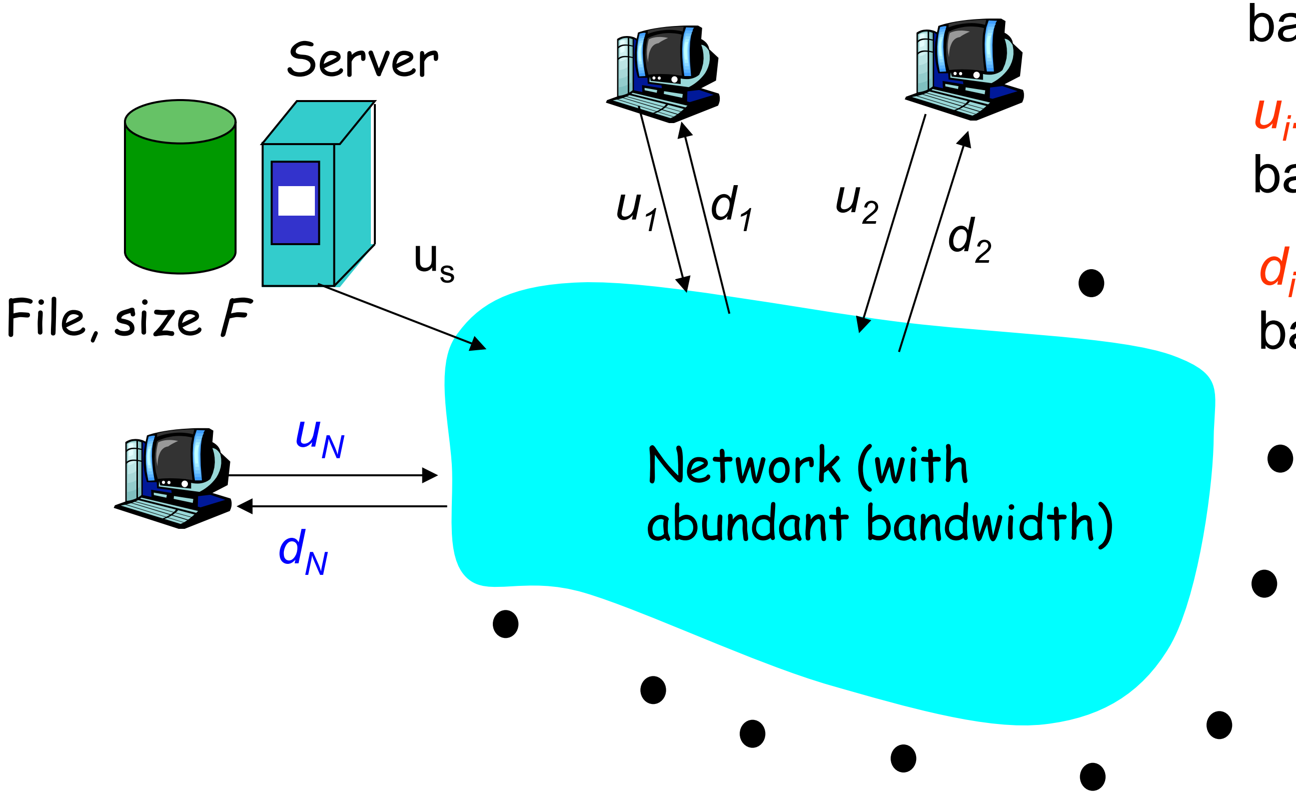
# Properties of Peer-to-Peer Systems

- No "always-on" server
- Arbitrary end systems directly communicate
- Peers are intermittently connected and change IP addresses



# File Distribution: Client-Server vs P2P

*Question* : How much time needed to distribute a file from one server to *N clients*?



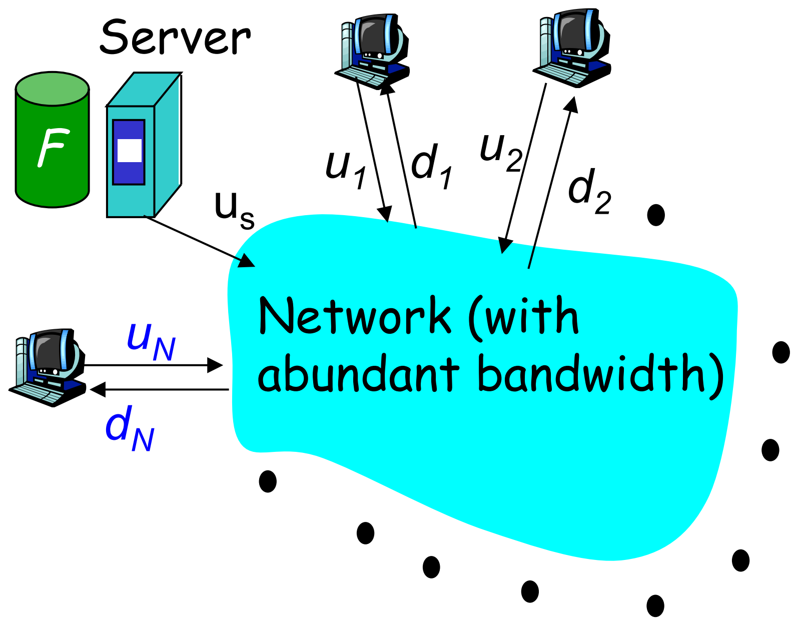

$u_s$: server upload bandwidth

$u_i$: peer i upload bandwidth

$d_i$: peer i download bandwidth

# File distribution time: client-server

- server sequentially sends (upload) N copies:
  - time to send one copy:
    - $F/u_s$
  - time to send N copies:
    - $N * F/u_s$ (lower bound)
- client i takes $F/d_i$ time to download

