Indian Institute of Information Technology, Sri City, Chittoor
(An Institute of National Importance under an Act of Parliament)

Computer Communication Networks

Application Layer

Dr. Raja Vara Prasad

Assistant Professor

IIIT Sri City

Application Layer

Network Applications

Network application development -- writing programs that run on

different end systems and communicate with each other over the network

Example:

Web application → two distinct programs that communicate with each other:

- the browser program running in the user's host (desktop, laptop, tablet, smartphone, and so on);
- the Web server program running in the Web server host.
- in P2P file-sharing system there is a program in each host that participates in the file-sharing community

Network Applications

- do not need to write software that runs on network core devices, such as routers or link-layer switches
- Network core devices do not function at the application layer
- function at lower layers— specifically at the network layer and below

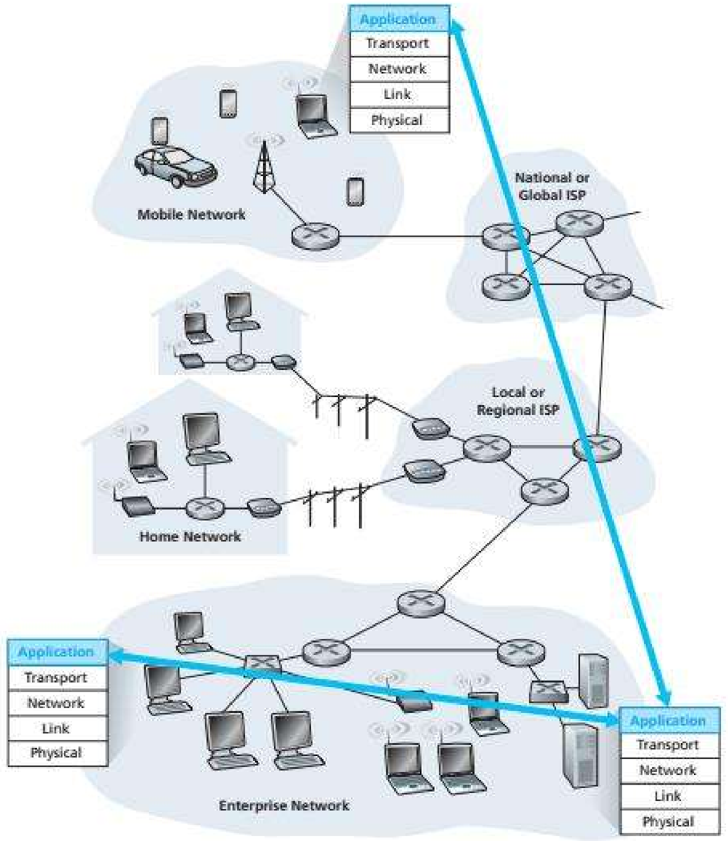


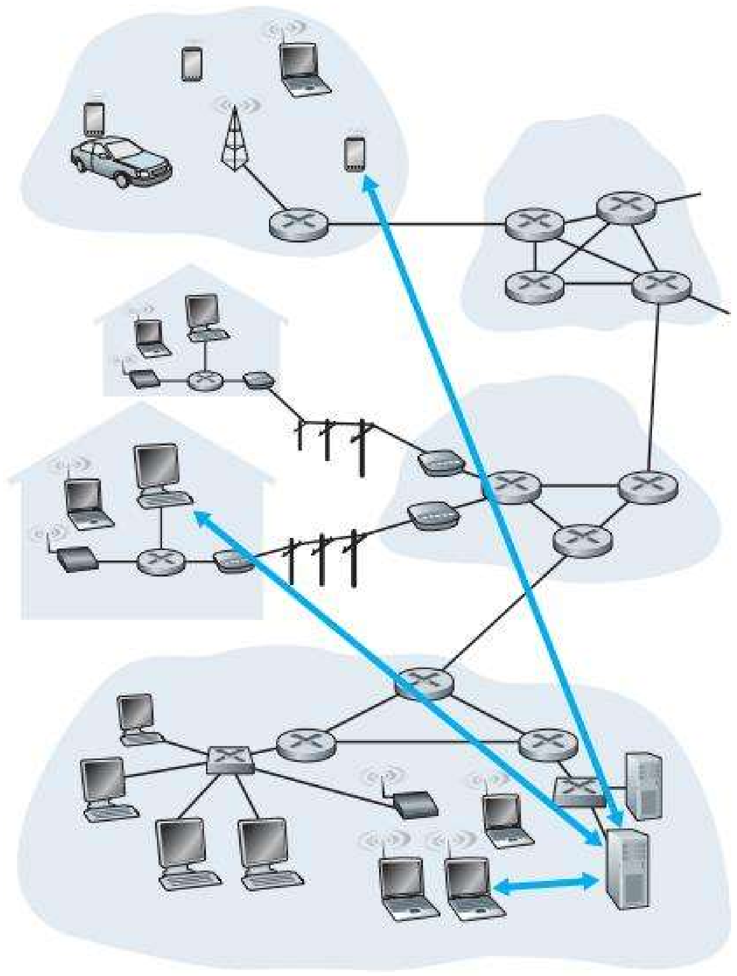
Figure 2.1 • Communication for a network application takes place between end systems at the application layer

