CS321: Computer Networks

Application Layer

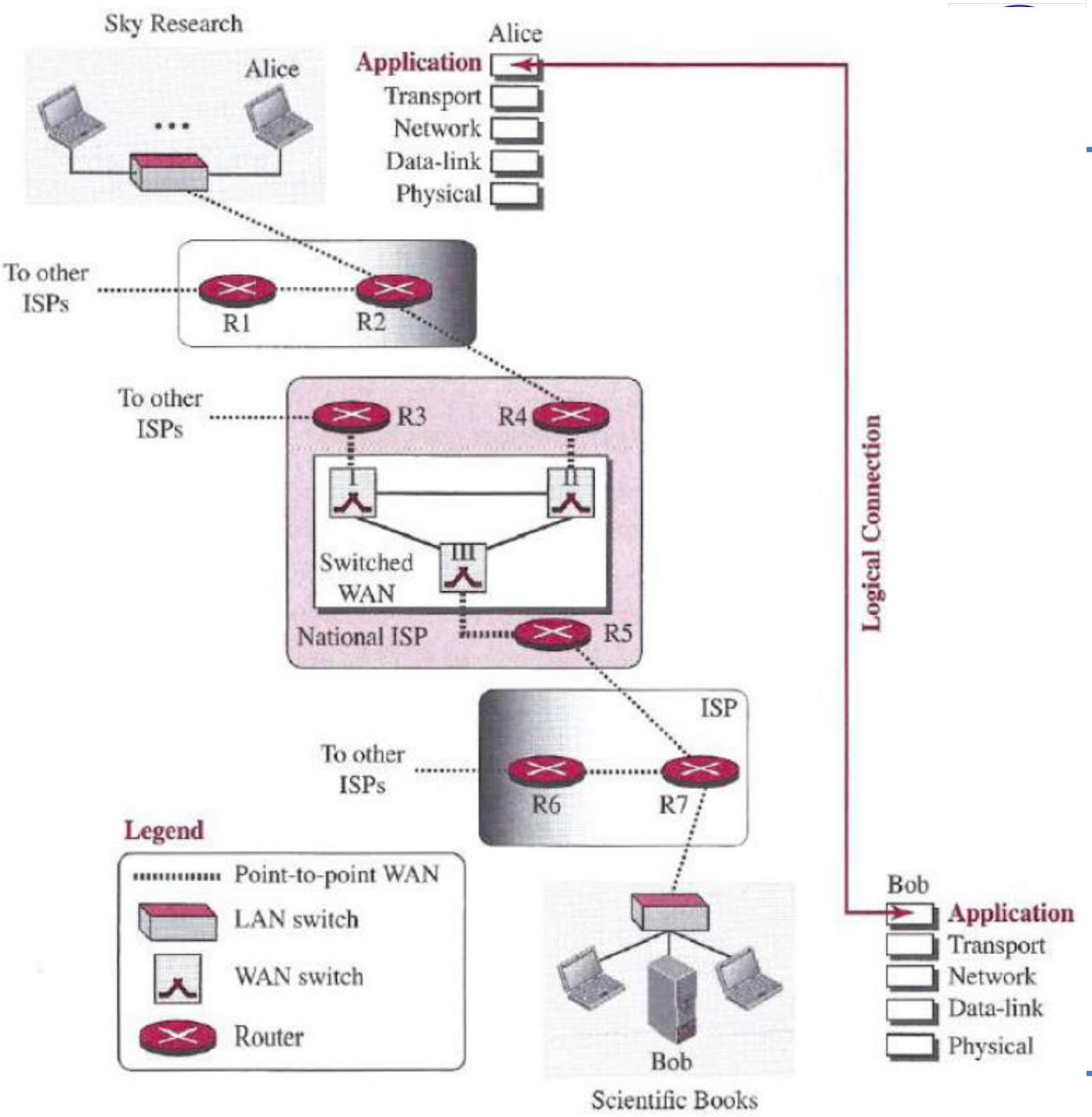
Dr. Manas Khatua
Assistant Professor
Dept. of CSE
IIT Jodhpur
E-mail: manaskhatua@iitj.ac.in

Basic

Application layer provides services to the user, and takes services from Transport layer.