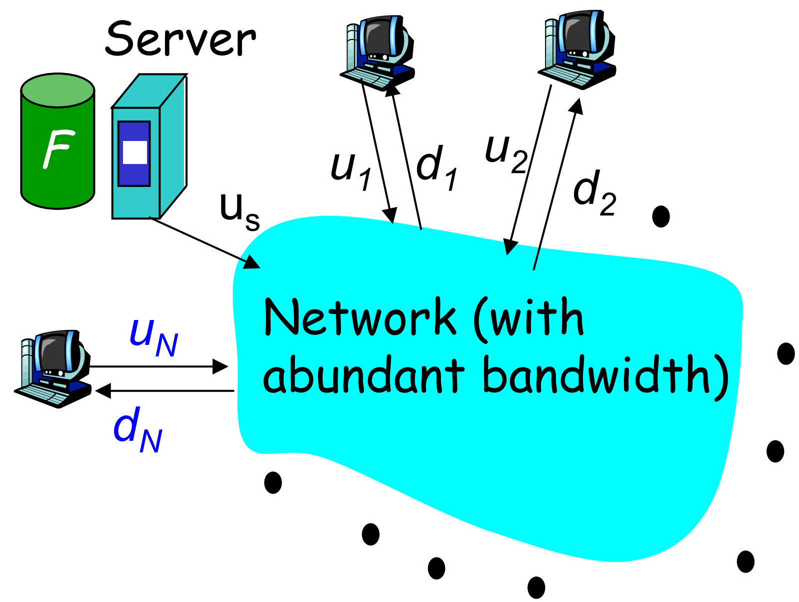



Time to distribute $F$ to $N$ clients using Client-server approach $= d_{cs} = \max \{ N * F/u_s, F/\min_i(d_i) \}$

increases linearly with N

# File distribution time: P2P

- server must send one copy:
  - Time to send one copy: $F/u_s$
- client *i* takes $F/d_i$ time to download
- Total N*F bits must be downloaded (aggregate)
  - max upload rate is $u_s + \sum u_i$




*time to distribute F to N clients using P2P approach*

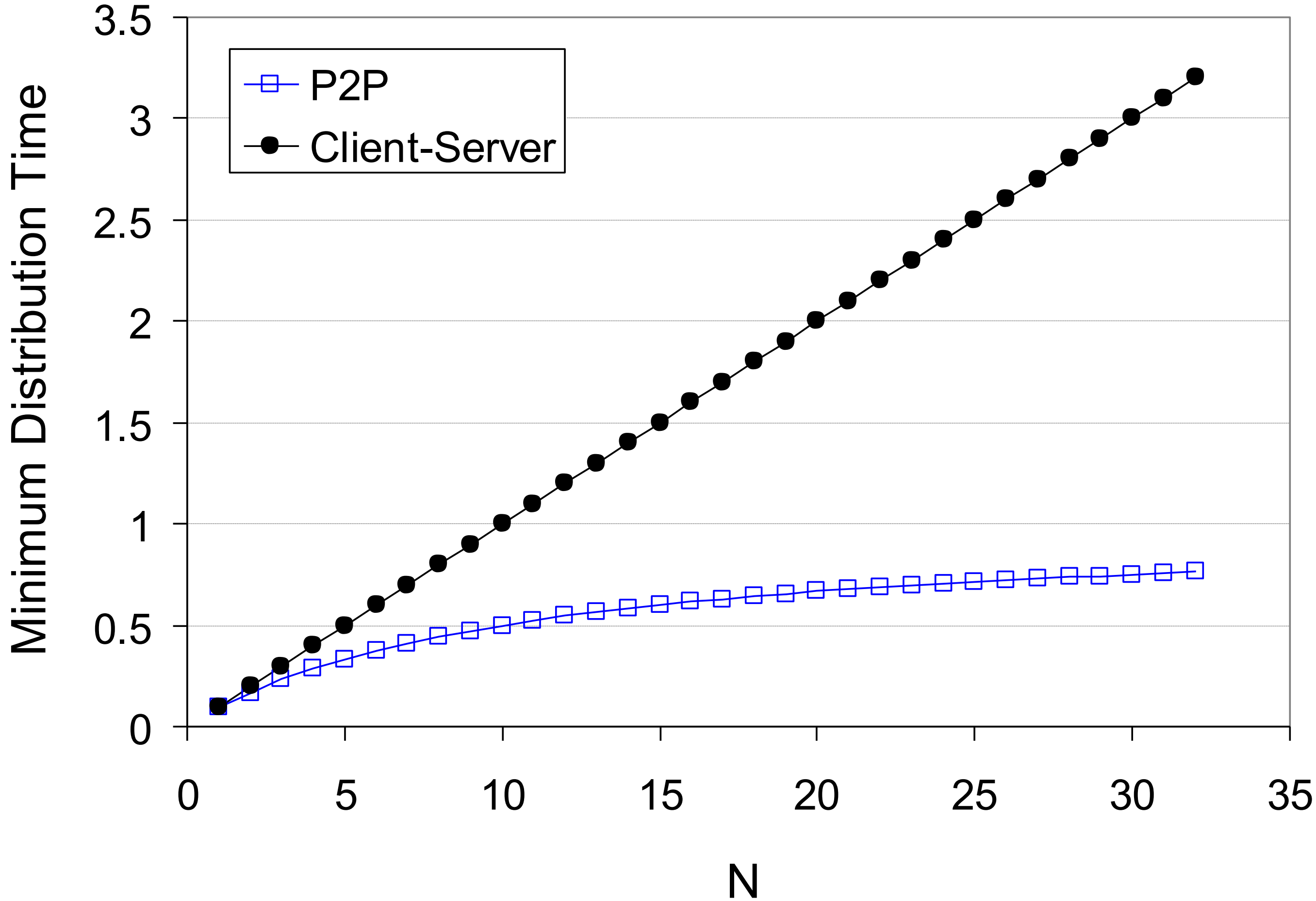
$$d_{p2p} \geq \max \{ F/u_s, F/\min(d_i), N{*}F/(u_s + \sum u_i) \}$$

