UNIT-2

Application Layer

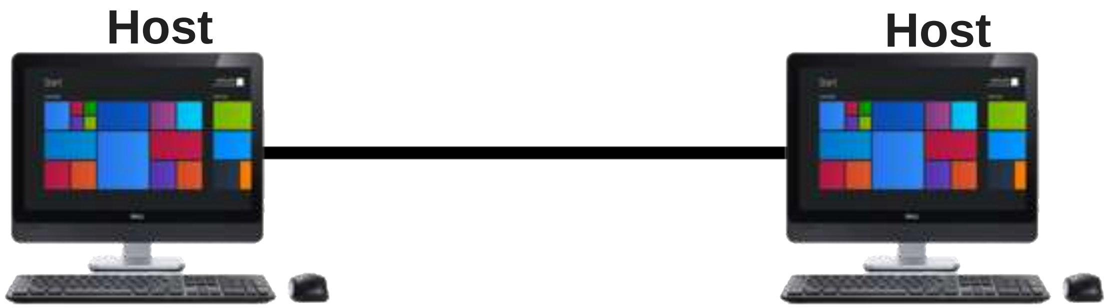
Outline

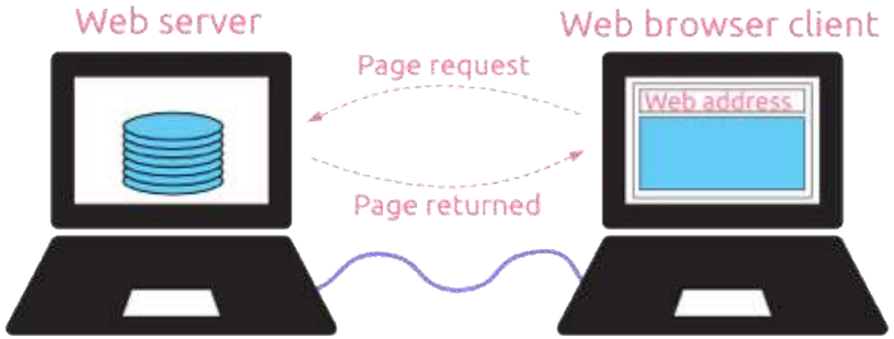
- Principles of Computer Applications
- Web
- HTTP
- E-mail
- DNS
- Socket programming with TCP and UDP

Network Applications

Network Applications

- A Network application is an application running on one host and provides a communication to another application running on a different host.
- A network application development is writing programs that run on different end systems and communicate with each other over the network.
- In the Web application there are two different programs that communicate with each other:
  - Browser program running in the user's host.
  - Web server program running in the Web server host.





Network Applications - Examples

- Email
- Web
- Remote Login
- P2P File Sharing
- Multi-user Network Games
- Streaming Stored Video (YouTube)
- Voice Over IP (Skype)
- Real-time Video Conference
- Social Networking

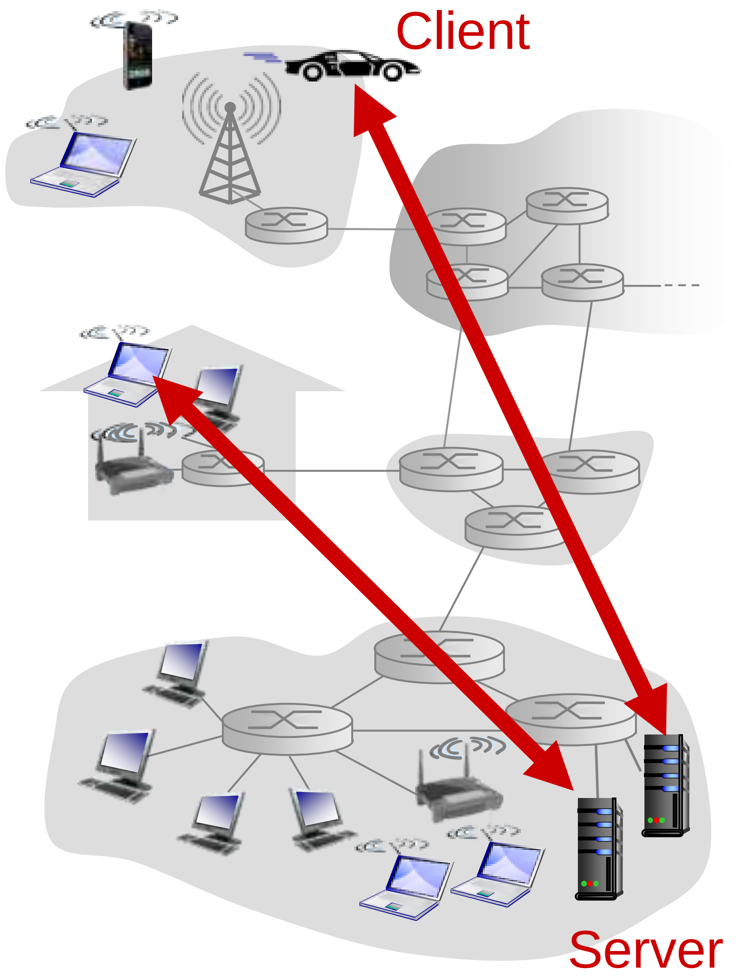
BearShare
FrostWire
Azureus
iMesh
BitTorrent™
eMule

