Chapter 2

Application Layer

Application Layer

- Application layer is the top most layer in the network model
- This is the layer where user actually works
- This layer is used by computer network applications like Google Chrome, Firefox etc. that use Internet

Principles of Network Applications

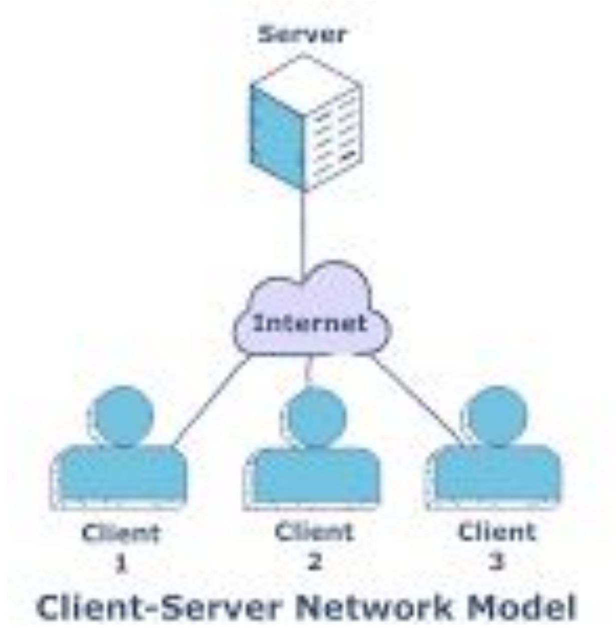
Network Application

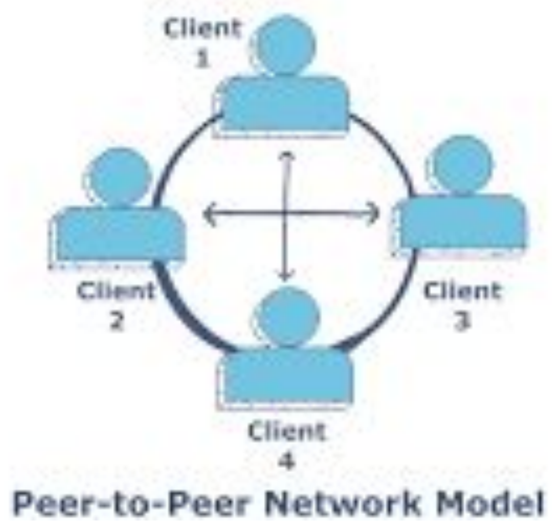
- Network Application is the program that run on different end systems and communicate with each other over the network
- **Examples**: **Browsers**

Principles of Network Applications (contd..)

<u>Network Application Architecture</u>

- 2 types of architecture,
  1. Client-Server architecture
  2. Peer-to-Peer architecture(P2P)





Principles of Network Applications (contd..)

1. Client-Server Architecture

- In client-server architecture there are two processes called client and server
- In client-server architecture, two client machines cannot directly communicate with each other instead the communication happen only through server

Server

- Always-on system
- Server system provides response to the request from the client machine
- Server has a fixed IP address

Client

- The system that sends request to server is called as client
- In client-server architecture two client machines cannot directly communicate with each other

Principles of Network Applications (contd..)

- In client-server architecture, single server is incapable of servicing request from all clients. So Data centers (housing a large number of servers) are used to create a powerful server
- A Data center can have hundreds to thousands of servers
- **Examples of client-server architecture:** Browsing in Internet, E-mail

Principles of Network Applications (contd..)

2. Peer-to-Peer Architecture

- In P2P architecture, there is no server
- Instead there is a direct connection between systems called peers
- peers request service from other peers, provide service in return to other peers
- **Example of Peer-to-Peer architecture:** Internet Telephony(Skype)
- Peer-to-Peer architecture is cost effective. No need to invest in high computing server

Principles of Network Applications (contd..)

Processes Communicating

- Process on two different end systems communicate with each other by exchanging messages across the computer network. There are two types of process: client and server process

1. Client and Server Process

Client process: the process that initiates the communication is called client process

Server process: The process that provides the response is called server process

Example:

- While browsing for any content in the web, browser process initiates the communication which is called client and server provides response which is called server process
- In P2P file sharing, Peer A asks Peer B to send a file, so Peer A is client process and Peer B is the server

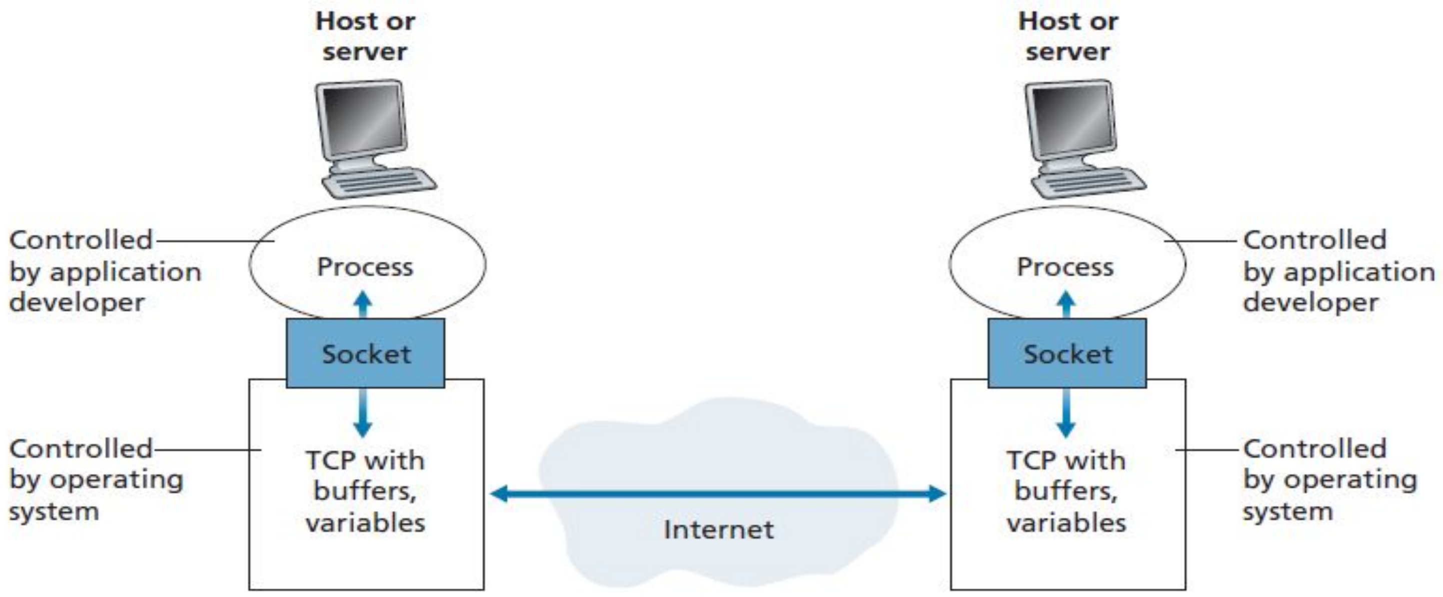
Note: applications with P2P architectures also have client processes & server processes

Principles of Network Applications (contd..)

2. Interface between process and Computer Network

- A process sends and receives messages through an interface called Socket

Analogy to socket: Assume process as a house and socket as a door in the house


Host or server
Host or server
Controlled by application developer
Process
Socket
TCP with buffers, variables
Controlled by operating system
Internet
Process
Socket
TCP with buffers, variables
Controlled by application developer
Controlled by operating system

- A socket is the interface between the application layer and transport layer of a system
- Socket is also called as the Application Programming Interface(API) between the application and network

Principles of Network Applications (contd..)

3. Addressing Processes

- In the internet, host is identified by the IP address
- Once the message reaches the receiver, the receiver process is identified by the Port Numbers

Principles of Network Applications (contd..)

Transport Services Available to Applications

- For the network application what is been developed, we have to choose the transport layer protocol
- The transport layer protocol is chosen based on the services provided by the transport layer protocols and the type of application
- The services provided by the transport layer protocols are,

1. Reliable Data transfer
2. Throughput
3. Timing
4. Security

Principles of Network Applications (contd..)

Transport Services Available to Applications

1.Reliable Data transfer

- Reliable Data transfer means guaranteed delivery of data to the destination. There should be no data loss
- One important property of transport layer protocol is the process-to-process reliable data transfer
- Applications that require reliable data transfer(No data loss) are E-mail, File transfer, Web documents transfer
- Applications like loss-tolerant applications (Multimedia applications like Video Conferencing, Live streaming of audio/video) can tolerate some amount of data loss

Principles of Network Applications (contd..)

Transport Services Available to Applications

2. Throughput

- Throughput means amount of data the sending process sends to the receiving process in a particular time
- This throughput fluctuate with time because of traffic in the network
- Transport layer protocol provide guaranteed throughput at some specific rate
- There are 2 types of applications based on throughput requirement,
    1. Bandwidth-sensitive applications
    2. Elastic Applications
- Applications that have throughput requirements are called as Bandwidth-sensitive applications. **Examples**: Multimedia applications
- Elastic Applications in contrast to Bandwidth-sensitive applications does not specify any throughput requirement. It is going to make use of available throughput. **Examples**: E-mail, File transfer, web transfers

Principles of Network Applications (contd..)

<u>Transport Services Available to Applications</u>

<u>3. Timing</u>

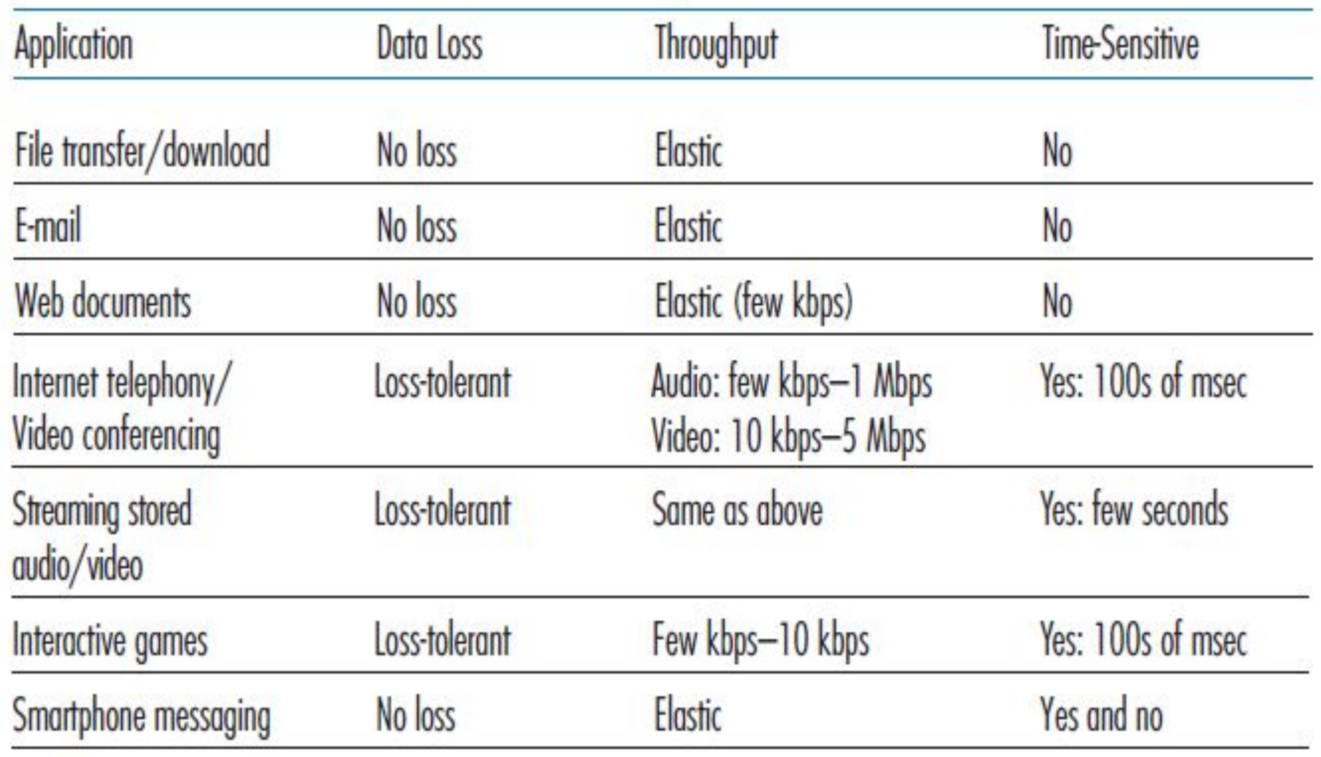
- Timing means, during data transfer the data should reach receiver within a specific amount of time
- Multimedia applications require this timing while applications like E-mail, File transfer a smaller amount of delay can be acceptable

<u>4. Security</u>

- Transport layer protocols provide one or more security services
- One such security feature is the encryption and decryption of data which provide data confidentiality between the two process

Principles of Network Applications (contd..)

<u>Transport Services Available to Applications</u>



Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1 Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Smartphone messaging	No loss	Elastic	Yes and no

Principles of Network Applications (contd..)

<u>Transport services provided by the Internet</u>

- There are two main transport-layer protocols:

    1. Transmission Control Protocol(TCP)
    2. User Datagram Protocol(UDP)

- With these protocols the following are the services provided by them

<u>TCP Services:</u>

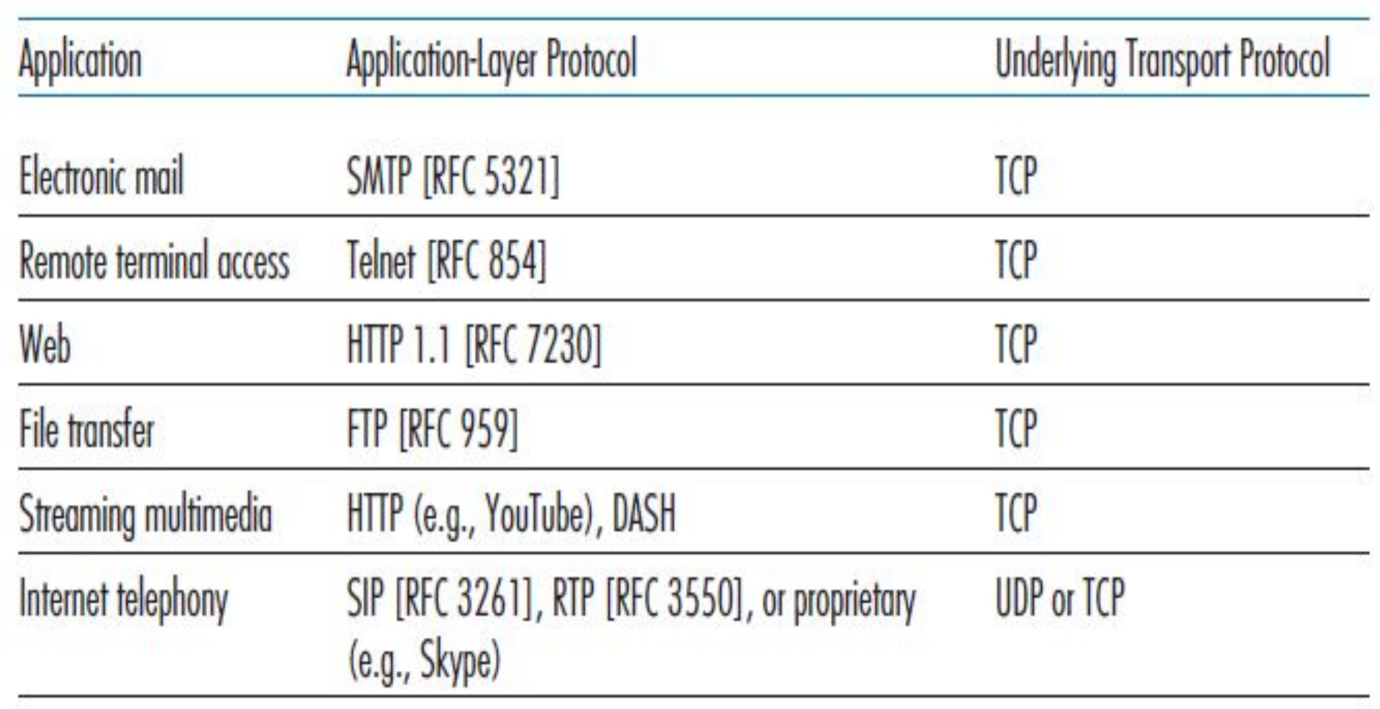
- Connection-oriented service
- Reliable Data Transfer

<u>UDP Services</u>

- Connectionless service

Principles of Network Applications (contd..)