increases linearly in *N* …

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

# Client-server vs. P2P: example

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



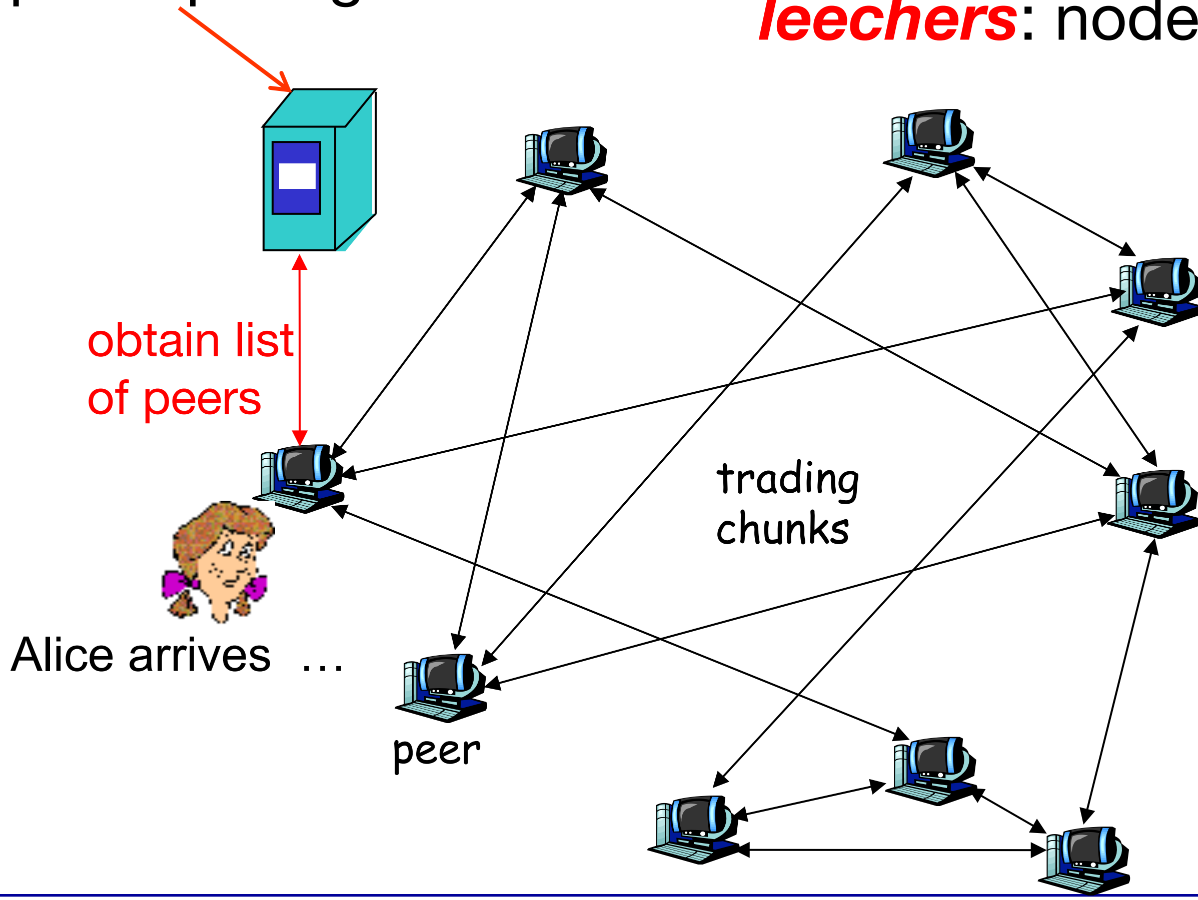
# P2P File distribution: BitTorrent

***tracker:*** *centralized* server that keeps track of participating nodes

***torrent:*** group of peers exchanging chunks of a file

***seeds***: node with entire content

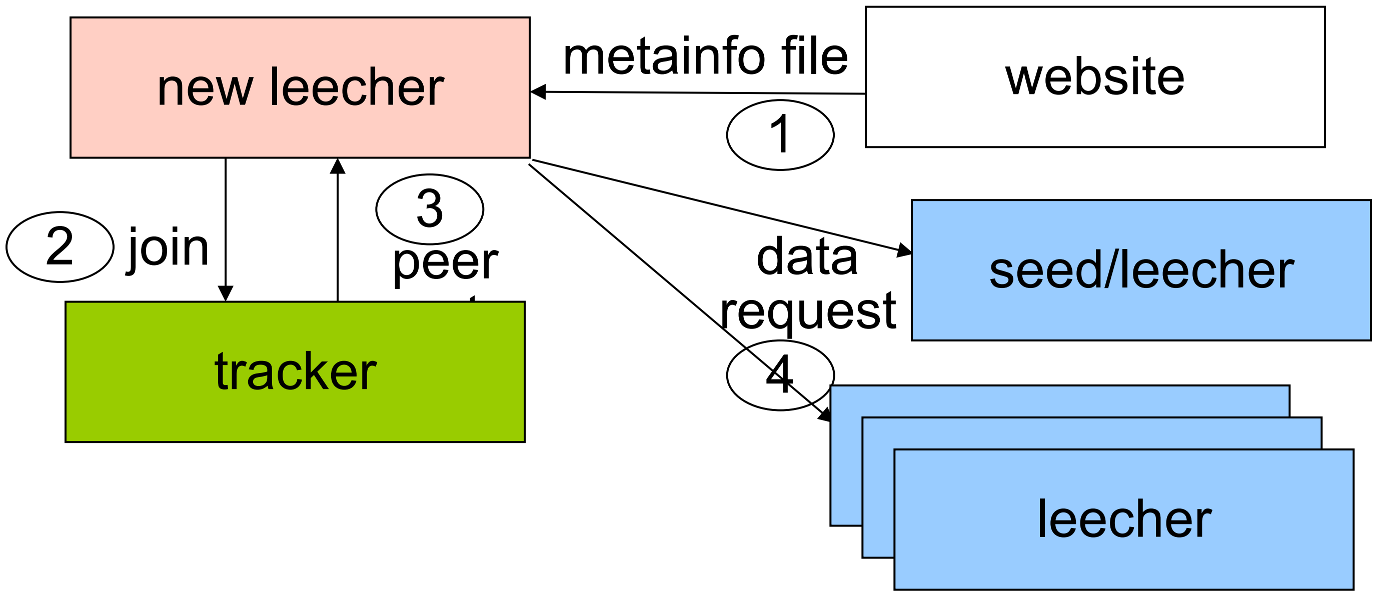
***leechers***: node still downloading




# BitTorrent Philosophy

- Author: Bram Cohen
- Based on Tit-for-tat
- Incentive - Uploading while downloading