Network Application Architecture

- Client-Server architecture
- P2P (Peer to Peer) architecture

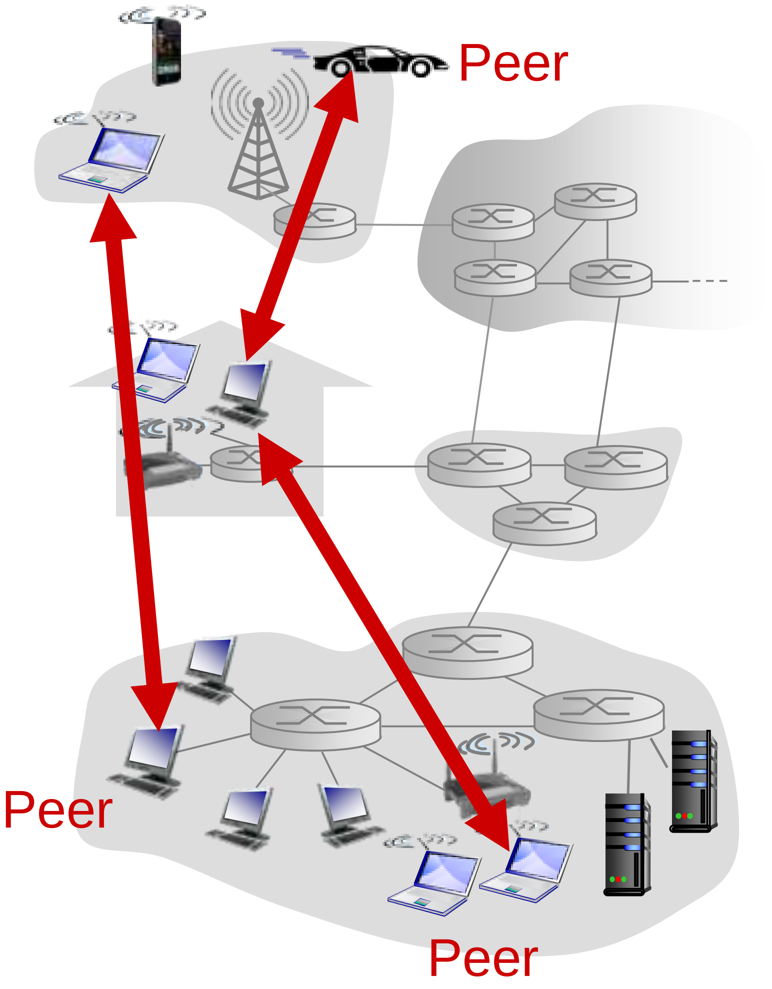
Client-Server Architecture

- Server:
  - Its always-on host.
  - It has a fixed IP address.
  - Large cluster of host – Data Centers.
  - E.g. Web Server
- Client:
  - It communicate with server.
  - Its not like continuously connected.
  - May have dynamic IP addresses.
  - Do not communicate directly with each other.
  - E.g. PCs, Mobiles



2. P2P Architecture

- Peers (end systems) directly communicate.
- Get peers request service from other peers, provide service to other peers.
  - ➥ Self Scalability – New peers bring new service capacity, as well as new service demands.
- Peers are alternatingly connected and change IP addresses.
  - ➥ Complex management



Process Communicating

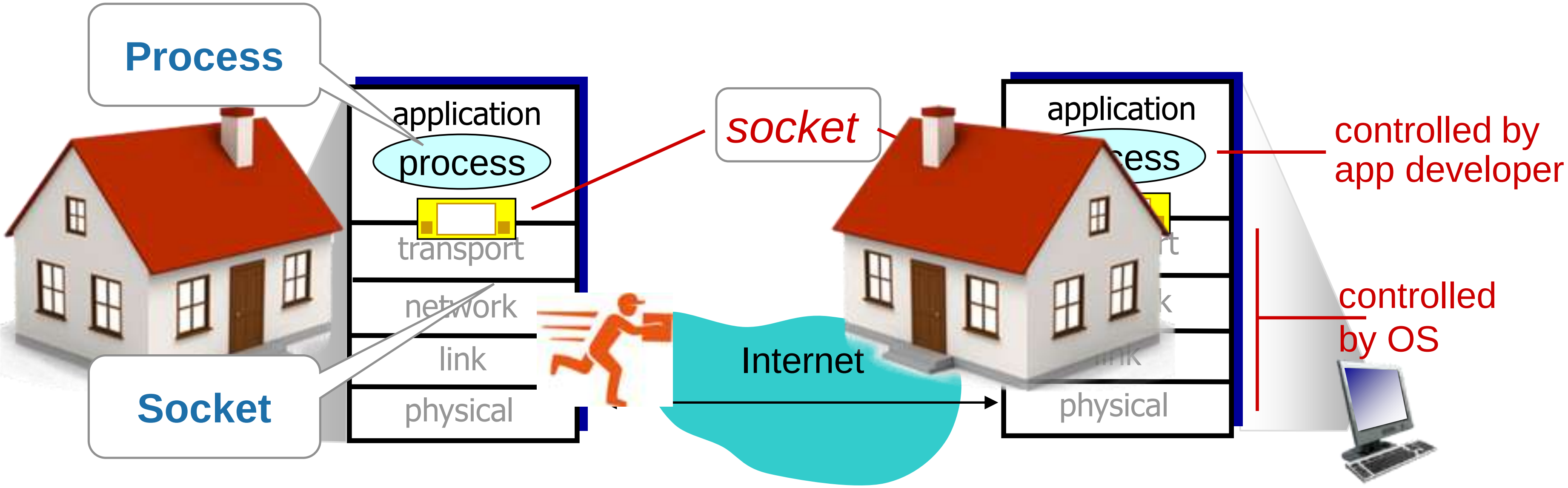
- What is Process?
  - A process is an instance of a program running in a computer.
- We can say that process is program under execution.
- Within same host, two processes communicate using inter-process communication (IPC).
- Process in different hosts communicate by exchanging messages.
- Client process: A process that initiates communication.
- Server process: A process that waits to be contacted.