Transport services provided by the Internet



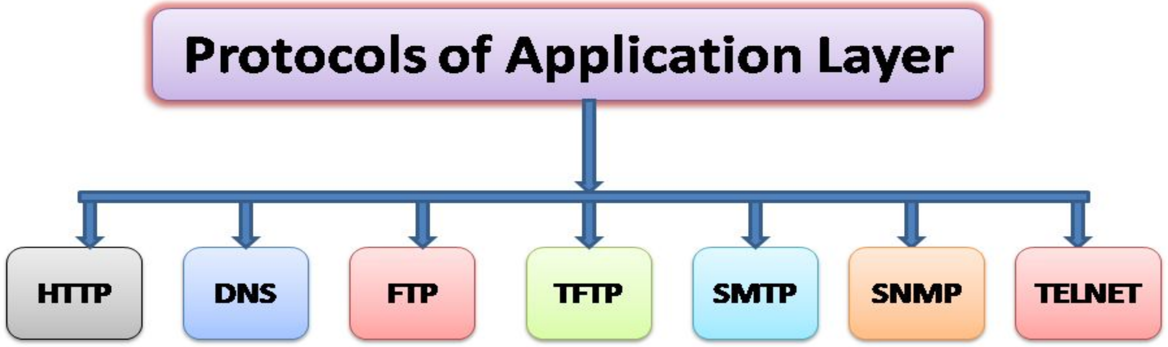
Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP 1.1 [RFC 7230]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube), DASH	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Principles of Network Applications (contd..)

Application Layer Protocols

- Application layer protocols defines how an application process running on different machines can pass messages to each other
- Application layer protocol defines:

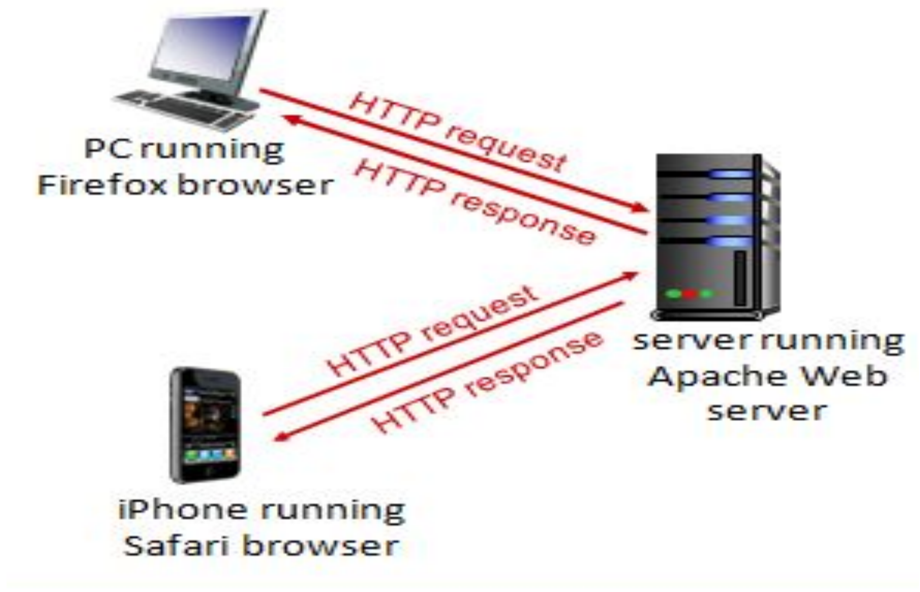
  1. Types of messages exchanged (Example request and response message)
  2. Syntax of various message types such as fields in the messages
  3. Meaning of various fields in the message
  4. Rules for determining when and how process send messages and responds to messages
- Various application layer protocols are,



1.The Web and HTTP

Overview of HTTP

- HTTP stands for Hyper Text Transfer Protocol(HTTP) is the heart of the web
- Messages is usually exchanged between client and server. The structure of that message is defined using this HTTP protocol
- HTTP uses TCP as the underlying Transport layer protocol


PC running Firefox browser
HTTP request
HTTP response
server running Apache Web server
HTTP request
HTTP response
iPhone running Safari browser

1.The Web and HTTP (contd..)

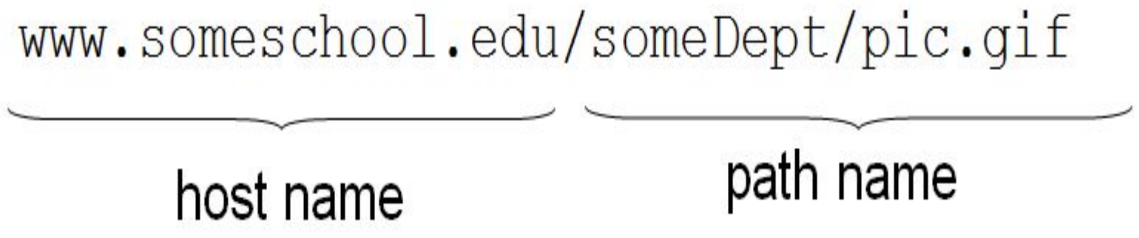
Overview of HTTP

Web page

- A Web page (also called a document) consists of objects. An object is simply a file such as an HTML file, a JPEG image, a Javascript file, a CCS style sheet file, or a video clip

URL

- Each web page is addressable by an URL
- Each URL has two components: the hostname of the server that houses the object and the object's path name



1.The Web and HTTP (contd..)

Overview of HTTP

- HTTP uses TCP as its underlying transport protocol
- The original version of HTTP is called HTTP/1.0
- As of 2020, the majority of HTTP transactions take place over HTTP/1.1
- HTTP is called stateless protocol because it stores no information about the client. When a particular client asks for the same object twice in a period of a few seconds, the server resends the object, as it has completely forgotten what it did earlier.

1.The Web and HTTP (contd..)

Types of connections in HTTP

1. Non-persistent connection
2. Persistent Connection

1. Non-persistent Connection

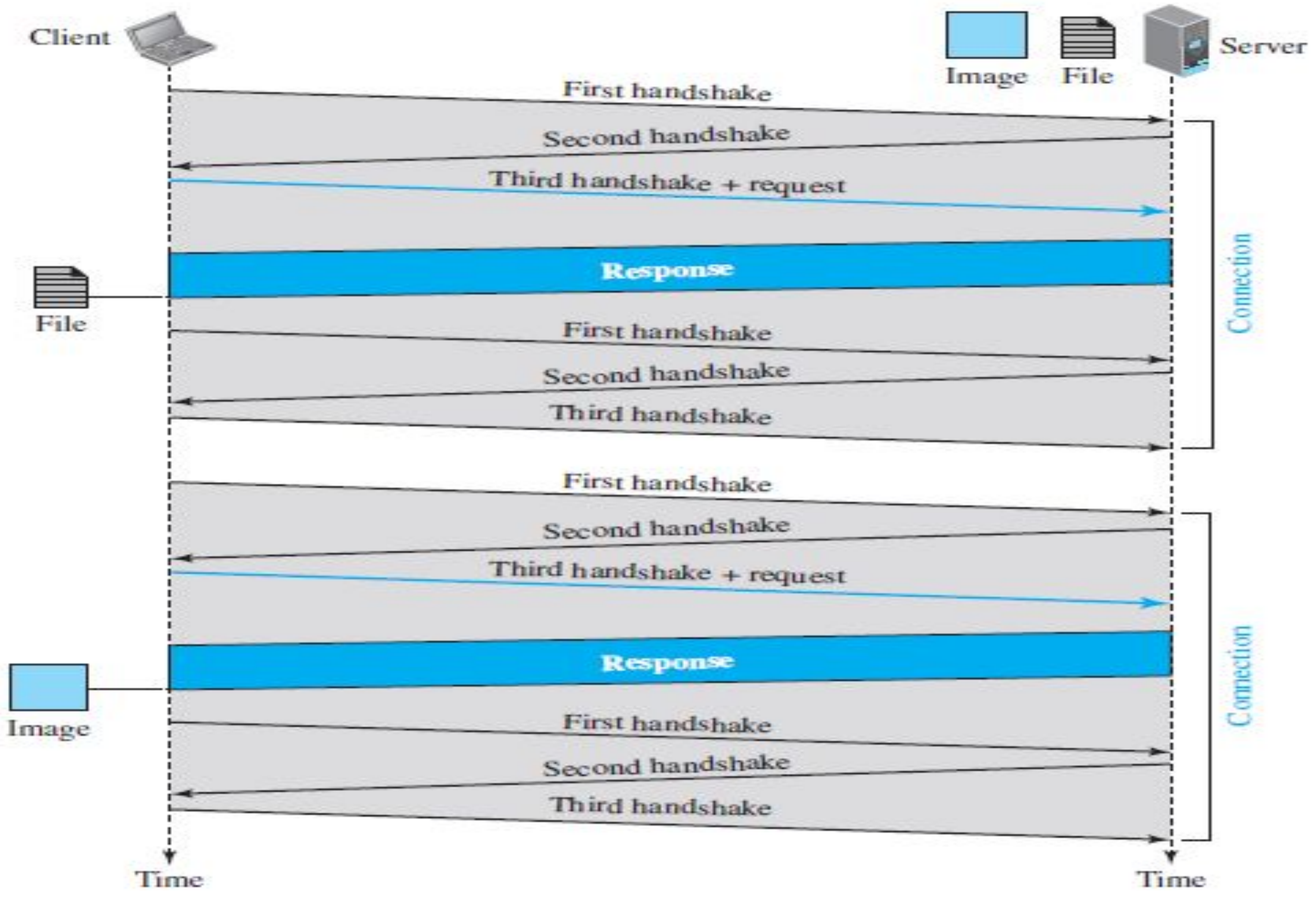
- One TCP connection is created for each request/response
- Imposes big overhead
- Used by HTTP/1.0

Example:

- The client needs to access a file that contains one link to an image. The text file and image are located on the same server. Here we need two connections.
- First the client has to create a connection establishment for accessing text file and then make connection termination
- Second the client access the image using link in that file using another connection establishment and termination

1.The Web and HTTP (contd..)

<u>Example(contd..)</u>



1.The Web and HTTP (contd..)

2. Persistent Connection

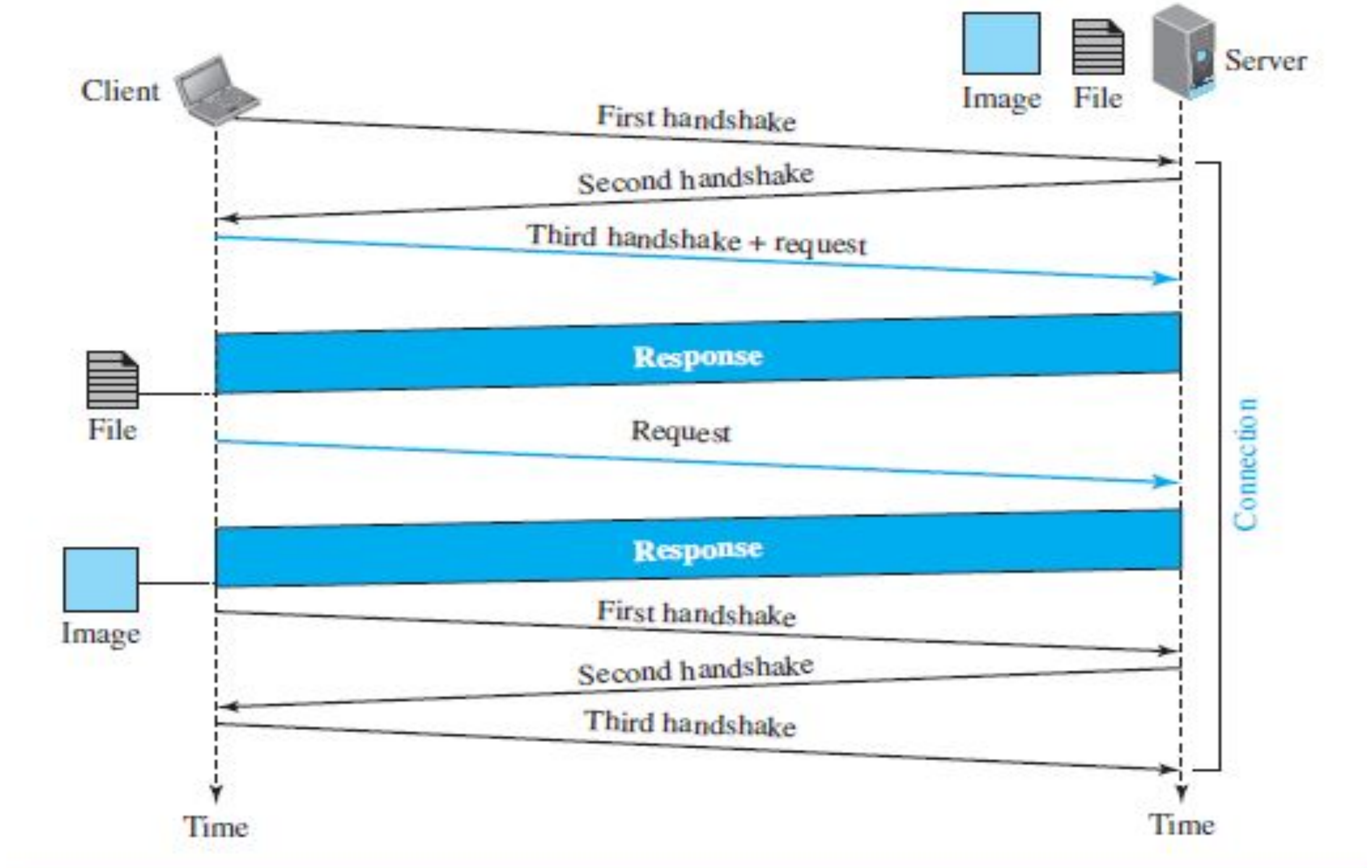
- In persistent connection , only one TCP connection is created for all request and response
- Once connection is created client can access all data using that connection.
- Default connection method for HTTP version 1.1

Example:

- The client needs to access a file that contains one link to an image. The text file and image are located on the same server. Here we need only one connection to access both file and image
- First the client has to create a connection establishment for accessing text file
- Using that connection it access both text file and image
- Finally client makes connection termination

1.The Web and HTTP (contd..)

Example(contd..)



1.The Web and HTTP (contd..)

<u>Example</u>

Refer for Problems in note for Types of HTTP connection

1.The Web and HTTP (contd..)

