CS6111 – Computer Networks Lab (V Semester - Q Batch)

Ms.J.Deepika Roselind

What is Computer Hardware?

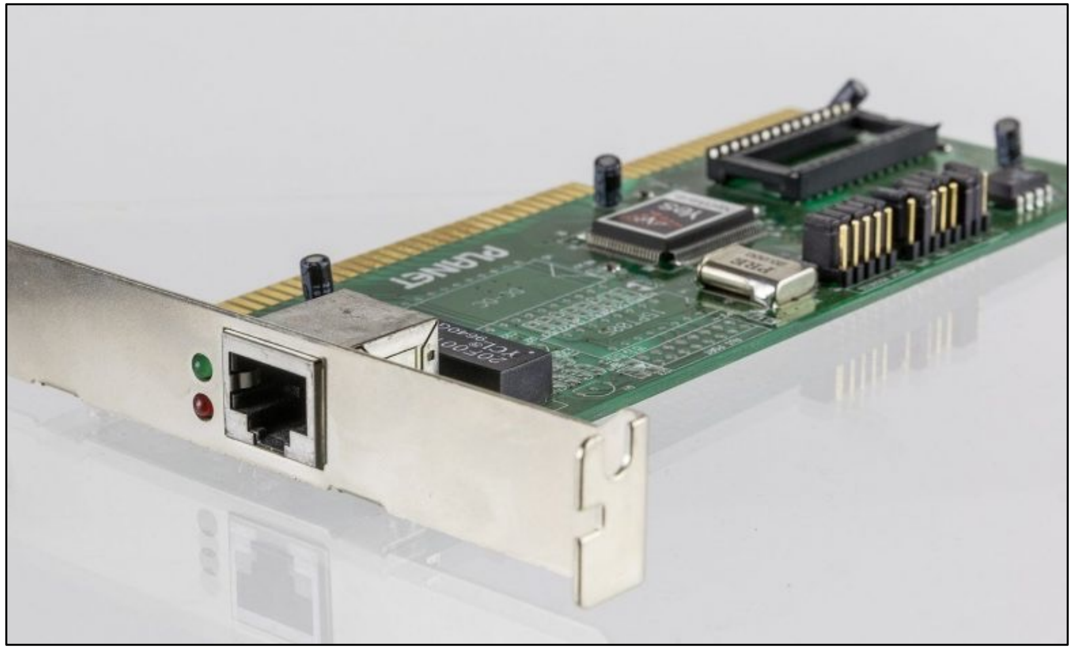
- Computer hardware describes the physical components of an analog or digital computer.
- It consists of written, machine-readable instructions or programs that tell physical components what to do and when to execute the instructions.
- Computer hardware can be categorized as being either **internal** or **external** components
  - ***Internal hardware*** *components are those necessary for the proper functioning of the computer*
  - ***External hardware*** *components are attached to the computer to add or enhance functionality*

What Are Internal Computer Hardware Components?

- Internal components collectively process or store the instructions delivered by the program or operating system (OS).
- These include the following:
    - Motherboard
    - CPU
    - RAM
    - Hard Drive
    - Solid State Drive (SSD)
    - Optical Drive
    - Graphics Processing Unit (GPU)
    - Network Interface Card (NIC)
    - Cooling Fan

Network Interface Controller

- A network interface controller is a computer hardware component that connects a computer to a computer network
- Implements the electronic circuitry required to communicate using a specific physical layer and data link layer standard such as Ethernet or Wi-Fi.