≡ File Edit Search Run Compile Debug Project Options Window Help
[■] FIRST.C 1-[↕]
4:19
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

```
#include<stdio.h>
#include<conio.h>
void main()
{
        clrscr();
        printf("Hello World...!");
        getch();
}
```

Socket

- A process sends messages into and receives messages from; the network through a software interface called a socket.
- A process is like a house and its socket is like its door.
  - Sending process passes message outdoor.
  - Sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process.



Transport Services to Applications

- Recall that a socket is the interface between the application process and the transport layer protocol.
- For develop an application, choose available transport layer protocol.
- Pick the protocol with the services that best match the needs of your application.
- Example: Choose either Train or Airplane transport for travel between two cities.
- Classify services with four parameters:

Reliable Data Transfer

Throughput

Timing

Security

Transport Services to Applications

- Reliable Data Transfer:
  - Many applications (e.g., email, file transfer, financial applications) require 100% reliable data transfer
  - Required guarantee that data sent by one end of application is delivered correctly and completely to the other end of application.
  - This guaranteed data delivery service is called Reliable Data Transfer.
  - When it will fail to deliver reliable data transfer, it is acceptable for loss-tolerant applications.
  - Loss-tolerant Applications (e.g., audio/video) can tolerate some loss.

Transport Services to Applications

- Throughput
  - Some apps (e.g., multimedia) require at least amount of throughput to be “effective”
  - Bandwidth sensitive application, specific throughput required.
  - Elastic application can use of as much, or as little, throughput as happens to be available.
- Timing
  - some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”
- Security
  - In the sending host, encrypt all data transmitted by the sending process.
  - In the receiving host, decrypt the data before delivering the data to the receiving process.

Internet Transport Protocols Services

TCP Service:

- Connection-Oriented: A setup required between client and server processes
- Reliable data transfer between sending and receiving process without error and proper order
- Congestion control: To control sender when network overloaded
- It does not provide, Timing, at least throughput guarantee (not preferred in real-time application)

UDP Services:

- Connectionless: No connection before two processes start to communicate.
- Unreliable data transfer between sending and receiving process
- It does not provide congestion control.
- It Does not provide. Reliability, flow control, throughput guarantee, security.

Internet Applications

- Popular internet applications with their application layer and their underlying transport protocol.

Applications	Application-Layer Protocol	Underlying Transport Protocol (Service)
Email	SMTP	TCP
Remote Terminal	Telnet	TCP
Web	HTTP	TCP
File Transfer	FTP	TCP
Streaming Media	HTTP(YouTube), RTP	TCP or UDP
Internet Telephony	SIP, RTP(Skype)	Typically UDP

Loss-tolerant

No loss, Elastic Bandwidth

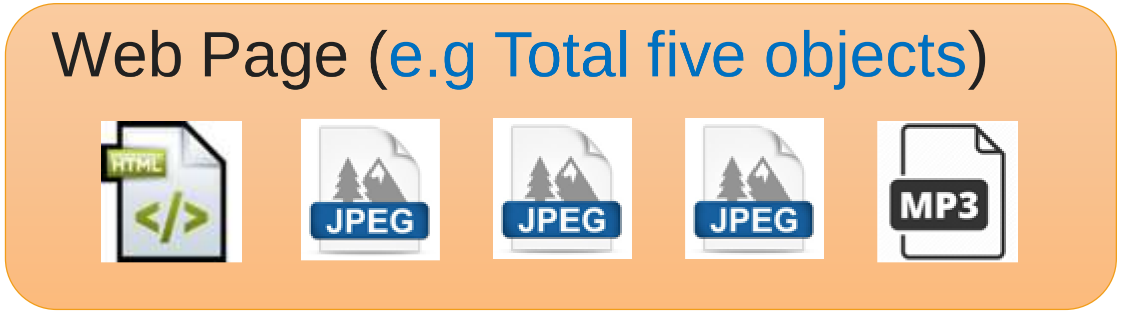
Web & HTTP

Web

- Early 1990, Internet was used only by researchers, academics, and university students.
- New application WWW arrived in 1994 by Tim Berners-Lee.
- World Wide Web - is an information where documents and other web resources are identified by URL, interlinked by hypertext links, and can be accessed via the Internet.
- On demand available, What they want, When they want it.
- Unlike TV and Radio.
- Navigate through Websites.

