

Computer Networks

Lecture 05

Application Layer

By
Dr. Khaled ELMenshawy

Lecture

05

LEARNING OUTCOMES

- Define network applications.
- Understand the difference between client-server and P2P paradigm
- Understand file distribution: Client-Server vs P2P.
- Understand the types of Hyper Text Transfer Protocol (HTTP) and HTTP message format.
- Apply non-persistent and persistent HTTP time calculations.
- Understand cookies and web cache.
- Define email and understand POP3 and SMTP interaction.
- Understand how DNS works and recognize DNS records, query types, cache, and DNS message components.
- Understand video Streaming and Content Distribution Networks (CDN).

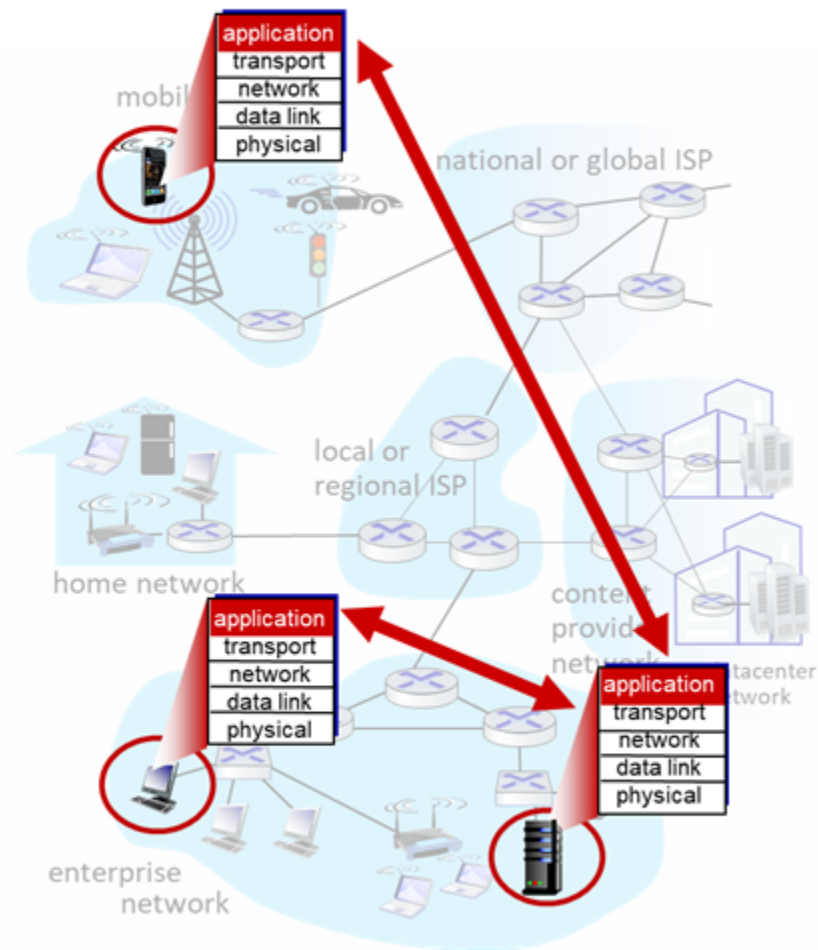
What are network applications?

Computer network applications are network software applications that utilize the Internet or other network hardware infrastructure to perform useful functions.



Characteristics of network applications:

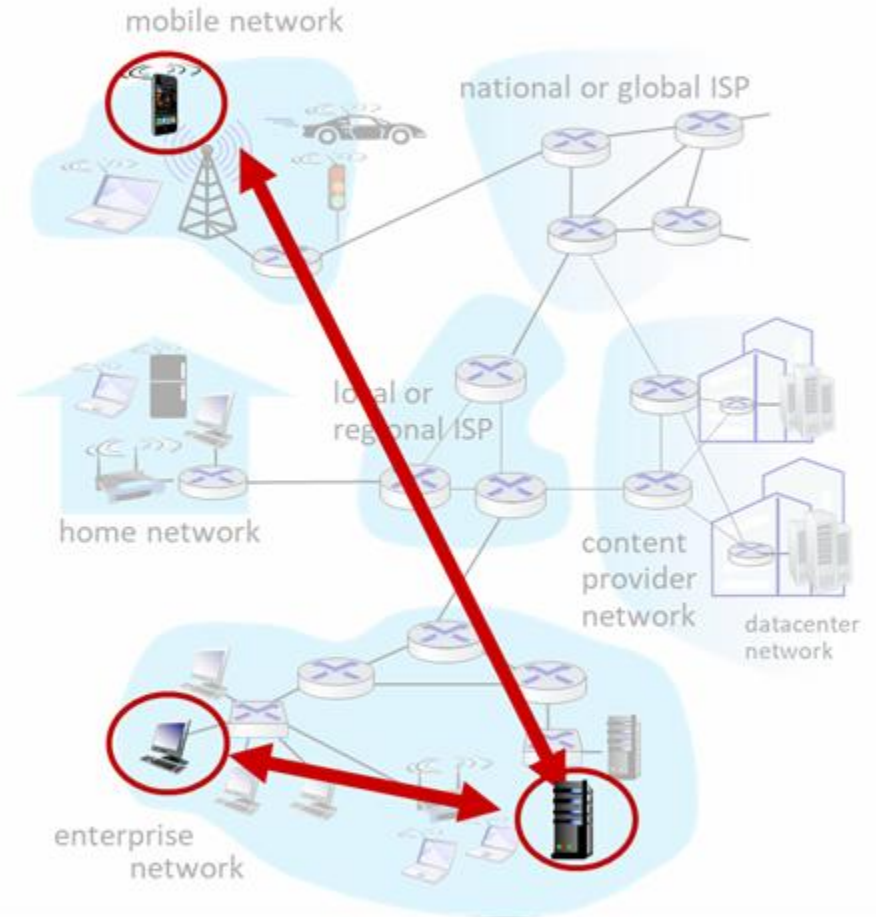
- Run on (different) end systems.
- Communicate over the network.
- Network-core devices do not run user applications .
- Applications on end systems allow for rapid app development, and propagation.



Client-server paradigm

Server:

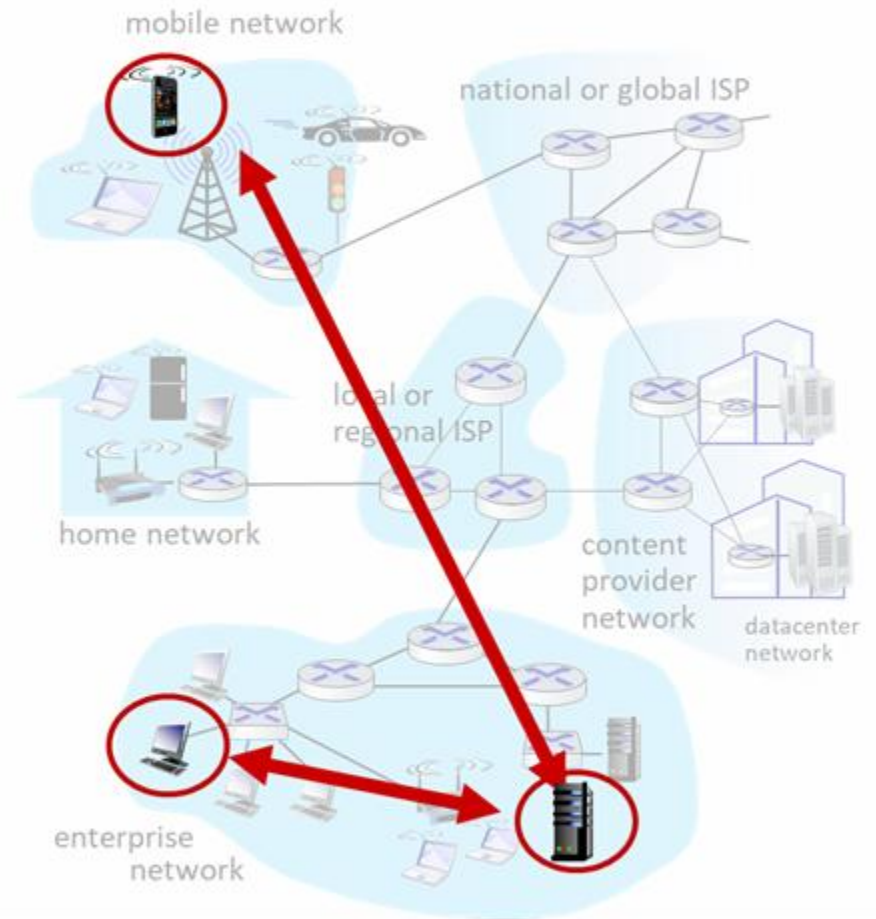
- The server is a high-performance computer that runs server operating systems and other network applications which offer services to client computers such as mail, file, database, web applications, and supporting remote applications besides other protocols such as DNS and DHCP.
- Servers run 24 hours/7 days, and their NICs hold a permanent IP address (usually a static IP address).



Client-server paradigm

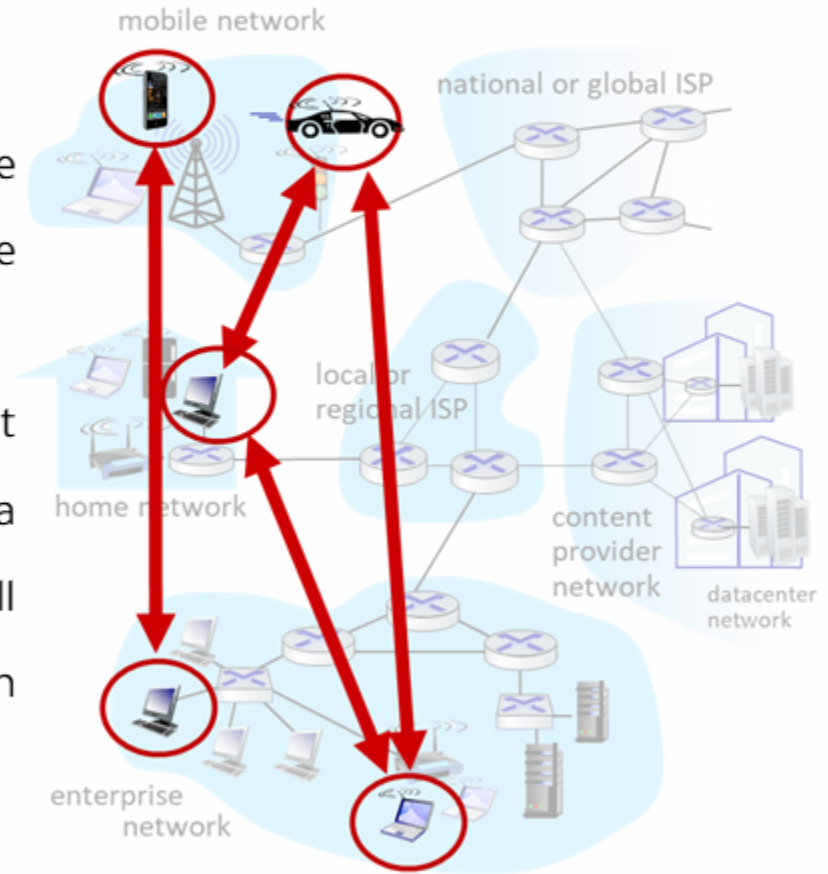
Client:

- Clients may be PCs, smartphones, tablets, or even smart TV.
- They are connected to the server and use the different services offered by the server.
- They may assign a dynamic IP address and do not communicate directly with each other but through the server.



