Computer Networks

BE Computer 7th Sem

Er. Anuj Sherchan

Assistant Professor , Department of Electronics and Computer Engineering

Pokhara Engineering College

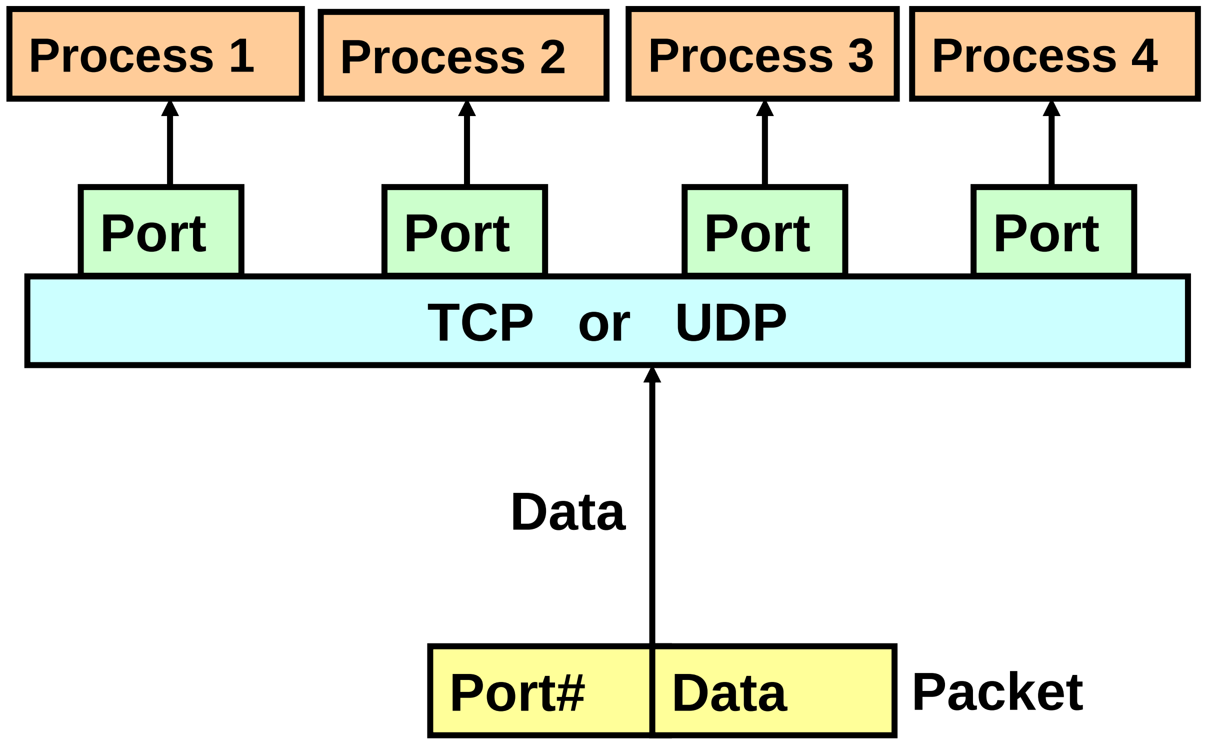
Unit 6 : Transport Layer

- Outline:
- Port Addressing overview
- Process to Process delivery : multiplexing and demultiplexing
- TCP services , features, segment headers , well known ports and Handshaking
- UDP services ,features , segment headers , well known ports
- Concept of Socket Programming

Port Addressing Overview

- Ports is the circuit connection point on a front end.
- Socket is the program associated with every process.
- Every process is identified by 16 bit port number.
- At Transport layer, address is needed to choose multiple process running on the destination host called port number .
- Destination port number for delivery.
- Source port number for reply.