Web and HTTP

- Web page consists of objects.
- Object can be HTML file, JPEG image, Java applet, audio file etc.…
- Web page consists of base HTML-file which includes several referenced objects.



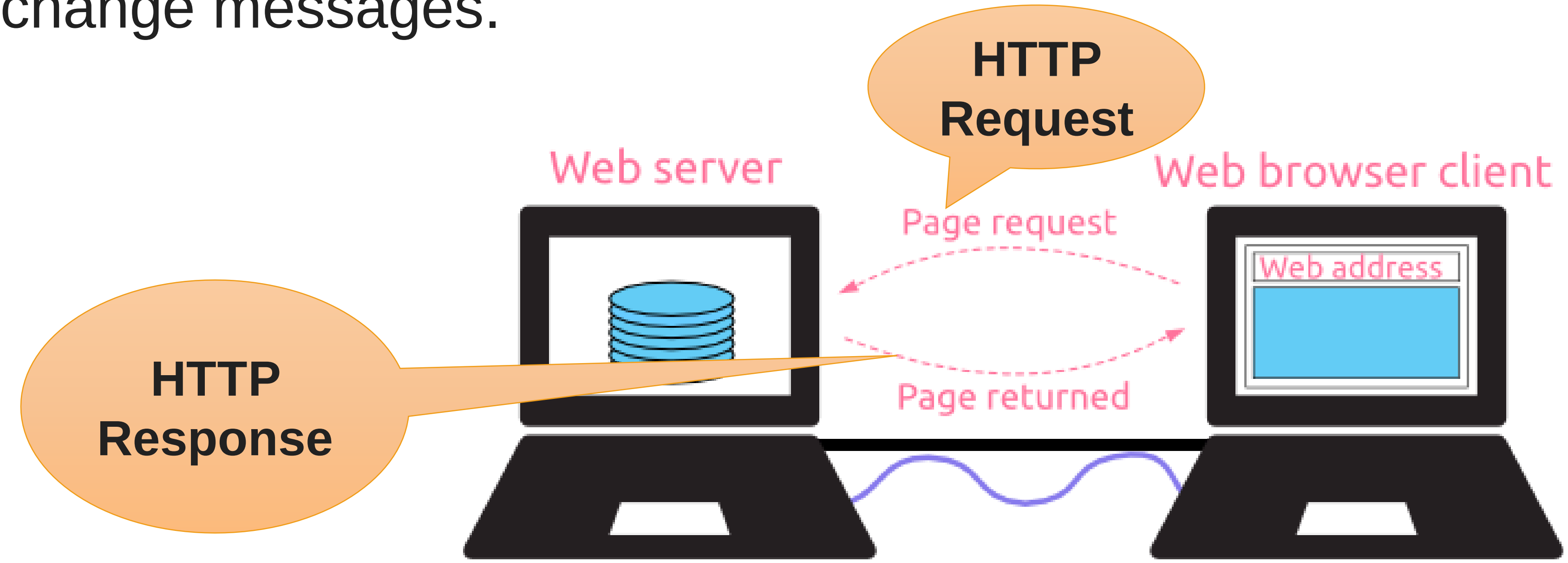
Web Page (e.g Total five objects)

- Each object is addressable by a Uniform Resource Locator (URL), like;

$$\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\underbrace{\texttt{/someDept/pic.gif}}_{\text{path name}}$$