- Pieces of files

# BitTorrent – joining a *torrent*




1. obtain the *metainfo file* from a website
2. contact the control server (tracker)
3. obtain a *peer set* (contains seeds & leechers)
4. contact peers in the peer set to request data

# BitTorrent: moving data

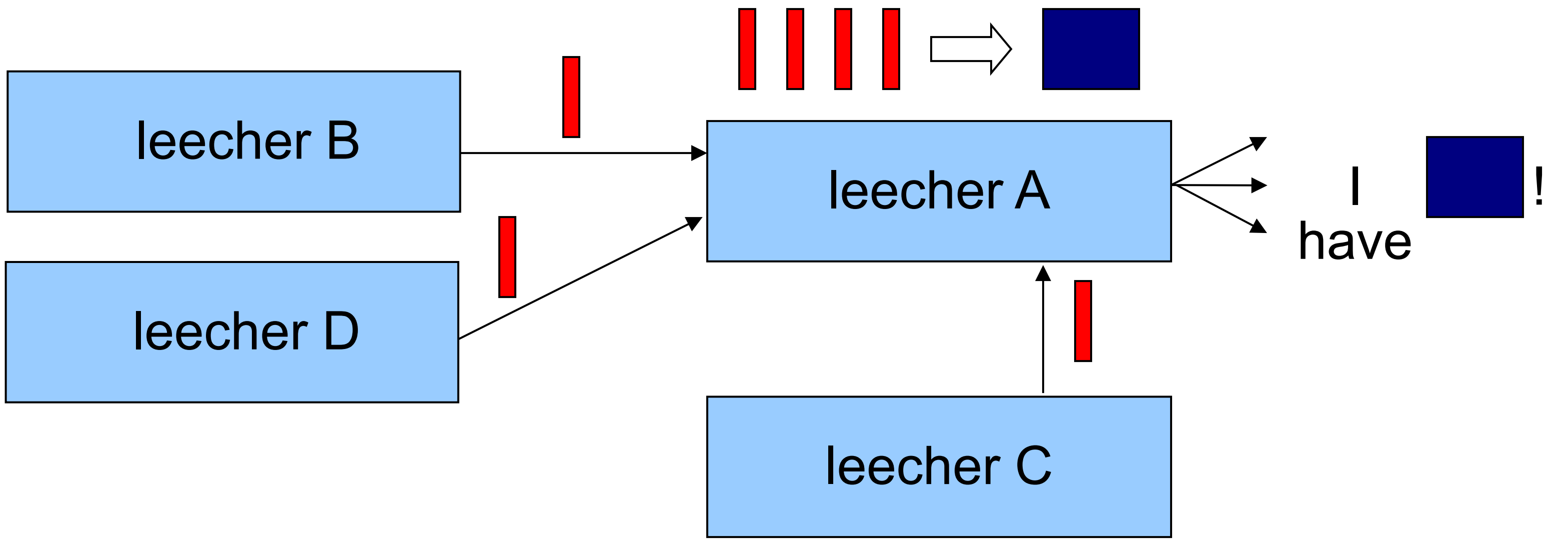
## Pulling Chunks

- file divided into 256KB *chunks*
- at any given time, different peers may have different subsets of file chunks
- periodically, a peer (Alice) asks each neighbor for the list of chunks they have
- Alice sends requests for her missing chunks
  - rarest first

