COMPUTER COMMUNICATION NETWORKS

Department of Electronics and Communication Engineering

COMPUTER COMMUNICATION NETWORKS

UNIT 1: INTERNET ARCHITECTURE AND APPLICATIONS –
Class 12 – Principles of Network Applications

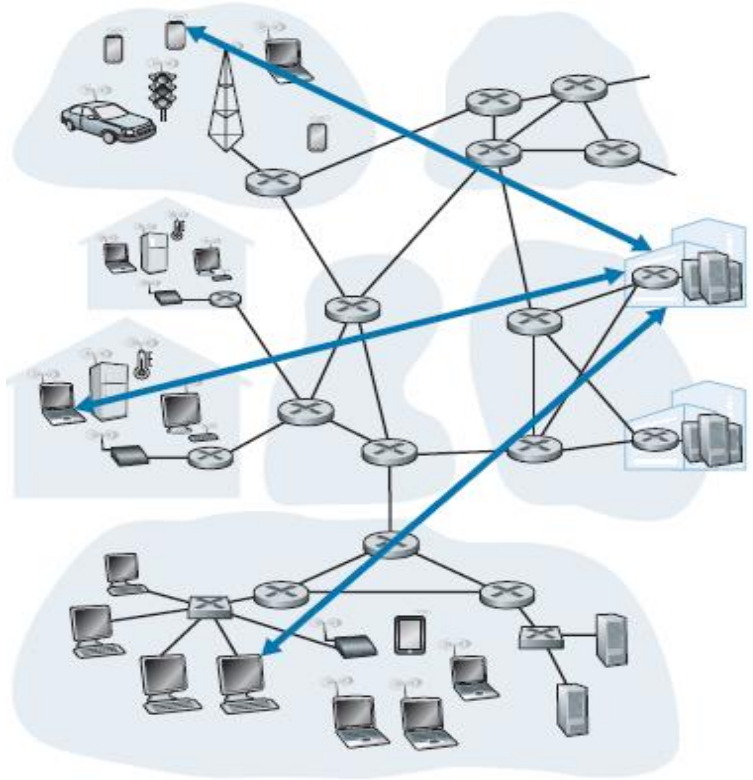
Department of Electronics and Communication Engineering

Unit 1 – Class 12 - Principles of network applications

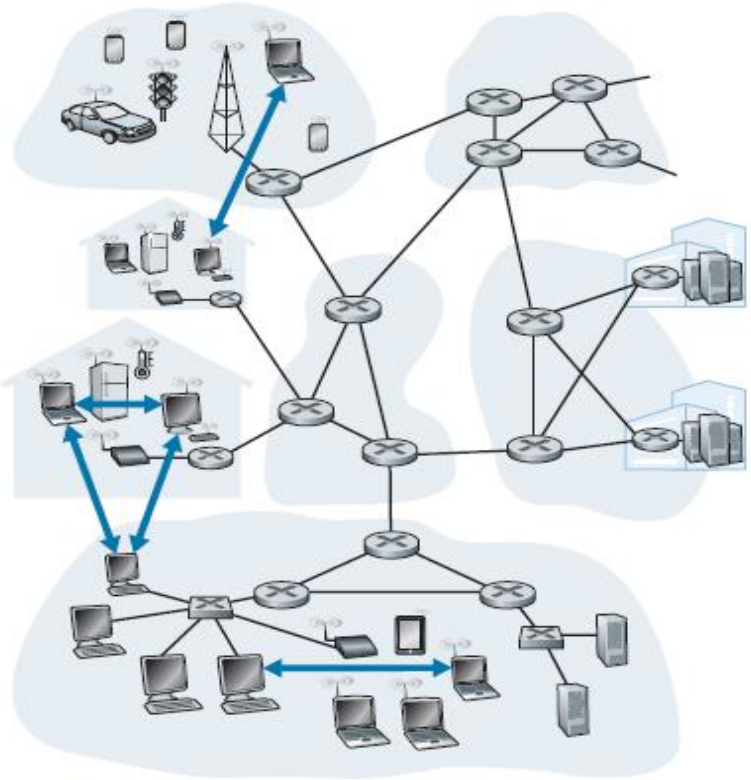
- Applications run on end-systems (e.g., computers, servers)
  - Examples: Skype, Whatsapp, Apple Pay, Youtube, Netflix
- Application developers often build a pair of programs which are coded in C, Java or Python
  - One program is referred to as client program while the other is referred to a server program (e.g., web browser and web server program)
  - **These programs are also referred to as processes**
- From the application developer's perspective, the network architecture is fixed and provides a specific set of services to applications.
  - Services: Reliability, throughput, security, timing, etc.