Communication is provided using a logical connection.

The actual communication takes place through several physical devices (Alice, R2, R4, R5, R7, and Bob) and channels.



Providing Services

- Application layer provides services to the user, and takes services from Transport layer.
- The application layer, however, is somewhat different from other layers.
- The protocols can be removed from this layer easily as they only receives services from Transport layer but does not provide any service to that layer.
- The protocols used in the first four layers of the *TCP/IP* suite need to be standardized and documented. So, normally comes with Operating Systems (OS).
- There are several application-layer protocols that have been standardized and documented by the Internet authority.
  E.g., DHCP, SMTP, FTP, HTTP, TELNET

Popular Applications

- The applications have been the driving force behind the Internet's success
- 1970s and 1980s: classic text-based applications - text email, remote access to computers, file transfers, newsgroups.
- mid-1990s: World Wide Web, encompassing Web surfing, search, and electronic commerce
- end-1990s: instant messaging and P2P file sharing
- Since 2000: voice and video applications, including: voice-over-IP (VoIP) and video conferencing over IP such as Skype; user-generated video distribution such as YouTube; and movies on demand such as Netflix.
- Recently: multi-player online games (e.g. World of Warcraft), social networking applications, such as Facebook and Twitter

Application Architecture

- designed by the application developer

- It dictates how the application is structured over the various end systems
  - What the relationship should be between these two programs?

- We use two predominant architectural paradigms
  - client-server
    - there is an always-on host, called the *server*, which services requests from many other hosts, called *clients*.
    - Classic example: Web, FTP, Telnet, SSH, and e-mail.
  - Peer-to-peer (P2P)
    - the application exploits direct communication between pairs of intermittently connected hosts, called *peers*
    - Classic example*:* Internet Telephony (e.g., Skype), file sharing (e.g., BitTorrent), and IPTV

Client-Server Paradigm

- the service provider is an application program, called the server process;
- Server process runs continuously, waiting for another application program, called the client process, to make a connection through the Internet and ask for service.

- The server process must be running all the time;
- the client process starts when the client needs to receive service.

