Chapters 2- Application layer

PART 2

Application layer, Chapter 2 goal

- Conceptual and implementation aspect of application-layer protocols
- Transport-layer service model
- Study popular Internet applications and their protocols
- Socket API (skip because it is covered in 3010)

Today's agenda (Sep 16th)

- Network application architecture
- Web application
- Application-level protocols (HTTP)

NETWORK APPLICATIONS

- Video streaming (YouTube, Netflix)
- Internet Browsing
- E-mail
- File storage and sharing (google drive. Dropbox, onedrive)
- Social media
- Voice over IP (Skype, WhatsApp)
- Remote login
- …

APPLICATION LAYER

1970s & 1980s: text email, remote access to computers, file transfers, and newsgroups.

Mid-1990s: the World Wide Web(Web surfing, search, and electronic commerce)

End of 1990s: instant messaging and P2P file sharing

2000s: voice over IP and video conferencing e.g, Skype, Facetime, and Google Hangouts

…

User generated video such as YouTube and movies on demand such as Netflix; multiplayer online games such as Second Life and World of Warcraft

Same time: New generation of social networking applications—such as Facebook, Instagram,Twitter, and WeChat, Tik Tok

Then smartphone and advancing location based mobile apps: check-in, dating, and road-traffic forecasting apps (such as Yelp, Tinder, Waz, and Yik Yak)

INTERNET APPLICATIONS

- Web
- Electronic mail
- Directory service
- Video streaming
- P2P applications

APPLICATION-LAYER PROTOCOL

The application-layer protocol specifies how an application's processes, running on different end systems, pass messages to each other. In particular:

- The types of messages exchanged: request messages and response messages
- The syntax of the different message types: the fields in the message and how the fields are outlined
- The semantics of the fields: the meaning of the information in the fields
- Rules: to determine when and how a process sends messages and responds to messages.

NETWORK APPLICATION ARCHITECTURES

- The application architecture is designed by the application developer and dictates how the application is structured over the various end systems.

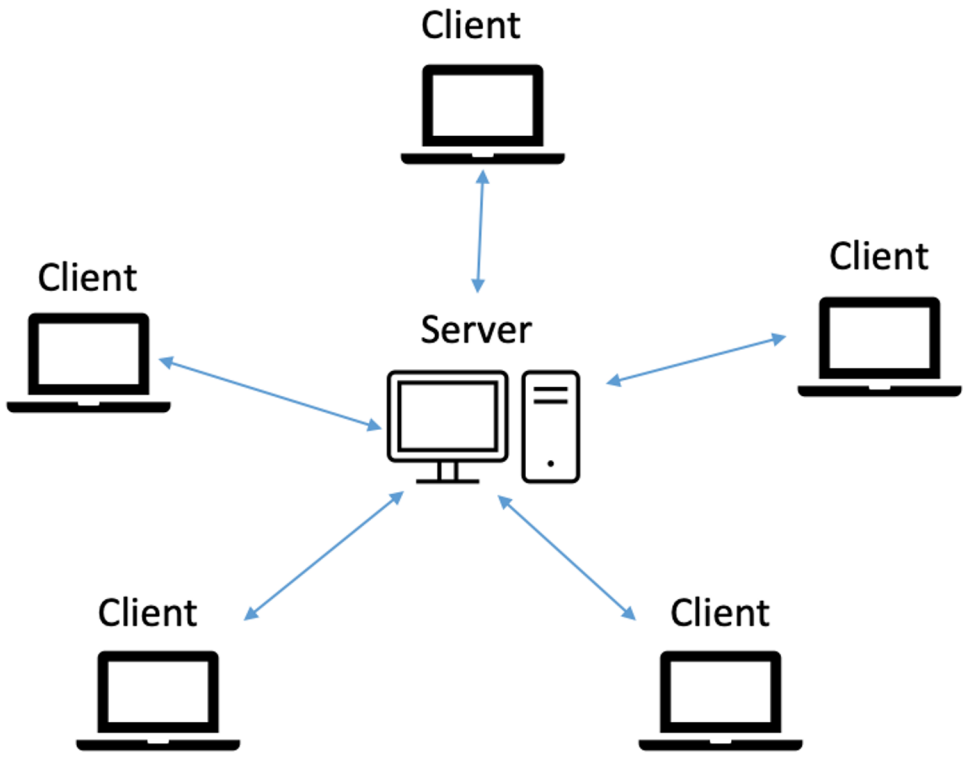
- Two application architectural paradigms:
  - Client-server Architecture
  - Peer to Peer architecture

CLIENT-SERVER ARCHITECTURE

Includes at least one server (one host) that provides services to multiple clients (other hosts.)