Back-of-the-envelope calculation

- It is the time between when client makes the request for a file and the entire file has been given as a response from server

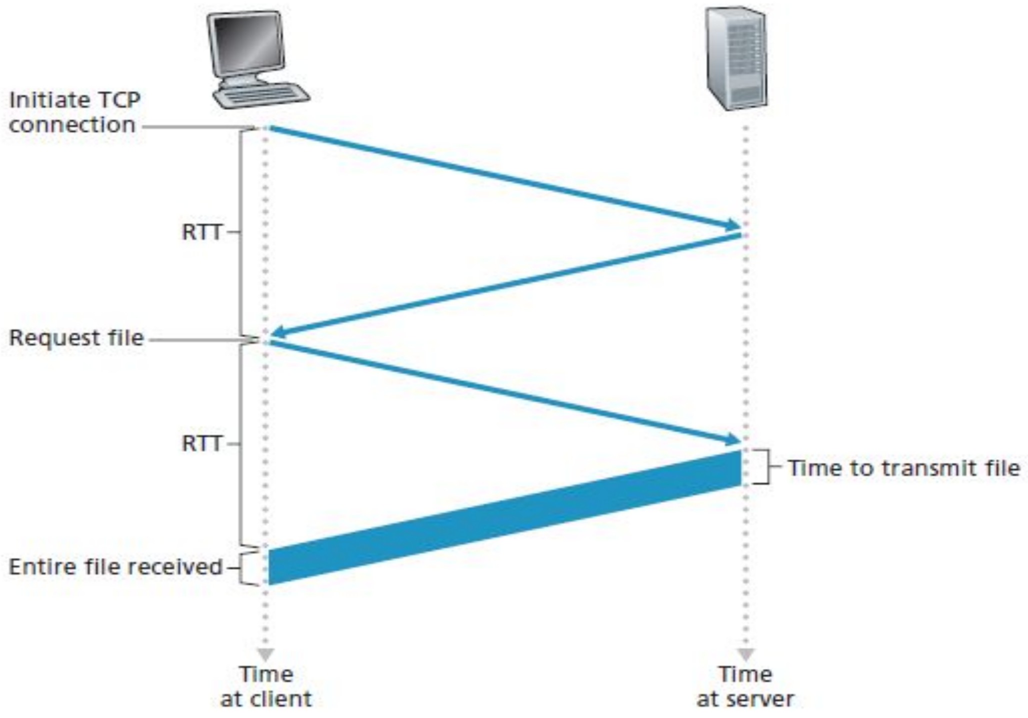


Figure 2.7 ♦ Back-of-the-envelope calculation for the time needed to request and receive an HTML file

1.The Web and HTTP (contd..)

<u>**Back-of-the-envelope calculation**</u>

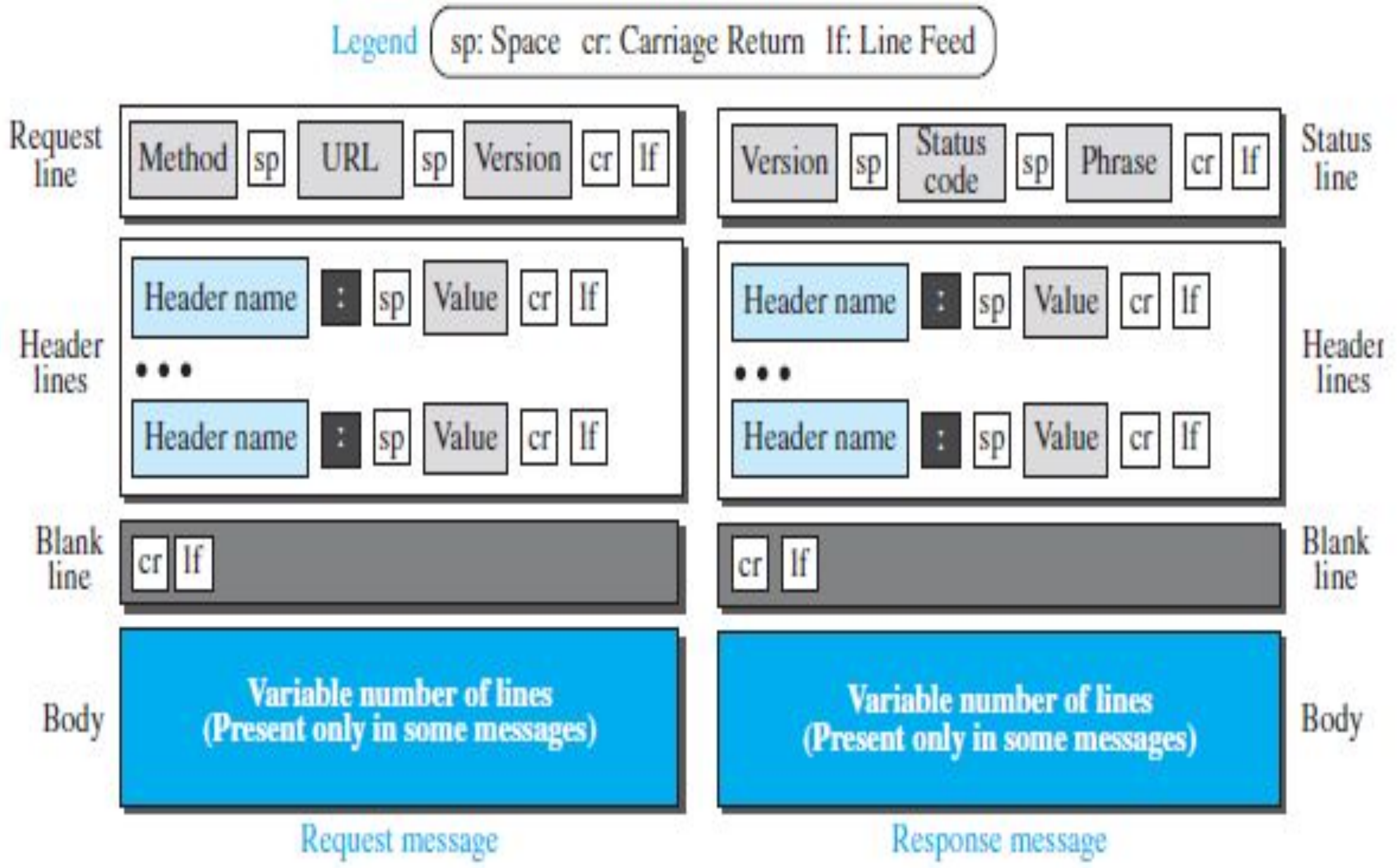
Back-of-the-envelope = 2 RTT’s + Transmission time of entire file from server

RTT (Round Trip Time) = Time to send some data to server and response from server

1. The Web and HTTP (contd..)

HTTP Message Format

- There are two HTTP messages request and response message



1. The Web and HTTP (contd..)

<u>1. Request Message</u>

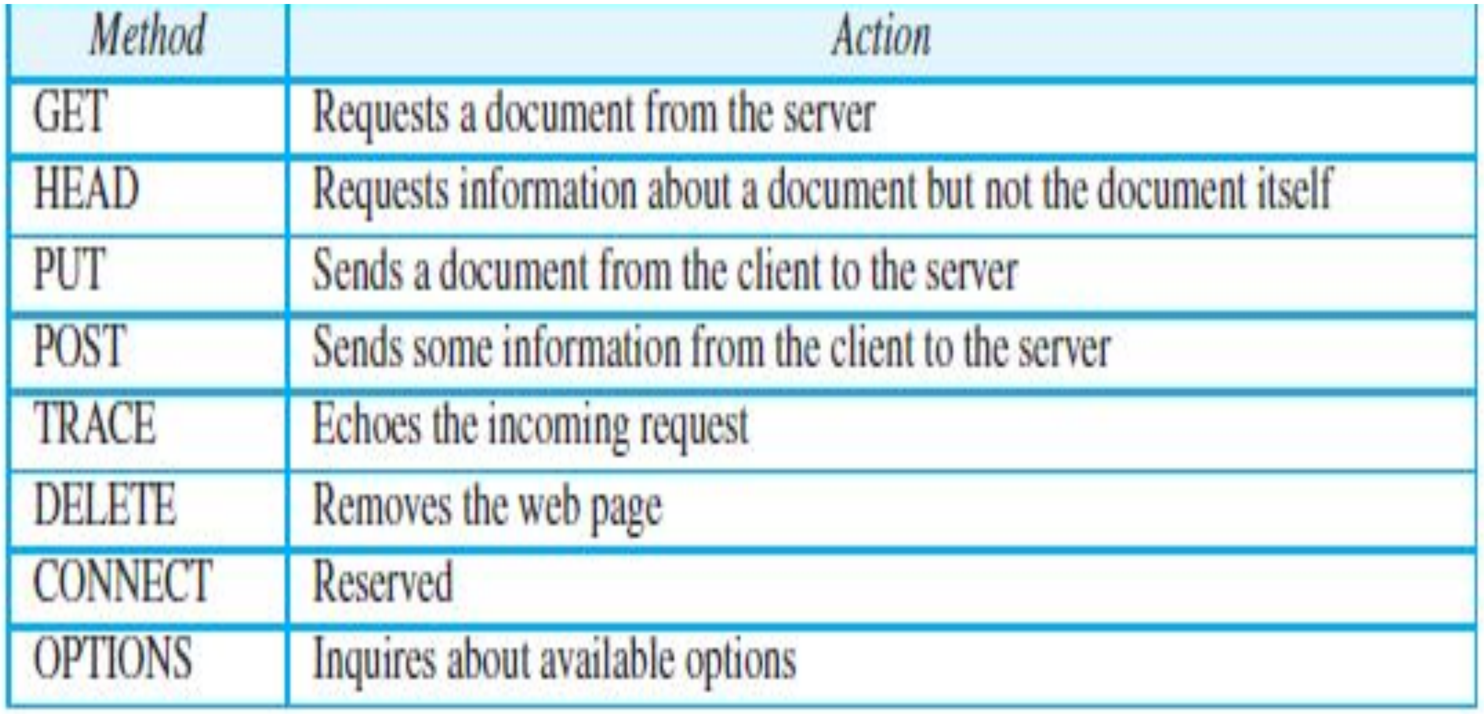
<u>Request Line</u>

- First Line in the request message is Request Line
- 3 fields in request line
  1. Method
  2. URL
  3. Version
- Method field defines the request types. Various values for this field is shown in table(next slide). Majority of request message uses GET method
- Second field defines URL
- Third field defines version and current version of HTTP

1. The Web and HTTP (contd..)

1. Request Message

Method types in Request line



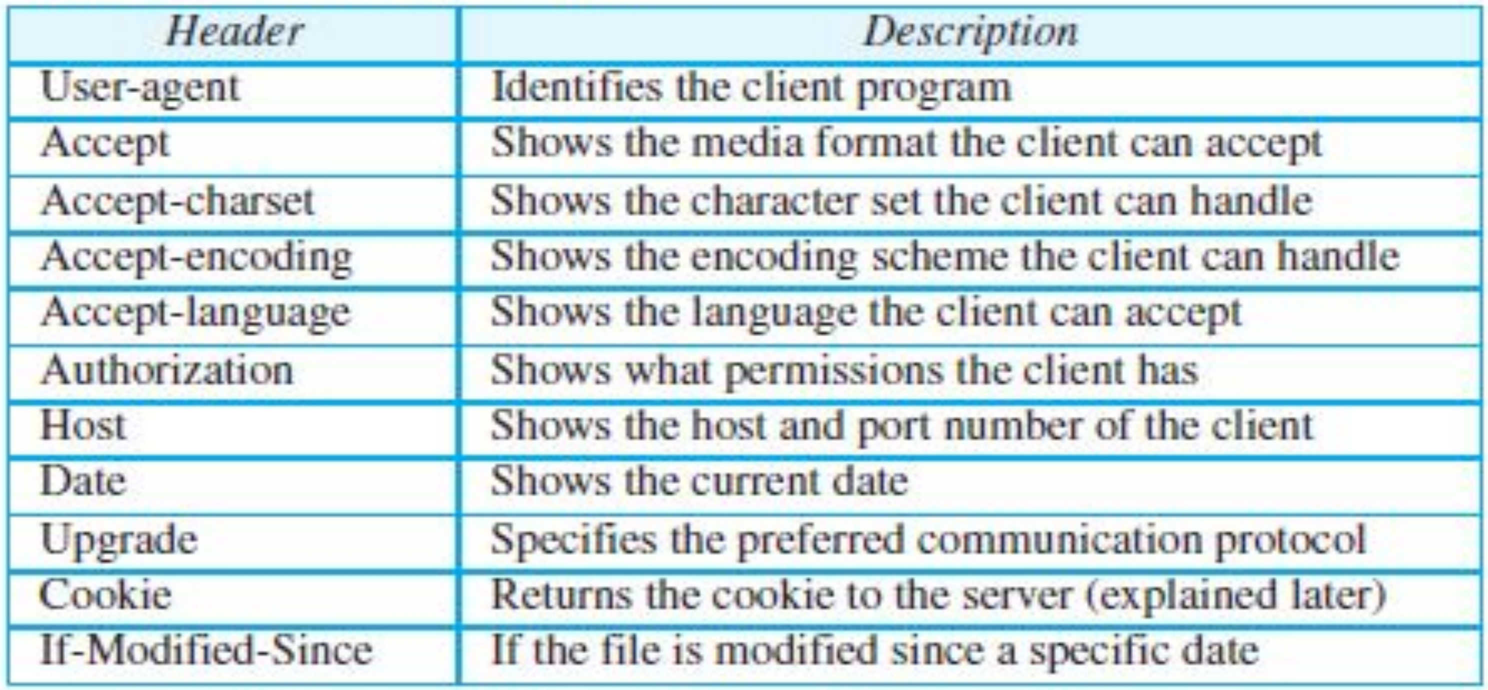
Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

1. The Web and HTTP (contd..)

1. Request Message

Header Line

- Below request line is zero or more header lines. Header line is used to give additional information from client to server
- Each header line has Header name followed by value
- The various header names are shown in below table



Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later)
If-Modified-Since	If the file is modified since a specific date

1. The Web and HTTP (contd..)

1. **Request Message**

Body of the Request message

- The body of the request message is empty with GET method but it is used in POST method
- HTTP client uses the POST method when user fills out a form or when user provides search words to a search engine. If the value of method field is POST, then body of the request message contains what user have entered into the form fields

1. The Web and HTTP (contd..)

2. Response Message

Status Line

- First Line in the response message is Status Line
- 3 fields in request line
  1. Version
  2. Status code
  3. Phrase
- Version field defines version of HTTP protocol
- Status code defines status of the request. It consists of 3 digits.

  code within 100 – used for informational purpose

  code within 200 – indicate successful request

  code within 300 – redirect client to another URL

  code within 400 – indicate an error at client site

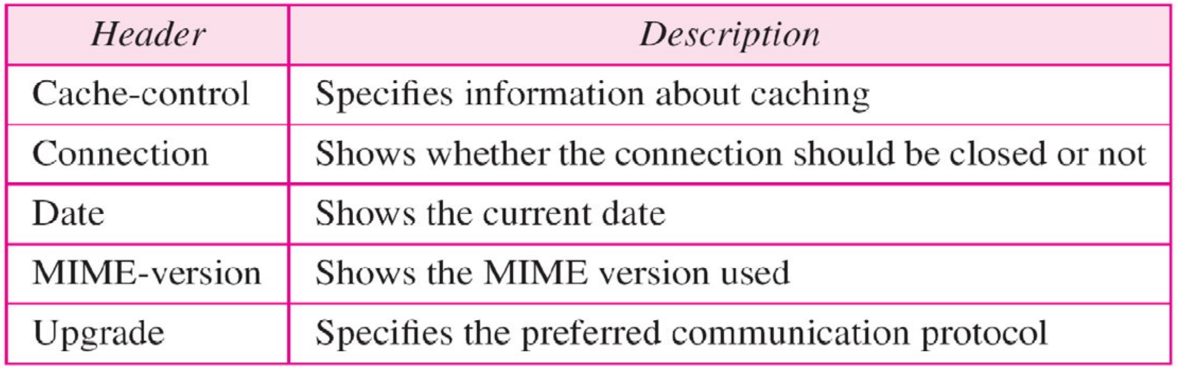
  code within 500 – indicate an error at server site
- Third field phrase gives explanation for status code