Network Applications

- Applications use the services of network (Transport layer)
- For an application developer, architecture and services of network are fixed
- Architectures of applications:
  - Client-Server architecture
  - Peer-to-Peer (P2P) architecture
- Application developer decides on the architecture and services of transport layer to be used.

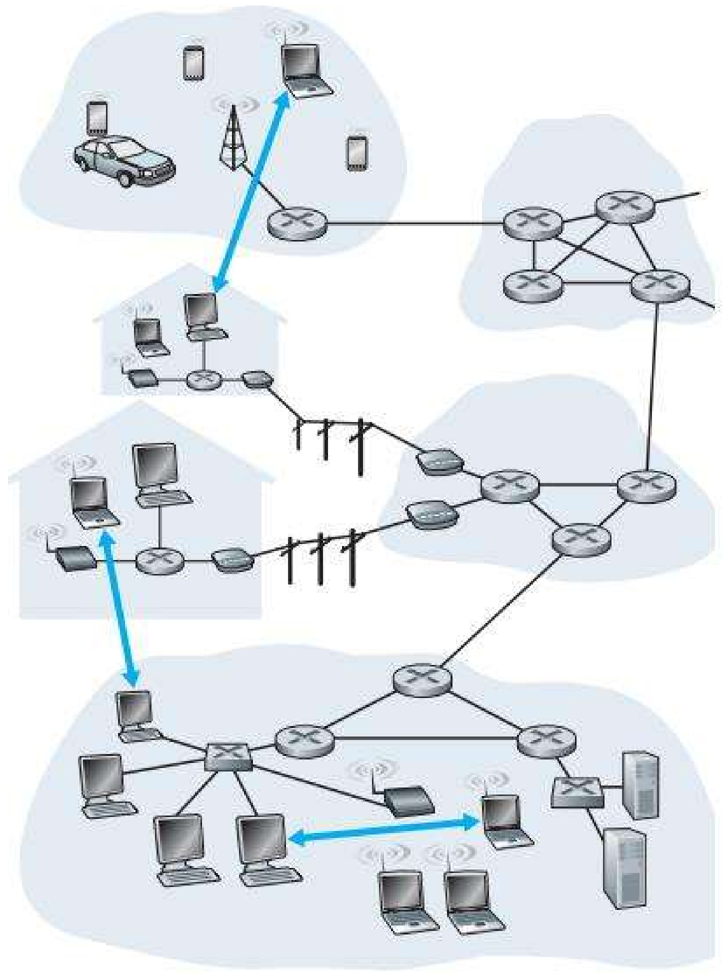
Client-Server Architecture

- Server: An end system that serves the requests from various hosts.
- A server is always ON.
- Client: An end system that requests a server for content.
- A client can be either ON-OFF or always ON.
- Example applications using this architecture: web, e-mail, file transfer, etc.


a. Client-server architecture

Peer-to-Peer Architecture

- End systems communicate by a direct connection.
- The end systems are called peers.
- Example applications: skype, internet telephony, torrents, etc
- Advantages:
  - File distribution
  - Self-scalable: can handle growth in traffic
  - Cost effective: no server infrastructure and server bandwidth.
- Challenges in P2P Architecture:
  - ISP friendly: asymmetric data traffic.
  - Security
  - Incentives: Peers should share bandwidth.


b. Peer-to-peer architecture

Processes Communicating

- A process is a program that is running within an end system.
- A client process is a process running on a client and a server process is process running on a server.
- It is the client process and server processes that are actually communicating.
- A process sends and receives messages to and from transport layer through a software interface known as socket.
- A socket is also known as Application Programming Interface (API).