Example: Web application that includes a web server services requests from browsers running on client hosts.

The server has a fixed, well-known address, called an IP address



DATA CENTER

For big networks a single-server host is incapable of keeping up with all the requests

from clients.

Data center network includes cabling, switches, routers that connect servers together, housing a large number of hosts to create a powerful virtual server.

A data center can have hundreds of thousands of servers, which must be powered and maintained

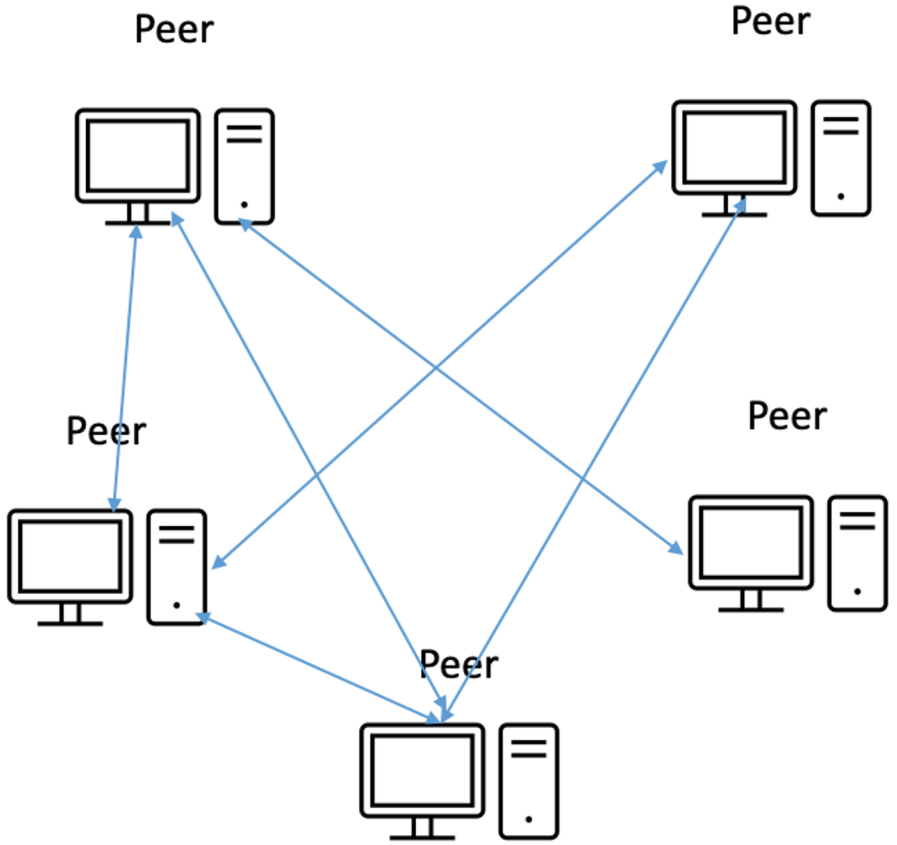
PEER TO PEER (P2P) ARCHITECTU

There is no specific client or a server. Each node(peer) can either be a client or a server.

The peer to peer network connects end systems so that every host shares all or some part of the resources.

Provide self-scalability

Traffic-intensive applications e.g, file sharing, peer-assisted download acceleration, video conference.



PROCESSES COMMUNICATING

From OS perspective, application's processes communicate between multiple hosts.