## Sending Chunks: tit-for-tat

- Alice sends chunks to four neighbors currently sending her chunks *at the highest rate*
  - re-evaluate top 4 every 10 sec
- every 30 secs: randomly select another peer, starts sending chunks
  - "optimistically unchoke"
  - The newly chosen peer may become one of Alice's top 4

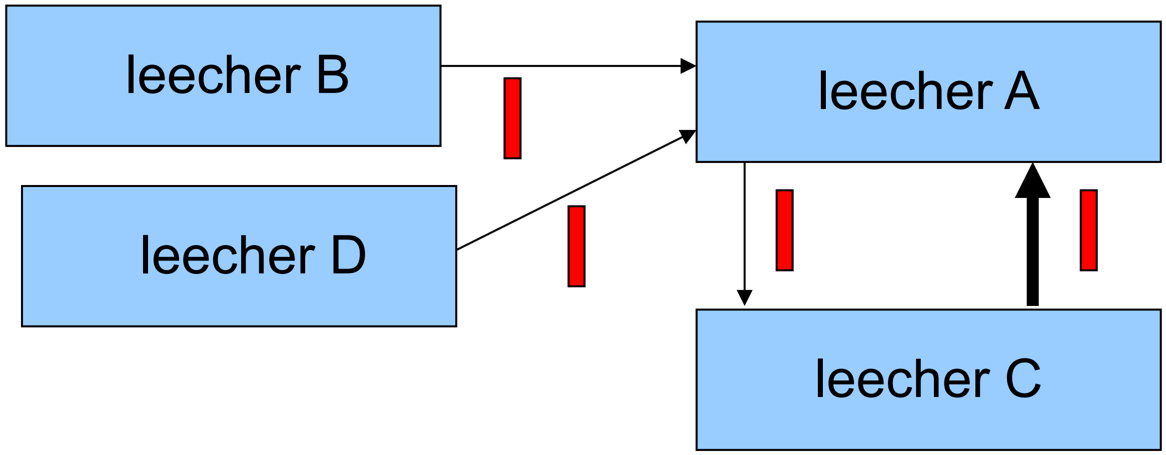
# BitTorrent – downloading data




After A learned peers B & C's chunk list:

- download the *rarest* missing piece first
- download data *blocks* in parallel
- verify the *piece* using hashes in the metainfo file
- advertise received pieces to the entire peer set

# BitTorrent – incentives mechanism




- all leechers periodically calculate download rates from others, and only upload to the fastest (*regular unchoke)*
  - allows new leechers to get their first data pieces
- *optimistic unchoke:* periodically select a peer at random and send it data

# BitTorrent Security

- Let's assume you have the "right" .torrent file

- Can peers modify parts of the downloaded file(s)?
- Can you be tracked?