NIC Functionalities

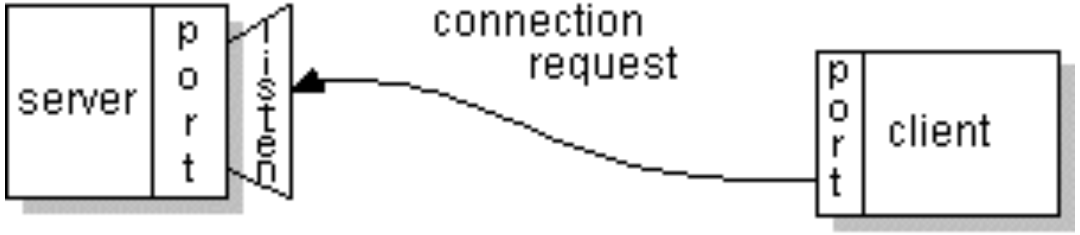
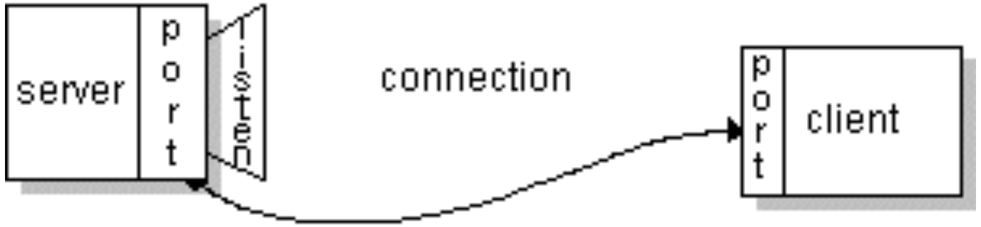
- Provides a base for a full network protocol stack, allowing communication among computers on the same local area network (LAN) and large-scale network communications through routable protocols, such as Internet Protocol (IP).
- It allows computers to communicate over a computer network, either by using cables or wirelessly.
- The NIC is both a physical layer and data link layer device
  - it provides physical access to a networking medium and, for IEEE 802 and similar networks
  - provides a low-level addressing system through the use of MAC addresses that are uniquely assigned to network interfaces.

Computer Network

- A **computer network** is a ***set of computers*** sharing resources located on or provided by network nodes.
- The nodes of a computer network can include *personal computers, servers, networking hardware, or other specialized or general-purpose hosts*.
- The computers use common communication protocols over digital interconnections to communicate with each other
- They are identified by network addresses, and may have hostnames.