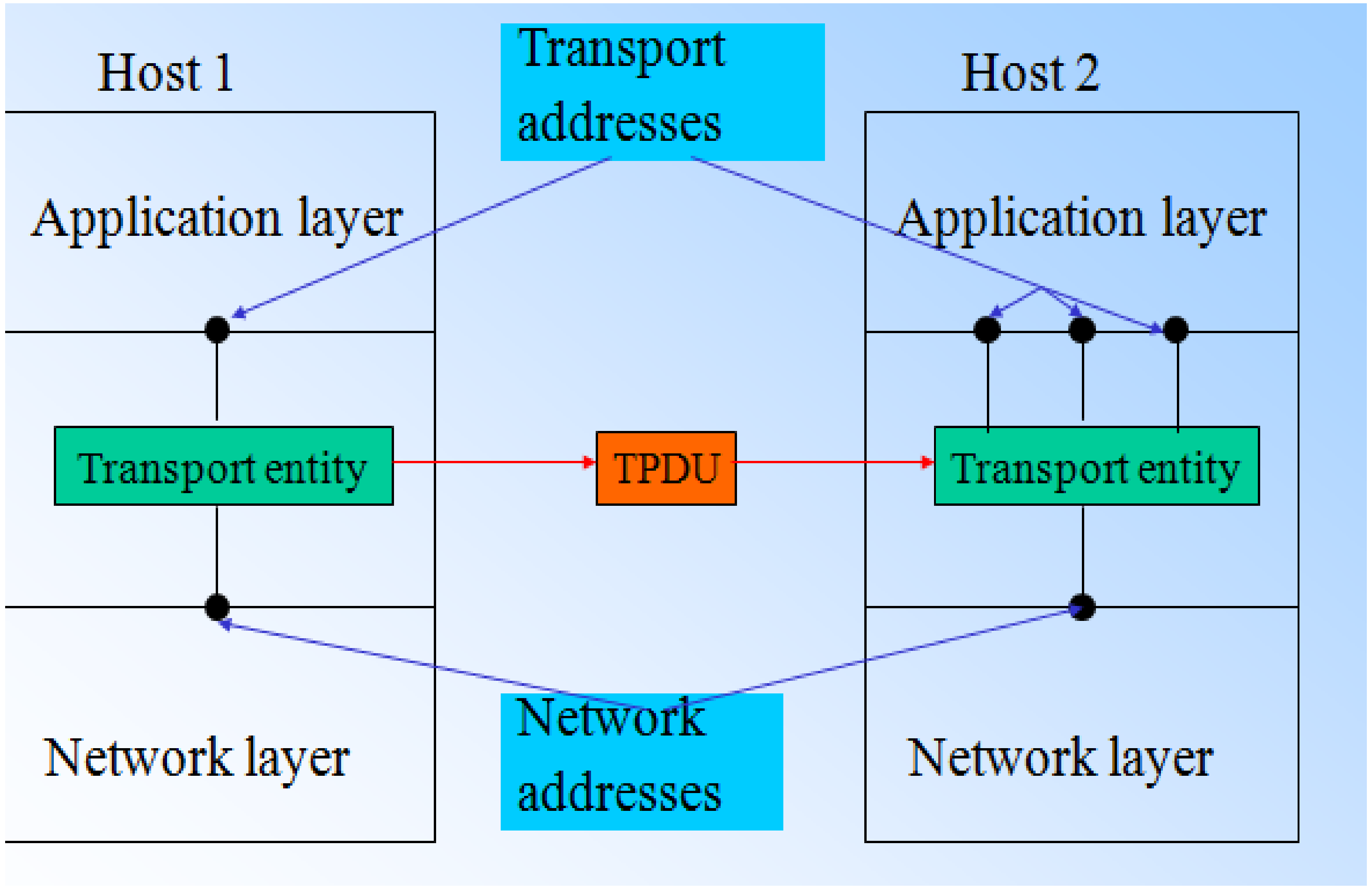
Port Addressing Overview



Port Addressing Overview

- Port is represented by 16 bit integer value between 0 to 65,535.
- Internet Assigned Number Authority(IANA) has divided port number into three ranges.
- 1. Well-known ports ranges from 0 to 1023.
- Assigned, Registered and controlled by IANA.
- Example: ftp 21/tcp, telnet 23/tcp, smtp 25/tcp, login 513/tcp.
- 2. Registered Ports.
- Ranges from 1024 to 49,151
- Not assigned and controlled, but registered by IANA.
- 3. Dynamic Ports.
- Ranges from 49,152 to 65,535.
- Neither registered nor controlled by IANA.
- Used by any process temporarily.