A process can be thought of as a program that is running within an end system. Within same host, two processes communicate using inter-process communication (defined by OS)

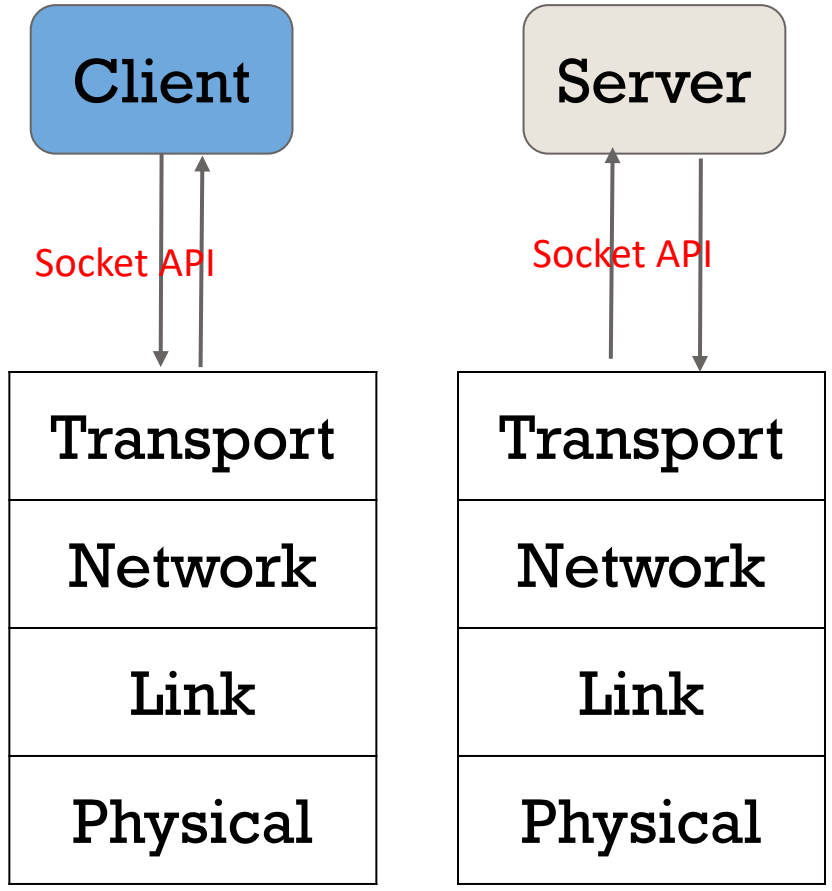
Processes on two different end systems communicate with each other by exchanging messages across the computer network.