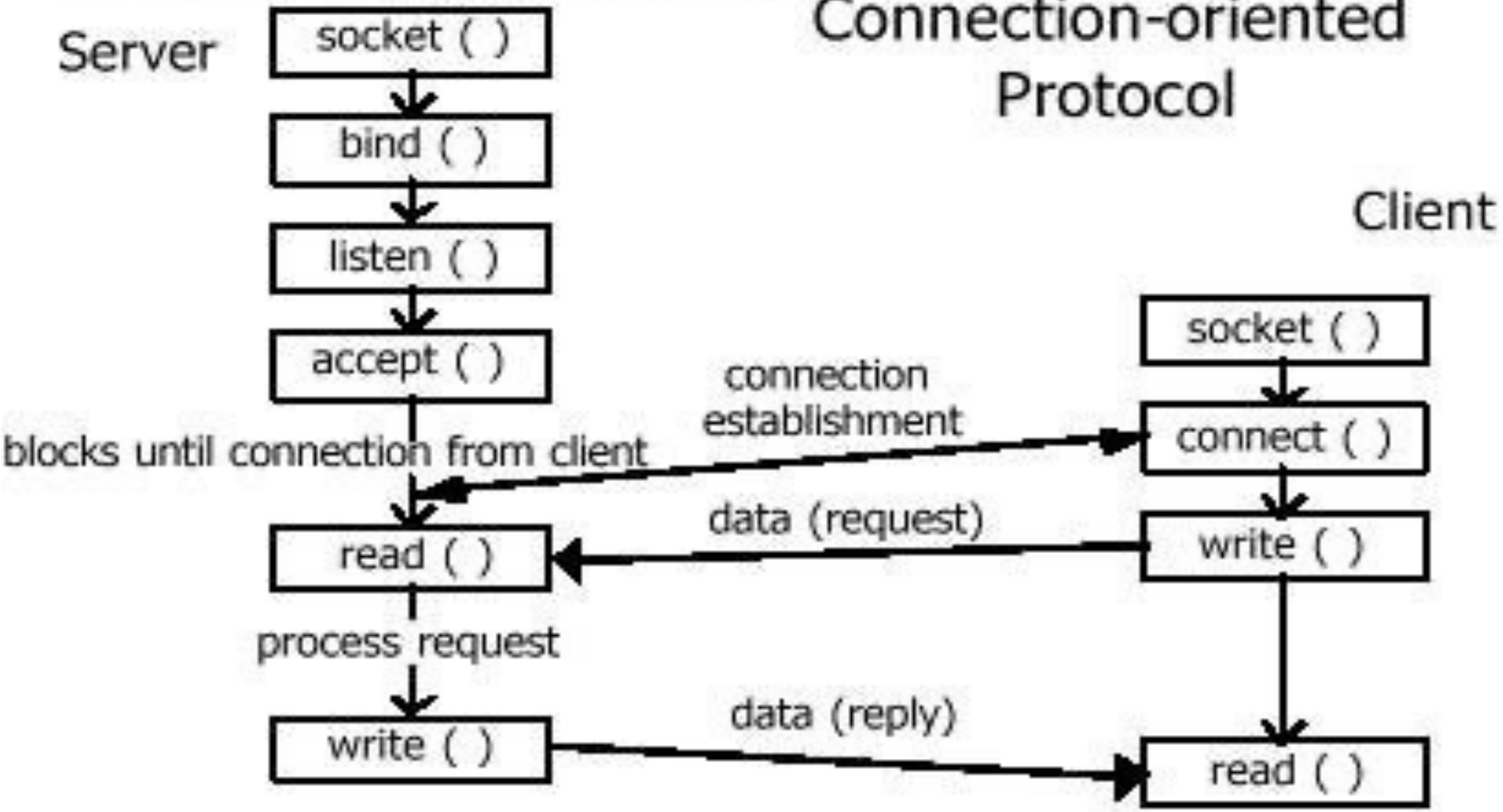
Socket Programming

- A *socket* is one endpoint of a two-way communication link between two programs running on the network.
- A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.
- An endpoint is a combination of an IP address and a port number.

Server-side	Client-side
1. The server just waits, listening to the socket for a client to make a connection request	The client knows the hostname of the machine on which the server is running and the port number on which the server is listening
 server port listen connection request port client	**2. To make a connection request, the client tries to connect with the server on the server's machine and port.**
	3. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection (assigned by the system).
4. The server accepts the connection	
5. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client.	
It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.	**6. A socket is successfully created, and the client can use the socket to communicate with the server**