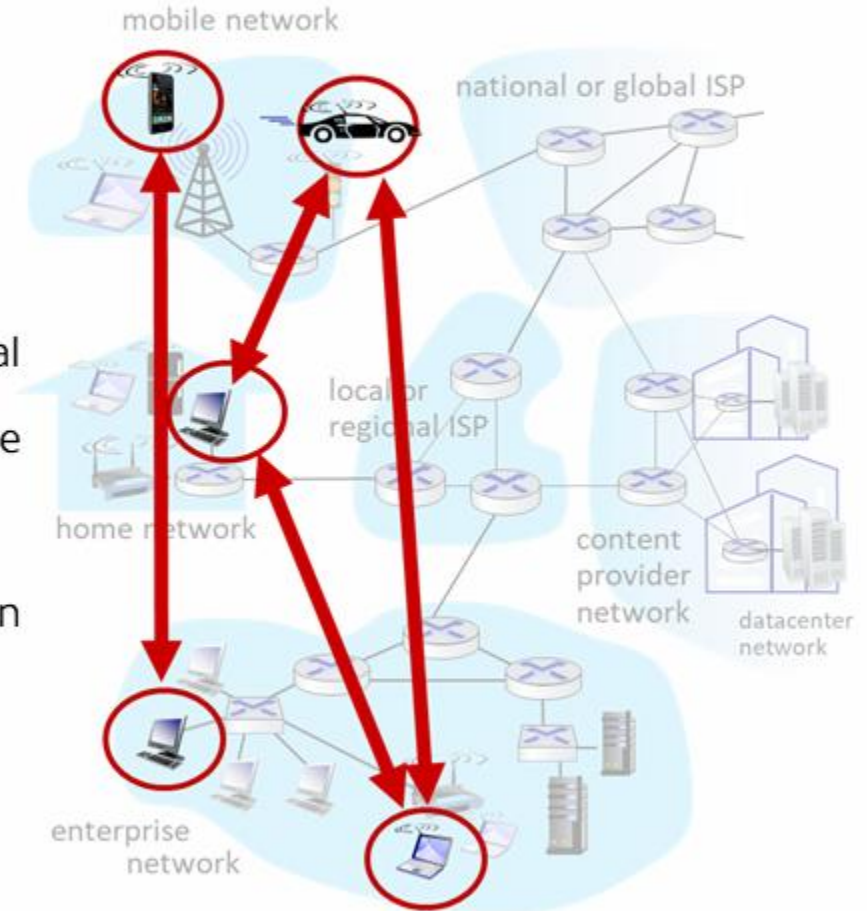
P2P paradigm

- In a peer-to-peer or P2P network, the tasks are allocated among all the members of the network. There is no real hierarchy among the computers, and all of them are considered equal.
- The peer-to-peer network does not use a central computer server that controls network activity. Instead, every computer on the network has a special software running that allows for communications between all the computers, each computer may act as both client and server, with the same responsibilities and status.



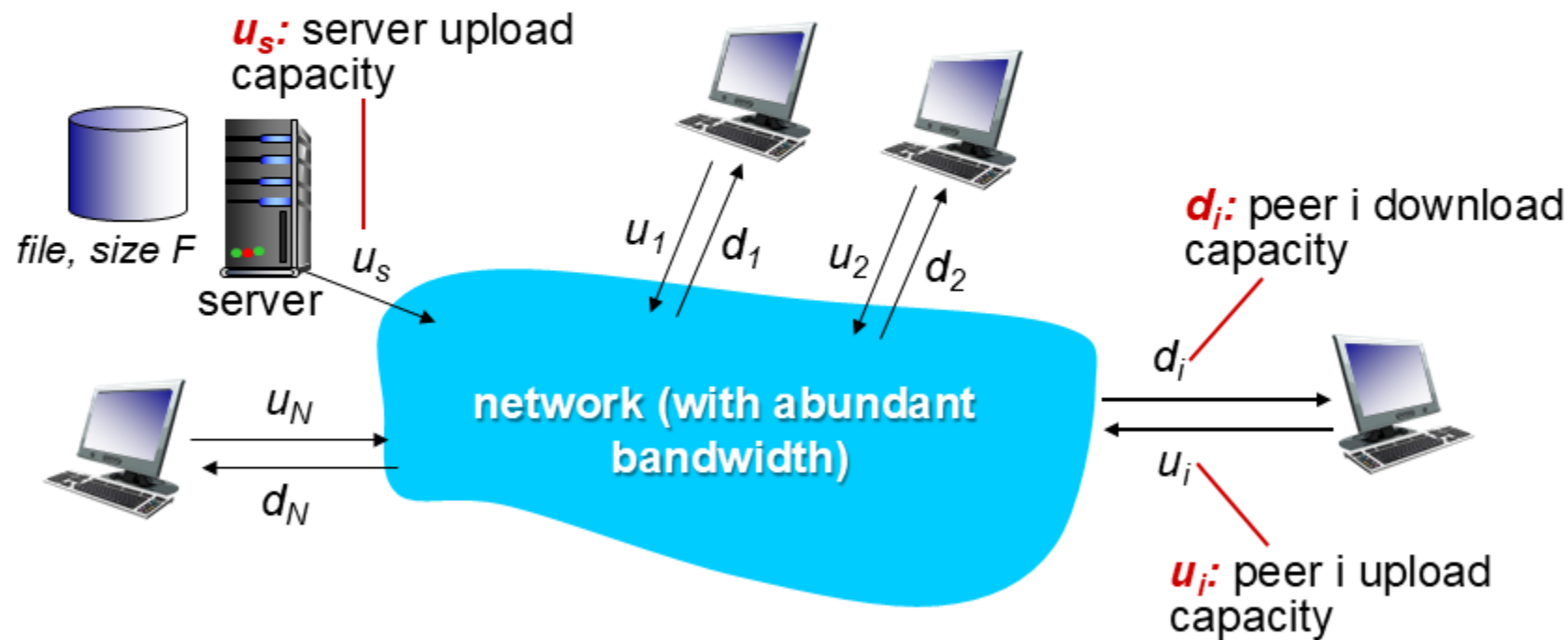
P2P paradigm

- Peer-to-peer is mostly used for file sharing.
- A peer-to-peer network is robust in the sense that if one or several of the individual computers stop working for some reason, the network continues to function.
- the quality of the network depends completely on the contribution of individual participants.



File distribution: Client-Server vs P2P

How much time to distribute file (size F) from one server to N peers?



File distribution: Client-Server

Server transmission:

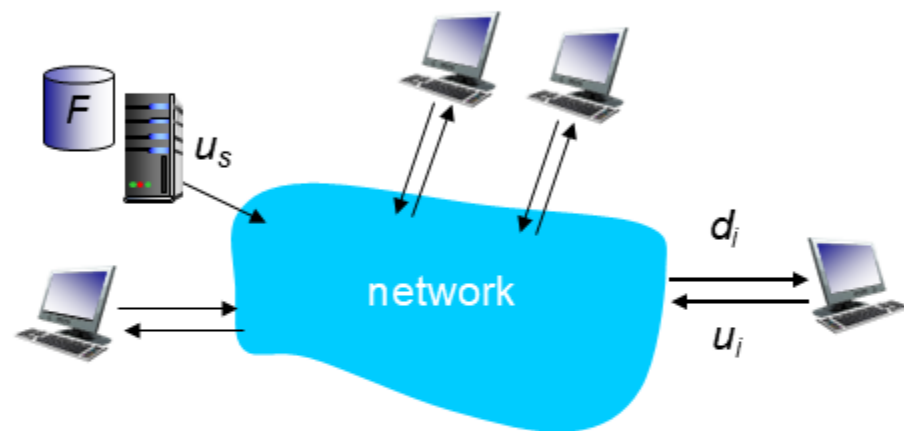
Sequentially send (upload) N file copies:

- Time to send one copy: F/u_s
- Time to send N copies: NF/u_s

Client:

each client must download file copy

- $d_{\min}$ = Minimum client download rate.
- Minimum client download time: $F/d_{\min}$



*Time to distribute F
to N clients using
client-server approach*

$$D_{c-s} \geq \max\{NF/u_s, F/d_{\min}\}$$

Increases linearly in N

File distribution: P2P**Server transmission:**

Must upload at least one copy

➤ Time to send one copy: F/u_s

Client:

Each client must download file copy

➤ Minimum client download time: $F/d_{\min}$

Clients:

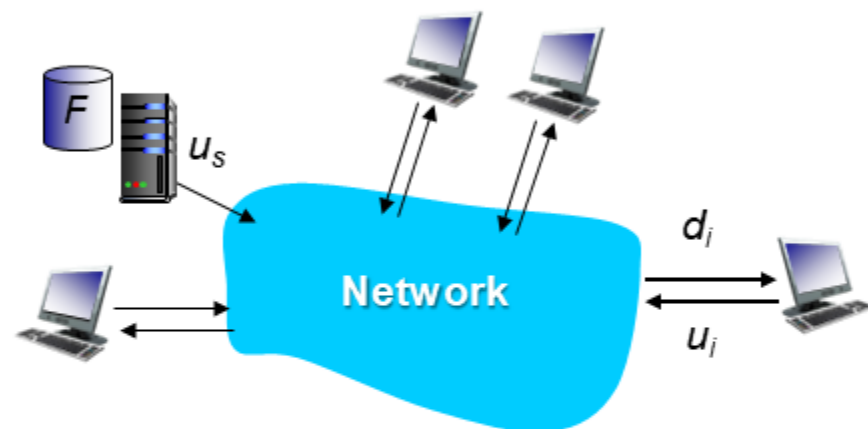
As aggregate must download NF bits

➤ Maximum upload rate (limiting max download rate) is $u_s + \sum u_i$

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

$$D_{P2P} \geq \max\{F/u_s, F/d_{\min}, NF/(u_s + \sum u_i)\}$$

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



Increases linearly in N ...

Example

The problem is to distribute a file of size $F = 9$ Gbits to each of these 9 peers. Suppose the server has an upload rate of $u = 64$ Mbps.

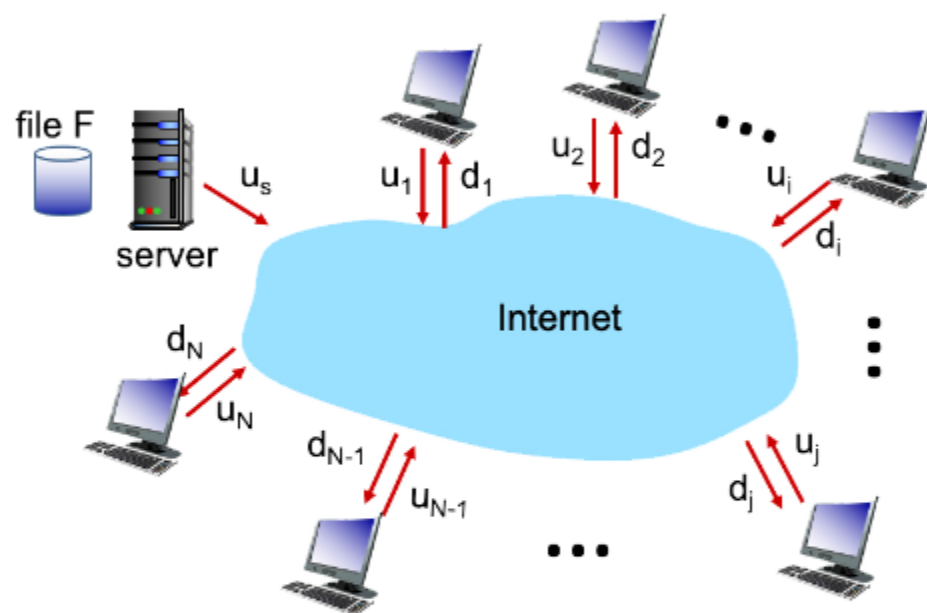
The 9 peers have upload rates of:

$u_1 = 19$ Mbps, $u_2 = 12$ Mbps, $u_3 = 17$ Mbps, $u_4 = 18$ Mbps, $u_5 = 29$ Mbps, $u_6 = 16$ Mbps, $u_7 = 26$ Mbps, $u_8 = 20$ Mbps, and $u_9 = 28$ Mbps

The 9 peers have download rates of:

$d_1 = 10$ Mbps, $d_2 = 38$ Mbps, $d_3 = 37$ Mbps, $d_4 = 36$ Mbps, $d_5 = 40$ Mbps, $d_6 = 36$ Mbps, $d_7 = 21$ Mbps, $d_8 = 10$ Mbps, and $d_9 = 15$ Mbps

What is the minimum time needed to distribute this file from the central server to the 9 peers using the client-server model?



$$D_{c-s} \geq \max\{NF/u_s, F/d_{\min}\}$$

$$D_{c-s} \geq \max\{9 \times 9 \times 10^9 / 64 \times 10^6, 9 \times 10^9 / 10 \times 10^6\} = \max\{1265.6 \text{ Sec.}, 900 \text{ Sec.}\}$$

Example

The problem is to distribute a file of size $F = 9$ Gbits to each of these 9 peers. Suppose the server has an upload rate of $u = 64$ Mbps.