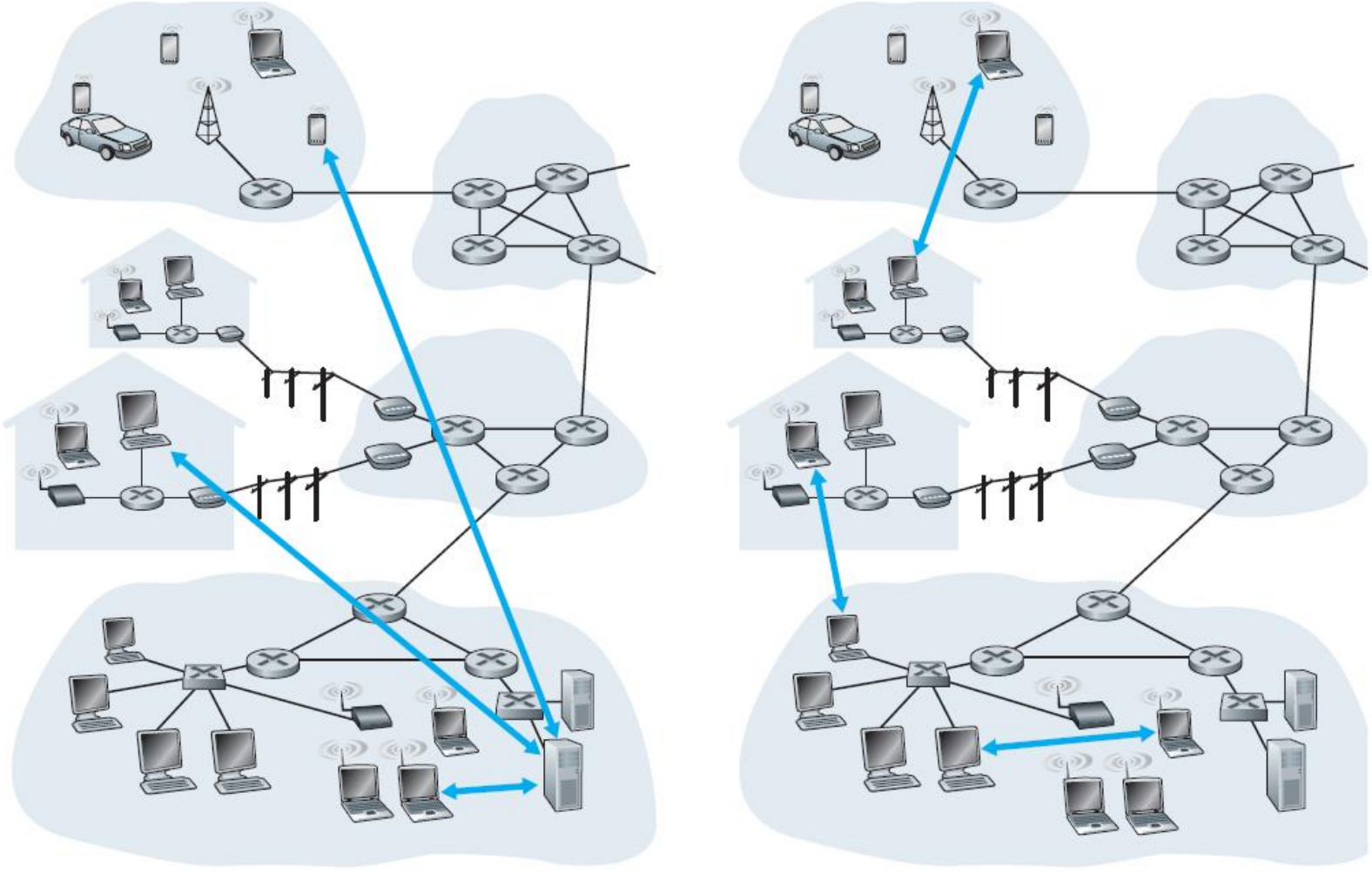
- Several traditional services are still using this paradigm, e.g., WWW, HTTP, FTP, SSH, E-mail, and so on.

- Problems:
  - the server should be a powerful computer
  - there should be a service provider willing to accept the cost and create a powerful server for a specific service

Peer-to-Peer Paradigm

- There is no need for a server process to be running all the time and waiting for the client processes to connect.
- The responsibility is shared between peers.
- E.g.: Internet telephony, BitTorrent, Skype, IPTV
- Advantages:
  - easily scalable and cost-effective in eliminating the need for expensive servers to be running and maintained all the time
- Challenges:
  - more difficult to create secure communication between distributed services
- Few instant messaging application use both client-server and P2P

Client-Server & P2P architectures



a. Client-server architecture

b. Peer-to-peer architecture

Figure 2.2 ♦ (a) Client-server architecture; (b) P2P architecture

Processes Communicating

- A process can be thought of as a program that is running within an end system.
- When processes are running on the same end system, they can communicate with each other with *interprocess communication*
- Processes on two different end systems communicate with each other by *exchanging messages* across the computer network.

Client-Server Programming

- Runs two processes: a client and a server

- A client is a running program that initializes the communication by sending a request;
- A server is another application program that waits for a request from a client.

- A client program is started and stopped by the user whenever it requires.
- A service provider continuously runs the server program

- lifetime of a server is infinite.
- lifetime of a client is finite.

Application Programming Interface

- How does a client process talk with a server process?

- Let, Computer Programming
  - Uses a computer language to define what to do?
  - Has set of instructions for mathematical operations, string manipulation, input/output operation, etc.