Interface Between the Process: API

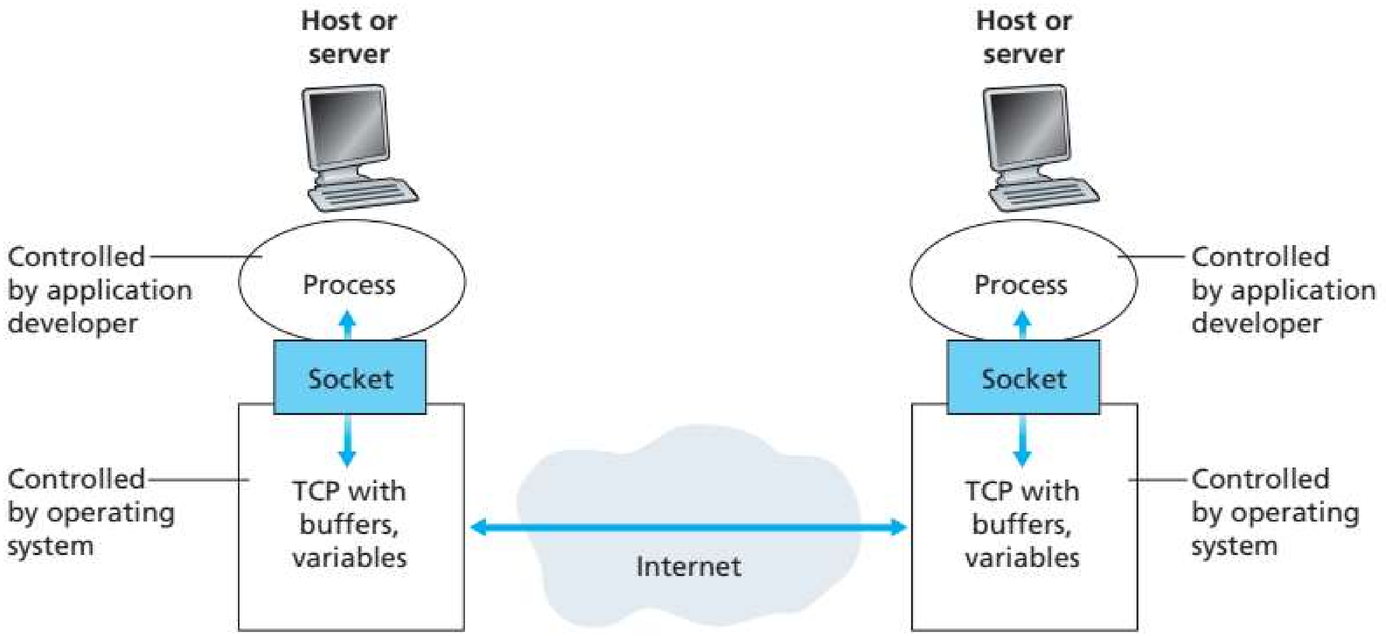