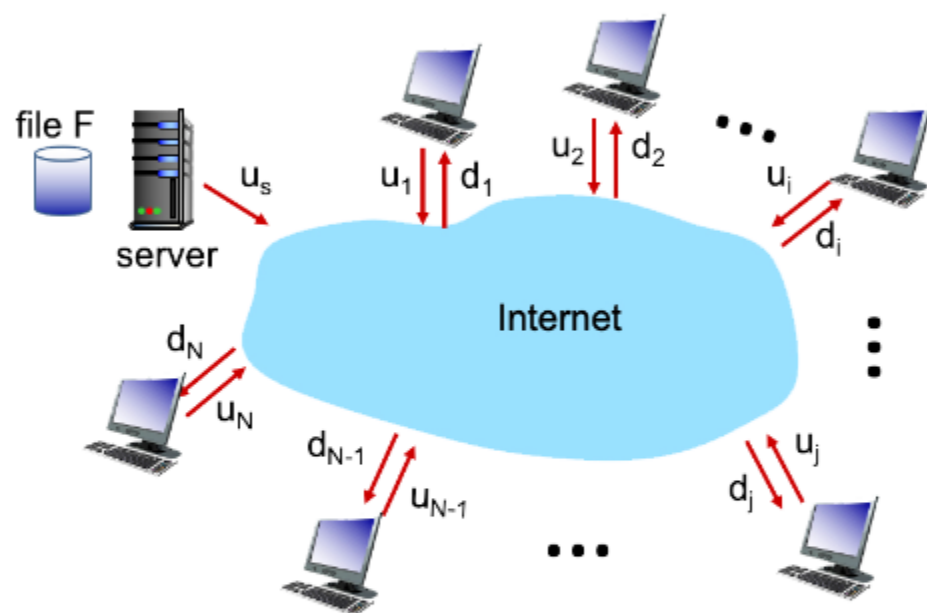
The 9 peers have upload rates of:

$u_1 = 19$ Mbps, $u_2 = 12$ Mbps, $u_3 = 17$ Mbps, $u_4 = 18$ Mbps, $u_5 = 29$ Mbps, $u_6 = 16$ Mbps, $u_7 = 26$ Mbps, $u_8 = 20$ Mbps, and $u_9 = 28$ Mbps

The 9 peers have download rates of:

$d_1 = 10$ Mbps, $d_2 = 38$ Mbps, $d_3 = 37$ Mbps, $d_4 = 36$ Mbps, $d_5 = 40$ Mbps, $d_6 = 36$ Mbps, $d_7 = 21$ Mbps, $d_8 = 10$ Mbps, and $d_9 = 15$ Mbps

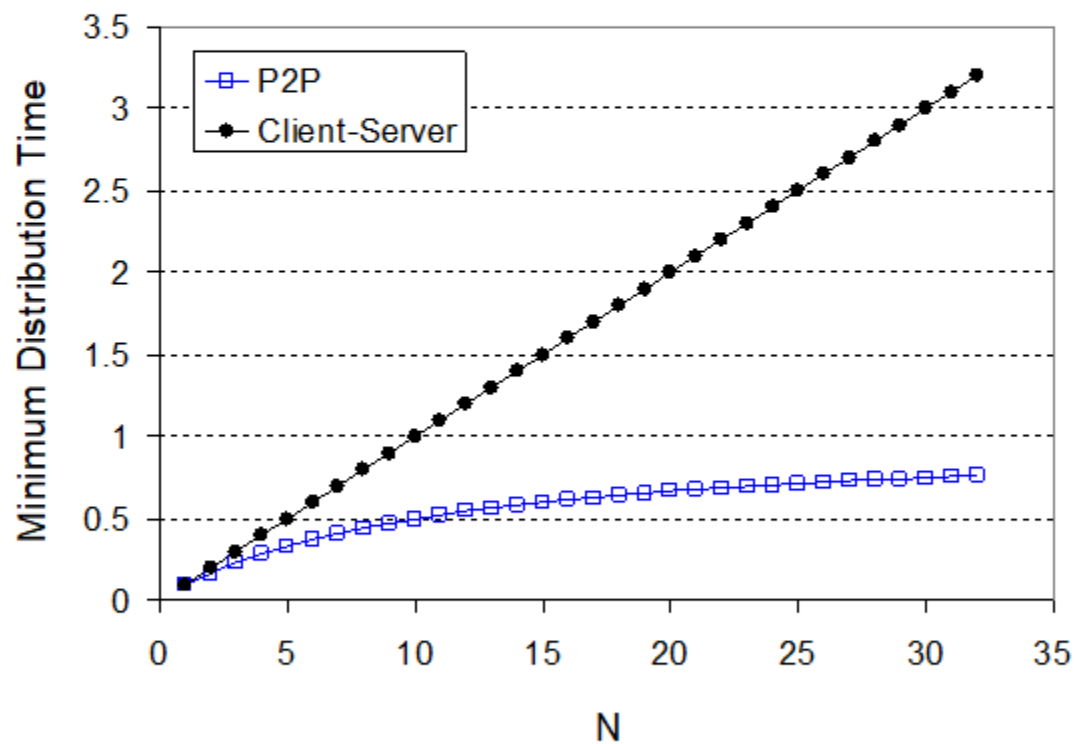
What is the minimum time needed to distribute this file from the central server to the 9 peers using the peer-to-peer model?



$$D_{p2p} \geq \max\{F/u_s, F/d_{\min}, NF/(u_s + \sum u_i)\} = D_{p2p} \geq \max\{9 \times 10^9 / 64 \times 10^6, 9 \times 10^9 / 10 \times 10^6, 9 \times 9 \times 10^9 / 64 \times 10^6 + 185 \times 10^6\}$$

$$= \max\{140.6 \text{ Sec.}, 900 \text{ Sec.}, 325 \text{ Sec.}\}$$

File distribution: Client-Server vs P2P

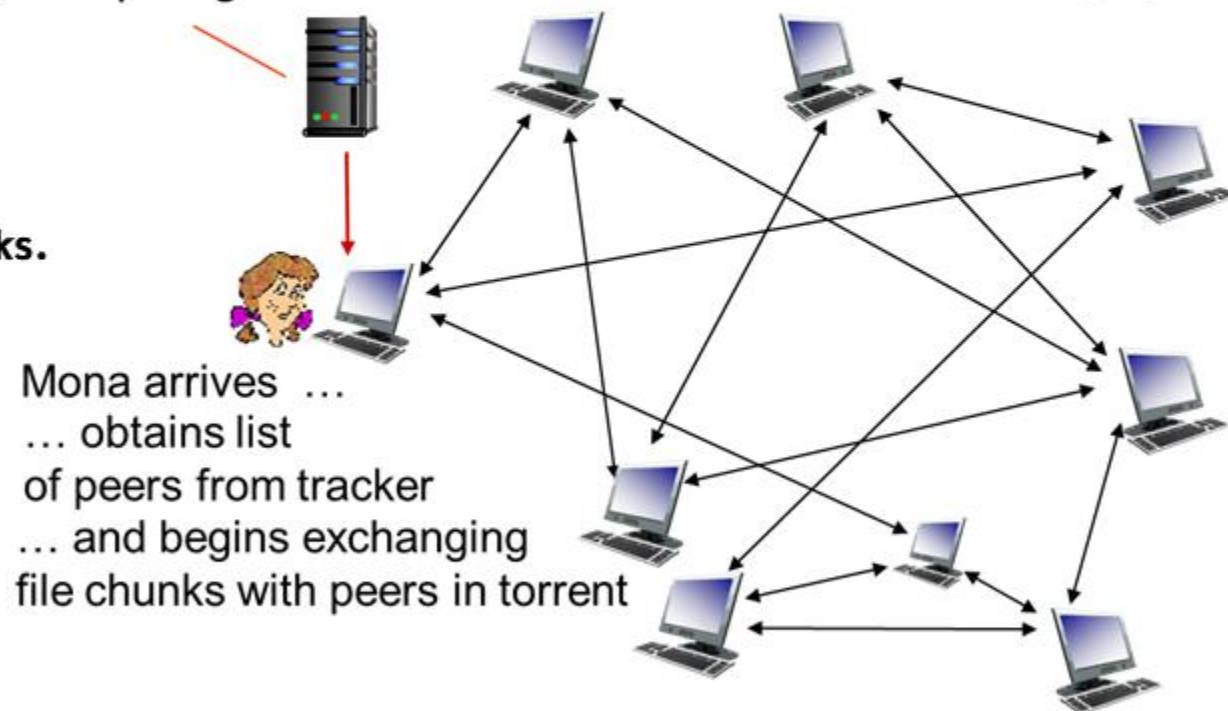


P2P file distribution: BitTorrent

Tracker: Tracks peers participating in torrent

Torrent: Group of peers exchanging chunks of a file

- File divided into 256KB chunks.
- Peers in torrent send/receive file chunks.



BitTorrent: requesting, sending file chunks

Requesting chunks:

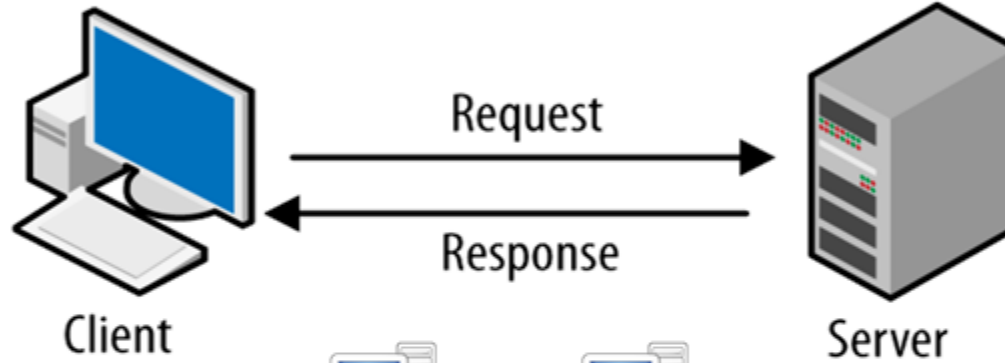
- At any given time, different peers have different subsets of file chunks.
- periodically, Mona asks each peer for list of chunks that they have.
- Mona requests missing chunks from peers, rarest first.

Sending chunks: tit-for-tat:

- Mona sends chunks to those four peers currently sending her chunks at highest rate:
 - Other peers are choked by Mona (do not receive chunks from her)
 - Re-evaluate top 4 every 10 secs
- Every 30 secs: randomly select another peer, starts sending chunks
 - "Optimistically unchoke" this peer
 - Newly chosen peer may join top 4

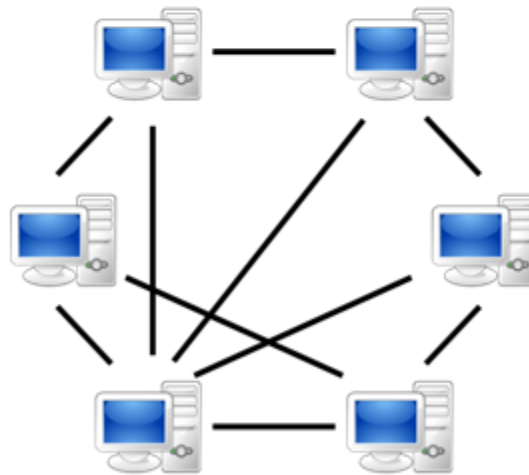
Processes communicating

A process that initiates communication



A process that waits to be contacted.

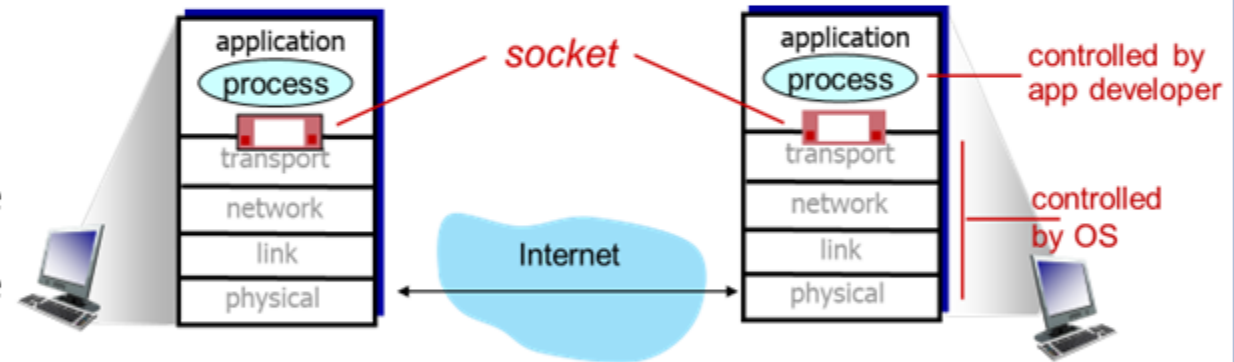
Applications with P2P architectures have client processes & server processes.