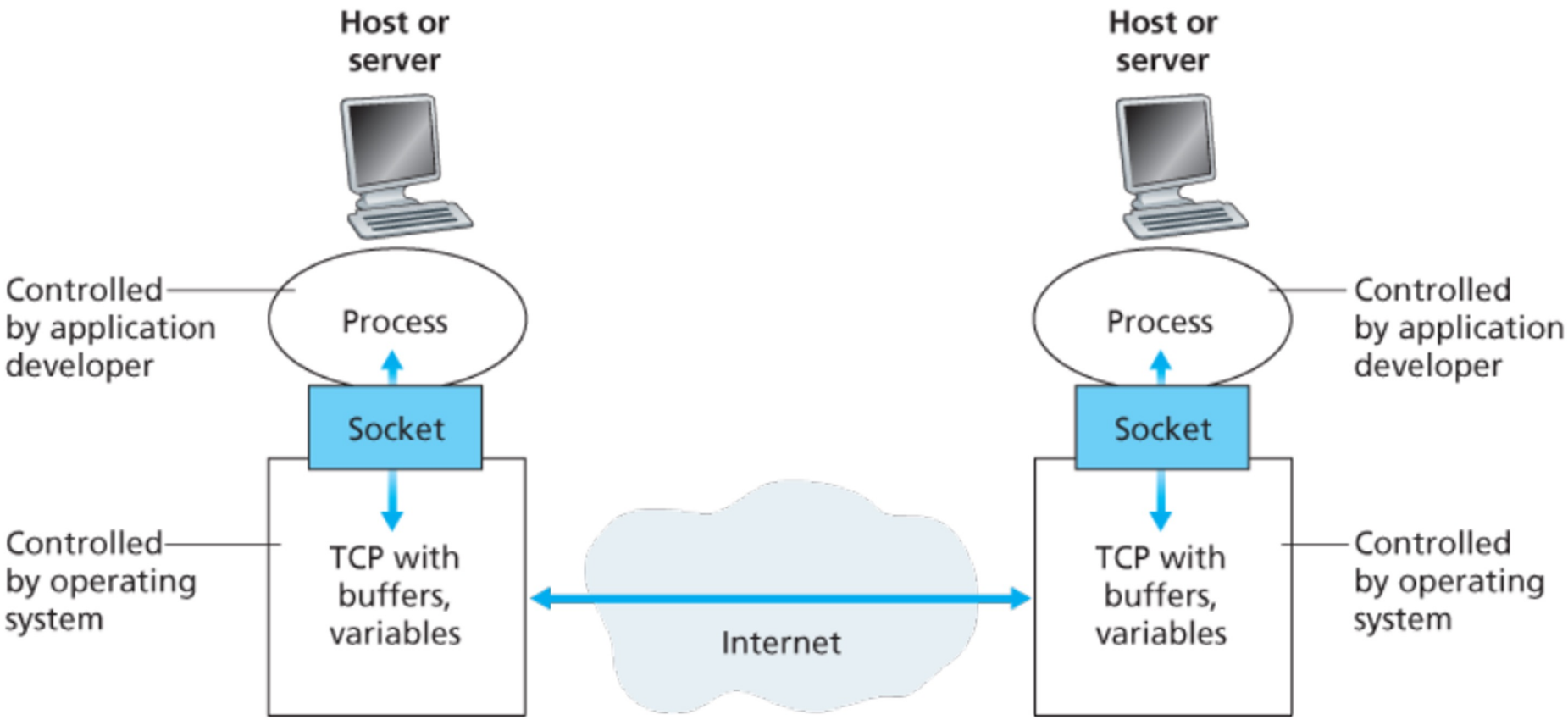
SOCKET

Socket is a software interface that allow communication between two different processes on the network.

A socket is the interface between the application layer and the transport layer within a host. It can be considered as Application Programming Interface (API) between the application and the network.

You have control on application layer side of the socket, but very limited on the transport-layer side of the socket. You can still choose the transport protocol and set up a few transport-layer parameters such as maximum buffer and maximum segment sizes.





Kurose, James F., and Keith W. Ross. "Computer Networking: A Top-Down Approach Edition." Addision Wesley (2017).

TRANSPORT LAYER SERVICES FOR APPLICATIONS

- Data integrity
  - Reliable data transfer
  - Tolerate some loss
- Timing
- Throughput
  - Require minimum amount of throughput
  - Elastic apps: make use of whatever throughput they get
- Security

TRANSPORT SERVICE REQUIREMENTS: COMMON APPS

application	data loss	throughput	time sensitive?
file transfer/download	no loss	elastic	no
e-mail	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	same as above	yes, few secs
interactive games	loss-tolerant	Kbps+	yes, 10's msec
text messaging	no loss	elastic	yes and no

WEB VS INTERNET

The Internet is a global network of networks.

The world wide web or web is one of the internet application.

Web provides a simple interface (a web page) that represent a collection of information.

A web page consist of an **objects** (files such as HTML, image, java applet, …) that are addressable by a **URL** (Uniform resource Locator).

Uniform Resource Locator (URL) Example

Hypertext Transfer Protocol

Path to web page's home page

http://www.cs.princeton.edu/llp/index.html

name of the machine (host) that serves the page

When you click such a URL:

- Up to six messages to translate the server name (www.cs.princeton.edu) into its Internet Protocol (IP) address (128.112.136.35)
- Three messages to set up a Transmission Control Protocol (TCP) connection between your browser and this server.
- Four messages for your browser to send the HTTP "GET" request and the server to respond with the requested page
- Four messages to tear down the TCP connection

IDENTIFIERS AND ADDRESSING PROCESS

Each selectable object on a web page is bound to an **identifier** (Uniform resource Locator (URL)) for the next page or next object.

Identifier is required to receive messages.

- IP address
- Port number

Host device has unique 32-bit IP address.

Port numbers associated with process on the host.

HTTP server port number: 80

Mail server port number: 25

WEB AND HTTP

Web was the first Internet application that attracted the general public.

- It operates in demand
- Navigate information
- Everyone can easily make information available

HyperText Transfer Protocol (HTTP): Web's application-layer protocol

HTTP is implemented on Both the client-side and server-side and establishes communication by exchanging messages.

HTTP defines the structure of messages and how they are exchanged.