Socket Programming

Connection-oriented Protocol



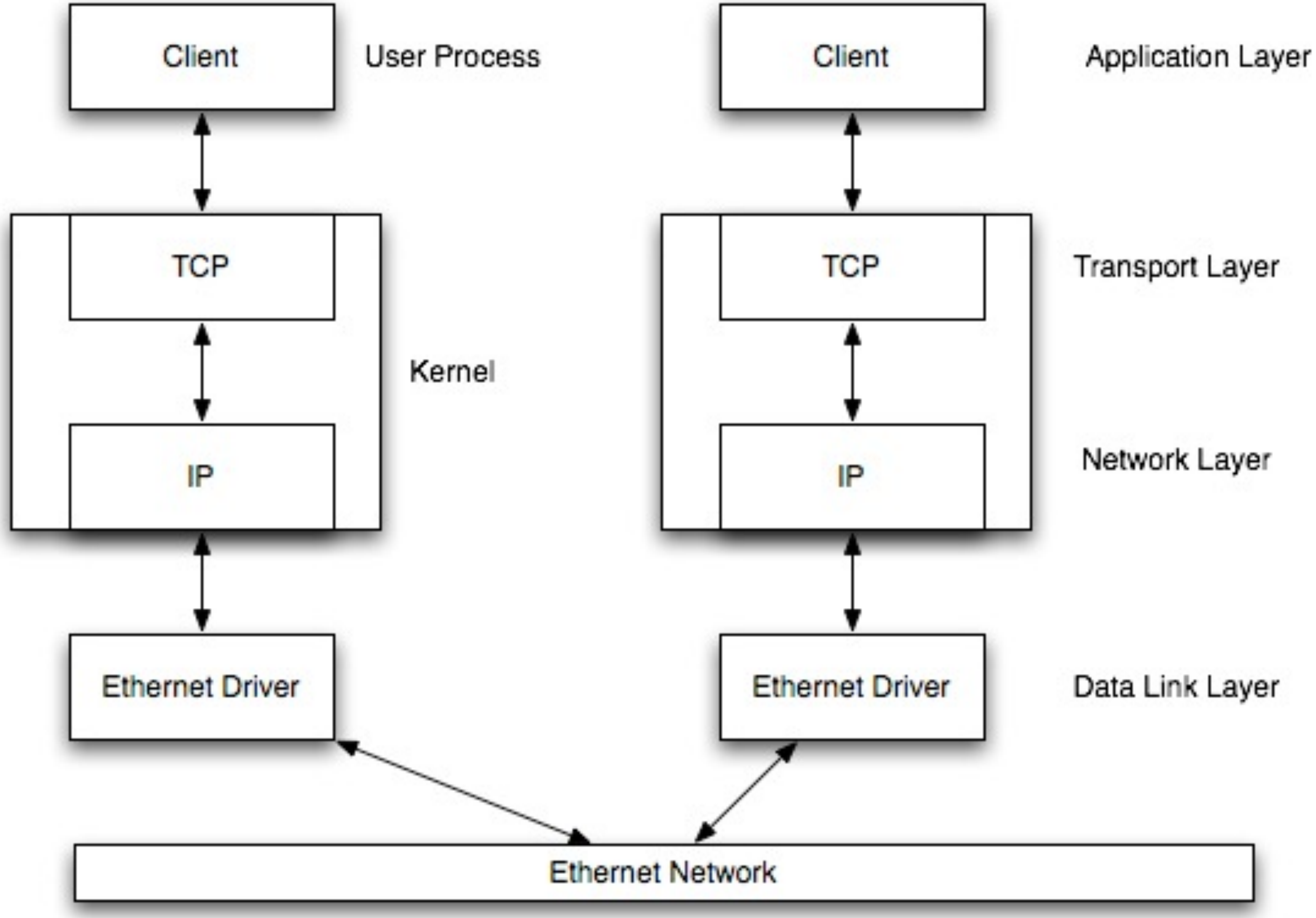
The Basics

- **Program:** A program is an executable file residing on a disk in a directory. A program is read into memory and is executed by the kernel
- **Process:** An executing instance of a program is called a *process*. Sometimes, *task* is used instead of process with the same meaning. UNIX guarantees that every process has a unique identifier called the *process ID*.
- **File descriptors:** File descriptors are normally small non-negative integers that the kernel uses to identify the files being accessed by a particular process. Whenever it opens an existing file or creates a new file, the kernel returns a file descriptor that is used to read or write the file.