Transport Layer: Introduction



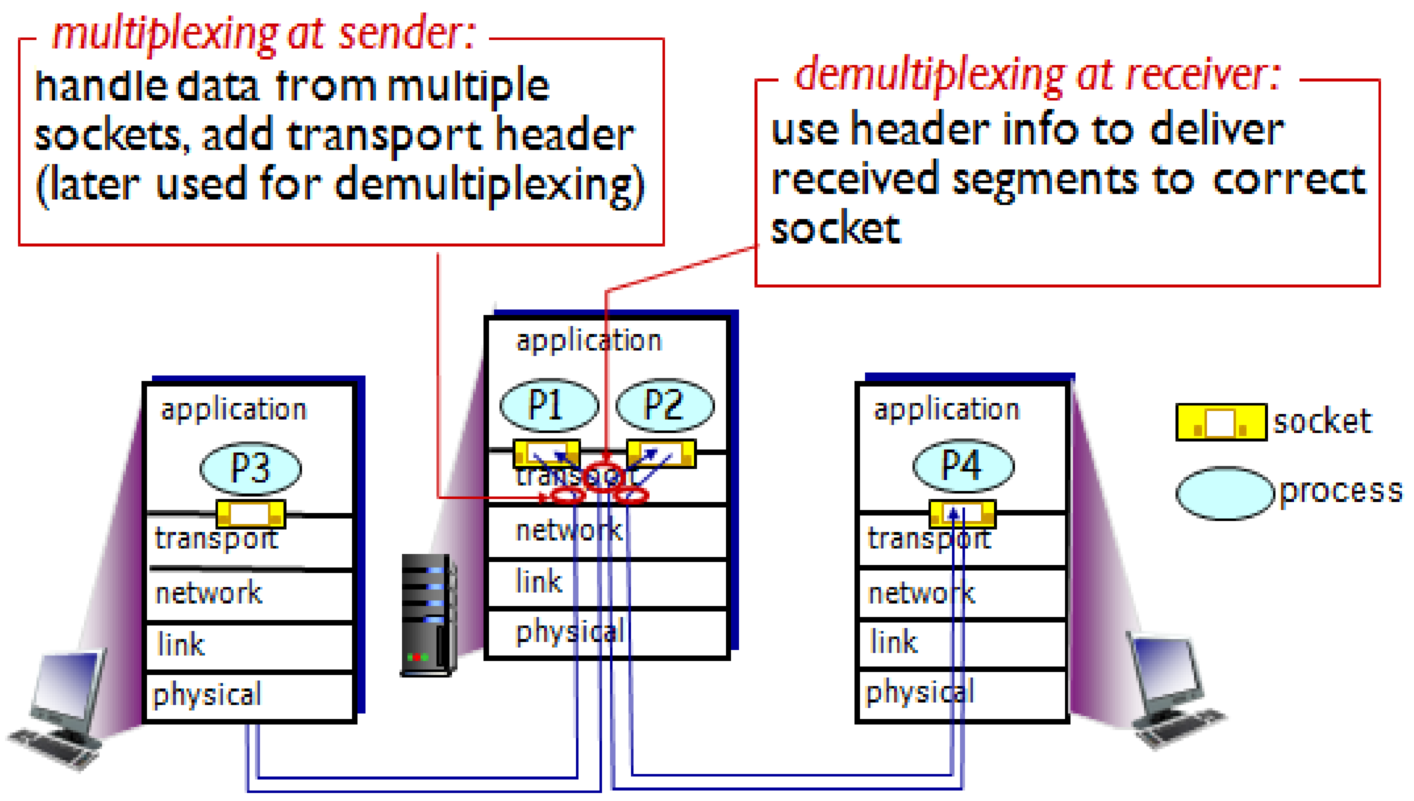
Transport Layer: Introduction

- Transport Layer together with the network layer is the heart of protocol hierarchy.
- Transport Layer provide reliable, cost-effective data transport from the source machine to the destination machine.
- Transport Layer is independent of the physical network or networks currently in use.

Transport Layer: Services

- Provides End to End Connection.
- Concerned with transportation issues between host.
- Receives information from upper layer and segment(divide)it into packets .
- Transport layer can be either connection oriented or connection less oriented.
- Assures that the entire message is correctly delivered at the receiving side.
- Transport layer at the destination re-assembles the packets.
- Fault detection and recovery.
- Information flow control. Protocol used are TCP and UDP.