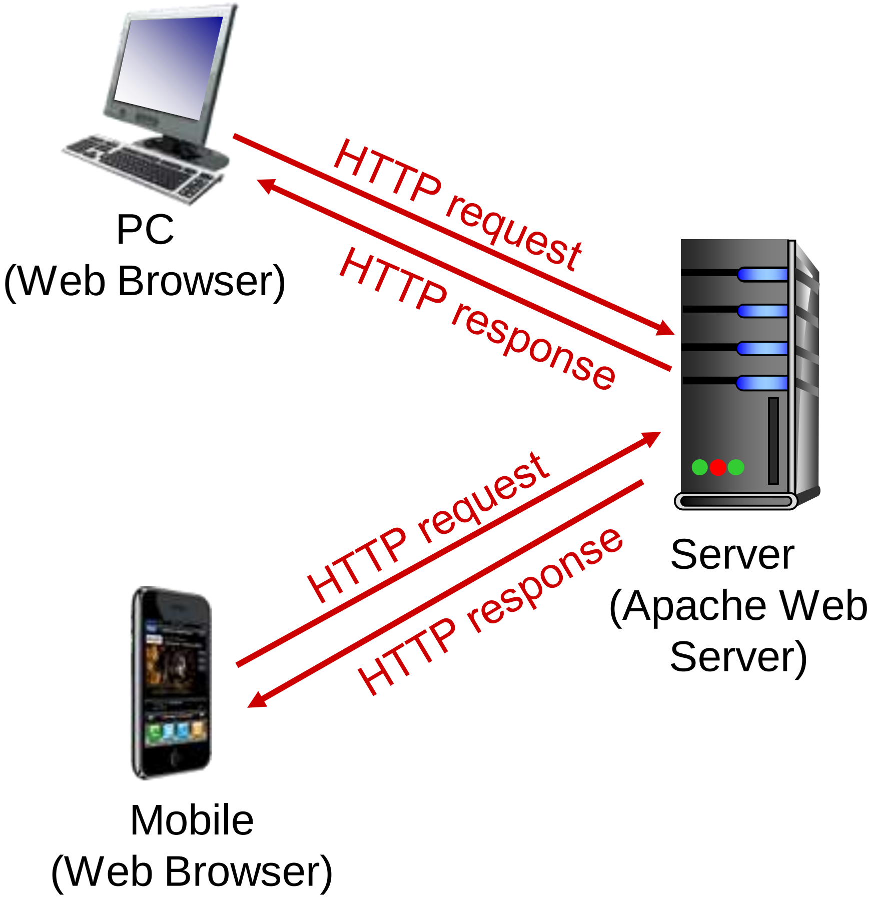
HTTP

- HyperText Transfer Protocol – Application layer protocol
- It is implemented in two programs.
  - Client Program
  - Server Program
- Exchanging HTTP message each others.
- HTTP defines the structure of these messages and how web client – web server exchange messages.



HTTP – Cont…

- HTTP
  - Hyper-Text Transfer Protocol
  - It is Application layer protocol
  - Client: A browser that requests, receives, (using HTTP protocol) and “displays” Web objects.
  - E.g. PC, Mobile
  - Server: Web server sends (using HTTP protocol) objects in response to requests.
  - E.g. Apache Web Server



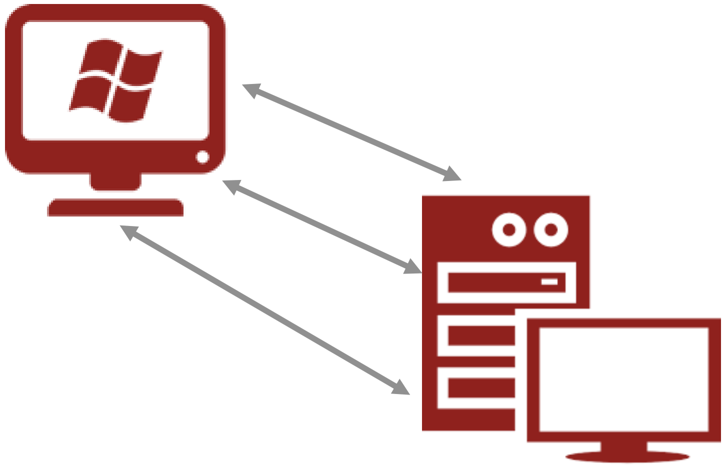
HTTP - Cont...

- A client initiates TCP connection (creates socket) to server using port 80.
- A server accepts TCP connection from client.
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server).
- HTTP is “stateless protocol”, server maintains no information about past client requests.
- HTTP connection types are:
  - Non-persistent HTTP
  - Persistent HTTP

Non-persistent HTTP & Persistent HTTP Connection

Non-persistent & Persistent Connection

- In Client-Server communication, Client making a series of requests to server, Server responding to each of the requests.
- Series of requests may be made back-to-back or periodically at regular time interval.
- So, Application developer need to make an important decision;
  - Should each request/response pair be sent over a separate TCP connection.
  - OR should all the requests and corresponding responses be sent over same TCP connection?