Unit 1 – Class 12 - Principles of network applications

- Application architecture dictates how the application developer views the interaction between the applications running on the end-systems



a. Client-server architecture



b. Peer-to-peer architecture

Unit 1 – Class 12 - Principles of network applications

- Client-server architecture
  - Client initiates the process communication
  - Server responds to requests from the clients
  - Server is always ON – a client can contact the server by sending a packet to the server's IP address
  - Server is well defined (e.g., IP address)
  - Server can handle concurrent connections (concurrent – Parallel)
  - Examples: Search engines, Internet commerce, Web-based email, Social media
  - Applications with a client-server architecture – Web, FTP, Telnet & e-mail

- Client-server architecture

  - Often in a client-server application, a single-server host is incapable of keeping up with all the requests from clients.
  - For example, a popular social-networking site can quickly become overwhelmed if it has only one server handling all of its requests.
  - For this reason, a data center, housing a large number of hosts, is often used to create a powerful virtual server.

- Client-server architecture
  - The most popular Internet services—such as search engines (e.g., Google, Bing, Baidu), Internet commerce (e.g., Amazon, eBay, Alibaba), Web-based e-mail (e.g., Gmail and Yahoo Mail), social media (e.g., Facebook, Instagram, Twitter, and WeChat)—run in one or more data centers.
  - Google has 19 data centers distributed around the world, which collectively handle search, YouTube, Gmail, and other services.
  - A data center can have hundreds of thousands of servers, which must be powered and maintained.
  - Additionally, the service providers must pay recurring interconnection and bandwidth costs for sending data from their data centers

- Peer-to-peer architecture
  - Any host can send and receive data
  - Hosts can join and leave the network any time
  - Hosts allocate resources to help each other
  - P2P architectures are self scalable – although each peer generates workload by requesting files, each peer also adds service capacity to the system by distributing files to other peers
  - Distributed algorithms are used for a) Maintaining state information and b) For file sharing
  - Examples: Bit Torrent, Skype