The Client-server Model

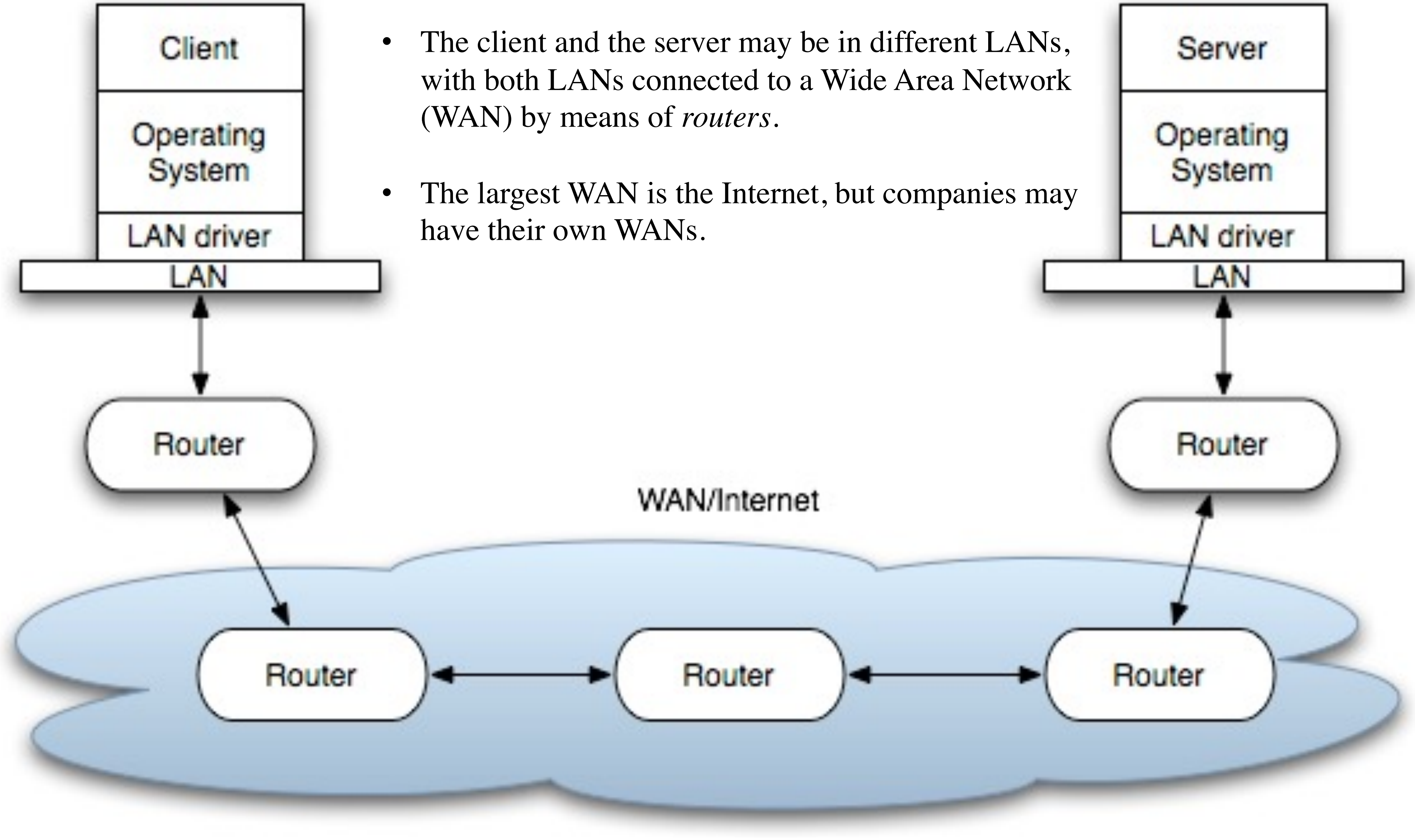
- The client-server model is one of the most used communication paradigms in networked systems.
- Clients normally communicates with one server at a time.
- A server communicates with multiple clients.
- Client and servers communicate by means of multiple layers of network protocols - TCP/IP protocol suite.

The Client-server Model



- The client and the server may be in different LANs, with both LANs connected to a Wide Area Network (WAN) by means of *routers*.

- The largest WAN is the Internet, but companies may have their own WANs.



Client and server on different LANs connected through WAN/Internet.

Server-side	Client-side
1. Create a socket with the socket() system call	1. Create a socket with the socket() system call
2. Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine	-
3. Listen for connections with the listen() system call	-
4. Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.	2. Connect the socket to the address of the server using the connect() system call
5. Send and receive data	3. Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls.

Socket Types

- **Stream Sockets** - Stream Sockets use TCP (Transmission Control Protocol)
- **Datagram Sockets** - Datagram Sockets use UDP (Unix Datagram Protocol)

Basis	Transmission control protocol (TCP)	User datagram protocol (UDP)
Type of Service	TCP is a connection-oriented protocol	UDP is the Datagram-oriented protocol
Reliability	TCP is reliable as it guarantees the delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
Error checking mechanism	TCP provides extensive error-checking mechanisms. It is because it provides flow control and acknowledgment of data.	UDP has only the basic error checking mechanism using checksums.
Acknowledgment	An acknowledgment segment is present.	No acknowledgment segment.
Sequence	Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver.	There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer.
Speed	TCP is comparatively slower than UDP.	UDP is faster, simpler, and more efficient than TCP.
Retransmission	Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in the User Datagram Protocol (UDP).
Weight	TCP is heavy-weight.	UDP is lightweight.
Handshaking Techniques	Uses handshakes such as SYN, ACK, SYN-ACK	It’s a connectionless protocol i.e. No handshake
Protocols	TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet.	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.
Stream Type	The TCP connection is a byte stream.	UDP connection is message stream.
Overhead	Low but higher than UDP.	Very low.

Sample code & Explanation

- Sockets Tutorial.pdf