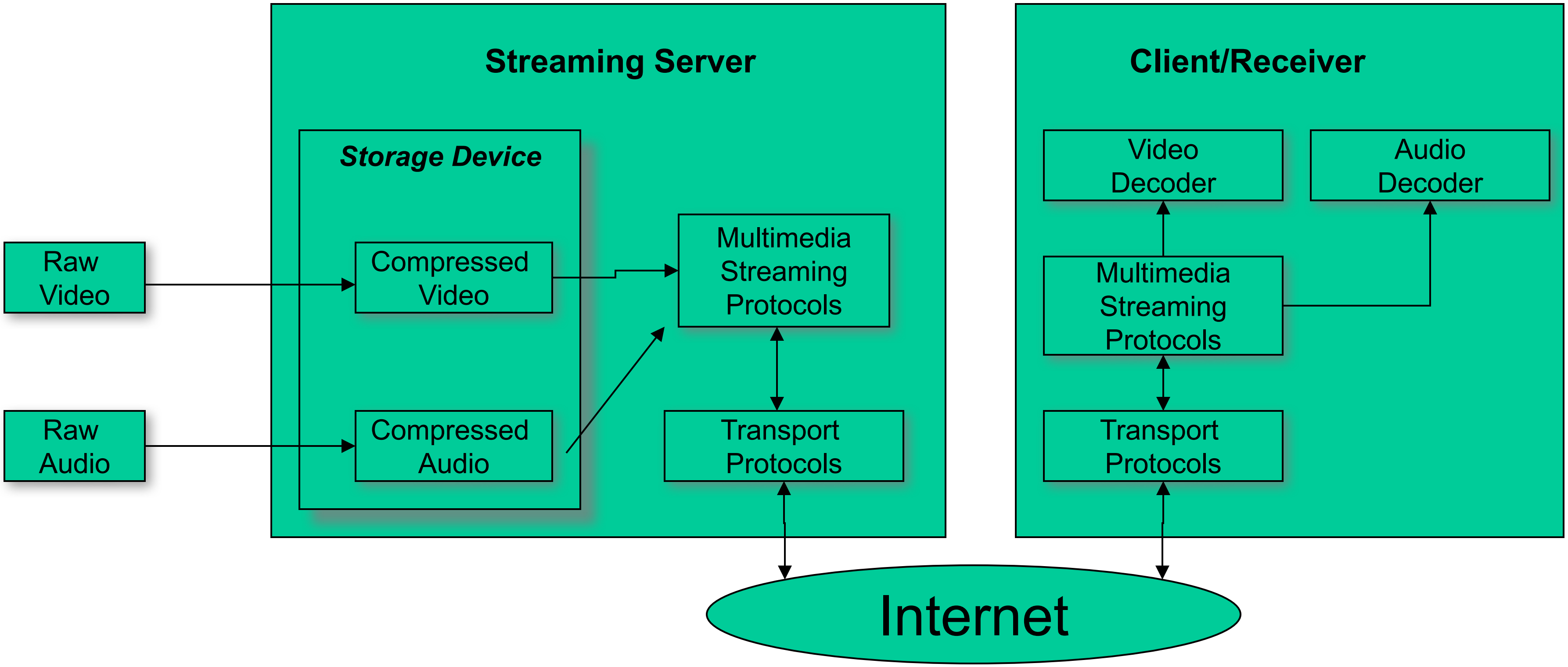
# Internet Video

# What is Multimedia Streaming?

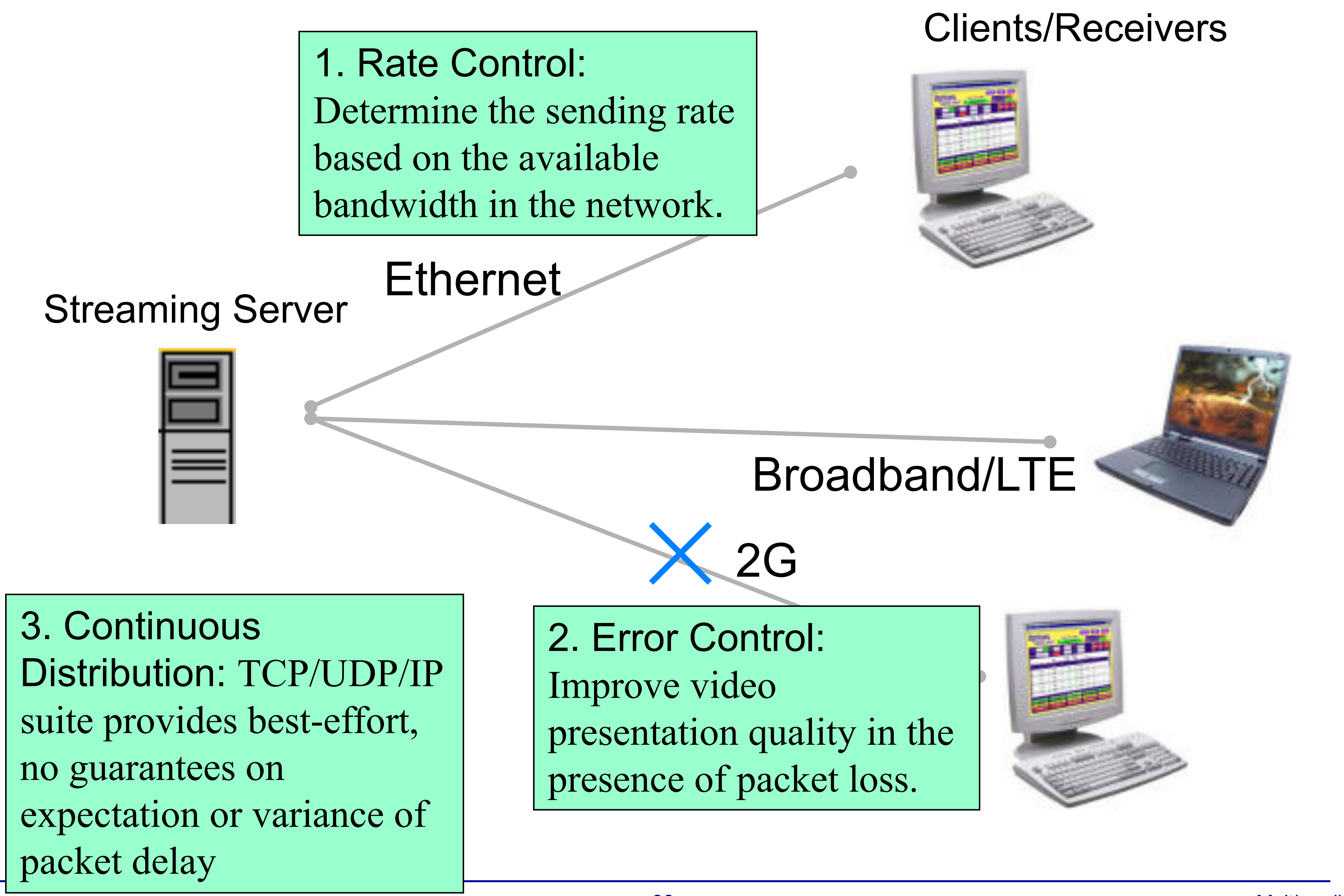
- Multimedia Streaming: Clients request audio/video files from servers and pipeline reception over the network and display

**User's perspective:**

- Quick start without waiting for full download.
- Coming continuously without interruption.
- VCR operation (pause, resume, fast forward, rewind, etc.)