1. The Web and HTTP (contd..)

2. Response Message

Header Line

- Below the status line is zero or more header lines. Header line is used to give additional information from server to client
- Each header line has Header name followed by value
- The various header names are shown in below table

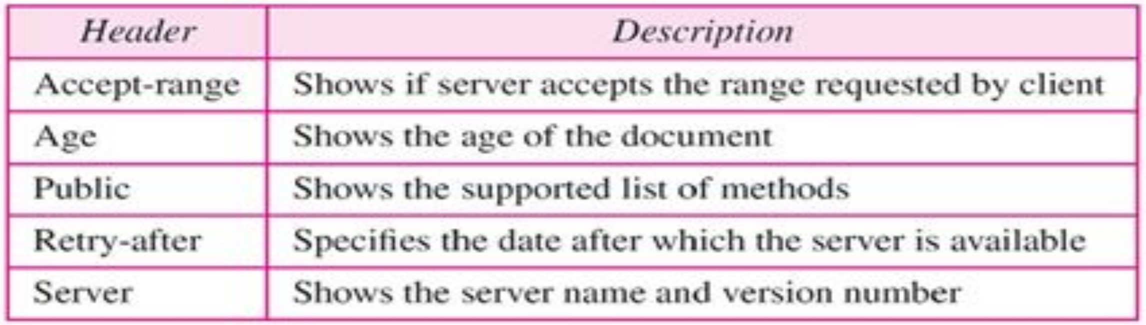


Header	*Description*
Cache-control	Specifies information about caching
Connection	Shows whether the connection should be closed or not
Date	Shows the current date
MIME-version	Shows the MIME version used
Upgrade	Specifies the preferred communication protocol

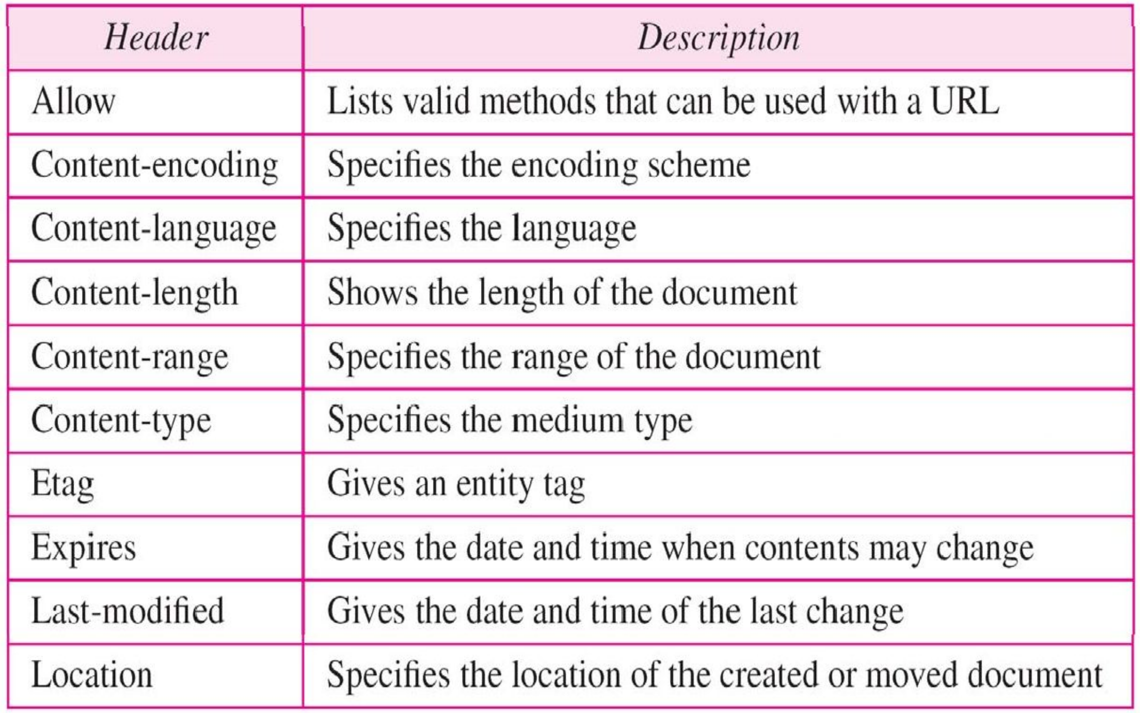
1. The Web and HTTP (contd..)

2. Response Message

- The various header names are shown in below table



Header	*Description*
Accept-range	Shows if server accepts the range requested by client
Age	Shows the age of the document
Public	Shows the supported list of methods
Retry-after	Specifies the date after which the server is available
Server	Shows the server name and version number



Header	*Description*
Allow	Lists valid methods that can be used with a URL
Content-encoding	Specifies the encoding scheme
Content-language	Specifies the language
Content-length	Shows the length of the document
Content-range	Specifies the range of the document
Content-type	Specifies the medium type
Etag	Gives an entity tag
Expires	Gives the date and time when contents may change
Last-modified	Gives the date and time of the last change
Location	Specifies the location of the created or moved document

1. The Web and HTTP (contd..)

1. **Response Message**

Body of the Response message

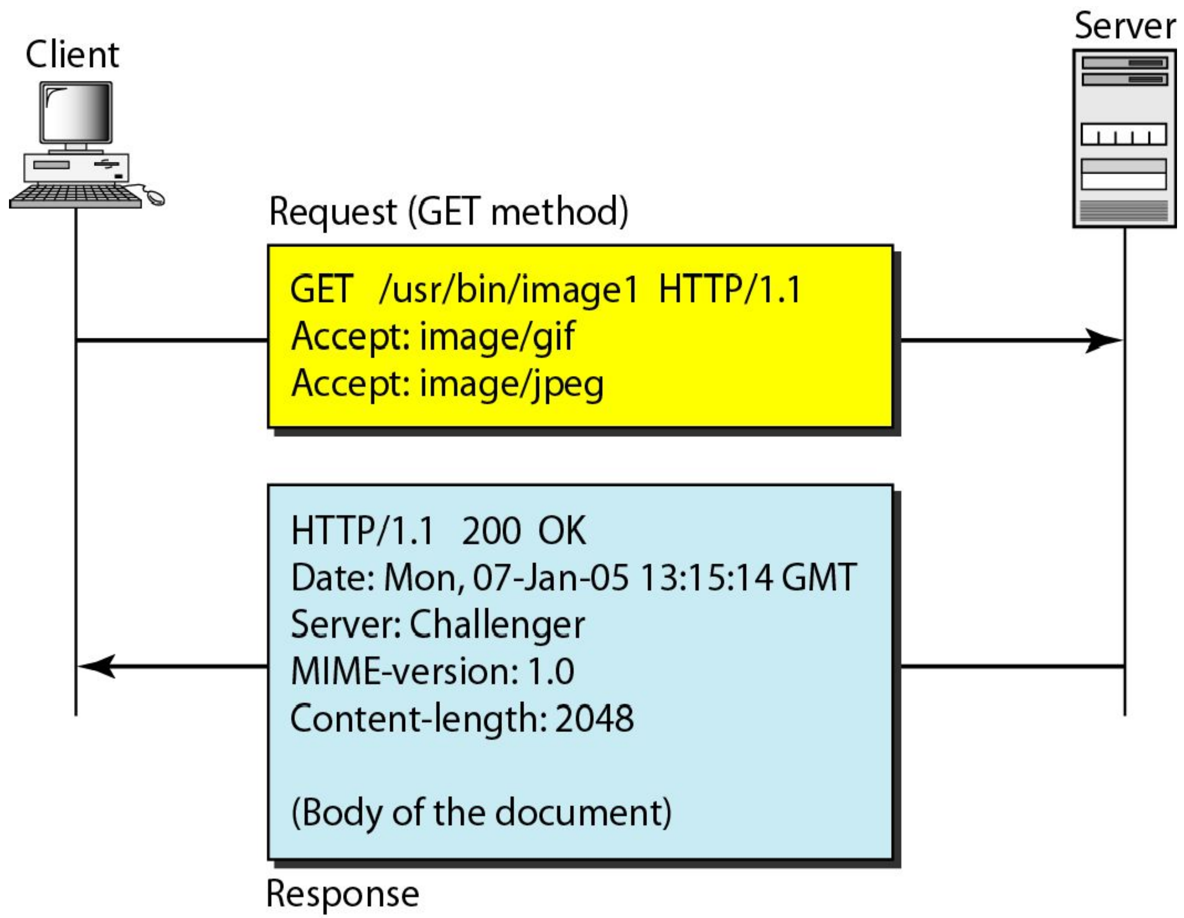
- The body of the response message contains the data that is requested by the client

1. The Web and HTTP (contd..)

HTTP Request and Response message

Example 1:

- In the below example, an image present in the path /usr/bin/image1 is retrieved using GET method



```
GET   /usr/bin/image1  HTTP/1.1
Accept: image/gif
Accept: image/jpeg
```

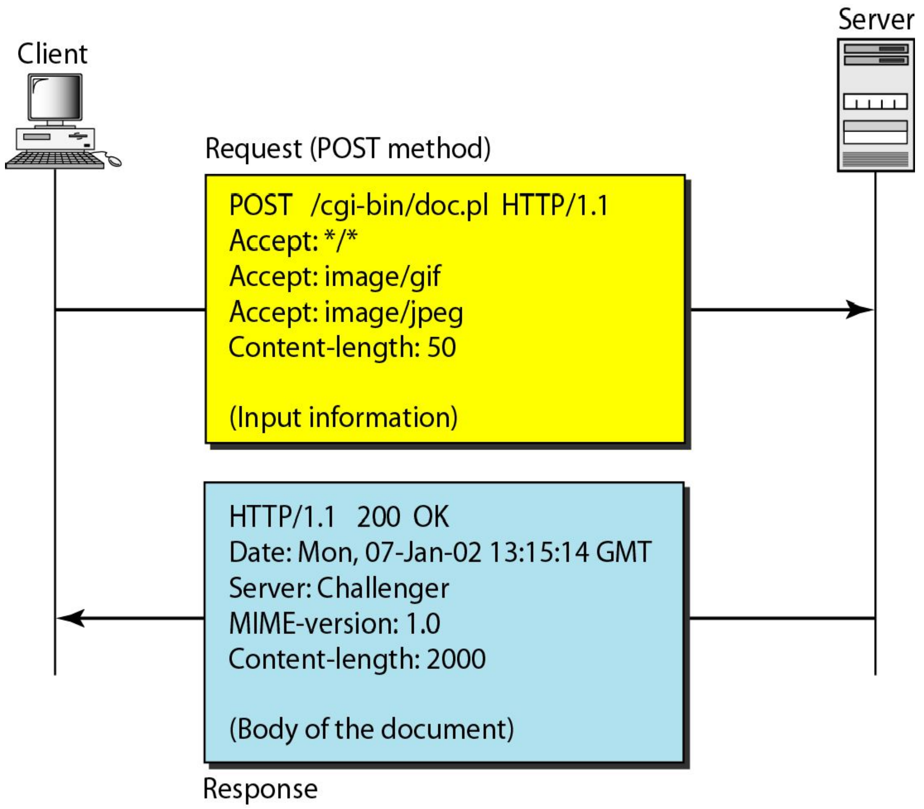
```
HTTP/1.1   200  OK
Date: Mon, 07-Jan-05 13:15:14 GMT
Server: Challenger
MIME-version: 1.0
Content-length: 2048

(Body of the document)
```

1. The Web and HTTP (contd..)

HTTP Request and Response message

Example 2:

- In the below example, client sends some data to the server using POST method



```
POST  /cgi-bin/doc.pl  HTTP/1.1
Accept: */*
Accept: image/gif
Accept: image/jpeg
Content-length: 50

(Input information)
```

```
HTTP/1.1   200  OK
Date: Mon, 07-Jan-02 13:15:14 GMT
Server: Challenger
MIME-version: 1.0
Content-length: 2000

(Body of the document)
```

1. The Web and HTTP (contd..)

Cookies

- **Cookies** are text files with small pieces of data
- Cookies are files created by websites we visit. They make your online experience easier by saving browsing information. With cookies, sites can keep us signed in, remember our site preferences, and give us locally relevant content.
- Each cookie is identified using an ID

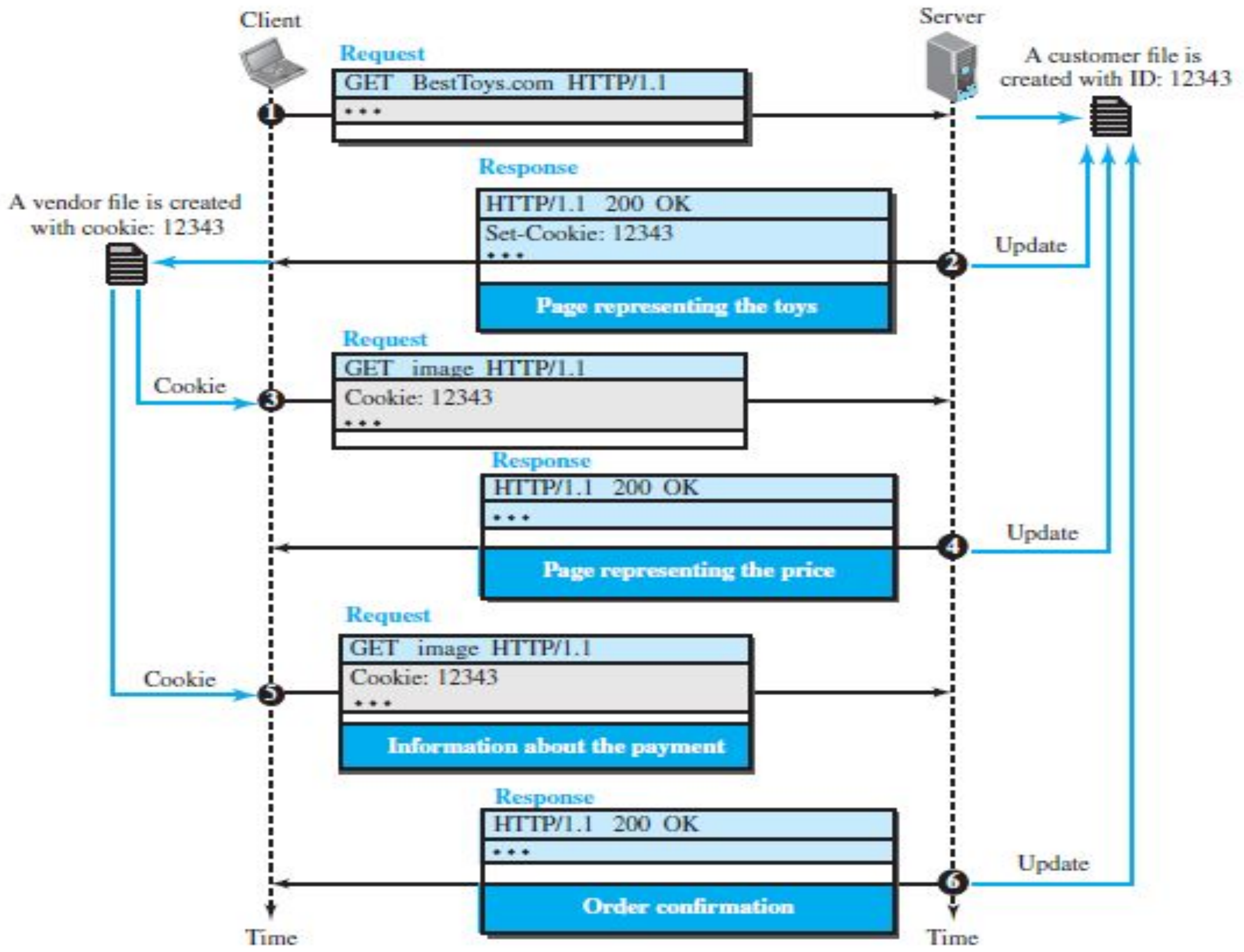
Creating and Storing Cookies

- In the diag shown in next slide, upon receiving request from client, server creates cookie for that client and set an ID. Server sends that cookie ID in its response message (Set-Cookie : 12343)
- When client receives response from server it gets the cookie ID. Hereafter all request send from that client to that server contains that cookie ID attached in order to know about that client