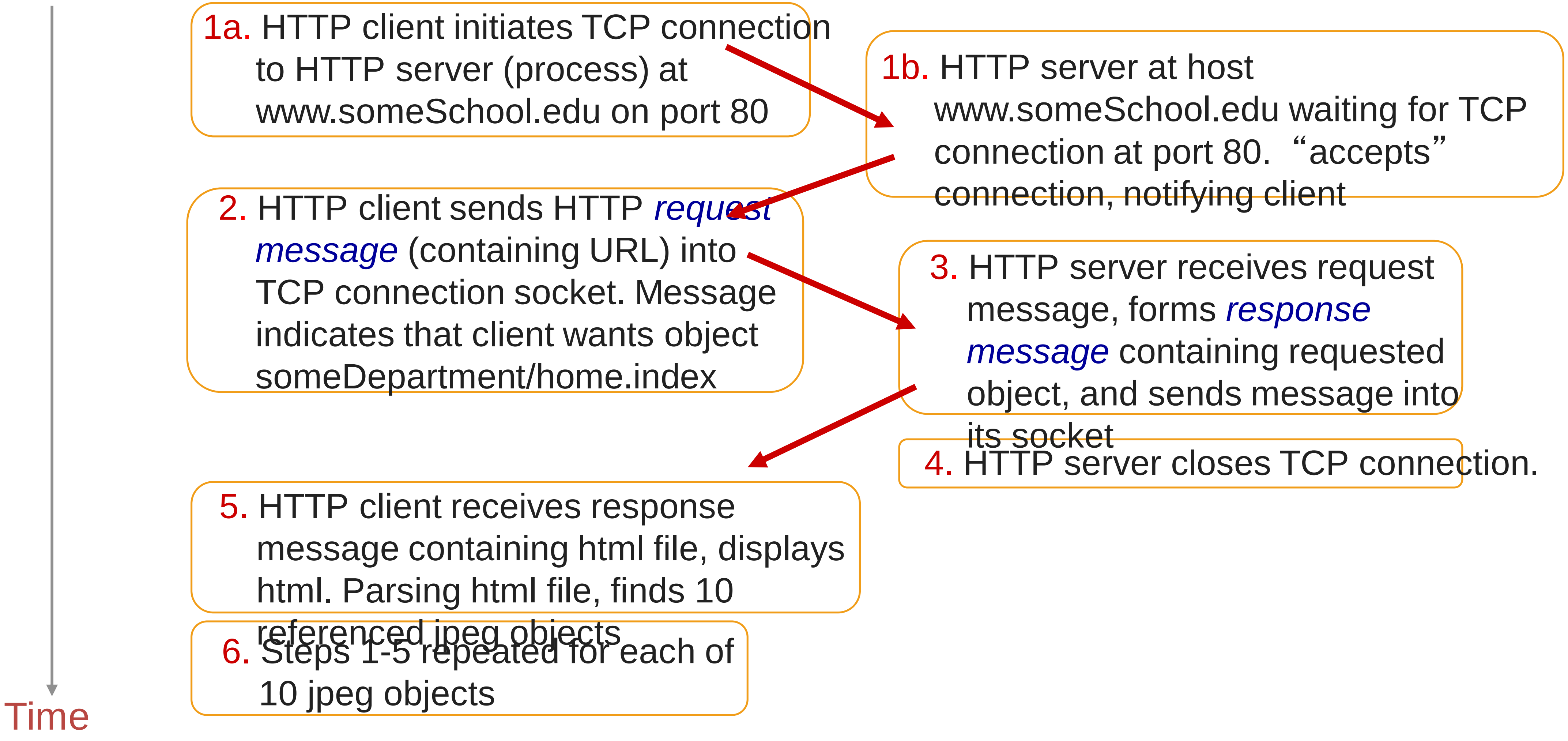


Non-persistent HTTP

- A non-persistent connection is closed after the server sends the requested object to the client.
- The connection is used exactly for one request and one response.
- For downloading multiple objects, it required multiple connections.
- Non-persistent connections are the default mode for HTTP/1.0.
- Example:
  - ↪ Transferring a webpage from server to client, webpage consists of a base HTML file and 10 JPEG images.
- Total 11 object are residing on server.

Non-persistent HTTP – Cont.…

- URL **`www.someSchool.edu/someDepartment/home.index`**



1a. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. “accepts” connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

Non-persistent HTTP: Response time

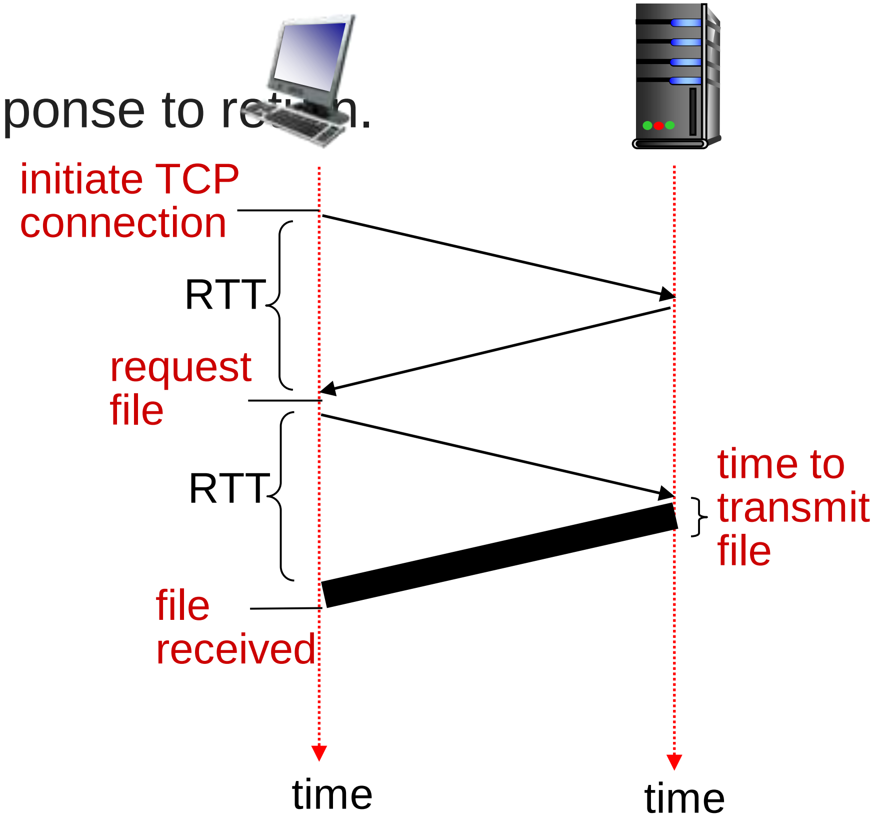
- RTT(round-trip time): A time for a small packet to travel from client to server and vice versa.
- HTTP response time:
  - One RTT to initiate TCP connection.
  - One RTT for HTTP request and first few bytes of HTTP response to return.
  - File transmission time