- Application Programming
  - Set of instructions to talk with the lowest four layers (in OS)
  - instructs to open a connection, send and receive data, close the connection
  - Set of instruction of this kind is API

Interface Between a Process & Network

- Several APIs have been designed for communication. Three most common APIs
  - Socket interface
  - Transport Layer Interface (TLI)
  - STREAM
- A process sends messages into, and receives messages from, the network through a software interface called a **socket**.

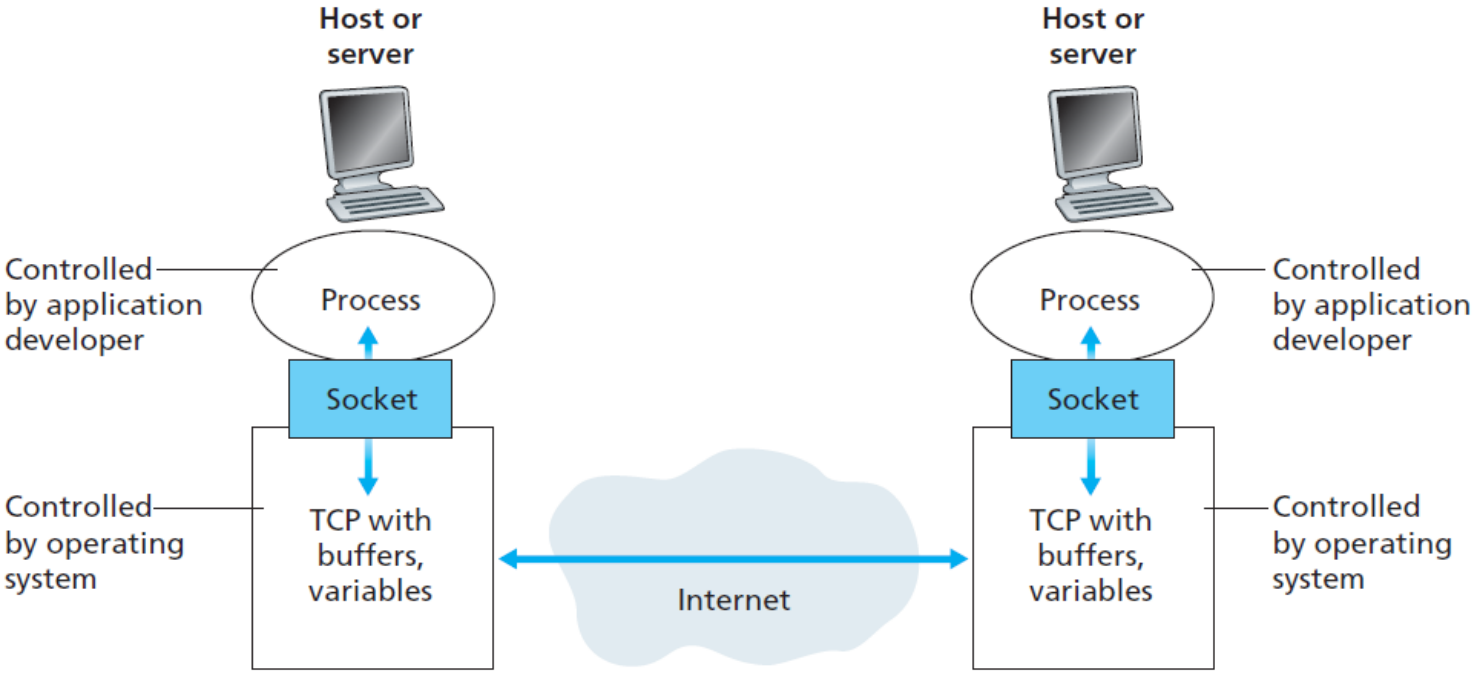


Figure 2.3 ♦ Application processes, sockets, and underlying transport protocol

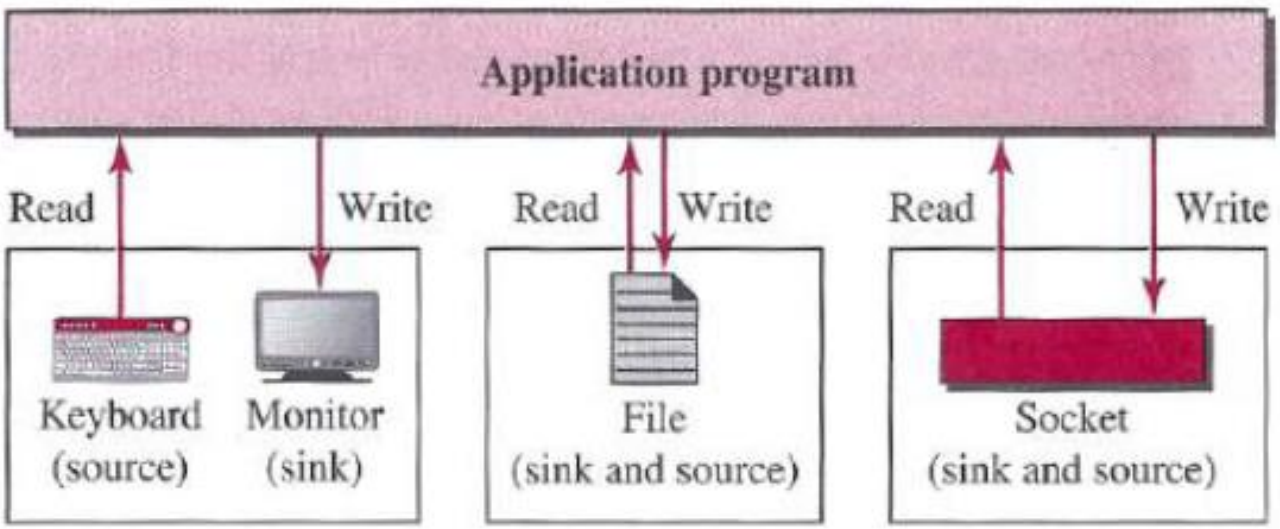
Socket Interface

- Socket interface started in the early 1980s at UC Berkeley as part of a UNIX environment.
- The socket interface is a set of instructions that provide communication between the application layer and the OS.

- The idea of socket allows us to use the set of all instructions already designed in a programming language for other sources and sinks.
- For example,
  - in C, C++, or Java, we have several instructions that can read and write data to other sources and sinks;
  - a keyboard (a source), a monitor (a sink), or a file (source and sink).

- We are adding only new sources and sinks to the programming language without changing the way we send or receive data.