1. The Web and HTTP (contd..)

Cookies

Example:



1. The Web and HTTP (contd..)

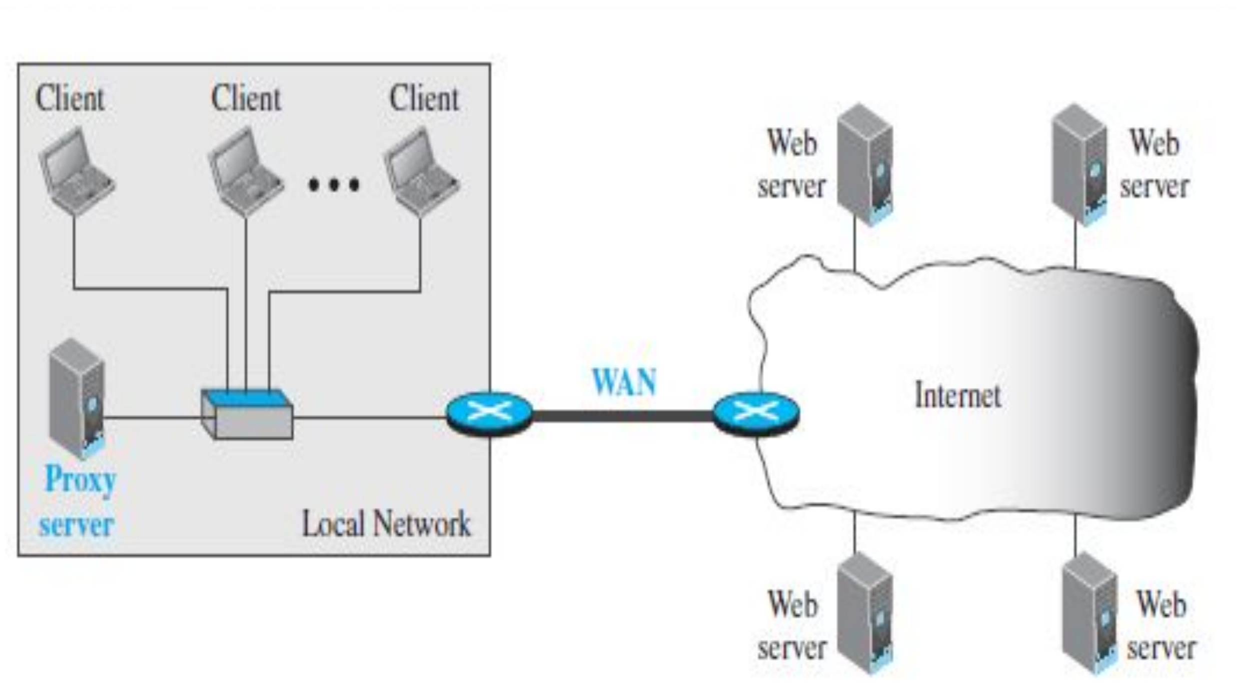
Web Caching: Proxy Servers

- HTTP supports proxy servers
- Proxy servers are used for Web caching
- Proxy server is a computer that keeps copies of responses to recent requests
- Client first sends request to proxy server. The proxy server checks it cache. If response is not stored in the cache, then proxy server send request to the corresponding server.
- Proxy server acts as both client/server. Upon responding to client is acts as server. Upon receiving data from target server it acts client

1. The Web and HTTP (contd..)

Proxy server Location

- Proxy servers are located normally at the client site
- Below diagram shows that the Proxy server is installed in the network of the client



1.The Web and HTTP (contd..)

Advantages of Web Caching:

- Reduce the response time for the client request
- Reduce traffic in the internet
- Proxy server reduces the load of original server

1.The Web and HTTP (contd..)

The Conditional GET

- Although web caching reduces response time, it introduces a new problem whether the data in the cache is updated one or not
- HTTP has a mechanism that allows a cache to verify that its data are up to date. This mechanism is called the **conditional GET**
- An HTTP request message is called conditional GET message if,
    1. the request message uses the GET method
    2. the request message includes an If-Modified-Since: header line

1.The Web and HTTP (contd..)

The Conditional GET

Example

- Initially the client sends the following request to the proxy server

```
GET   /fruit/kiwi.gif   HTTP/1.1
Host: www.exotiquecuisine.com
```

- The proxy server forwards the request to the original server
- The original server sends the following response to the proxy server

```
HTTP/1.1 200 OK
Date: Sat, 3 Oct 2015 15:39:29
Server: Apache/1.      3.0 (Unix)
Last-Modified: Wed, 9 Sep 2015 09:23:24
Content-Type: image/gif!
(data data data data data ...)
```

1.The Web and HTTP (contd..)

The Conditional GET

Example (contd..)