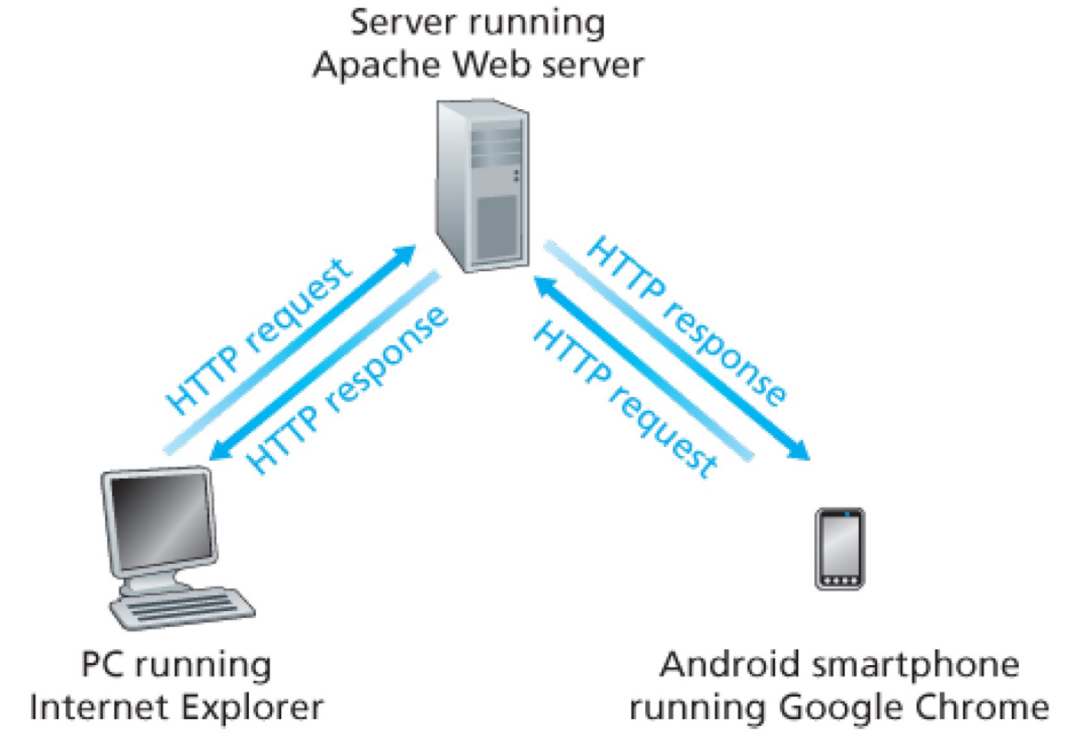
Web uses the client-server application architecture.

A Web server (and its secondary server) is always on, with a fixed IP address, and it services requests from many browsers.

HTTP uses **TCP**:

- client initiates TCP connection (creates socket) to a server, port 80
- server accepts TCP connection HTTP messages exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is a **stateless protocol** as an HTTP server maintains no information about the clients. It is easier, as maintaining states are complex.