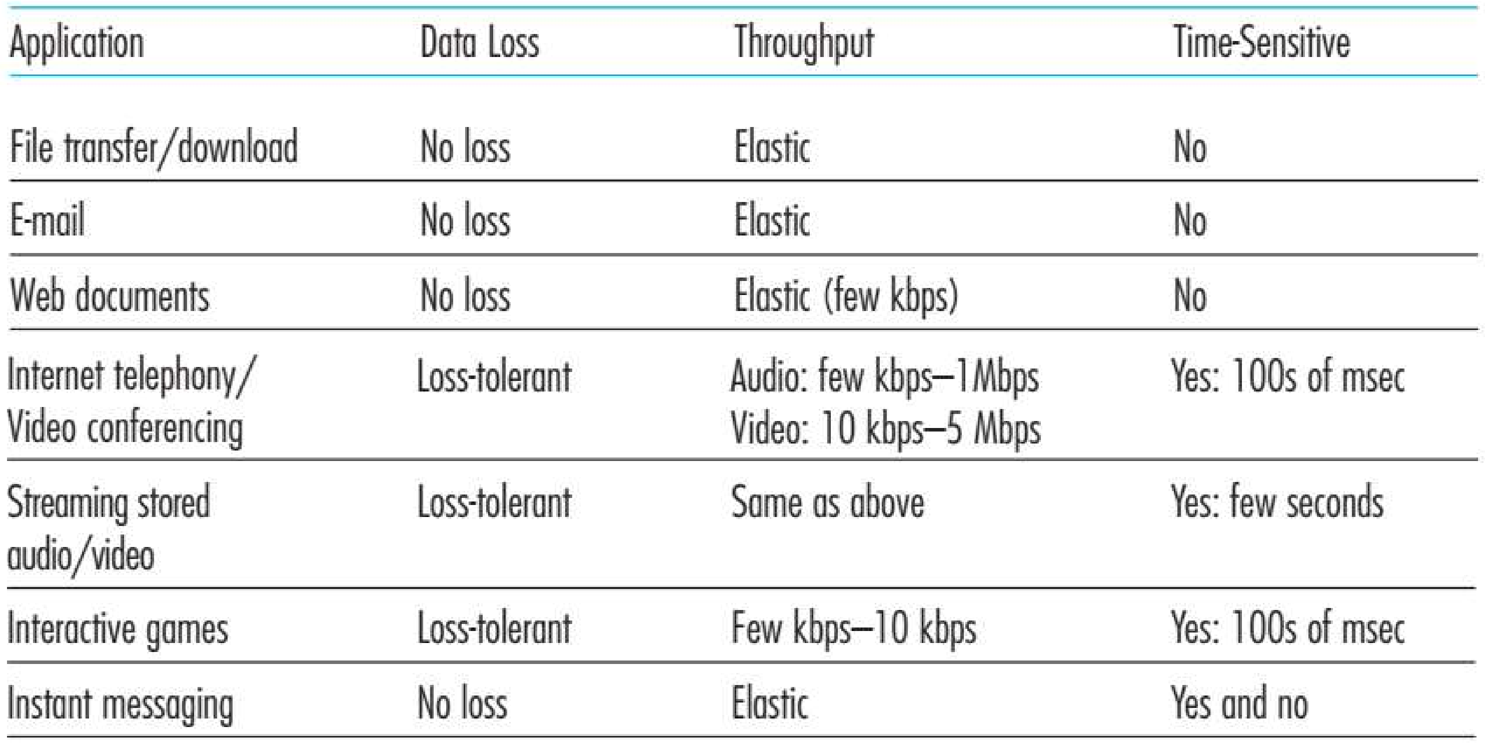