- Proxy server forwards that response to the client and also stores the data in its database
- Importantly, the cache also stores the last-modified date along with the data
- One week later, another browser requests the same object via the cache and the object is still in the cache. Since this object may be modified at the Web server in the past week, the cache performs an up-to-date check by issuing a conditional GET. Specifically, the cache sends:

```
GET    /fruit/kiwi.gif        HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

- the value of the If-modified-since: header line is exactly equal to the value of the Last-Modified: header line that was sent by the server one week ago

1.The Web and HTTP (contd..)

The Conditional GET

Example (contd..)

- This conditional GET is telling the server to send the object only if the object has been modified since the specified date
- Suppose the object has not been modified since 9 Sep 2015 09:23:24. Then the Web server sends a following response message to the cache:

  HTTP/1.1 304 Not Modified

  Date: Sat, 10 Oct 2015 15:39:29

  Server: Apache/1.3.0 (Unix)!

  (empty entity body)
- In above response message 304 Not Modified in the status line, tells the cache that it can go ahead and forward its cached copy of the object to the requesting client.
- If the object is modified then server sends the modified data using normal response message

1.The Web and HTTP (contd..)

HTTP/2

- HTTP/1.1 was standardized in 1997 and HTTP/2 was standardized in 2015
- The following are the added features in HTTP/2 over HTTP/1.1
  1. Break down an HTTP message into independent frames and interleave them
  2. Message Prioritization
  3. Server Pushing

1.The Web and HTTP (contd..)

HTTP/2

1. Break down an HTTP message into independent frames and interleave them

Head of Line (HOL) blocking problem

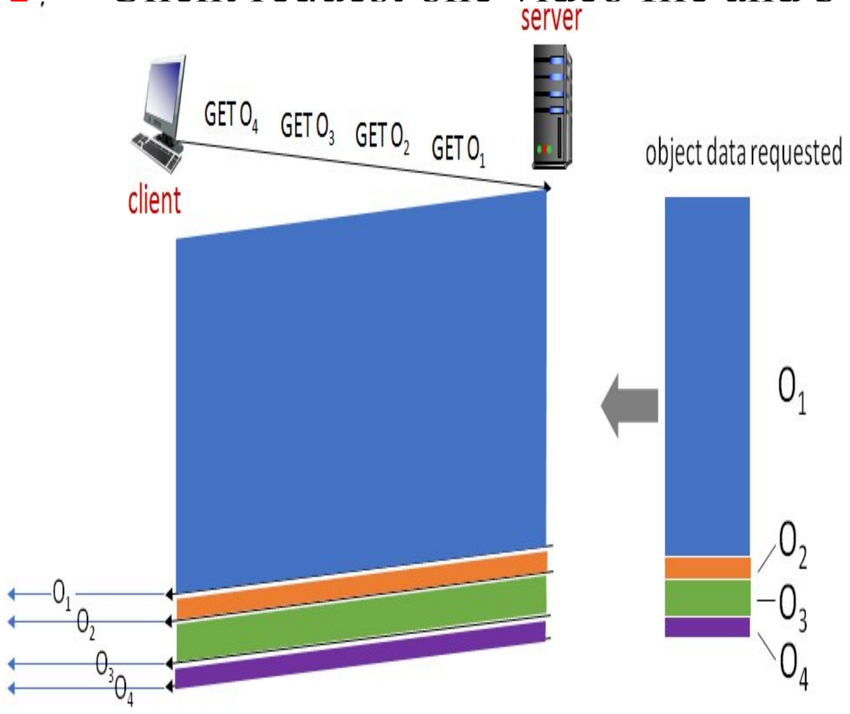
- Consider a Web page that includes an HTML base page, a large video clip near the top of Web page, and many small objects below the video
- Using a single TCP connection, the video clip will take a long time to pass through the link, while the small objects are delayed as they wait behind the video clip; that is, the video clip at the head of the line blocks the small objects behind. This problem is called Head of Line(HOL) blocking problem
- To overcome this problem in HTTP/2, the message is split into frames and interleave the request and response messages on the same TCP connection
- This mechanism helps to decrease the user perceived delay

1.The Web and HTTP (contd..)

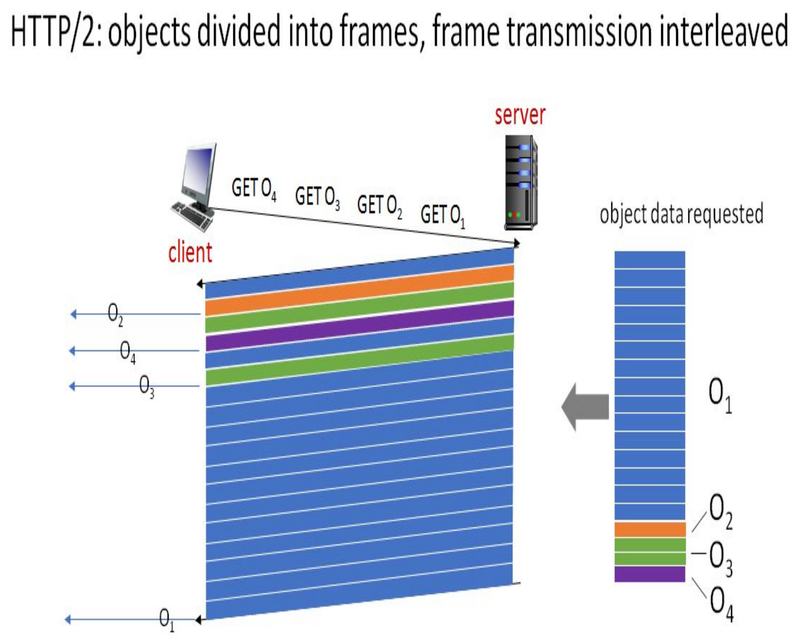
HTTP/2

1. Break down an HTTP message into independent frames and interleave them

- Client request one video file and 3 smaller objects



Without Interleaving



HTTP/2: objects divided into frames, frame transmission interleaved
server
client
GET O4 GET O3 GET O2 GET O1
object data requested
O2
O4
O3
O1
O1
O2
O3
O4

With Interleaving

1.The Web and HTTP (contd..)

HTTP/2

2. Message Prioritization

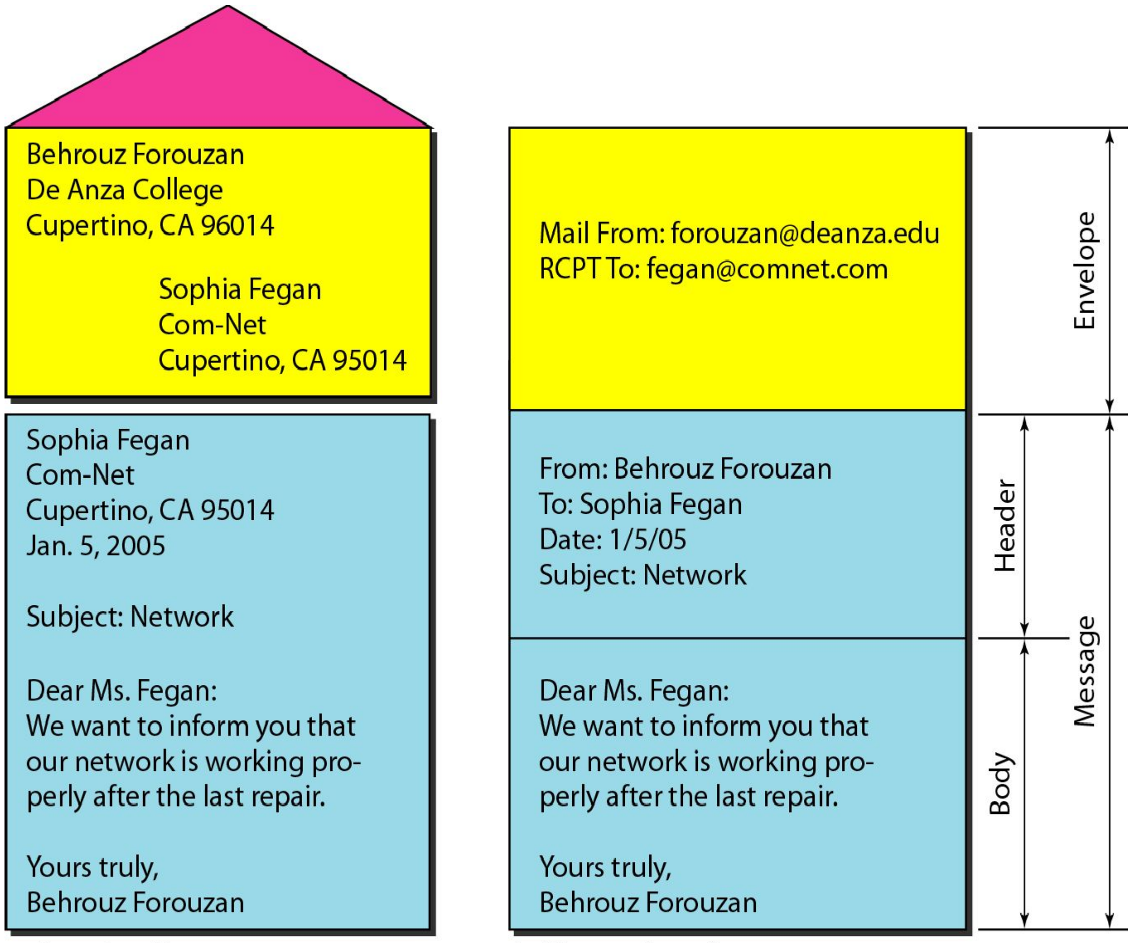
- Client can prioritize the responses it is requesting by assigning a weight between 1 and 256 to each message
- The higher number indicates higher priority
- Using these weights, the server can send first the frames for the responses with the highest priority

3. Server Pushing

- It is the ability for a server to send multiple responses for a single client request.
- That is, in addition to the response to the original request, the server can push additional objects to the client, without the client having to request each one. This is possible since the HTML base page indicates the objects that will be needed to fully render the Web page.
- Instead of waiting for the HTTP requests for these objects, the server can analyze the HTML page, identify the objects that are needed, and send them to the client before receiving explicit requests for these objects. Server push eliminates the extra latency due to waiting for the requests.

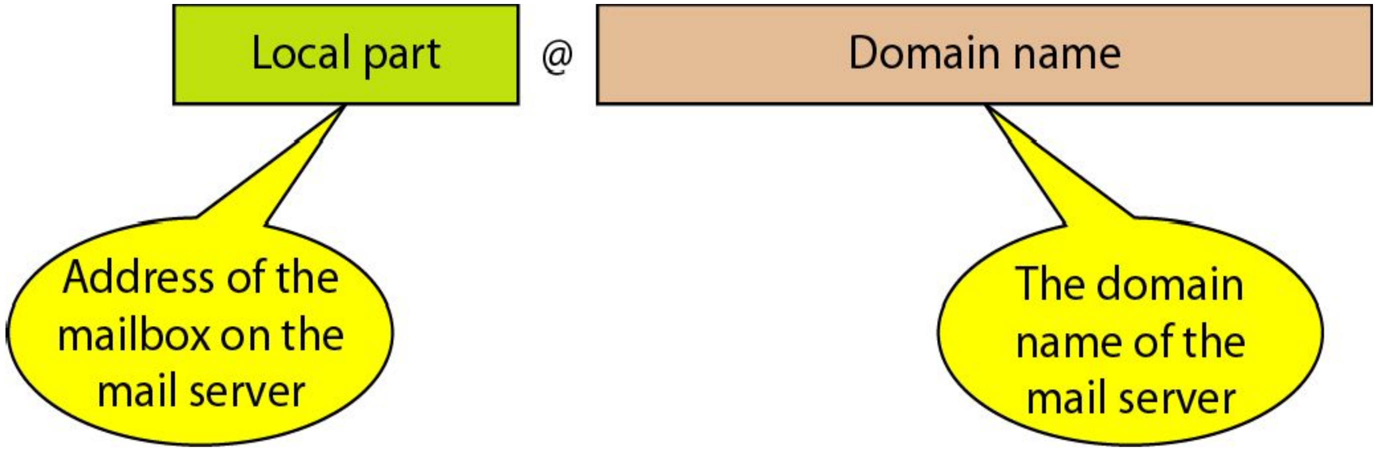
Electronic Mail (E-mail)

Postal Letter vs E-mail

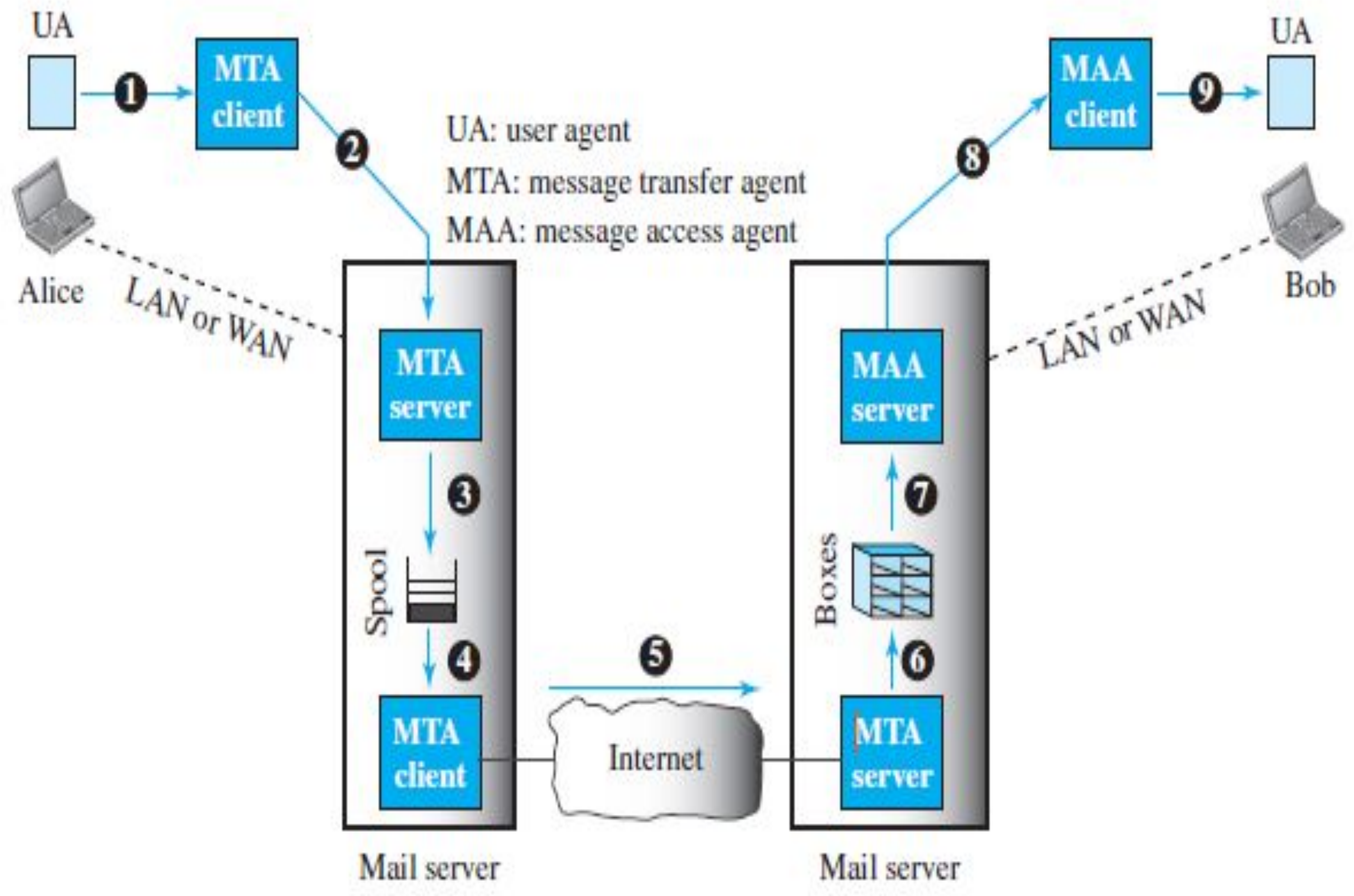


a. Postal mail b. Electronic mail

Format of E-mail address

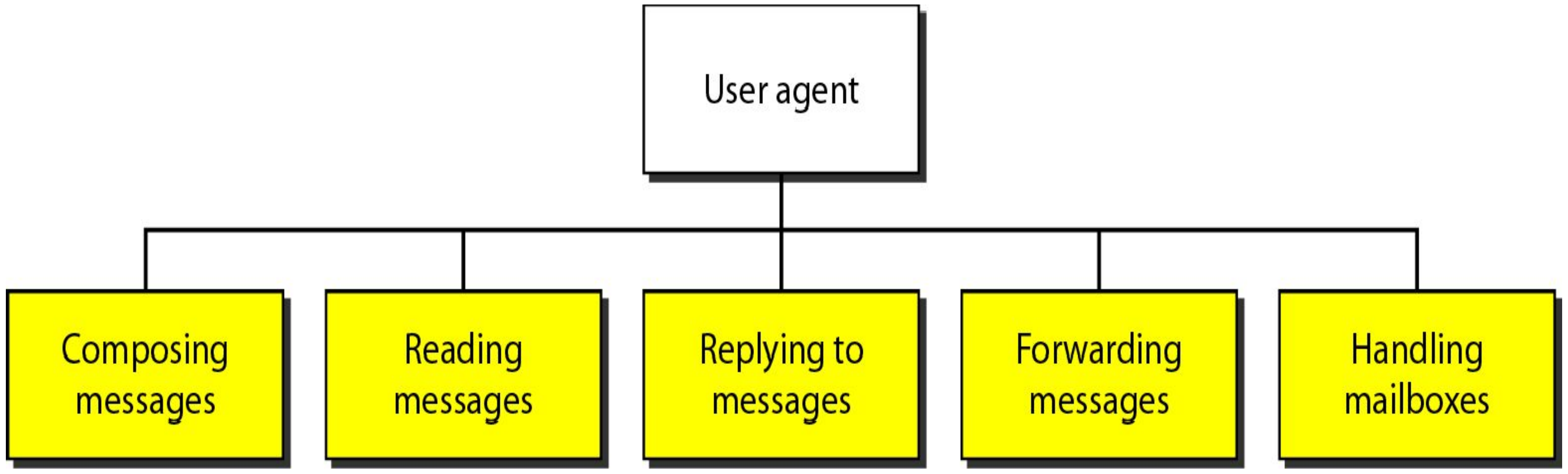

Local part
@
Domain name
Address of the mailbox on the mail server
The domain name of the mail server

Architecture



User Agent

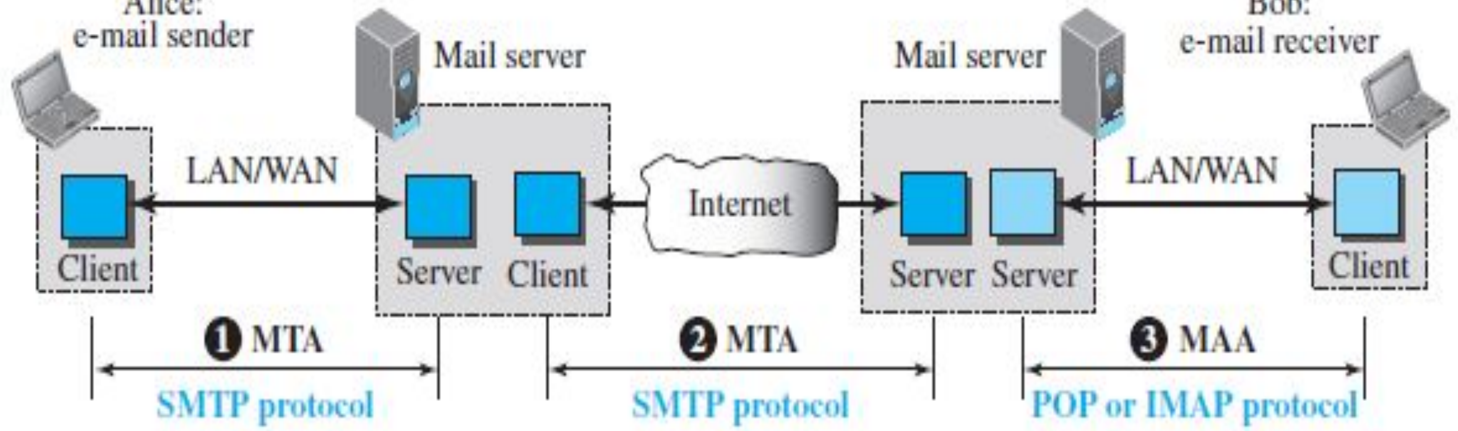
- User agents allow users to read, reply to, forward, save, and compose messages.
- Examples of user agents for e-mail include Microsoft Outlook, Apple Mail, Web based Gmail, the Gmail App running in a smartphone, and so on.



Mail Server

- Mail servers form the core of the e-mail infrastructure.
- A typical message starts its journey in the sender’s user agent, then travels to the sender’s mail server, and then travels to the recipient’s mail server, where it is deposited in the recipient’s mailbox.

Message Transfer Agent: SMTP



Message Transfer Agent: SMTP

- SMTP (Simple Mail Transfer Protocol)
- Used to send emails from a sender’s email to the mail server
- To forward emails between mail servers
- NOT used for receiving emails
- It is called as Push protocol (initiates email sending)

Message Transfer Agent: SMTP

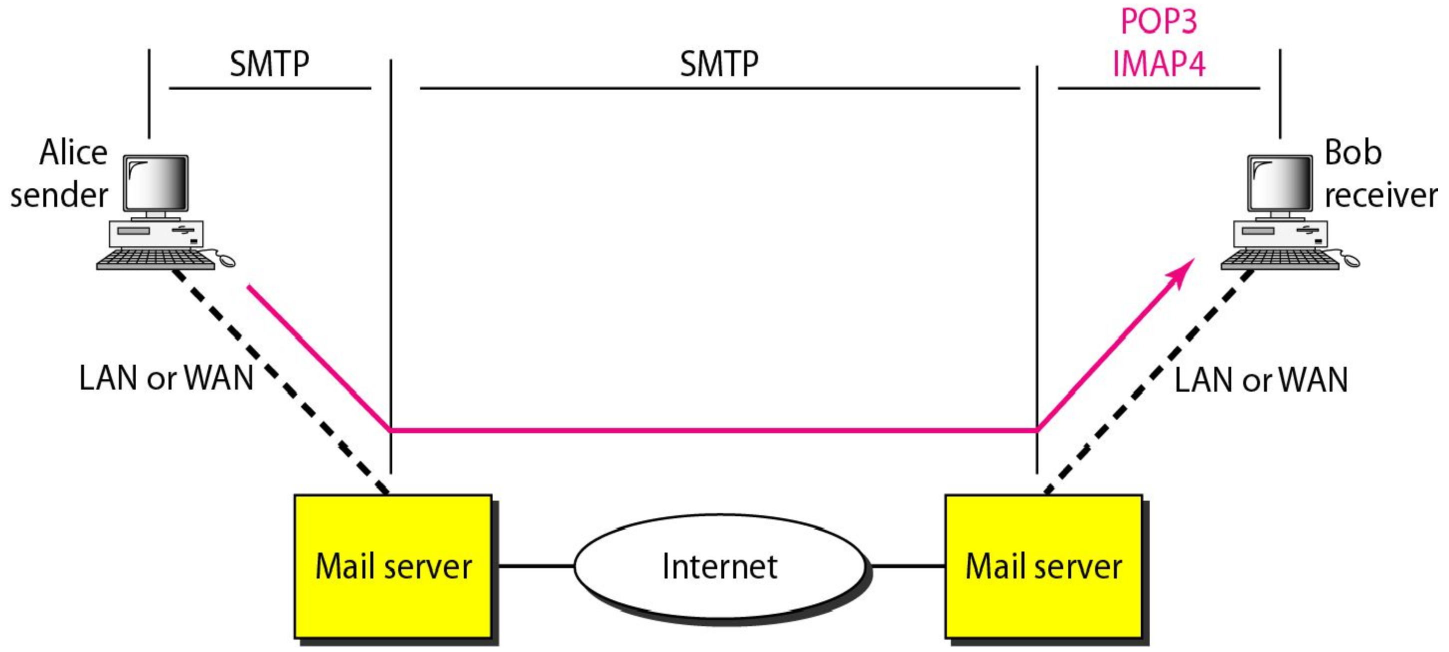
Example:

Suppose Alice wants to send Bob a simple ASCII message.

1. Alice invokes her user agent for e-mail, provides Bob’s e-mail address (for example, bob@someschool.edu), composes a message, and instructs the user agent to send the message.
2. Alice’s user agent sends the message to her mail server, where it is placed in a message queue.
3. The client side of SMTP, running on Alice’s mail server, sees the message in the message queue. It opens a TCP connection to an SMTP server, running on Bob’s mail server.
4. After some initial SMTP handshaking, the SMTP client sends Alice’s message into the TCP connection.
5. At Bob’s mail server, the server side of SMTP receives the message. Bob’s mail server then places the message in Bob’s mailbox.
6. Bob invokes his user agent to read the message at his convenience.

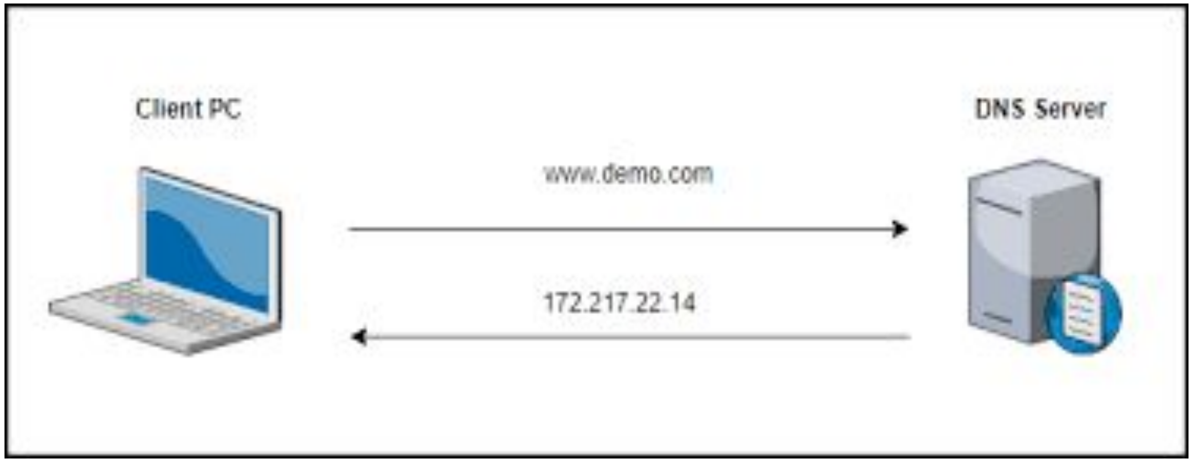
POP3 and IMAP4

- **POP3**: Post Office Protocol, version 3
- **IMAP4**: Internet Mail Access Protocol, version 4
- POP3 and IMAP4 are called as mail access protocols used at receiver side to retrieve emails from a mail server



DNS(Domain Name System)

- IP addresses are used to uniquely identify a system in the network
- However remembering IP address of every machine is not possible
- For that purpose names are used to identify machines instead of IP address
- These names are called as Host Name
- Mapping between Host name to IP address is done with the help of Domain Name System (DNS) Protocol



- In above diagram www.demo.com is the host name which is been converted to IP address 172.217.22.14 by the DNS server

DNS(contd..)

DNS server provides the following services to the user,

1<u>. Mapping Hostname to IP Address</u>

<u>2. Host aliasing</u>

- A host with a complicated or lengthy hostname can have one or more alternative(alias) short names
- That complicated or lengthy host name is called as canonical hostname
- User uses the alias name(short names)
- DNS can be used to obtain canonical hostname for the given alias name

<u>Ex:</u>

- relay1.west-coast.enterprise.com is the canonical hostname which have alias names like enterprise.com, www.enterprice.com

DNS(contd..)

DNS server provides the following services to the user,

3. Mail server aliasing

- Similar to servers having alias names, Mail servers also have alias name
- DNS can be used to obtain canonical name for the given alias name

Ex:

- An Email id with Hotmail mail server is alice@hotmail.com. But the mail server has a canonical hostname like relay1.west-coast.hotmail.com

4. Load Distribution