Non-persistent HTTP response time =

2RTT

+

file transmission time



Persistent HTTP

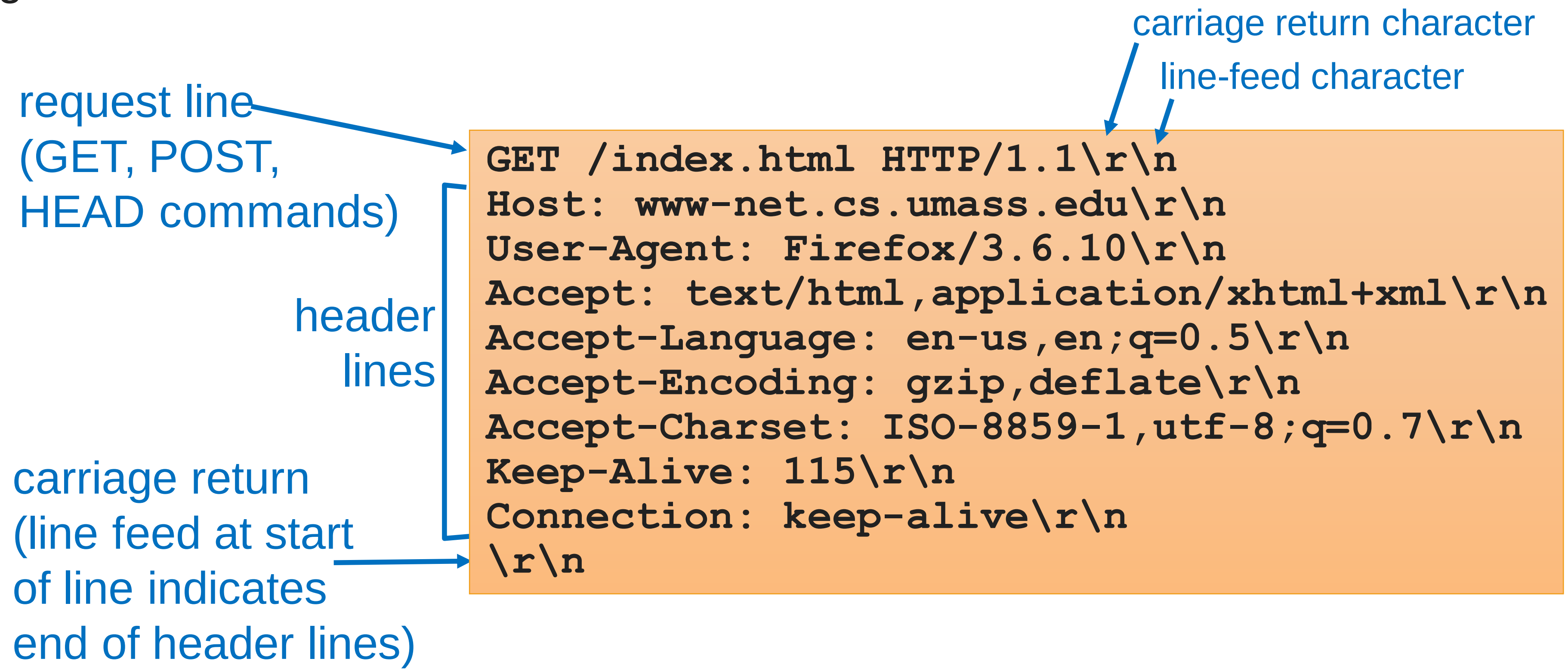
- Server leaves the TCP connection open after sending responses.
- Subsequent HTTP messages between same client and server sent over open connection.
- The server closes the connection only when it is not used for a certain configurable amount of time.
- It requires as little as one round-trip time (RTT) for all the referenced objects.
- With persistent connections, the performance is improved by 20%.
- Persistent connections are the default mode for HTTP/1.1.

HTTP Message Format

- Two types:
  1. Request Message
  2. Response Message

HTTP Request Message

- It is in ASCII format which means that human-readable format.
- HTTP request message consist three part:
  - Request line
  - Header line
  - Carriage return



carriage return character

line-feed character

request line (GET, POST, HEAD commands)

header lines

carriage return (line feed at start of line indicates end of header lines)