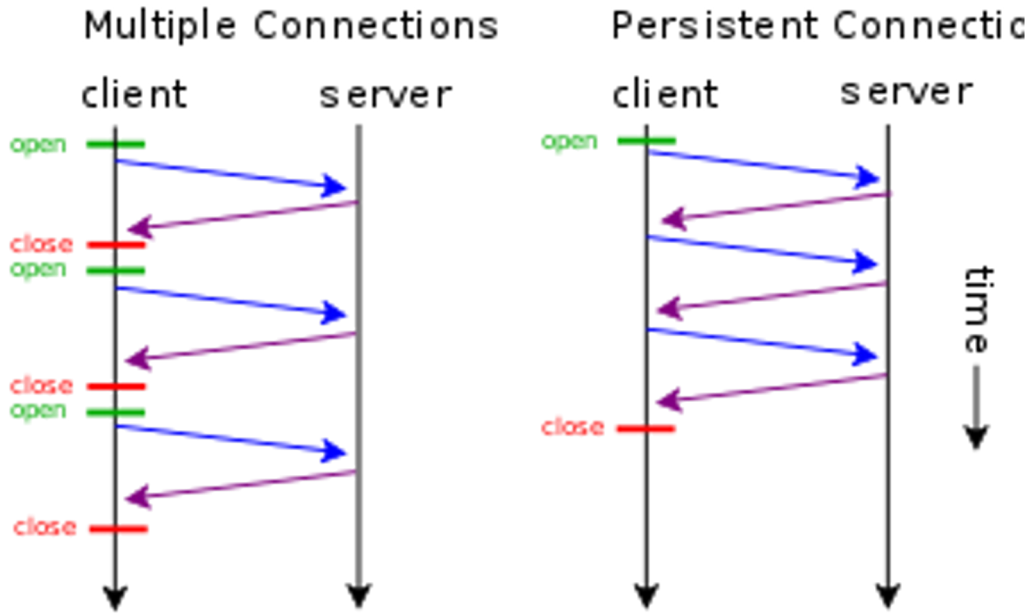


Kurose, James F., and Keith W. Ross. "Computer Networking: A Top-Down Approach Edition." Addision Wesley (2017).

NON-PERSISTENT AND PERSISTENT CONNECTIONS

- Non-Persistent: After the client receives one object, the connection is immediately closed.
- Persistent Connections: ensures the transfer of multiple objects over the same TCP connection.

HTTP use both non-persistent connections and persistent connections (default mode).



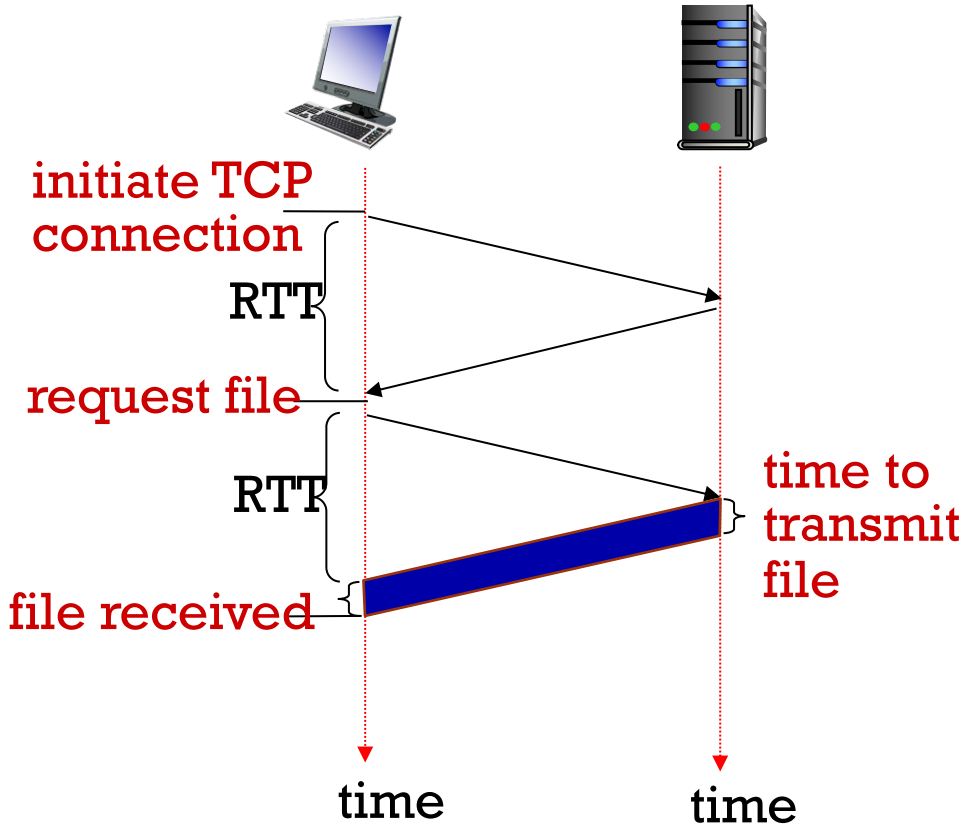
https://en.wikipedia.org/wiki/HTTP_persistent_connection

NON-PERSISTENT HTTP: RESPONSE TIME

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time (per object):

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- object/file transmission time



Non-persistent HTTP response time = 2RTT+ file transmission time

PERSISTENT CONNECTIONS

Advantages of persistent connections :

1) Lower CPU and memory usage as there are less number of connections.

2) Allows HTTP pipelining of requests and responses.

3) Reduced network congestion (because of fewer TCP connections).

4) Reduced latency in subsequent requests (no handshaking).

Disadvantages of persistent connections :

Resources may be kept occupied even when not needed and may not be available to others.

Most of the modern browsers like Chrome and Firefox use persistent connections.