Host or server
Host or server
Controlled by application developer
Process
Socket
Controlled by operating system
TCP with buffers, variables
Internet
Process
Socket
TCP with buffers, variables
Controlled by application developer
Controlled by operating system

Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

Services of Transport Layer

- Reliable data transfer: Guaranteed data delivery service.
- Throughput
- Timing: for example, it is guaranteed that a packet will be delivered no more than 100 msec later.
- security: end-point authentication, encryption and decryption.

Requirements of Applications



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–1Mbps 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
Instant messaging	No loss	Elastic	Yes and no

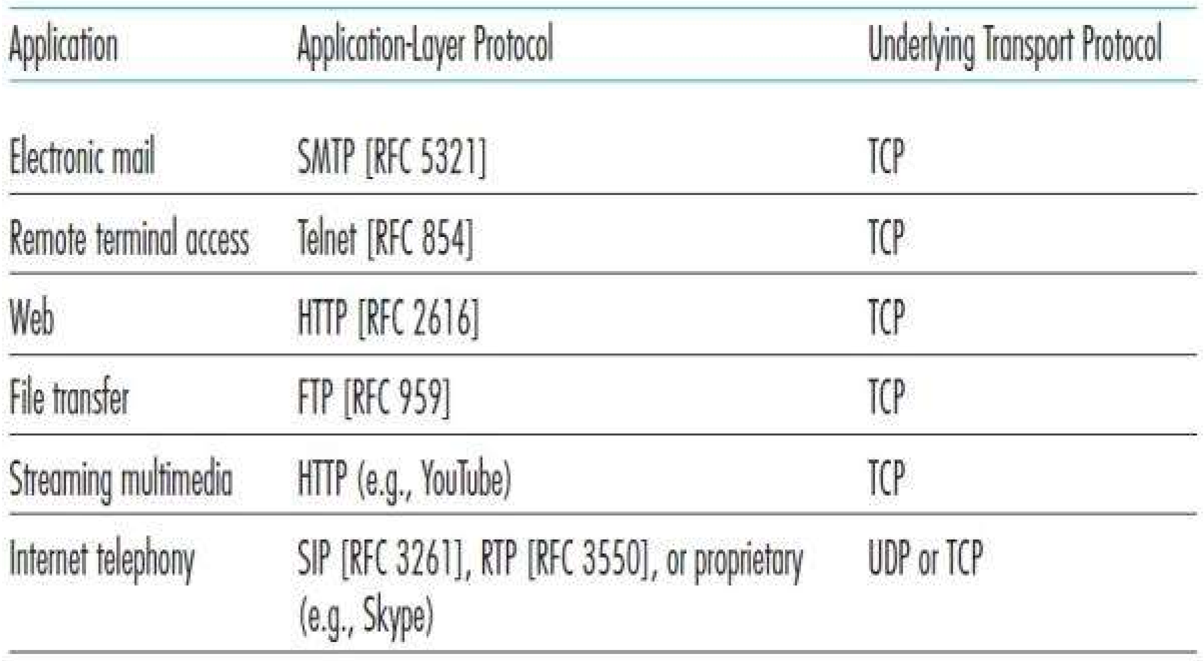
Figure 2.4 ♦ Requirements of selected network applications

Transport protocols

- Transmission Control Protocol (TCP)
  - Connection oriented service: handshaking, full-duplex connection
  - Reliable data transfer service: packets get delivered without error and in proper order.
  - Congestion control
- User Datagram Protocol (UDP)
  - Connectionless
  - Unreliable data transfer service.
  - No congestion control

- can often provide satisfactory service to time-sensitive applications,
- cannot provide any timing or throughput guarantees

Applications



Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Addressing Processes

- There are many processes running on a host, how to identify the destination process?
- We identify host by IP address.
- We identify processes by port numbers!
- For example, web server is identified by port number 80, mail server is identified by port number 25.

Application Layer - Introduction

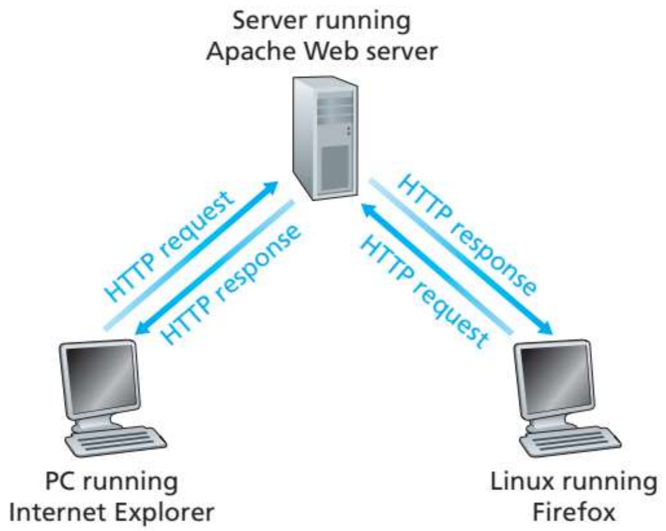
Application-layer protocol defines:

- The types of messages exchanged, for example, request messages and response messages
- The syntax of the various message types, such as the fields in the message and how the fields are delineated
- The semantics of the fields - meaning of the information in the fields
- Rules for determining when and how a process sends messages and responds to messages

Application-layer protocol is only one piece of a network application

Web and HTTP

- A web page is a document and consists of objects
- An object is nothing but a file such as HyperText Markup Language (HTML) file, an image file, applet or video clip.
- If a web page contains a basic html file and ten images, we say the web page contains 11 objects.
- HyperText Transfer Protocol (HTTP) is the web's application layer protocol
- HTTP uses client-server architecture with TCP.
- The client program and server program talk to each other by exchanging HTTP messages.