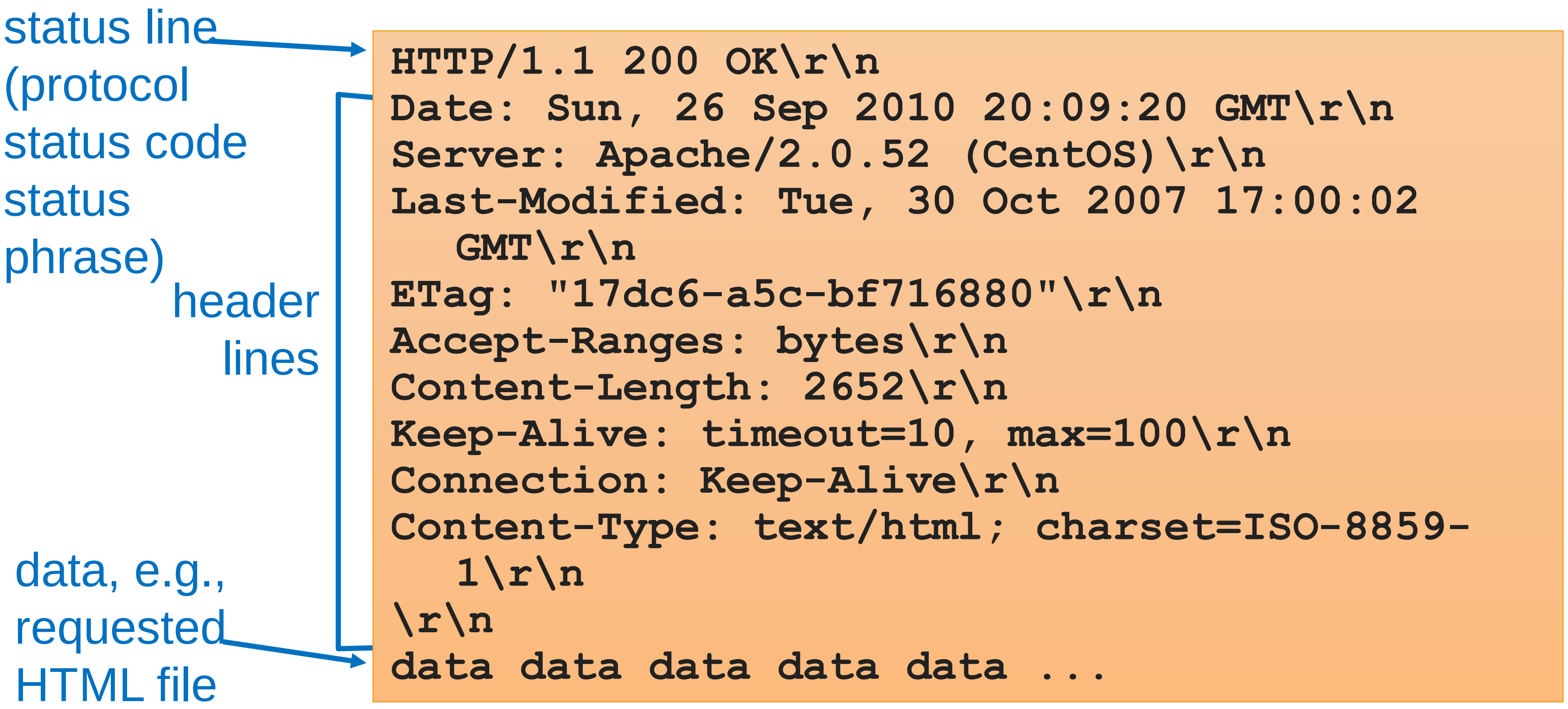
```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP Request Message - Format

- The request line has three fields: Method field, URL field, and HTTP version field.
- The method field can take on several different values, including GET, POST, HEAD, PUT, and DELETE.
- In above message, browser is requesting the object /somedir/page.html and version is self-explanatory; browser implements version HTTP/1.1.
- The header line Host: www-net.cs.umass.edu specifies the host on which the object resides.
- User agent indicate browser name and version.

HTTP Response Message

- HTTP response message consist of three part:
  1. Status line
  2. Header line
  3. Data (Entity body)


status line (protocol status code status phrase)
header lines
data, e.g., requested HTML file

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02
   GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-
   1\r\n
\r\n
data data data data data ...
```

HTTP Response Message - Format

- The status line has three fields: protocol version field, status code and corresponding status message.
- In below example, the status line indicates that the server is using HTTP/1.1 and that everything is OK.

```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-8859-1\r\n \r\n
data data data data data ...
```

Content-Type: header line indicates that the object in the entity body is HTML text.

Content-Length: header line indicates the number of bytes in the object being sent.

Last-Modified: header line indicates the time and date when the object was created or last modified.

Server: header line indicates that the message was generated by an Apache Web server.

Date: header line indicates the time and date when the HTTP response was created and sent by the server.

HTTP Response Status Codes

- A status code appears in 1st line in server-to-client response message.
- Some sample codes:
  - 200 OK
    - Request succeeded, requested object later in this message
  - 301 Moved Permanently
    - Requested object moved, new location specified later in this message(Location)
  - 400 Bad Request
    - Request message not understood by server
  - 404 Not Found
    - Requested document not found on this server
  - 505 HTTP Version Not Supported
    - Requested http version not support

Outline - Summary

- Principles of Computer Applications
  - Browser, Web Server, Email, P2P Applications etc…
- Application Layer (TCP – UDP Services)
- Web (Web Pages – Objects like html, jpeg, mp3, etc…)
- HTTP (TCP connection, port-80, persistent & non-persistent conn.), Request & Response Message format, Cookies, Web caches, FTP, Port-21
- E-mail (User agent, Mail Server, SMTP port - 25), POP3, IMAP
- DNS (Domain names to IP Address), hierarchy structure
- Socket programming with TCP and UDP (TCP – Sock_Stream, UDP – Sock_DGram)