Processes communicating

The Application layer sends/receives messages to/from its socket which is analogous to a door

- Sending process shoves message outdoor.
- Sending process relies on transport infrastructure on another side of the door to deliver the message to the socket at receiving process
- two sockets involved: one on each side



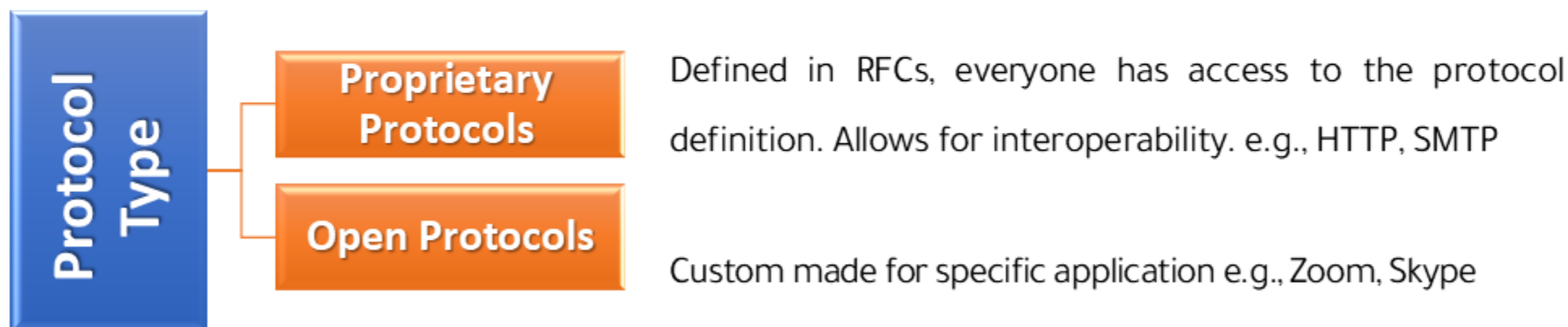
An application-layer protocol defines:

Types of messages exchanged. (e.g., request, response)

Message syntax; (what fields in messages & how fields are delineated)

Message semantics; the meaning of information in fields.

Rules; for when and how processes send & respond to messages.



What transport service does an application need?

Data integrity and security; some applications (e.g., file transfer, web transactions) require 100% reliable data transfer while other applications (e.g., audio) can tolerate some loss. Also, some applications require encryption (e.g., money transactions, personal accounts).

Timing; some apps (e.g., Internet telephony, interactive games) require a low delay to be “effective”.

Throughput; some apps (e.g., multimedia) require a minimum amount of throughput to be “effective” while other apps (“elastic apps”) make use of whatever throughput they get.

Application	Data loss	Throughput	Time-sensitive
File transfer/download	No loss	elastic	no
email	No loss	elastic	no
Web documents	No loss	elastic	no
Real-time audio/video	loss-tolerant	audio: 5Kbps-1Mbps video:10Kbps-5Mbps	yes, 10's msec
Streaming audio/video	loss-tolerant		
Interactive games	loss-tolerant	3-6Mbps	Depends on the game
Text messaging	no loss	elastic	yes, a few secs

What is HTTP?

- HTTP stands for Hyper Text Transfer Protocol.
- It is a protocol used to [access the data on the World Wide Web](#) (www).
- It can be used to transfer the data in the form of plain text, hypertext, audio, video, etc.
- It used in a hypertext environment where there are rapid jumps from one document to another document.
- It based on [Client/Server architecture](#) where the clients are the web browsers (Edge, Chrome, Firefox, etc.) or any applications use HTTP that can request web pages, and the servers are web servers that host web pages and respond to the clients' requests (e.g., Apache, IIS).

A web page consists of objects (or resources), each of which can be stored on different Web servers, the object can be HTML file, JPEG image, Java applet, audio file, etc. Clients can access to web pages using URL

A **Uniform Resource Locator (URL)**, is the complete, unique address of a Web page, and is a reference to a web resource that specifies its location on a computer network.

http://www.intel.com/pressroom/index.ht

ml

Protocol

Domain name

Path directory / file name



HTTP is "stateless"

- Server maintains no information about past client requests.