# Challenges in Media Streaming Protocols



# Techniques in Multimedia Streaming Protocols (1)

- Rate Control
  - Scalable compression
    - Base substream and enhancement substreams.
    - SNR scalability / spatial scalability / temporal scalability
  - Rate filter
    - Frequency filter
    - Frame-dropping filter
    - Re-quantization filter
  - QoS Feedback, e.g. RTCP.

# Techniques in Multimedia Streaming Protocols (2)

- Error Control
  - Add redundant data in coding
    - MDC, (Multiple Description Coding)
    - FEC (Forward Error Coding)
  - Receiver End Error Concealment
    - Receiver conceal data loss.
    - Spatial interpolation, used in intra-coded frame.
    - Temporal interpolation, used in inter-coded frame.

# A General View of the Classic Intenet Multimedia Streaming Protocols

- Stream description SDP, SMIL...

  Describe the session and content

- Stream control RTSP

  Remote control the session

- Media transport RTP

  Error control and flow control

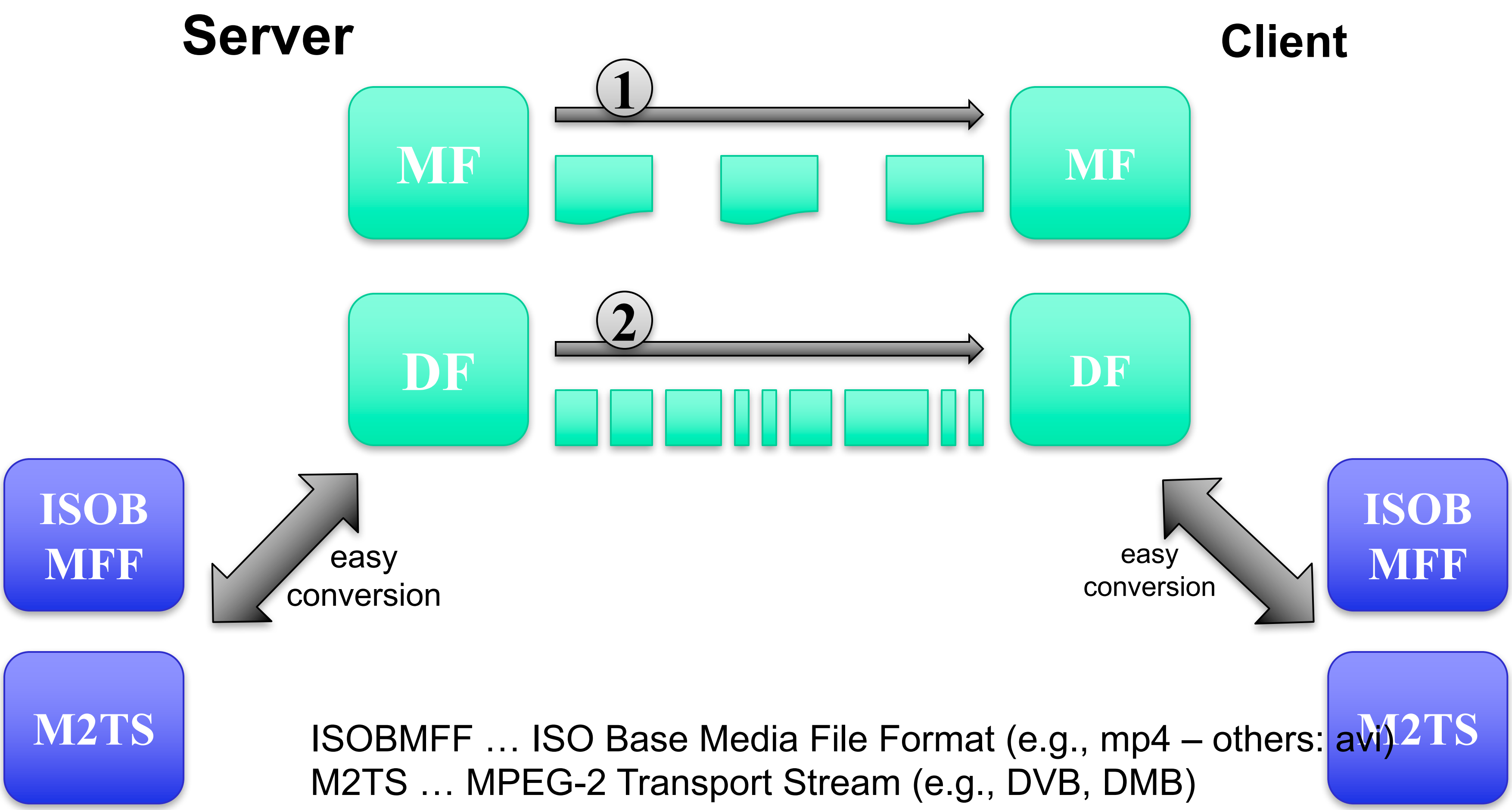
- Resource reservation (if any!): RSVP, DiffServ

  provide QoS for media streaming packets

# HTTP-Based Internet Video

Use HTTP to scale video distribution

# HTTP Streaming of Media




ISOBMFF … ISO Base Media File Format (e.g., mp4 – others: avi)
M2TS … MPEG-2 Transport Stream (e.g., DVB, DMB)
MF … Manifest Format (e.g., MPD, FMF)
DF … Delivery Format (e.g., F4F, 3gs)

# Adaptive Streaming in Practice

move networks, inc.