HTTP request-response behavior

Uniform Resource Locater

- An object should be addressable by a URL.
- Each URL consists of hostname and objects path name
- For example, http://www.iiits.ac.in/wp-content/uploads/2017/05/Untitled-design-15.png is url for an image.
- www.iiits.ac.in is host name
- wp-content/uploads/2017/05/Untitled-design-15.png is path name.
- Client side of HTTP is implemented in Web browser and server side is implemented in Web server.
- Examples: Apache and Microsoft Internet Information server.

✓ base HTML file plus objects

✓ base HTML file references the other objects in the page with the objects' URLs.

HTTP

- HTTP client initiates a connection with HTTP server (handshaking).
- Once the connection is established, client and server exchange messages through socket interface.
- Client sends an HTTP request and receives HTTP messages through its socket
- Server receives HTTP requests and sends HTTP responses through its socket interface.
- Client/server need not worry about packets (does not have any control) after sending through their socket.
- Server sends requested files without storing state information of client. Thus HTTP is a stateless protocol.

HTTP Connection

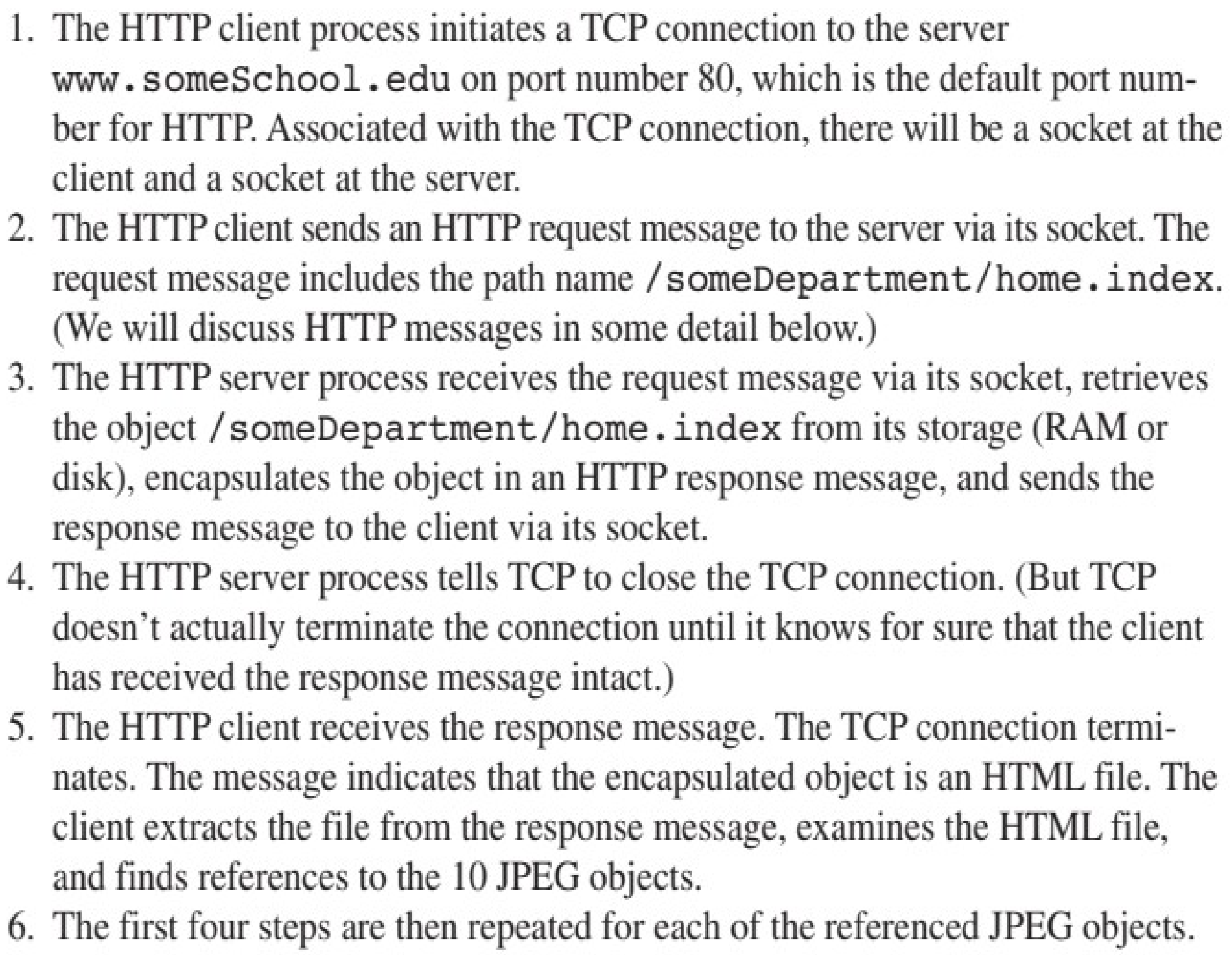
- Let us say, a web page has one html file and 10 images.
- How does client retrieve the web page?
- Nonpersistent and Persistent
- Nonpersistent: one TCP connection for each file
- Persistent: one TCP connection for all files