HTML file format

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Computer Networks</title>
  </head>
  <body>
    <h1>Welcome to computer networks course </h1>
    <h2>Course description</h2>
    <br> course introduces fundamental concepts in the design and implementation of
computer communication networks and their protocols.
    <br> <a href="https://esu.ac.ae">University link</a>
    <br>
</html>
```

Welcome to computer networks course

Course description

course introduces fundamental concepts in the design and implementation of computer communication networks and their protocols.

[University link](https://esu.ac.ae)



HTTP

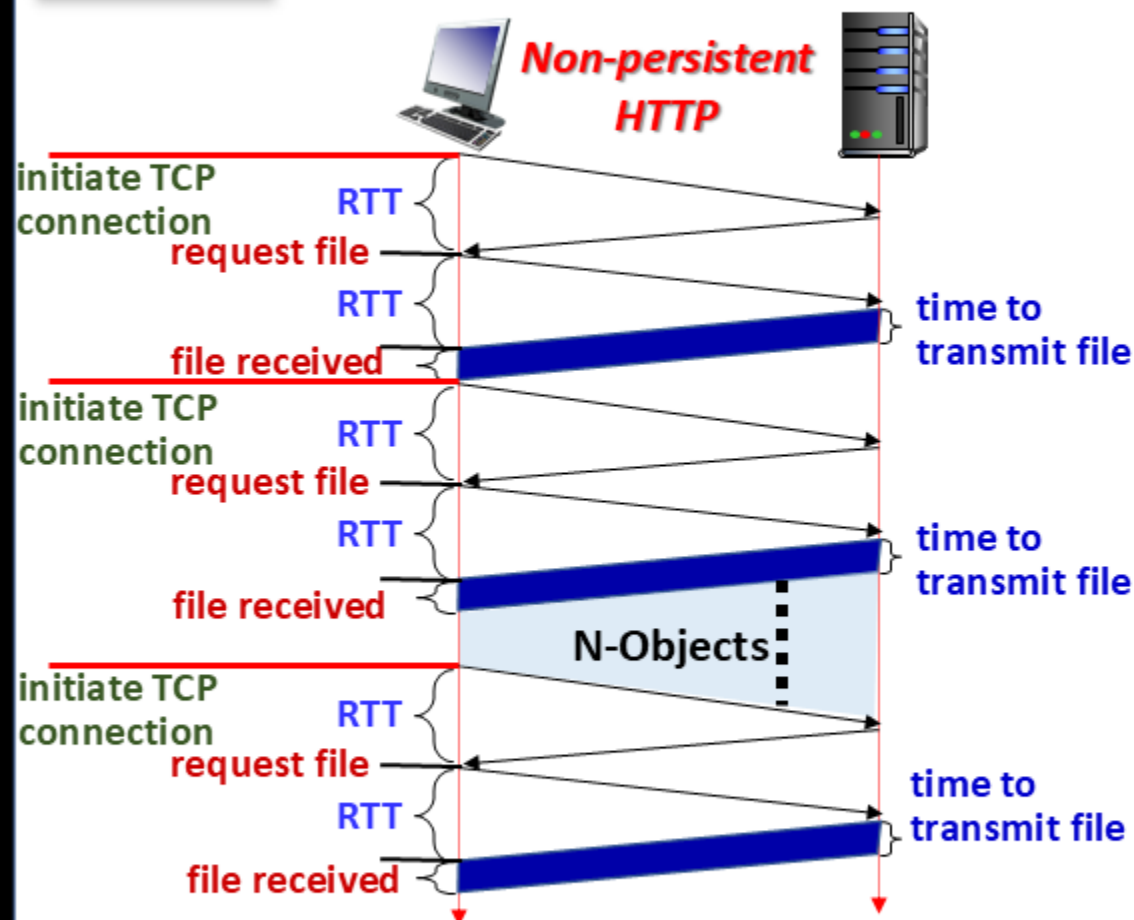
Non-persistent

1. TCP connection opened.
2. At most one object sent over TCP connection.
3. TCP connection closed.

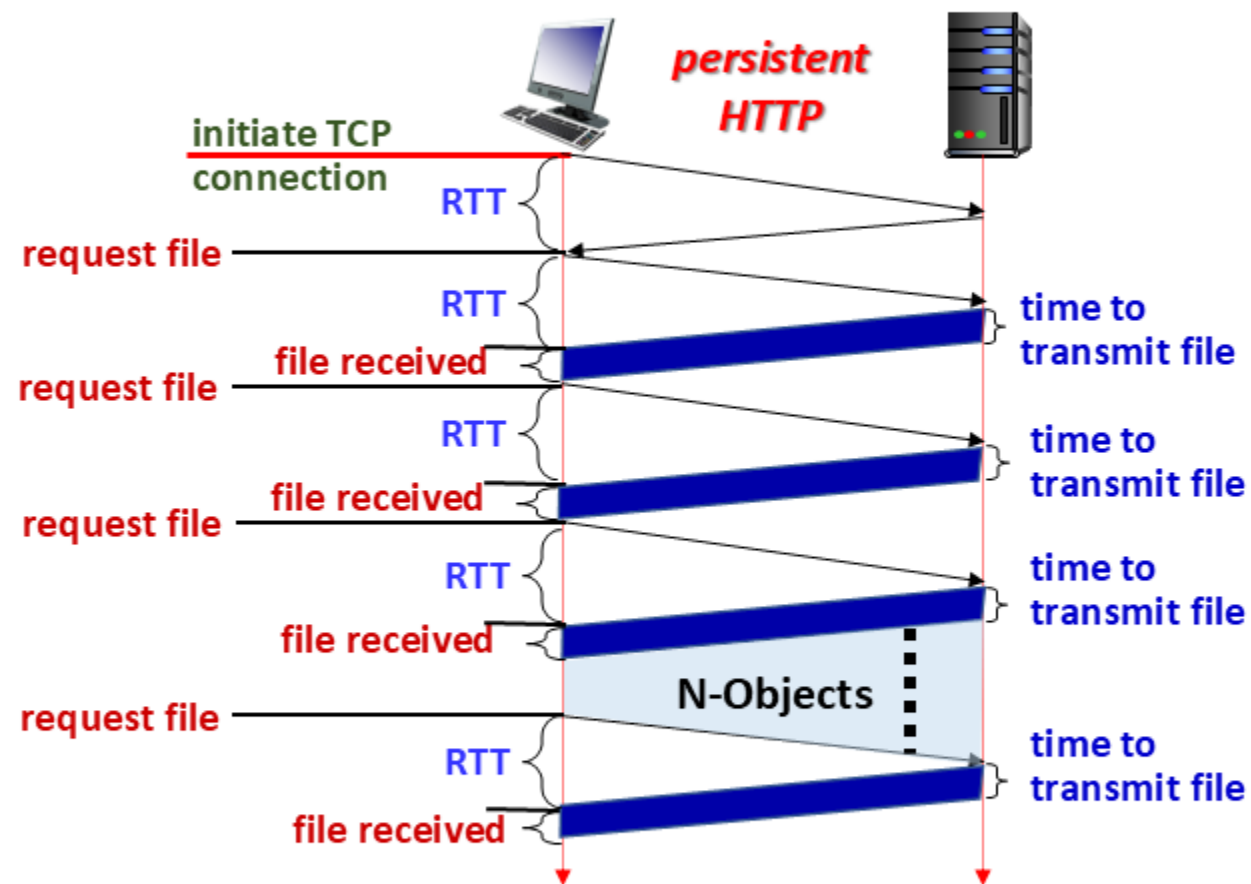
[downloading multiple objects required multiple connections](#)

Persistent

1. TCP connection opened to a server.
2. Multiple objects can be sent over single TCP connection between client, and that server.
3. TCP connection closed.



$$\text{Time} = \sum_{x=1}^N RTT + RTT + TTF_x = 2N \times RTT + \sum_{x=1}^N TTF_x$$



$$\text{Time} = RTT + \sum_{x=1}^N RTT + TTF_x = (N+1) \times RTT + \sum_{x=1}^N TTF_x$$

Example:

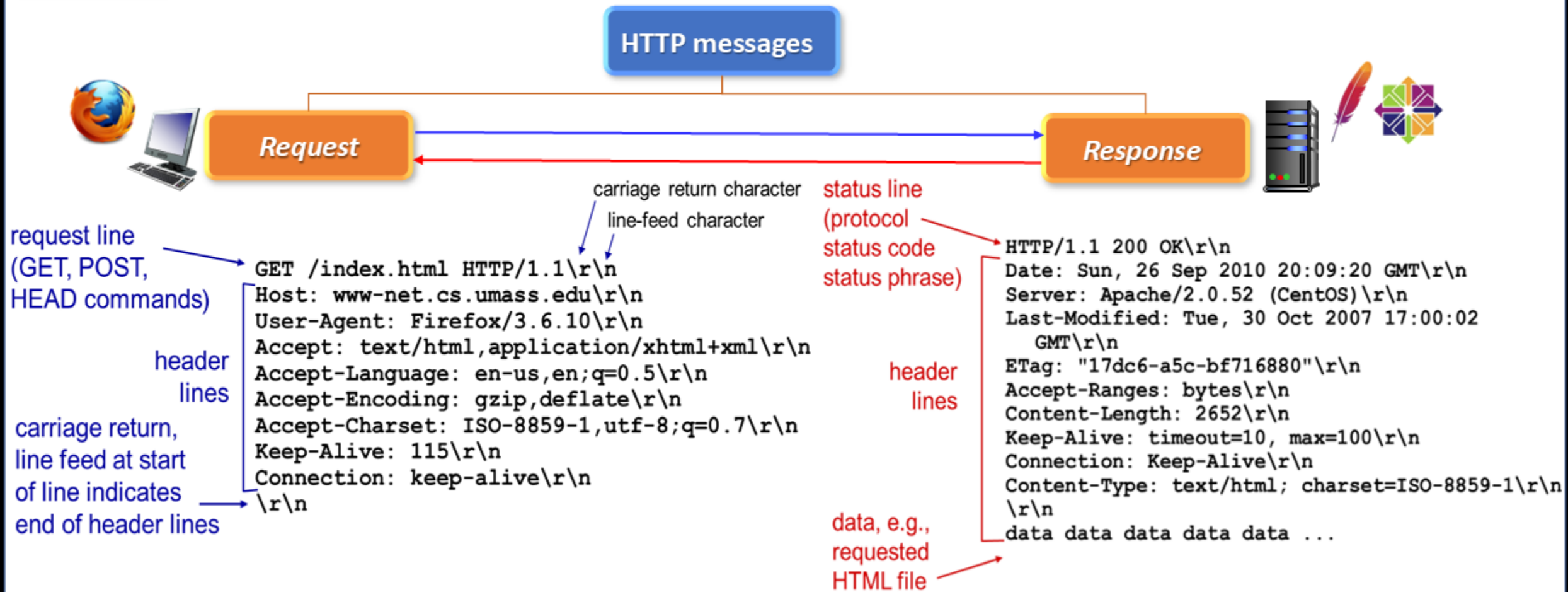
A client requests a web page from a server. The web page includes five images to download with average download time two seconds for each image, and the round-trip time is 150 ms. Calculate the Non-persistent and persistent HTTP

Non-persistent HTTP

$$\text{Time} = 2N \times RTT + \sum_{x=1}^N TTF_x = 2 \times 5 \times 0.15 + 5 \times 2 = 11.5 \text{ sec}$$

Persistent HTTP

$$\text{Time} = (N+1) \times RTT + \sum_{x=1}^N TTF_x = (5+1) \times 0.15 + 5 \times 2 = 10.9 \text{ sec}$$



Other HTTP request messages

POST method:

- web page often includes form input
- user input sent from client to server in entity body of HTTP POST request message

GET method (for sending data to server):

- include user data in URL field of HTTP GET request message (following a '?'):
`www.somesite.com/animalsearch?monkeys&banana`

HEAD method:

- requests headers (only) that would be returned *if* specified URL were requested. with an HTTP GET method.

PUT method:

- uploads new file (object) to server.
- completely replaces file that exists at specified URL with content in entity body of POST HTTP request message.

HTTP response status codes

Code	Text	Meaning
200	OK	Request succeeded, requested object later in this message
201	Created	The request succeeded, and a new resource was created as a result. This is typically the response sent after POST requests, or some PUT requests.
301	Moved Permanently	Requested object moved, new location specified later in this message (in Location: field)
400	Bad Request	Request msg not understood by server
403	Forbidden	The client does not have access rights to the content; that is, it is unauthorized
404	Not Found	Requested document not found on this server
505	HTTP Version Not Supported	The HTTP version used in the request is not supported by the server.

User-server state: cookies

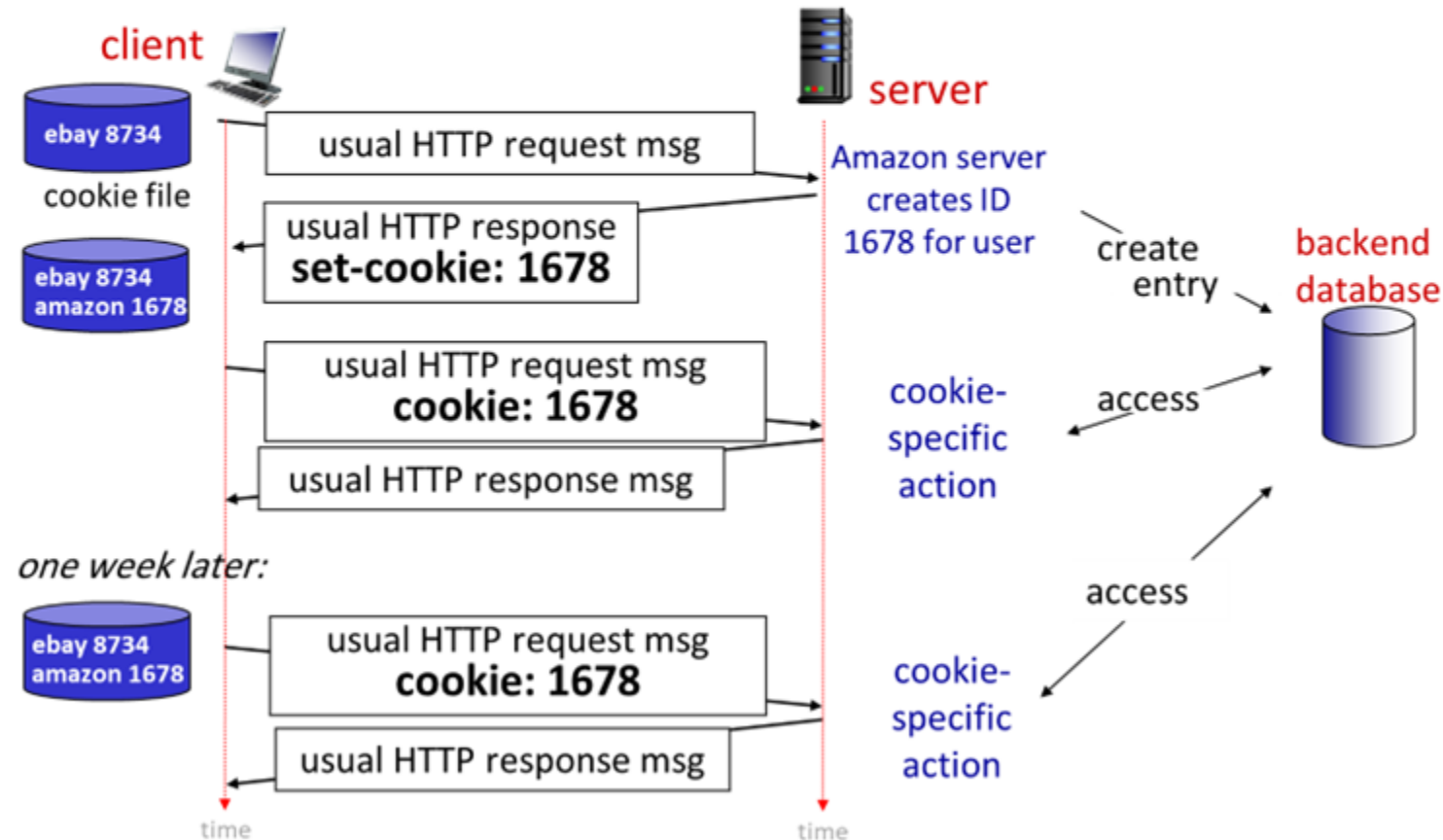
What is Cookies?

An HTTP cookie is a small piece of data that a server sends to a user's web browser. The browser may store the cookie and send it back to the same server with later requests. It offers insight into fine details such as where the user goes first when landing on the home page, what interface settings they prefer, and how long they are active on the site.

HTTP cookies has four components:

- Cookie header line of HTTP response message.
- Cookie header line in next HTTP request message.
- Cookie file kept on user's host, managed by user's browser.
- Back-end database at Web site.

User-server state: cookies



User-server state: cookies

What cookies can be used for?

- Authorization
- Shopping carts
- Recommendations
- User session state; keep track of a user session during a series of HTTP requests.

Cookies and privacy:

cookies are used to gather information about you without your authorization, they present a real threat to your online privacy.

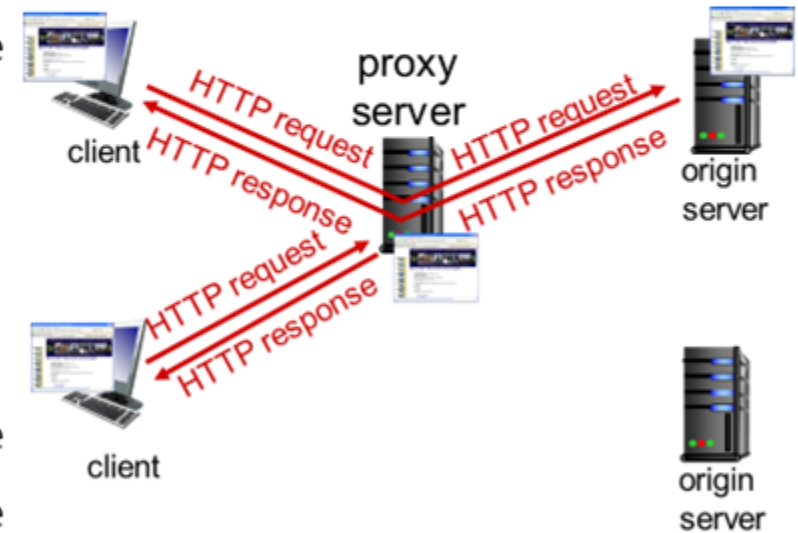
Web caches (proxy server)

Goal: satisfy client request without involving origin server.

Web caching is the activity of storing data for reuse, data is stored the first time a user visits the page and the next time a user requests the same page, a cache will serve the copy, which helps keep the origin server from getting overloaded.

A typical web cache flow looks like this:

- A user accesses a website.
- The browser sends an HTTP request to the web cache (ISP Proxy).
- If the requested object **IS** stored in the cache, the web cache responds with the object. If the requested object **IS NOT** stored in the cache, the web cache requests the object from the origin server and sends the response to the browser.



cache: specify date of cached copy in HTTP request

If-modified-since: <date>

server: response contains no object if cached copy is up-to-date:

HTTP/1.0 304 Not Modified

cache: specify date of cached copy in HTTP request

If-modified-since: <date>

server: response contains the modified object

HTTP/1.0 200 OK

client



server

HTTP request msg
If-modified-since: <date>

object
not
modified
before
<date>

HTTP response
HTTP/1.0
304 Not Modified

HTTP request msg
If-modified-since: <date>

object
modified
after
<date>

HTTP response
HTTP/1.0 200 OK
<data>

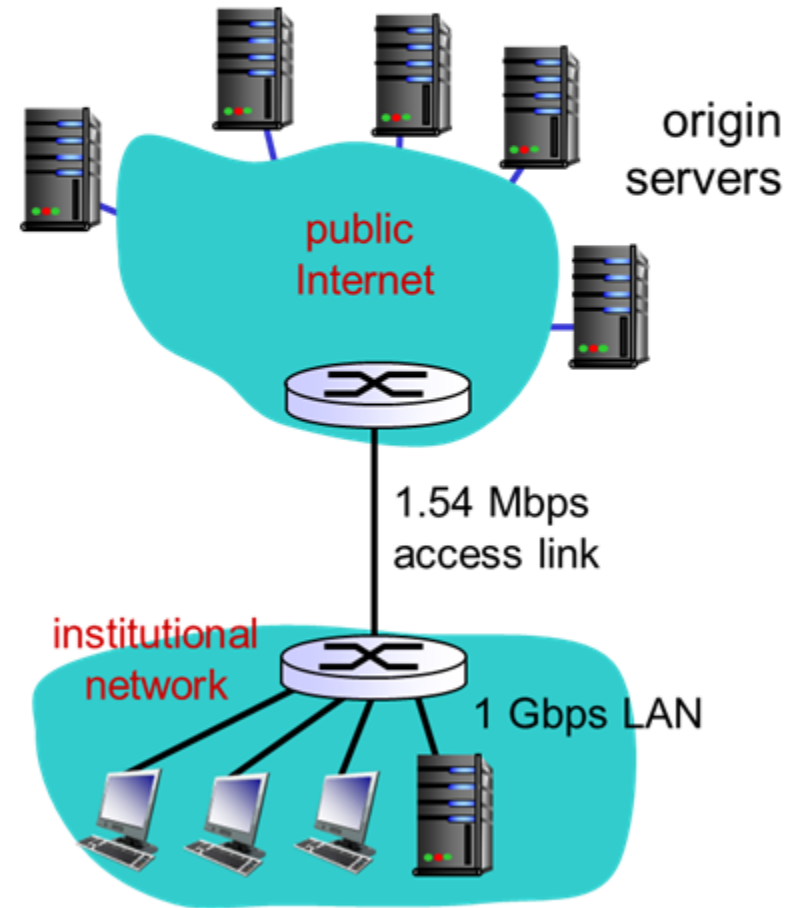
Caching Example

Assumptions:

- avg object size: 100K bits
- avg request rate from browsers to origin servers: 15/sec
- avg data rate to browsers: 1.50 Mbps
- RTT from institutional router to any origin server: 2 sec
- access link rate: 1.54 Mbps

Consequences:

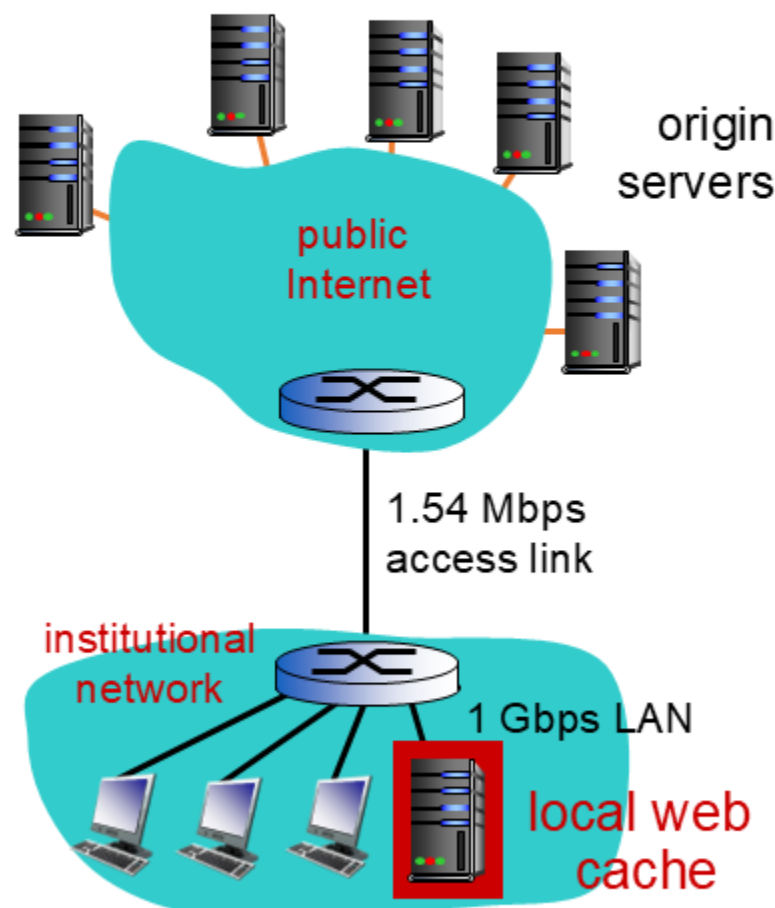
- LAN utilization: 15%
- access link utilization = 99%
- total delay = Internet delay + access delay + LAN delay
= 2 sec + minutes + μ secs



Caching Example

Calculating access link utilization, delay with cache:

- suppose cache hit rate is 0.4
40% requests satisfied at cache; 60% requests satisfied at origin
- Access link utilization:
60% of requests use access link
- Data rate to browsers over access link
 $= 0.6 * 1.50 \text{ Mbps} = 0.9 \text{ Mbps}$
utilization $= 0.9 / 1.54 = 0.58$
- Total delay
 - $= 0.6 * (\text{delay from origin servers}) + 0.4 * (\text{delay when satisfied at cache})$
 - $= 0.6 (2.01) + 0.4 (\mu\text{secs}) = \sim 1.2 \text{ secs}$



Email

User Agent

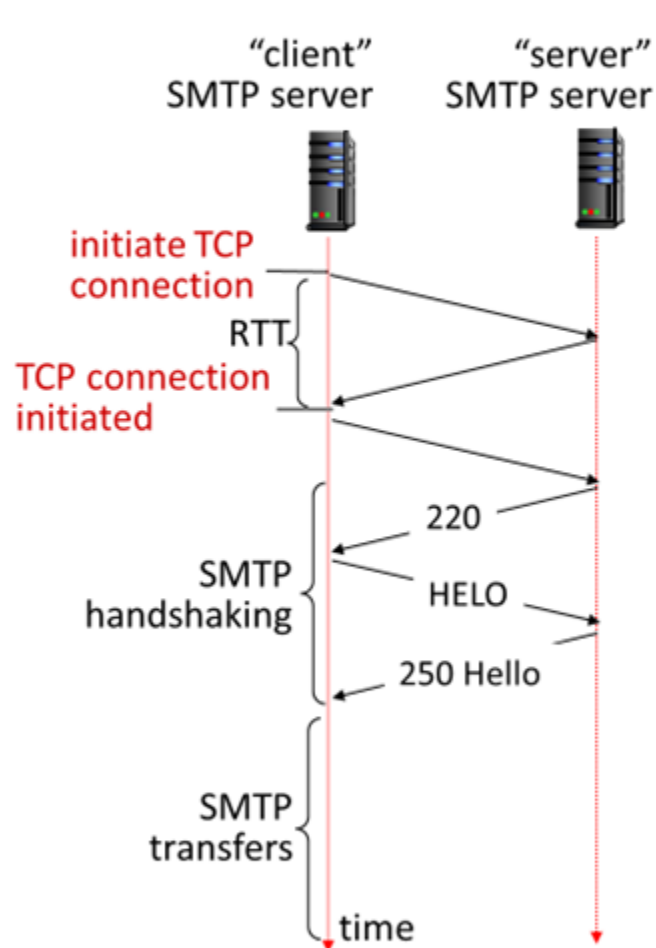
- email address [user@esu.ac.ae](#)
- composing, editing, reading mail messages applications.
e.g., Outlook, Thunderbird
- Messages must be in [7-bit ASCII](#)

Mail Server

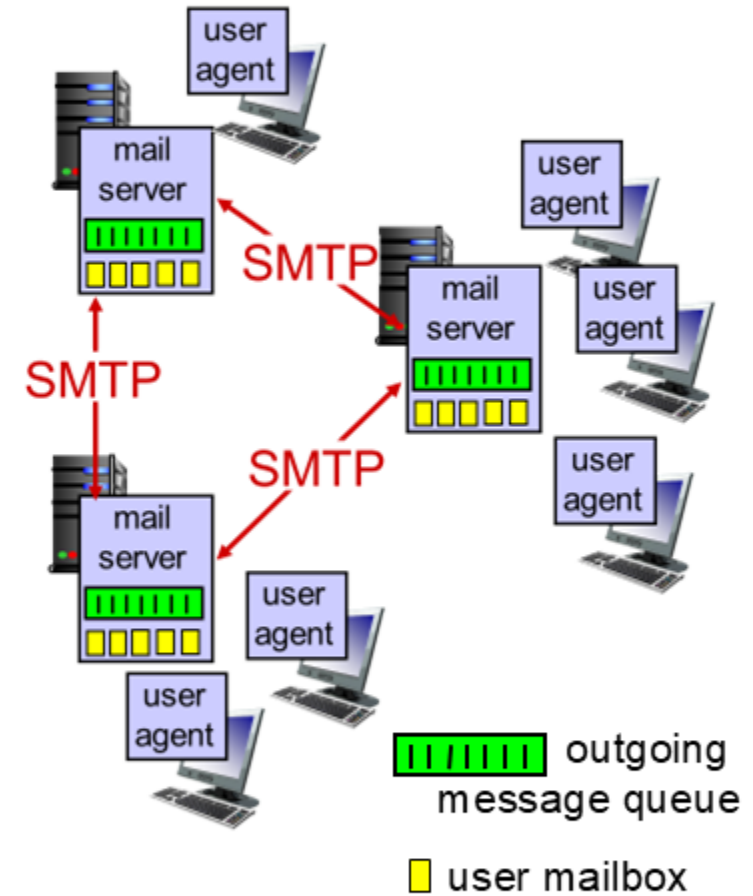
- [Web-based](#); Gmail, Yahoo, etc.
- [Client-based](#); any POP3/IMAP that runs on any mail client such as Outlook, Mail, or Thunderbird.

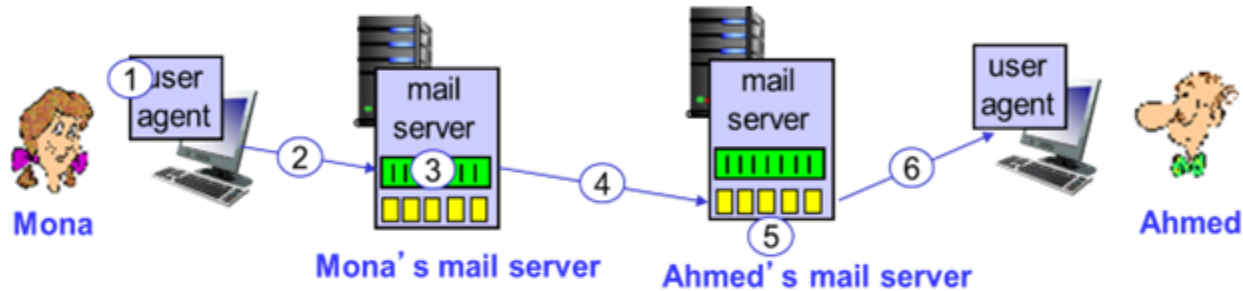
Protocols

- Simple mail transfer protocol ([SMTP](#)), Post Office Protocol v3.0 ([POP3](#)), and Internet Mail Access Protocol ([IMAP](#))
- [HTTPS](#) for web-based emails e.g., Hotmail, Gmail, and yahoo.



- Mailbox contains incoming messages for user.
- Message queue of outgoing (to be sent) mail messages.
- SMTP protocol between mail servers to send email messages.
- Uses TCP to reliably transfer email message from client (mail server initiating connection) to server, port 25.
- Three phases of transfer:
 - SMTP handshaking (greeting)
 - SMTP transfer of messages
 - SMTP closure





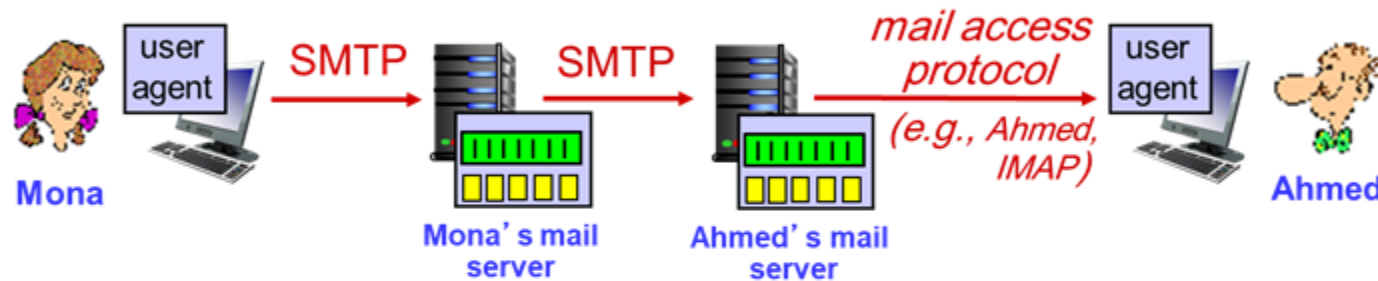
- 1) Mona uses UA to compose message "to" ahmed@esu.ac.ae
- 2) Mona's UA sends message to her mail server; message placed in message queue.
- 3) Client side of SMTP opens TCP connection with Ahmed's mail server.
- 4) SMTP client sends Mona's message over the TCP connection
- 5) Ahmed's mail server places the message in Ahmed's mailbox.
- 6) Ahmed invokes his user agent to read message.

Sample SMTP interaction

SMTP handshake
SMTP transfer of message
SMTP closure