HTTP Live Streaming

Microsoft Silverlight

Smooth Streaming

Adobe HTTP Dynamic Streaming

Combined A/V streams only

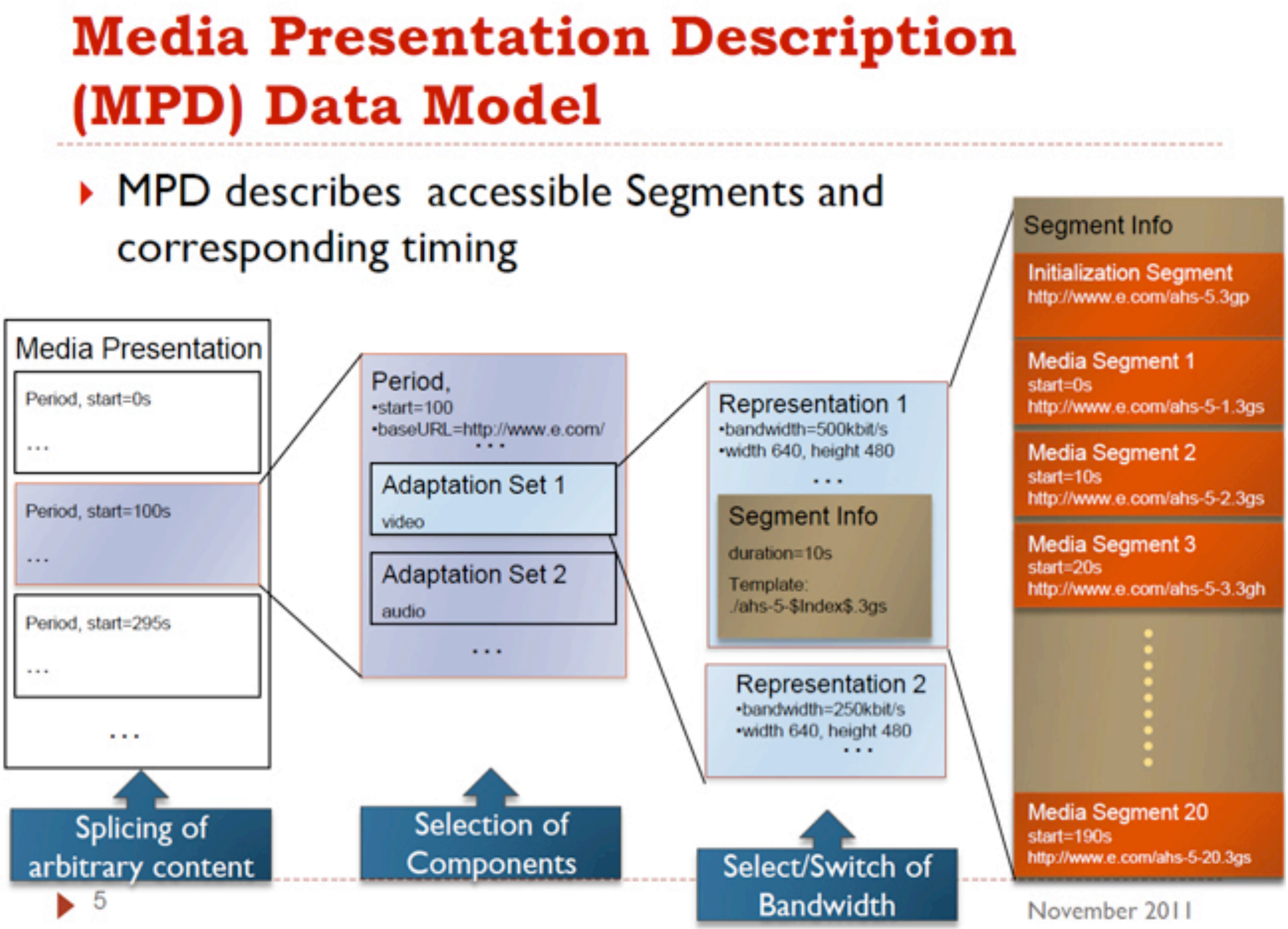
NETFLIX

MPEG DASH

Separate Audio/Video

Ack & ©: Mark Watson 2010/05/02

# MPEG DASH Data Model



# Media Presentation Description

- Redundant information of Media Streams for the purpose to initially select or reject Groups or Representations
  - Examples: Codec, DRM, language, resolution, bandwidth
- Access and Timing Information
  - HTTP-URL(s) and byte range for each accessible Segment
  - Earliest next update of the MPD on the server
  - Segment availability start and end time in wall-clock time
  - Approximated media start time and duration of a Media Segment in the media presentation timeline
  - For live service, instructions on starting playout such that media segments will be available in time for smooth playout in the future
- Switching and splicing relationships across Representations
- Relatively little other information