Nonpersistent Connection

- For each file:
  - HTTP client initiates a TCP connection to the server on port number 80
  - Client sends its HTTP request and it includes the path name to the file
  - HTTP server receives the request and retrieves the file and sends the HTTP response to the client
  - HTTP server tells TCP to close the connection.
- TCP connections can be serial or parallel depending on browser's configuration

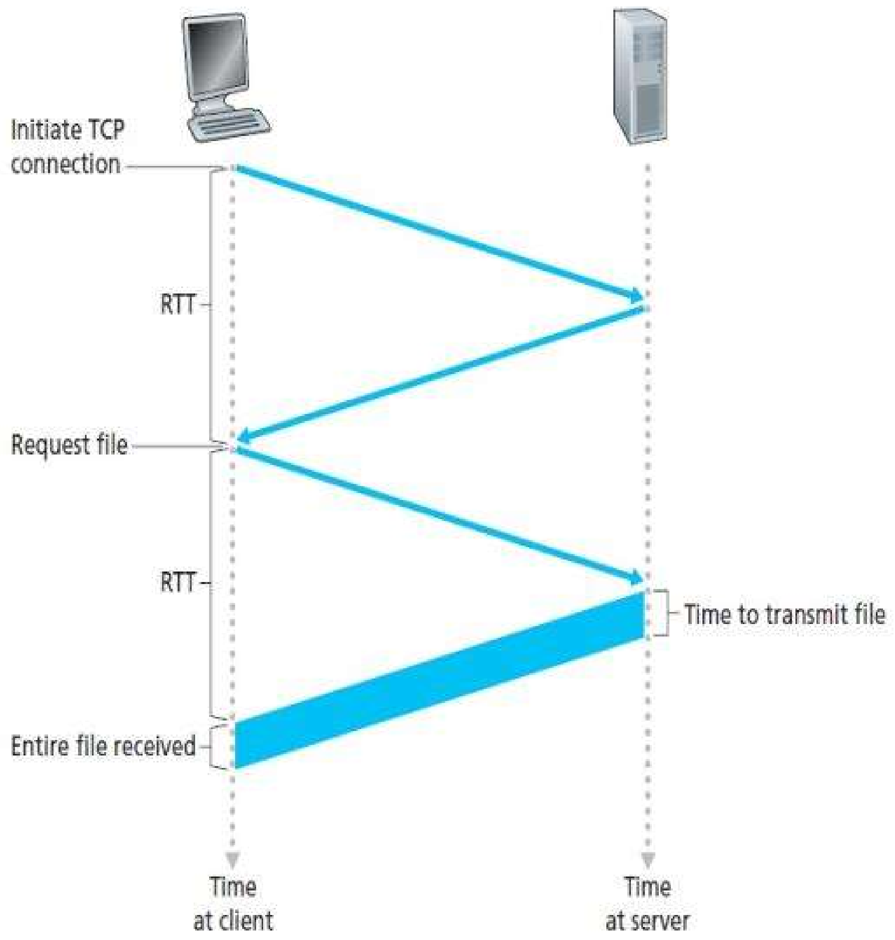
Example: Non-Persistent

- steps of transferring a Web page from server to client for the case of non-persistent connections.
- page consists of a base HTML file and 10 JPEG images → 11 objects reside on the same server.
- URL for the base HTML file is: http://www.someSchool.edu/someDepartment/home.index