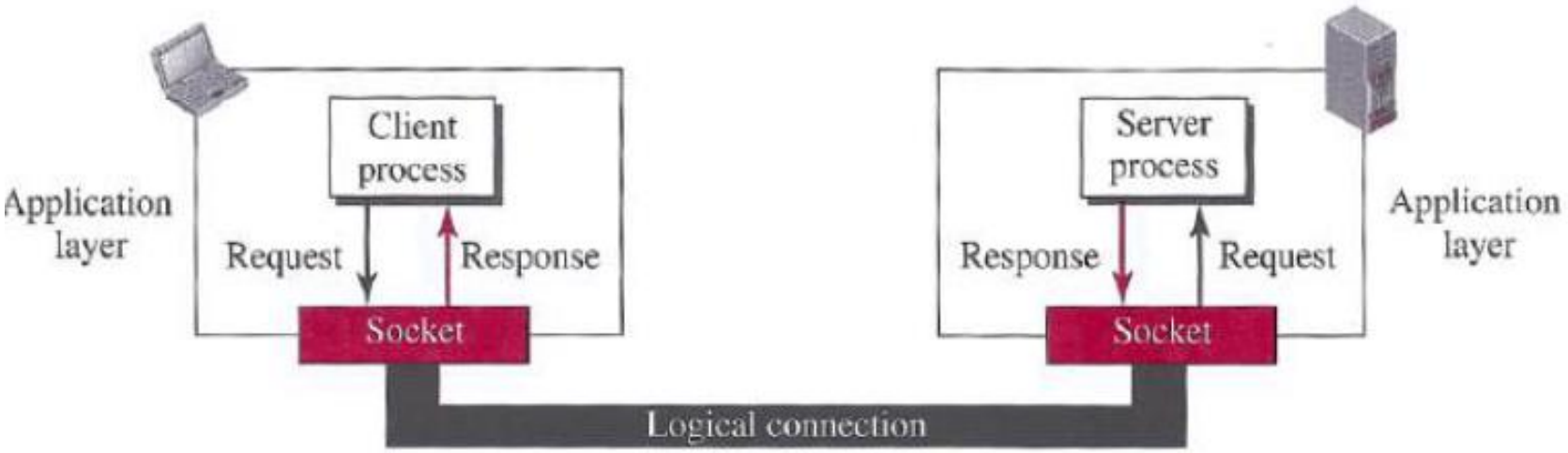
Cont…



- Socket is not a physical entity like files, keyboard, etc.; it is an abstraction

Cont…

- Communication between a client process and a server process is nothing but communication between two sockets



- Need a pair of socket addresses for communication:
  - a local socket address and a remote socket address.

Cont…

- A socket address should
  - **first** define the computer on which a client or a server is running.
    - a computer in the Internet is uniquely defined by its IP address
  - **Then**, we need another identifier to define the specific client or server involved in the communication
    - an application program can be defined by a port number
    - Popular applications have been assigned specific port numbers
    - Few port numbers: Web server=80, Mail Server=25
- So, socket address = {IP address, port number}

Socket address

Finding Socket Addresses

- How can a client / server find a pair of socket addresses for communication?

- In Server Site:
  - The server needs a local (server) and a remote (client) socket address for communication.

- In Client Site:
  - The client also needs a local (client) and a remote (server) socket address for communication.

Cont…

- In Server Site
  - *Local Socket Address (server)*: Provided by the OS; IP and Port number needs to be defined. For standard services, port numbers are well-known.
  - *Remote Socket Address (client)*: The server can find this socket address from the REQ packet when a client tries to connect to the server.

- In Client Site
  - *Local Socket Address (client)*: Provided by the OS; The port number is assigned to a client process each time the process needs to start the communication; the ephemeral port numbers are assigned to client
  - *Remote Socket Address (server)*: We know port-number of standard application, but don't know IP. We know only URL (e.g. www.gmail.com) , and DNS gives server socket address corresponding to URL.

Transport Layer Services

- The choice of the transport layer protocol seriously affects the capability of the application processes.
- broadly classify the possible transport layer services along four dimensions:
  - Reliable data transfer
  - Throughput
  - Timing
  - Security

- use UDP
  - if it is sending small messages
  - if the simplicity and speed is more important for the application than reliability
  - for lightweight transport protocol, providing minimal services
- use TCP
  - if it needs to send long messages and require reliability
  - for providing security it use SSL (Secure Socket Layer)

Cont…

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

Figure 2.5 ♦ Popular Internet applications, their application-layer protocols, and their underlying transport protocols

N/W Application vs Application Layer Protocol

- An application-layer protocol is only one piece of a network application.

- Example:
  - Web application consists of many components, including
    - a standard for document formats (e.g. HTML),
    - Web browser (e.g. Firefox, Chrome),
    - Web server (e.g. Apache, Microsoft Server), and
    - application-layer protocol (e.g. HTTP) which defines the format and sequence of messages exchanged between browser and Web server.

  - Internet e-mail application also has many components, including
    - mail servers that house user mailboxes;
    - mail clients (such as Microsoft Outlook, Gmail) that allow users to read and create messages;
    - a standard for defining the structure of an e-mail message; and
    - application-layer protocol (e.g. SMTP) that define how messages are passed between servers, how messages are passed between servers and mail clients, and how the contents of message headers are to be interpreted.

Thanks!