- Many reputed servers (like Google server) have multiple server machines each having one IP address. But the hostname for all the servers is same(Google.com)
- When client request with DNS servers for Google.com, the DNS server is going to return the IP addresses of all servers having the hostname Google.com

DNS(contd..)

DNS server provides the following services to the user,

<u>4. Load Distribution (contd..)</u>

- Since client takes only the 1st IP address from that response message, each time DNS servers rotates the IP address of Google server in order to maintain the load to all servers

DNS(contd..)

- If there is a single centralized DNS server, then the following problems will arise,

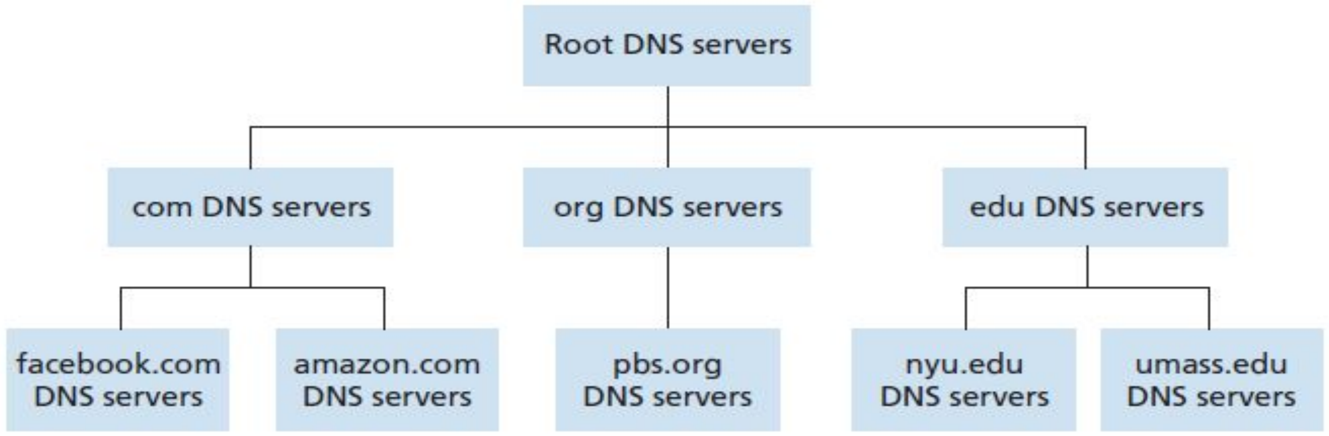
1. A single point of failure
2. All Traffic volume to the same server
3. Distant centralized database (DNS server in US and client in Australia)
4. Maintenance (Updating all data in same server is a difficult and time taking process)

- In order to overcome from above problem there is no single centralized DNS server, instead the DNS server is distributed and is in hierarchical manner

DNS(contd..)

A Distributed, Hierarchical DNS servers

- DNS servers are distributed across world and are present in hierarchical manner. There are 3 levels of DNS servers,
  1. Root DNS servers
  2. Top-level Domain(TLD) servers
  3. Authoritative DNS servers



- When user requests for IP address of www.amazon.com, request first forwarded to root DNS server which gives IP address of .com TLD server. .com TLD server gives IP address of amazom.com authoritative DNS servers. Finally amazon.com authoritative DNS server gives IP address for ww.amazon.com to client

DNS(contd..)

<u>Root DNS servers</u>

- This DNS server gives IP address of Top-level domain servers

<u>Top-level Domain(TLD) DNS servers</u>

- These servers are responsible for top-level domains like com, org, net, edu, gov and country level domains like uk, in, fr etc
- TLD servers are responsible for giving IP addresses of authoritative DNS servers

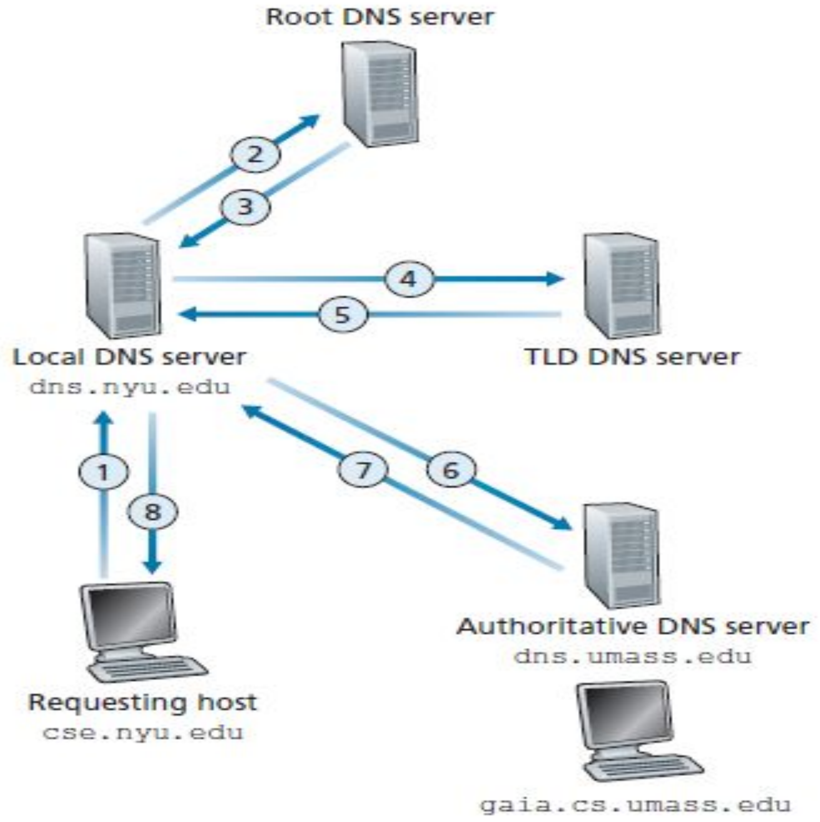
<u>Authoritative DNS servers</u>

- This authoritative servers are responsible for giving IP address for the hostname given by the user

- Apart from above servers there is Local DNS servers which is maintained nearer to the client network. All requests from client are first given to local DNS server and then only forwarded to other DNS servers of required

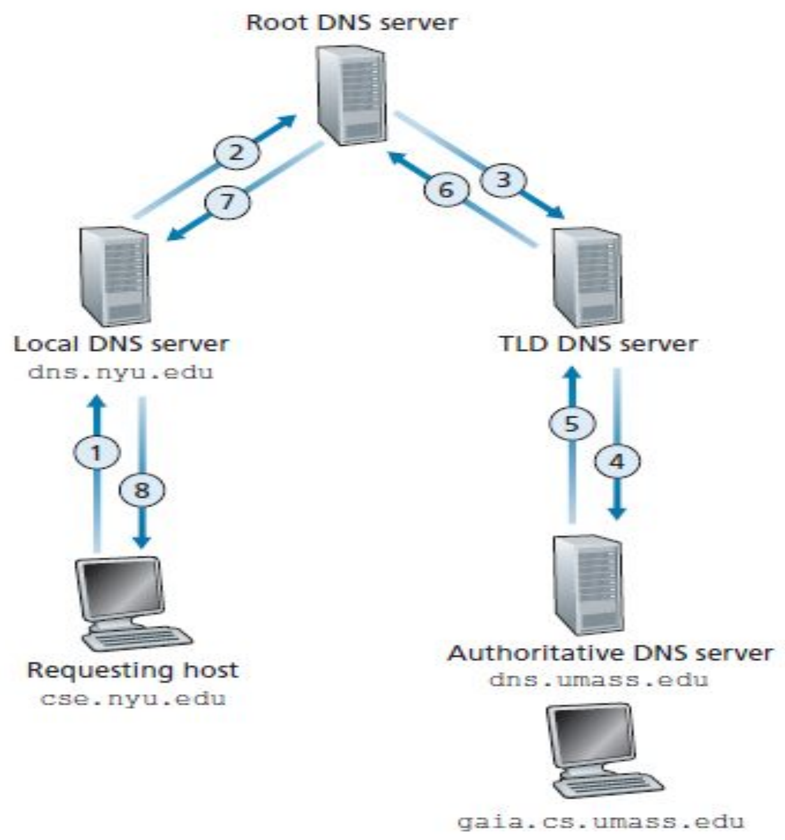
DNS(contd..)

Interaction between various DNS servers (Iterative manner)



DNS(contd..)

<u>Interaction between various DNS servers (Recursive manner)</u>



DNS(contd..)

<u>DNS caching</u>

- Like web documents caching, DNS servers also do caching in order to avoid delay and reduce DNS messages travelling over network
- When a DNS server receives response from any DNS servers in its chain, it does caching in its memory
- For example Local DNS server may cache the response from any DNS servers and when client requests the IP address for same hostname it will be given by Local DNS server instead of requesting to root DNS servers

DNS(contd..)

DNS Records

- Every DNS server maintains data in the form of Resource Records(RR)
- Every Resource record have following 4 fields:

  (Name, Value, Type , TTL)
- TTL(Time To Live field represents how long the resource record is valid)
- There are 4 types of resource records:

Type = A:

- The resource record of this type contains Hostname in Name field and its corresponding IP Address in Value field

Type = NS:

- The resource record of this type contains Domain name (Ex:foo.com) in Name field and name of Authoritative server in Value field

DNS(contd..)

DNS Records

Type = CNAME:

- The resource record of this type contains Canonical hostname in Name field and alias name in Value field

Type = MX:

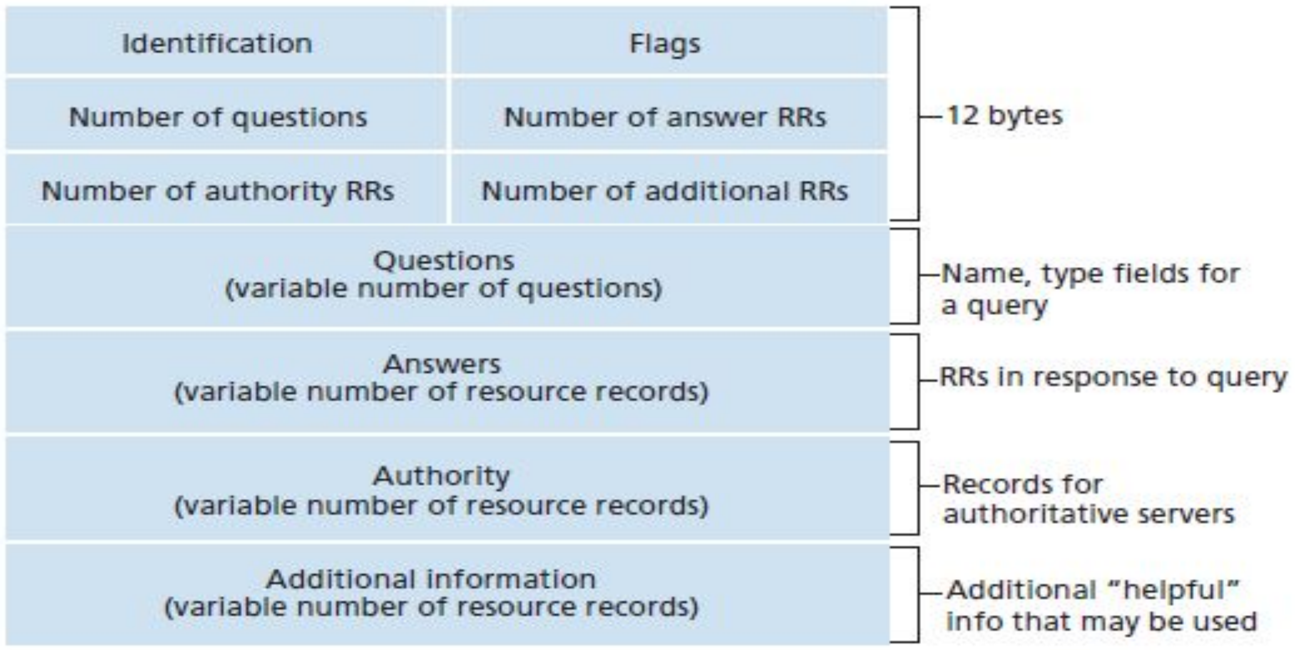
- The resource record of this type contains Canonical hostname of mail server in Name field and alias name in Value field

- Authoritative DNS servers contain Type A records
- Other DNS servers contains Type NS and Type A records

DNS(contd..)

DNS Messages Format

- DNS Query (Request) and Response message both have the same format as shown in below diagram
- DNS messages uses UDP as the transport layer protocol



Identification
Flags
Number of questions
Number of answer RRs
Number of authority RRs
Number of additional RRs
12 bytes
Questions (variable number of questions)
Name, type fields for a query
Answers (variable number of resource records)
RRs in response to query
Authority (variable number of resource records)
Records for authoritative servers
Additional information (variable number of resource records)
Additional "helpful" info that may be used

DNS(contd..)

<u>DNS Messages Format</u>

<u>Identification field</u>

- This field is used to identify the message

<u>Flag field</u>

- It totally contains 16 bits in which
  - 1 bit represents its Query/Response message
    - 1 bit in flag is set if reply comes from Authoritative server
    - 1 bit represents Recursion-desired
    - 1 bit represents Recursion-available
- Below flag field represents number of records in questions, answers sections and number of records from other authoritative servers and number of additional resource records

DNS(contd..)

DNS Messages Format

Question section

- This field contains the request that is been made by the client.
- It contains Name, Type fields. Name field contains Hostname and Type field contains Type of record being asked

Answer section

- This field contains the response for the request that is been made by the client.

Authoritative section

- This field contains the records from other authoritative servers

Additional section

- This field contains other helpful records

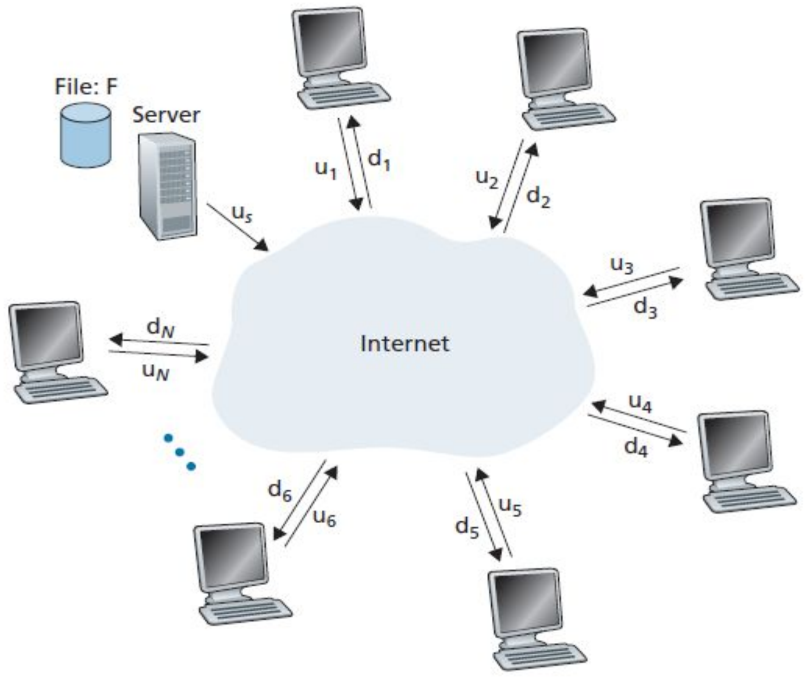
Peer to Peer File Distribution

- Instead of client-server architectures which always relies on always-on servers, in a P2P architecture, there is minimal (or no) reliance on always-on servers
- Pairs of connected hosts, called peers, communicate directly with each other
- In P2P file distribution, each peer can redistribute any portion of the file it has received to any other peers
- The most popular P2P file distribution protocol is BitTorrent

Peer to Peer File Distribution (contd..)

Scalability of P2P architectures

- To compare client-server architectures with peer-to-peer architectures consider a simple case of distributing a file to a fixed set of peers in both architectures



In above diagrai

u_s - upload rate of the server's access link

u_i - upload rate of the ith peer's access link

d_i - download rate of the ith peer's access link

F- Size of the file to be distributed (in bits)

N - number of peers

Peer to Peer File Distribution (contd..)

Scalability of P2P architectures

- The distribution time for the client-server architecture is denoted by $D_{cs.}$ The distribution time is the time it takes to get a copy of the file to all *N* peers.

$$D_{cs} = \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\}$$

- The server must transmit one copy of the file to each of the N peers. Thus, the server must transmit NF bits. Since the server's upload rate is u_s, the time to distribute the file must be atleast NF/ u_s
- Let d_{min} denote the download rate of the peer with the lowest download rate.

$$d_{min} = \min\{d_1, d_p, \ldots, d_N\}.$$

- The peer with the lowest download rate cannot obtain all F bits of the file in less than F/d_{min} seconds. Thus, the minimum distribution time is at least F/d_{min}.

Peer to Peer File Distribution (contd..)

Scalability of P2P architectures

- The distribution time for the peer-to-peer architecture is denoted by $D_{p2p.}$ The distribution time is the time it takes to get a copy of the file to all N peers.

$$D_{\mathrm{P2P}} = \max\left\{ \frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i} \right\}$$