```

Server : 220 esu.ac.ae
Client : HELO gmail.com
Server : 250 Hello gmail.com , pleased to meet you
Client : MAIL FROM: <Mona@gmail.com>
Server : 250 Mona@gmail.com ... Sender ok
Client : RCPT TO: <ahmed@esu.ac.ae>
Server : 250 ahmed@esu.ac.ae... Recipient ok
Client : DATA
Server : 354 Enter mail, end with "." on a line by itself
Client : Did you finish the short exam?
Client : Is it difficult?
Client : .
Server : 250 Message accepted for delivery
Client : QUIT
Server : 221 esu.ac.ae closing connection
  
```



Authorization phase

➤ Client commands:

user: declare username
pass: password

➤ Server responses:

+OK
 -ERR

Transaction phase, client:

- **list**: list message numbers
- **retr**: retrieve message by number
- **dele**: delete
- **quit**

Sample POP3 interaction

POP3
authorization
phase

```
Server : +OK POP3 server ready
Client : user Ahmed
Server : +OK
Client : pass Academic20
Server : +OK user successfully logged on
```

POP3 transaction phase

```
Client : list
Server : 1 498
Server : 2 912
Server : .
Client : retr 1
Server : <message 1 contents>
Server : .
Client : dele 1
Client : retr 2
Server : <message 2 contents>
Server : .
Client : dele 2
Client : quit
Server : +OK POP3 server signing off
```

SMTP vs HTTP

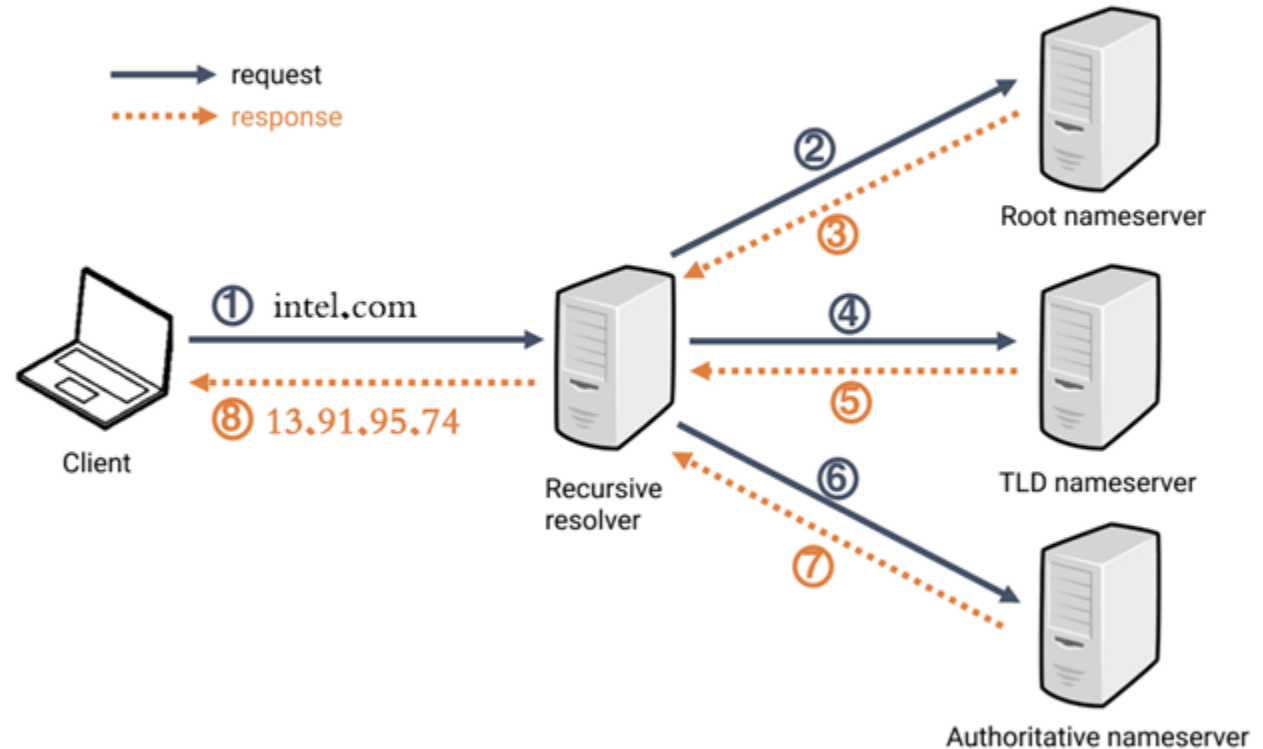
SMTP	HTTP
Uses port number 25	Uses port number 80
Persistence connection	Persistence and non-persistence connection
It is a push protocol , i.e., the sending mail server pushes the data onto the receiving mail server by initiating a TCP connection.	It is a pull protocol , i.e., a client pulls the information available on a server by initiating a TCP connection.
Require 7-bit ASCII	Dose not require 7-bit ASCII
It places all the objects into a single message. It uses Multipurpose Internet Mail Extensions (MIME) type which exchange different kinds of data files, including audio, video, images and application programs, over email.	It places each object in its own HTTP message.

POP3 vs IMAP

POP3	IMAP
The POP server listens on port 110, and the POP with SSL secure server listens on port 995	The IMAP server listens on port 143, and the IMAP with SSL secure server listens on port 993.
To read the mail it must be downloaded on the local system (All the messages are downloaded at once).	The mail content can be read partially before downloading (The Message header can be viewed prior to downloading.).
It is unidirectional i.e., all the changes made on a device do not affect the content present on the server.	It is Bi-directional i.e., all the changes made on the server or device are made on the other side too.
A user cannot organize the emails on the server using POP3.	IMAP allows its users to organize their available emails on the server.

What is DNS?

The Domain Name System (DNS) is a distributed database comprised of all the websites on the Internet and their corresponding IP addresses used to resolve host names (e.g., www.intel.com) into machine-readable IP addresses (e.g., 13.91.95.74). DNS uses both TCP and UDP for communication through port 53.



How Does
DNS Work?*Local DNS*

Also referred to as a DNS resolver, receives the query from the DNS client. Then it communicates with other DNS servers to find the right IP address.

*Root
nameservers*

It answers requests by sending back a list of the authoritative nameservers that go with the correct TLD.

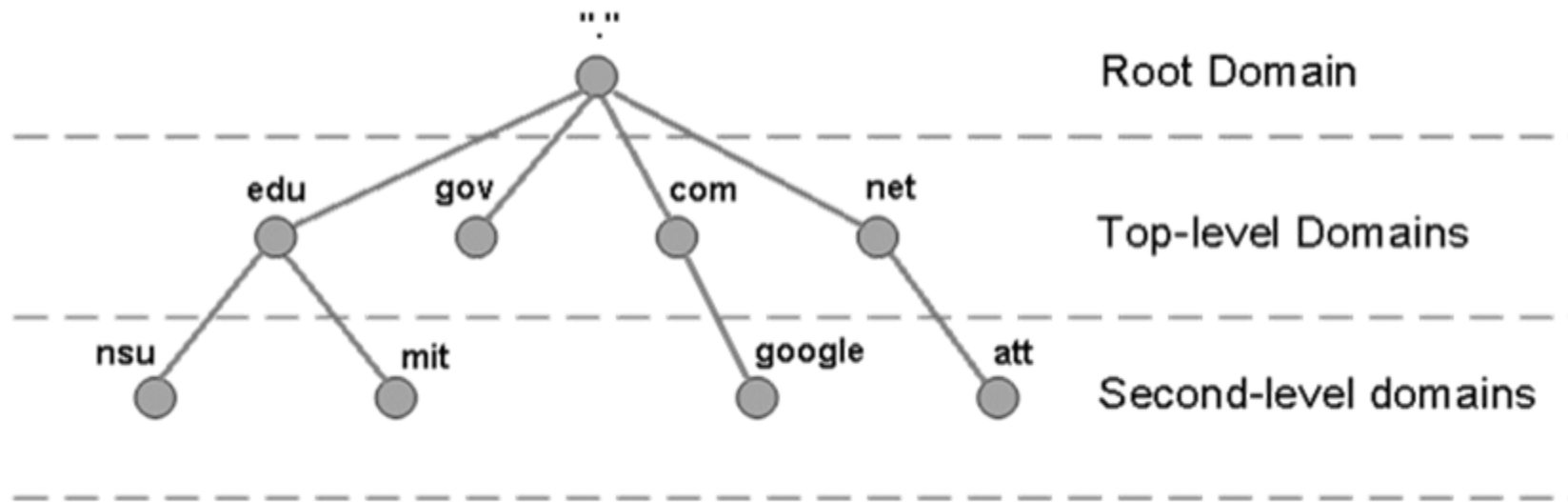
*TLD
nameservers*

keeps the IP address of the second-level domain contained within the TLD name.

*Authoritative
nameservers*

It is what gives the real answer to a DNS query. There are two types; **Primary** which keeps the original copies of the zone records and **Secondary** which keeps the exact copy of the primary server.

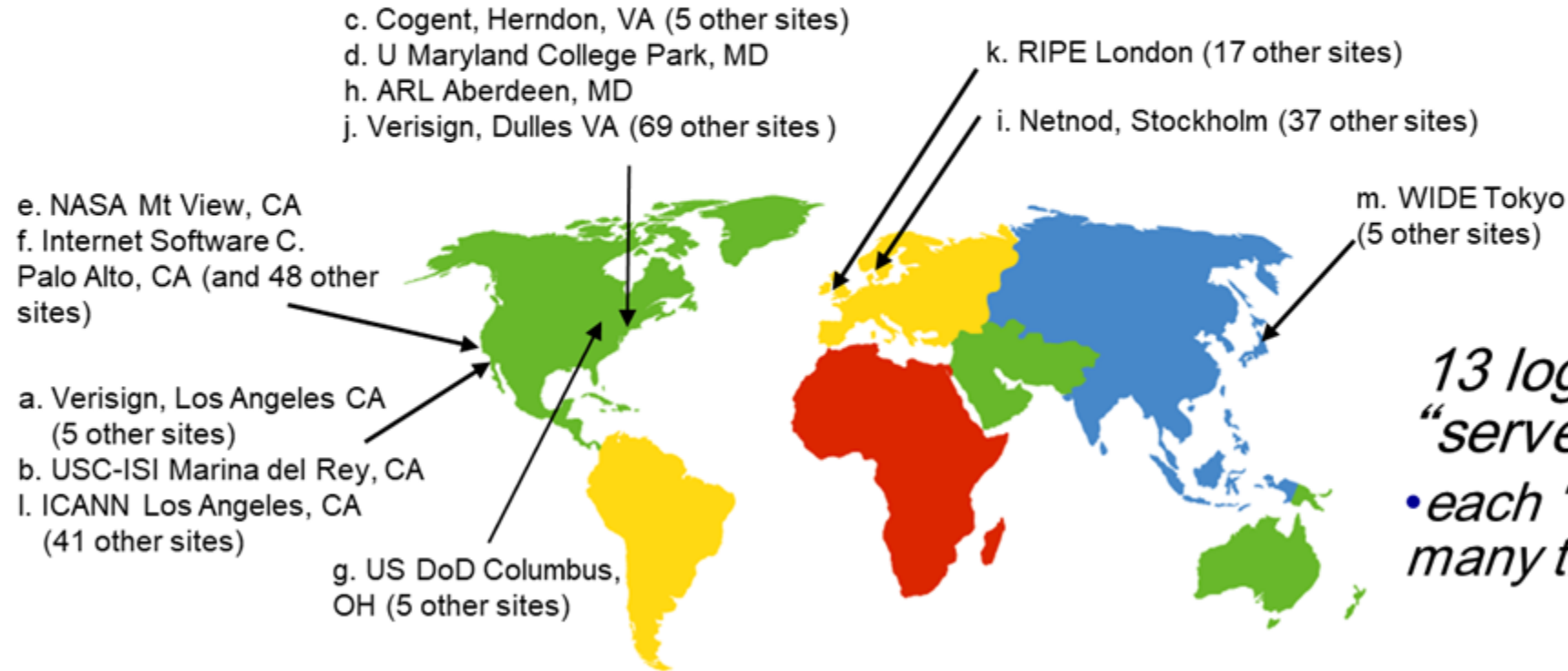
Domain Name System (DNS)



Client wants IP for *www.mit.edu*:

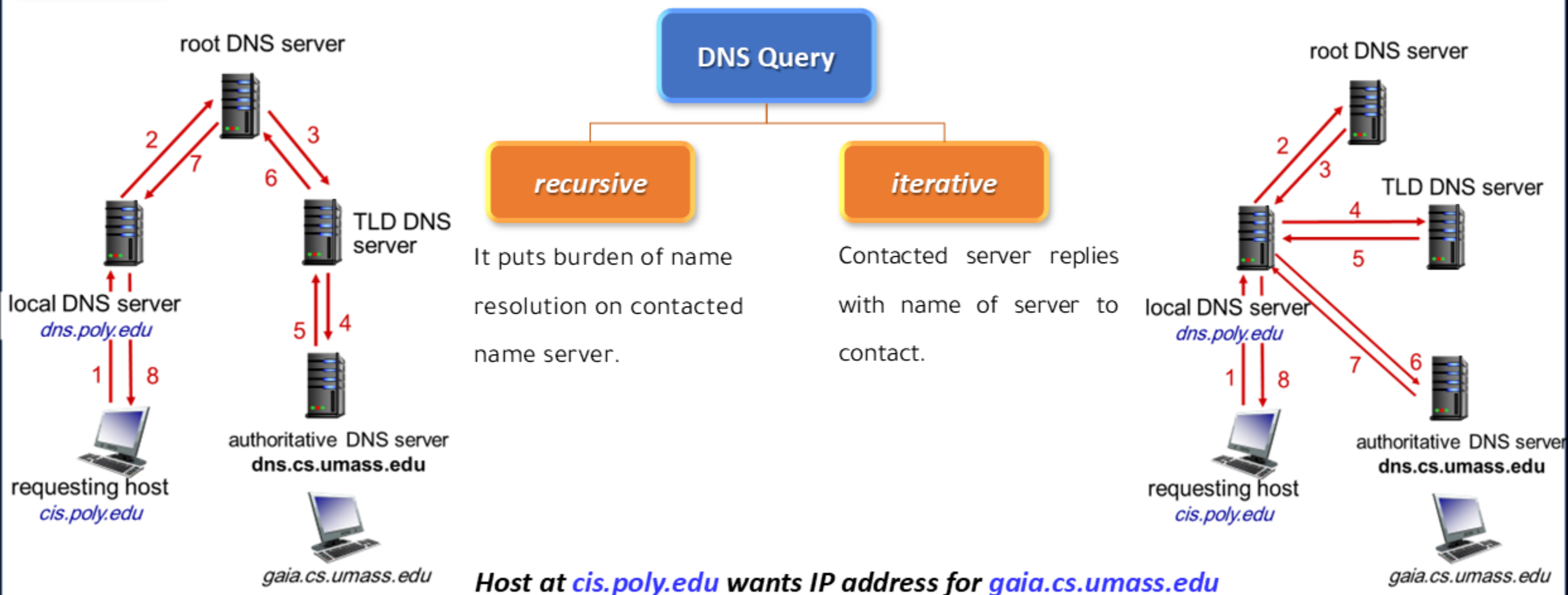
- Client queries *root server* to find com DNS server
- Client queries *.edu* DNS server to get *mit.edu* DNS server
- Client queries *mit.edu* DNS server to get IP address for *www.mit.edu*

Root Name Servers



*13 logical root name
“servers” worldwide*

- *each “server” replicated many times*



What is DNS cache?

- A DNS cache is a temporary database, maintained by a computer's operating system, that contains DNS records of all the recent visits and attempted visits to websites and other internet domains.
- A DNS cache becomes poisoned when unauthorized domain names or IP addresses are inserted into it by a computer viruses or other network attacks that insert invalid DNS entries into the cache.