Transport Layer: Multiplexing



multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)
demultiplexing at receiver:
use header info to deliver received segments to correct socket
application
P1
P2
transport
network
link
physical
application
P3
transport
network
link
physical
application
P4
transport
network
link
physical
socket
process

Transport Layer: Multiplexing

- Multiplexing: several conversations onto connections, virtual circuits, and physical links plays a role in several layers of the network architecture.
- In the transport layer the need for multiplexing can arise in a number of ways.
- For example, if only one network address is available on a host, all transport connections on that machine have to use it.
- When a TPDU (Transport Protocol Data Unit)comes in, some way is needed to tell which process to give it to.
- This situation is called “Upward multiplexing”

Transport Layer: Multiplexing

- For example, that a subnet uses virtual circuits internally and imposes a maximum data rate on each one.
- If a user needs more bandwidth than one virtual circuit can provide, a way out is to open multiple network connections and distribute the traffic among them on a round-robin basis.
- This mode of operation is called “Downward multiplexing”.

Transport Layer: Multiplexing

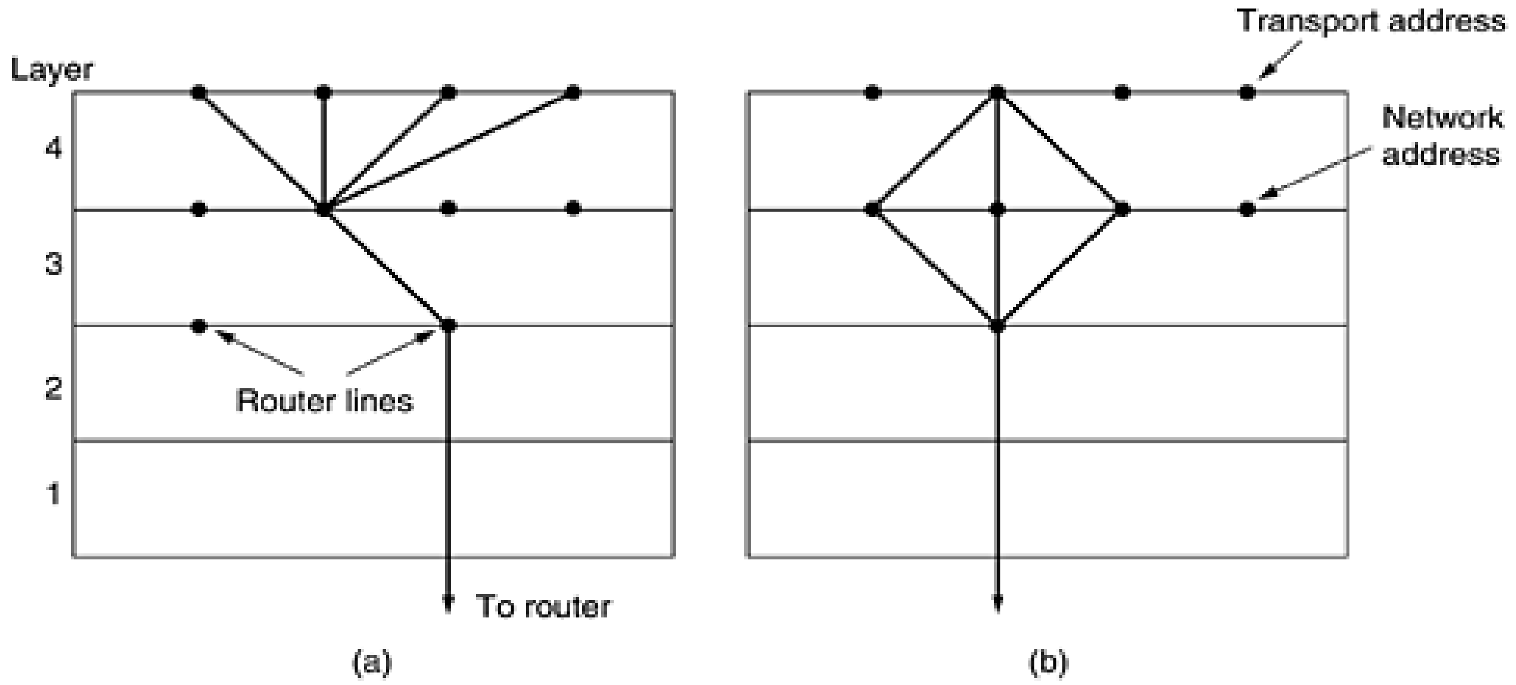
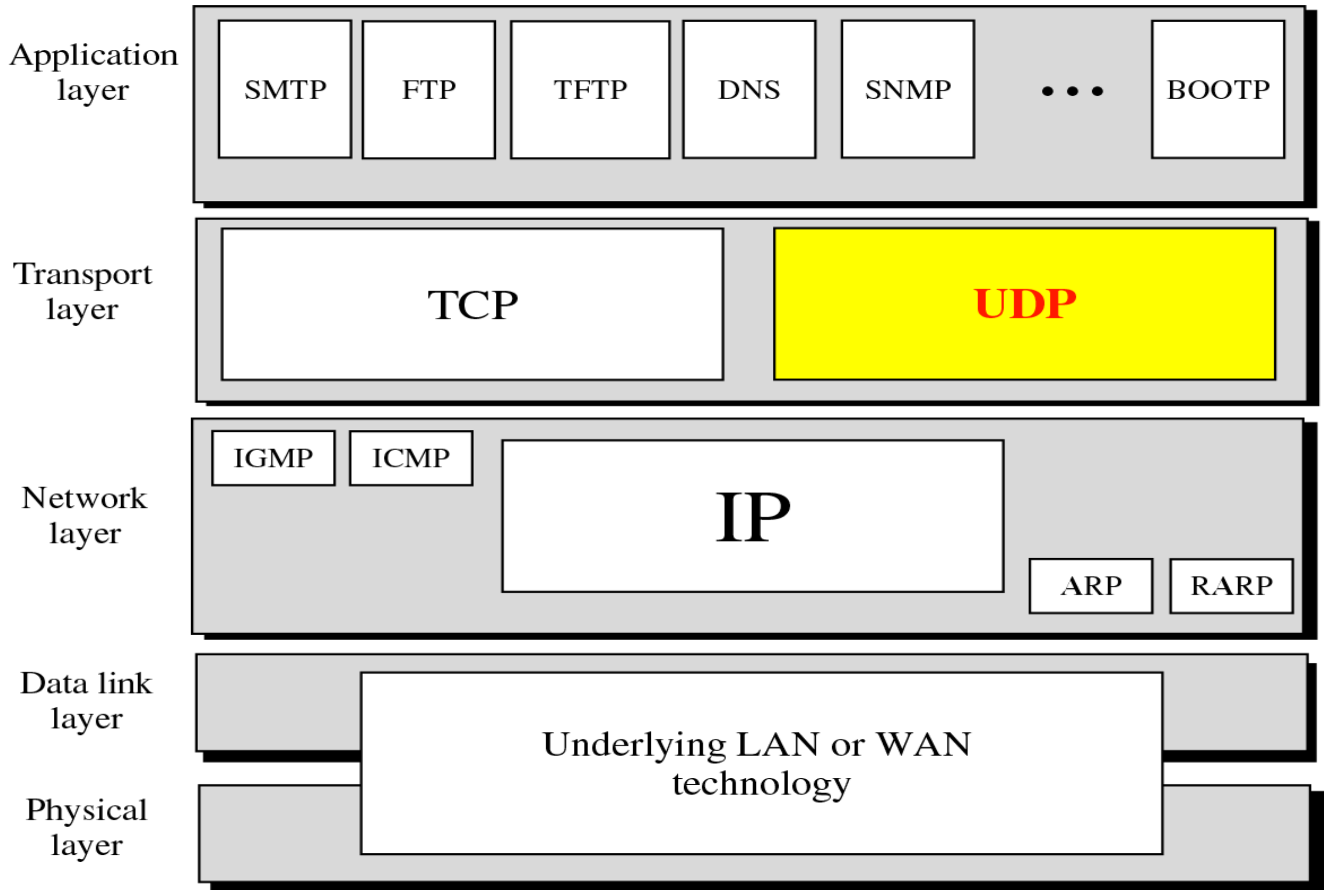


Fig a: Upward Multiplexing. **Fig b :Downward Multiplexing**.