1. The HTTP client process initiates a TCP connection to the server `www.someSchool.edu` on port number 80, which is the default port number for HTTP. Associated with the TCP connection, there will be a socket at the client and a socket at the server.
2. The HTTP client sends an HTTP request message to the server via its socket. The request message includes the path name `/someDepartment/home.index`. (We will discuss HTTP messages in some detail below.)
3. The HTTP server process receives the request message via its socket, retrieves the object `/someDepartment/home.index` from its storage (RAM or disk), encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.
4. The HTTP server process tells TCP to close the TCP connection. (But TCP doesn't actually terminate the connection until it knows for sure that the client has received the response message intact.)
5. The HTTP client receives the response message. The TCP connection terminates. The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file, and finds references to the 10 JPEG objects.
6. The first four steps are then repeated for each of the referenced JPEG objects.

Round-Trip Time



Persistent Connection

- Server leaves the connection after sending the HTTP response
- Pipelining: A browser can request for files without waiting for the reception of pending requests.
- TCP closes after some idle period
- Default mode HTTP: Persistent connection with pipelining.

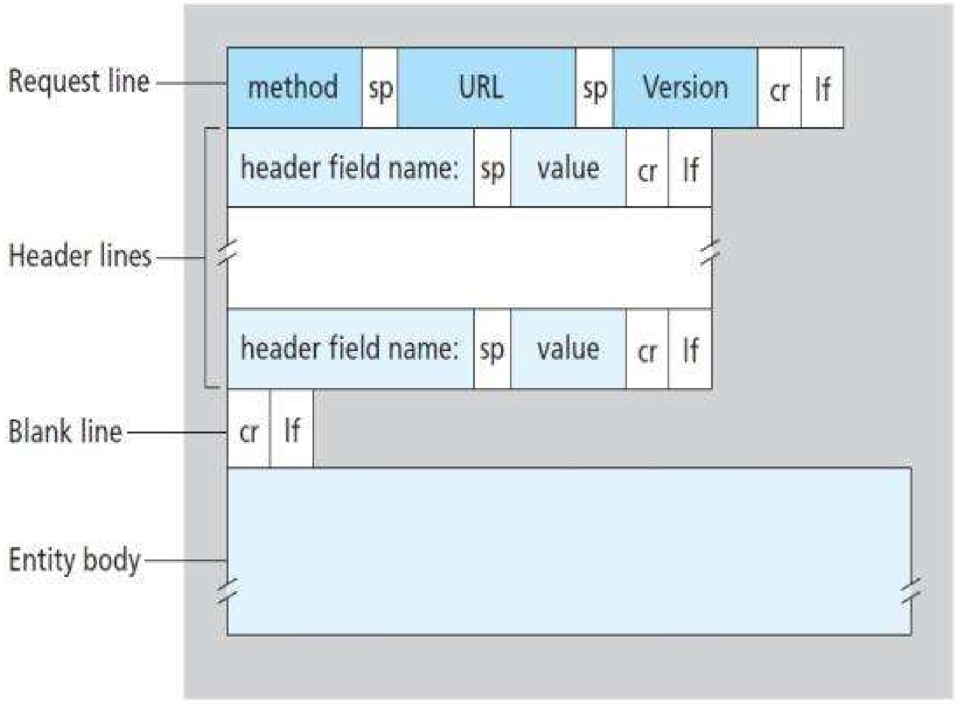
HTTP Request Format

- HTTP request message:

```
GET /somedir/page.html HTTP/1.1
Host: www.iitm.ac.in
Connection: close
User-agent: Mozilla/4.0
Accept-language: En
```

- Methods: GET, PUT, POST, HEAD, DELETE

HTTP Request


Request line
method
sp
URL
sp
Version
cr
lf
Header lines
header field name:
sp
value
cr
lf
header field name:
sp
value
cr
lf
Blank line
cr
lf
Entity body