- DNS Flushing is clearing any DNS records from operating system cache, usually used to troubleshooting cache poisoning.
- DNS TTL (time to live) is a setting that tells the DNS resolver how long to cache a query before requesting a new one.

DNS Records

DNS: distributed database storing resource records (RR)

RR format: (name, value, type, ttl)

**A, AAAA
Record**

Name: is a hostname
Value: is an IP address
(A; for IPV4 and
AAAA for IPv6)

NS Record

Name: is a domain
Value: is a hostname of
authoritative name
server for this domain

**CNAME
Record**

Name: is an alias name
for some "canonical" (the
real) name.
Value: is canonical name.

MX Record

Name: is a domain
Value: is a name of mail
server associated with
domain.

DNS protocol, messages

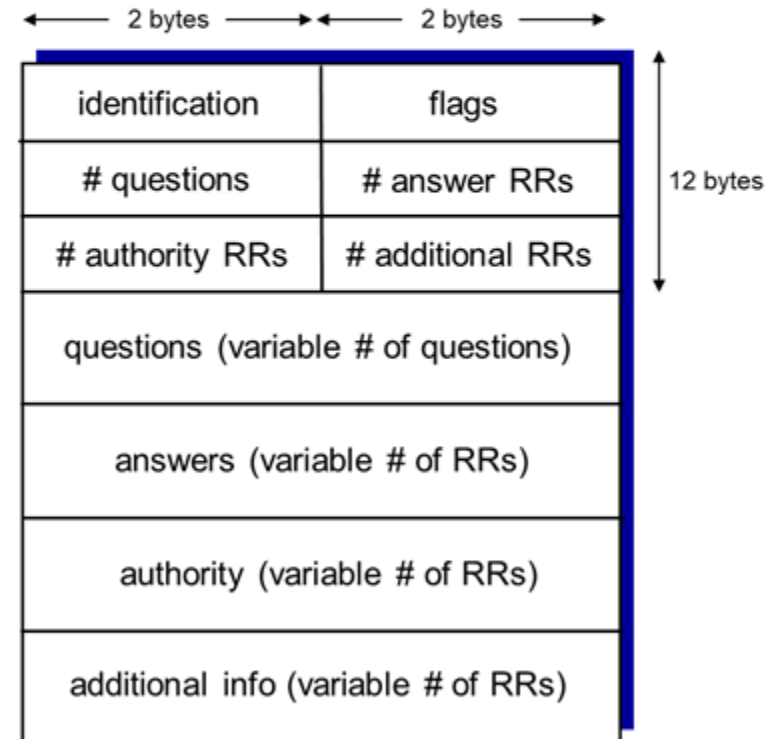
query and reply messages, both with same message format

Message header:

- **Identification:** 16 bit # for query, reply to query uses same #

Flags:

- Query or Reply
- Recursion desired
- Recursion available
- Reply is authoritative (*the answer that originates from the DNS Server which has the information about the zone file.*)



**Attacking
DNS****DDoS Attack**

The attacker's traffic isn't coming from a single source. Instead, the target gets hit from thousands of different IP addresses, which are usually part of a botnet.

Targets: Root name servers, and TLD, Local DNS

**Redirect
Attack**

Cache poisoning; The attacker corrupts a DNS server by replacing a legitimate IP address in the server's cache with that of a rogue address to redirect traffic to a malicious website.

Man-in-Middle; Intercept DNS queries and redirect them to malicious website.

Video Streaming and Content Distribution Networks (CDN)

What is Video Streaming?

- Video streaming is a continuous transmission of video files from a server to a client.
- In video streams, content is sent in a compressed form over the internet and is displayed by the viewer in real time. The media is sent in a continuous stream of data and is played as it arrives. The user needs a player, which is a special program that uncompressed and sends video data to the display and audio data to speakers.



Video Streaming and Content Distribution Networks (CDN)

Video coding

Spatial

*Video: sequence of images displayed at constant rate
e.g., 24 images/sec*

Instead of sending N values of same color (all purple), send only two values: color value (purple) and number of repeated values (N)



Temporal

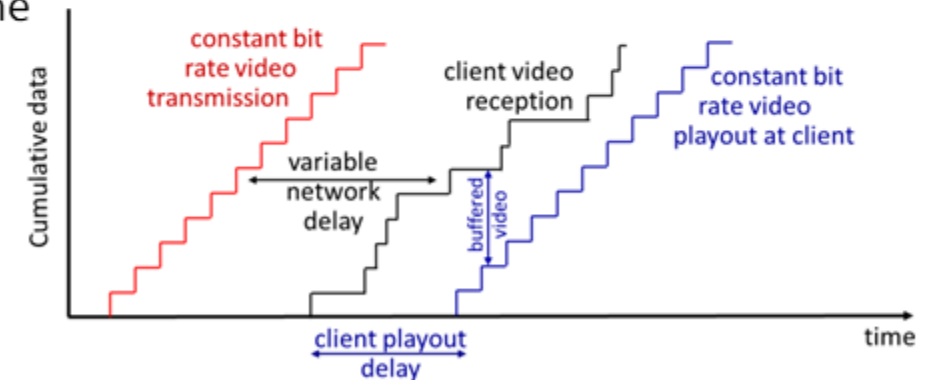
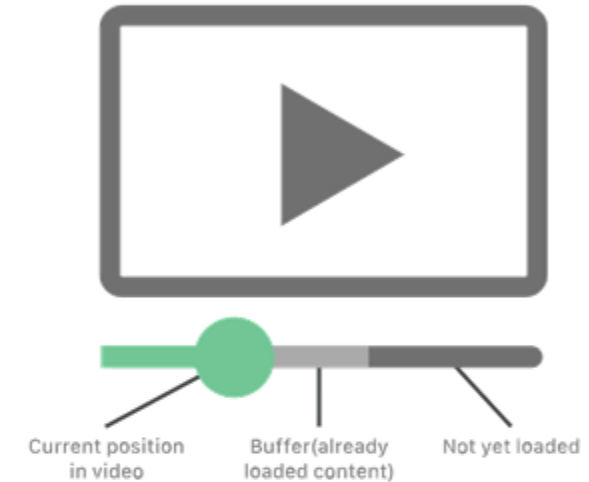
Instead of sending complete frame at $i+1$, send only differences from frame i .



Video Streaming and Content Distribution Networks (CDN)

What is Video Buffering?

- A video buffer is a portion of a computer's physical memory that is used to store video or graphics information as it moves from the renderer (video card) to the display screen or monitor.
- Video streaming suffer from server-to-client **bandwidth** will vary over time besides **packet loss**, **slower performing client devices**, or **Network latency**.
- Client-side buffer is used to overcome the previous challenges.



Video Streaming and Content Distribution Networks (CDN)

Streaming multimedia: DASH

What is Streaming multimedia: DASH?

➤ *It is a streaming method. DASH stands for "Dynamic Adaptive Streaming over HTTP."*

Server:

- Divides video file into multiple chunks.
- Each chunk encoded at multiple different rates.
- Different rate encodings stored in different files.
- Files replicated in various CDN nodes.
- *manifest file*: provides URLs for different chunks.

Client:

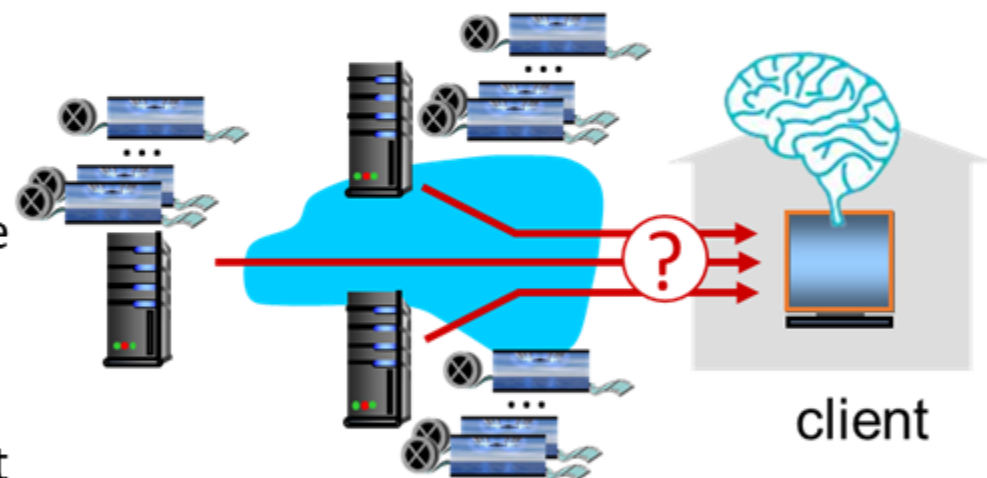
- Periodically estimates server-to-client bandwidth
- Consulting manifest, requests one chunk at a time
 - Chooses maximum coding rate sustainable given current bandwidth
 - Can choose different coding rates at different points in time (depending on available bandwidth at time), and from different servers

Video Streaming and Content Distribution Networks (CDN)

Streaming multimedia: DASH

➤ 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).



Streaming video = encoding + DASH + playout buffering

Video Streaming and Content Distribution Networks (CDN)

What is Content Distribution Networks (CDN)?

- **CDN** is store/serve multiple copies of videos at multiple geographically distributed sites in order to stream content to hundreds of thousands of simultaneous users.