- Initially only the server has the file. To get this file into the community of peers, the server must send each bit of the file at least once into its access link. Thus, the 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. Thus, the minimum distribution time is at least F/d_{min}
- Finally, 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

$$u_{total} = u_s + u_1 + \ldots + u_N.$$

- The system must deliver (upload) F bits to each of the N peers, thus delivering a total of NF bits. This cannot be done at a rate faster than u_{total}. Thus, the minimum distribution time is also at least $NF/(u_s + u_1 + \ldots + u_N)$.

Peer to Peer File Distribution (contd..)

Scalability of P2P architectures

- Below graph compares the minimum distribution time for the client-server and P2P architectures
- For the client-server architecture, the distribution time increases linearly and without bound as the number of peers increases

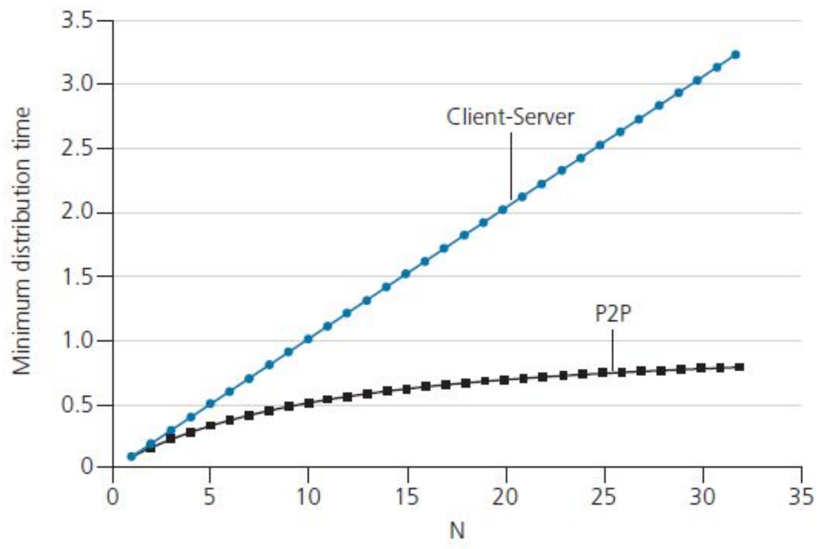


Figure 2.23 ♦ Distribution time for P2P and client-server architectures

Peer to Peer File Distribution (contd..)

Bit Torrent

- BitTorrent is a popular P2P protocol for file distribution
- In BitTorrent, the 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, with a typical chunk size of 256 KBytes.
- When a peer first joins a torrent, it has no chunks. Over time it accumulates more and more chunks
- While it downloads chunks it also uploads chunks to other peers. Once a peer has acquired the entire file, it may leave the torrent, or remain in the torrent and continue to upload chunks to other peers
- Each torrent has an infrastructure node called a tracker.
- When a peer joins a torrent, it registers itself with the tracker and periodically informs the tracker that it is still in the torrent
- Given torrent may have fewer than ten or more than a thousand peers participating at any time
- Diagram shown in next slide

Peer to Peer File Distribution (contd..)

Bit Torrent

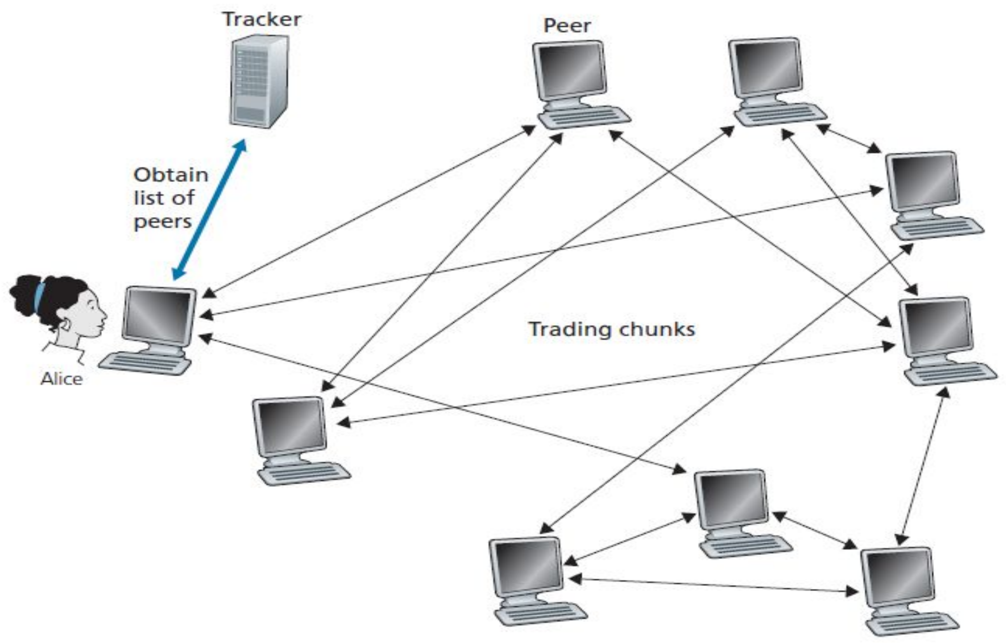


Figure 2.24 ♦ File distribution with BitTorrent

- When a new peer, Alice, joins the torrent, the tracker randomly selects a subset of peers and sends the IP addresses of those peers to Alice
- Alice attempts to establish concurrent TCP connections with all the peers on this list

Video Streaming and Content Distribution Networks

- Many streaming video including Netflix, YouTube and Amazon Prime account for about 80% of Internet traffic
- How popular video streaming services are implemented in today's Internet is discussed here

Internet Video

- In streaming stored video applications, the underlying medium is prerecorded video such as movie, a television show, a prerecorded user-generated video (Videos seen on YouTube)
- These prerecorded videos are placed on servers, and users send requests to the servers to view the videos on demand
- Video is a sequence of images, typically being displayed at a constant rate, for example, at 24 or 30 images per second
- One of the characteristic of video is that it can be compressed.
- Compressed Internet video typically ranges from 100 kbps for low-quality video to over 4 Mbps for streaming high-definition movies

Video Streaming and Content Distribution Networks (contd..)

HTTP Streaming and DASH

- In HTTP streaming, the video is simply stored at an HTTP server.
- The client establishes a TCP connection with the server and issues an HTTP GET request and the server then sends the video file, within an HTTP response message
- The major disadvantage of above HTTP streaming is that all clients receive the same encoding of the video, despite the large variations in the amount of bandwidth available to a client, both across different clients and also over time for the same client
- To overcome above problem new type of HTTP streaming referred as Dynamic Adaptive Streaming over HTTP (DASH) is developed.

Video Streaming and Content Distribution Networks (contd..)

HTTP Streaming and DASH

- In DASH, the video is encoded into several different versions, with each version having a different bit rate
- The 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 a high-rate version; and when the available bandwidth is low, it naturally selects from a low-rate version.
- The client selects different chunks one at a time with HTTP GET request messages
- DASH allows clients with different Internet access rates to stream in video at different encoding rates.
- Clients with low-speed 3G connections can receive a low bit-rate (and low-quality) version, and clients with fiber connections can receive a high-quality version.
- DASH also allows a client to adapt to the available bandwidth if the available end-to-end bandwidth changes during the session

Video Streaming and Content Distribution Networks (contd..)

Content Distribution Networks (CDN)

- Internet video companies are distributing on-demand multi-Mbps streams to millions of users on a daily basis
- <u>**Example**</u>

YouTube with a library of hundreds of millions of videos, distributes hundreds of millions of video streams to users around the world every day

- In order to meet the challenge of distributing massive amounts of video data to users distributed around the world, all major video-streaming companies make use of Content Distribution Networks (CDNs).
- A CDN manages servers in multiple geographically distributed locations, stores copies of the videos in its servers, and attempts to direct each user request to a CDN location that will provide the best user experience

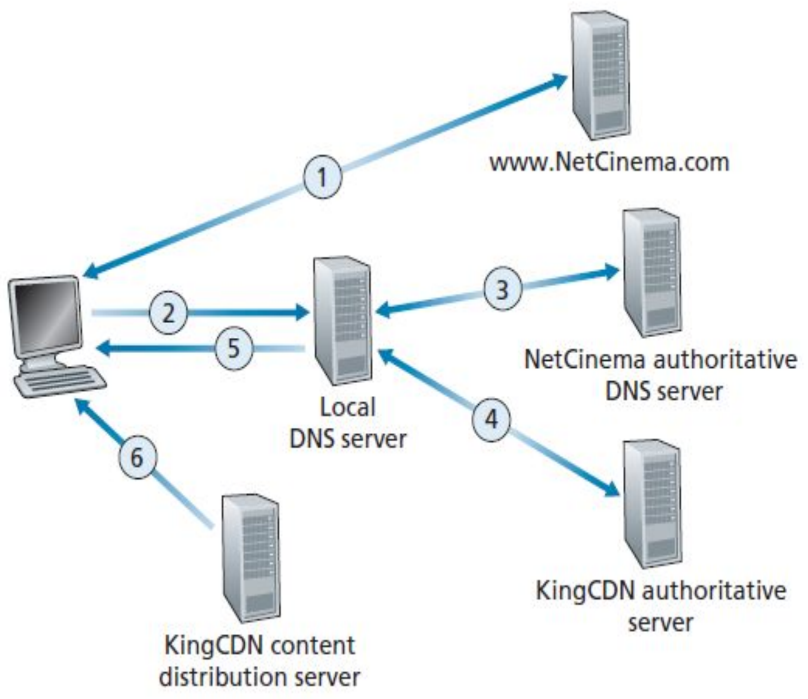
Video Streaming and Content Distribution Networks(contd..)

CDN Operation

- When a browser in a user's host is instructed to retrieve a specific video, CDNs take advantage of DNS to intercept and redirect requests

Example:

- Suppose a content provider, NetCinema, employs the third-party CDN company, KingCDN, to distribute its videos to its customers. On the NetCinema Web pages, each of its videos is assigned a URL that includes the string "video" and a unique identifier for the video itself



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

Figure 2.25 ♦ DNS redirects a user's request to a CDN server

Video Streaming and Content Distribution Networks (contd..)

CDN Operation

1. The user visits the Web page at NetCinema
2. When the user clicks on the link http://video.netcinema.com/6Y7B23V, the user’s host sends a DNS query for video.netcinema.com
3. The user’s Local DNS Server (LDNS) relays the DNS query to an authoritative DNS server for NetCinema, which observes the string “video” in the hostname video.netcinema.com. To “hand over” the DNS 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, for example a1105.kingcdn.com.
4. From this point on, the DNS query enters into KingCDN’s private DNS infrastructure. The 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. It is thus here, within the KingCDN’s DNS system, that the CDN server from which the client will receive its content is specified

Video Streaming and Content Distribution Networks (contd..)

CDN Operation

5. The LDNS forwards the IP address of the content-serving CDN node to the user's host.
6. 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 and issues an HTTP GET request for the video. If DASH is used, the server will first send to the client a manifest file with a list of URLs, one for each version of the video, and the client will dynamically select chunks from the different versions.

Video Streaming and Content Distribution Networks (contd..)

Cluster Selection strategies

- cluster selection strategy is a mechanism for dynamically directing clients to a server cluster or a data center within the CDN
- simple strategy is to assign the client to the cluster that is geographically closest

Socket Programming using TCP and UDP

Refer Lab Program