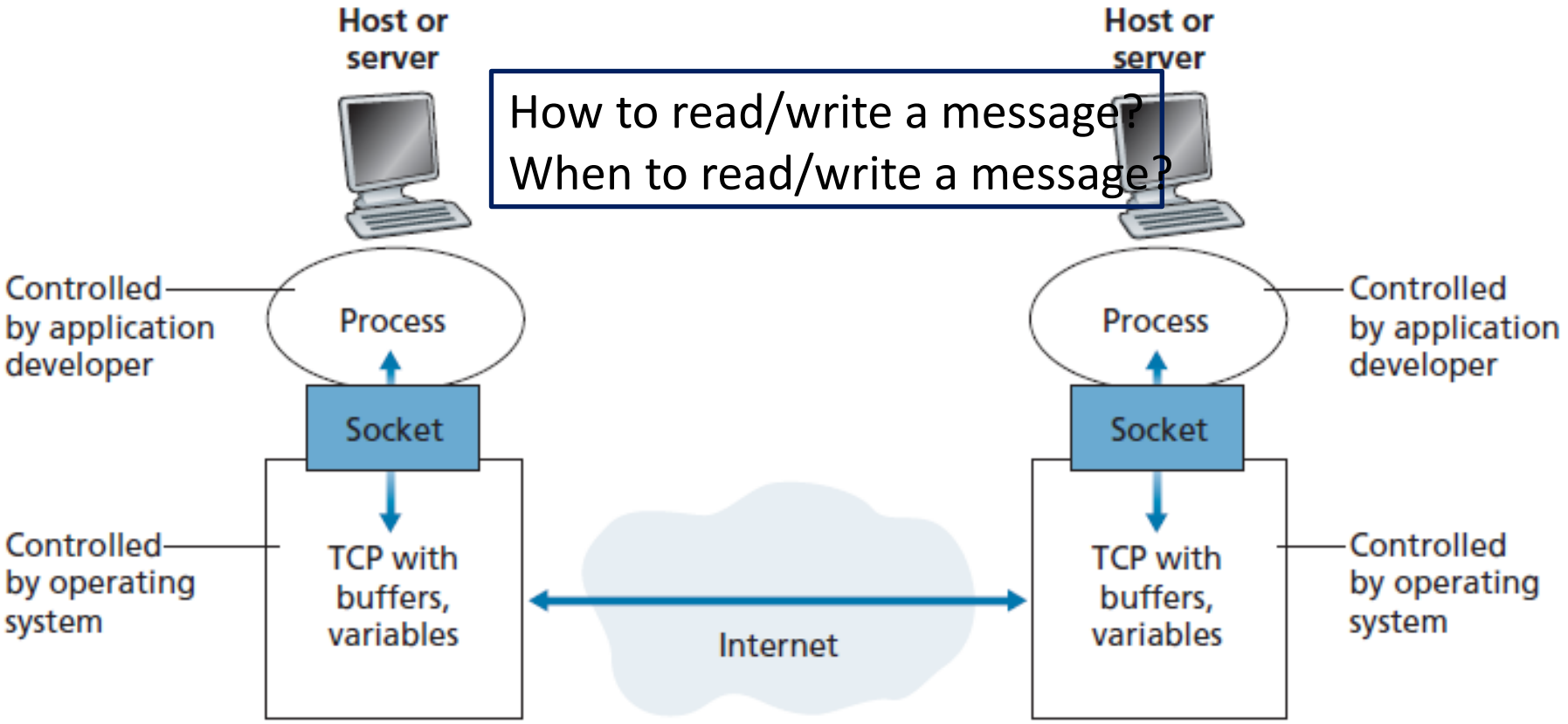
- Peer-to-peer architecture
  - P2P architectures are also cost effective, since they normally don’t require significant server infrastructure and server bandwidth (in contrast with clients-server designs with datacenters).
  - However, P2P applications face challenges of security, performance, and reliability due to their highly decentralized structure.

- Client & Server Processes
- In Web application a client browser process exchanges messages with a Web server process.
- In a P2P file-sharing system, a file is transferred from a process in one peer to a process in another peer.
- With the Web, a browser is a client process and a Web server is a server process.
- With P2P file sharing, the peer that is downloading the file is labeled as the client, and the peer that is uploading the file is labeled as the server.

- Client & Server Processes
- You may have observed that in some applications, such as in P2P file sharing, a process can be both a client and a server.
- Indeed, a process in a P2P file-sharing system can both upload and download files.
- We define the client and server processes as follows:
- “In the context of a communication session between a pair of processes, the process that initiates the communication (that is, initially contacts the other process at the beginning of the session) is labeled as the client. The process that waits to be contacted to begin the session is the server”

Processes communicating

- Processes exchange messages with one another using the rules governed by the end-systems operating system



Processes communicating

- Figure illustrates socket communication between two processes that communicate over the Internet.
- Figure assumes that the underlying transport protocol used by the processes is the Internet's TCP protocol.
- As shown in this figure, a socket is the interface between the application layer and the transport layer within a host.
- It is also referred to as the Application Programming Interface (API) between the application and the network, since the socket is the programming interface with which network applications are built.

Processes communicating

- The application developer has control of everything on the application-layer side of the socket but has little control of the transport-layer side of the socket.
- The only control that the application developer has on the transport layer side is (1) the choice of transport protocol and (2) perhaps the ability to fix a few transport-layer parameters such as maximum buffer and maximum segment sizes.
- Once the application developer chooses a transport protocol (if a choice is available), the application is built using the transport-layer services provided by that protocol.

Transport layer services

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

Transport layer services

- Applications and the supported protocols

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

Application–layer protocols

- Application layer protocols define the following
- 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, that is, the meaning of the information in the fields
- Rules for determining when and how a process sends messages and responds to messages.

THANK YOU

Department of Electronics and communication engineering