Transport Layer: Protocol

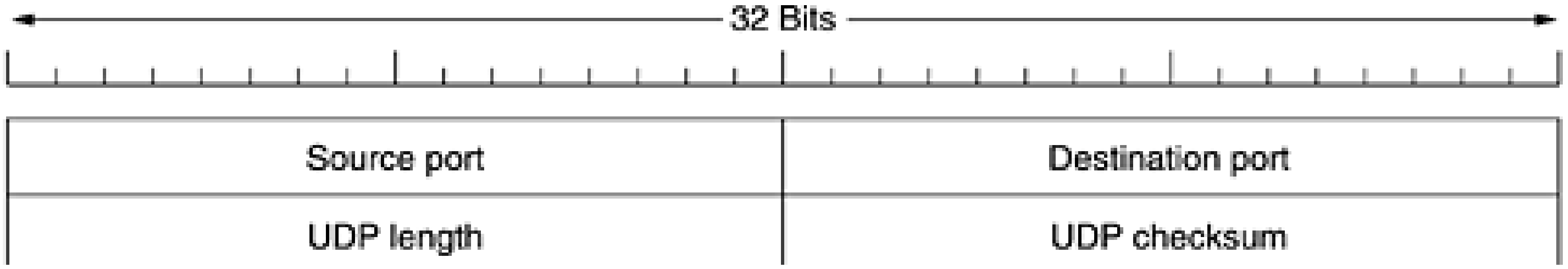
- The transport layer also defines end-to-end connectivity between host applications.
- The transport service is implemented by a transport protocol used between the two transport entities.
- In some ways, transport protocols resemble the data link protocols.
- Transport layer protocols are Transmission Control (TCP) and User Datagram Protocol (UDP).
- The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. In the following sections we will study both of them.

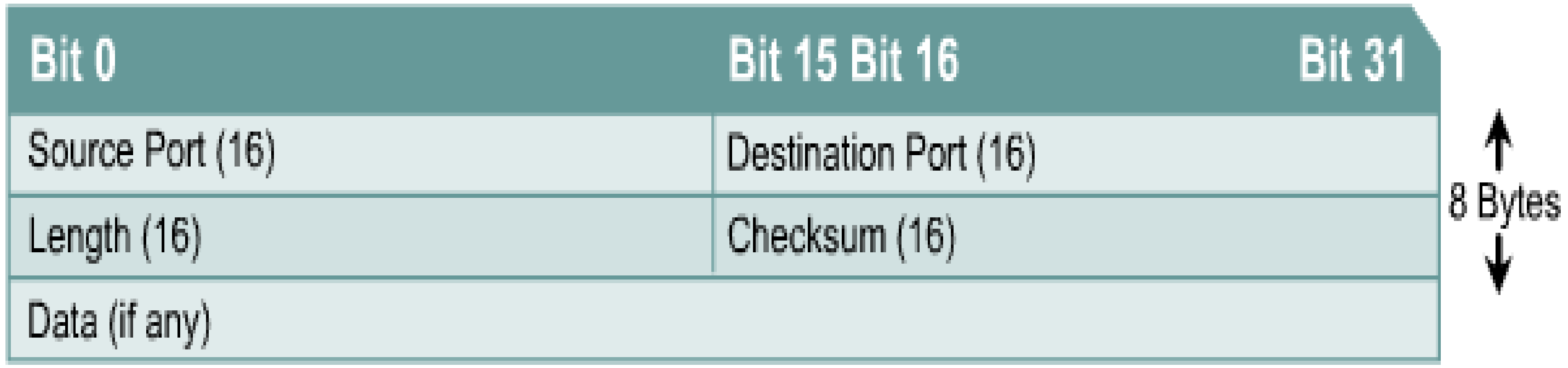
Transport Layer: Protocol



Transport Layer : User Datagram Protocol(UDP)

- User Datagram protocol is connectionless transport protocol.
- UDP is a simple protocol that exchanges datagram without guaranteed delivery.
- It relies on higher-layer protocols to handle errors and retransmit data.





Transport Layer : User Datagram Protocol(UDP)

- Source port – Number of the port that sends data.
- Destination port – Number of the port that receives data.
- Length – Number of bytes in header and data.
- Checksum – Calculated checksum of the header and data fields.
- Data – Upper-layer protocol data.

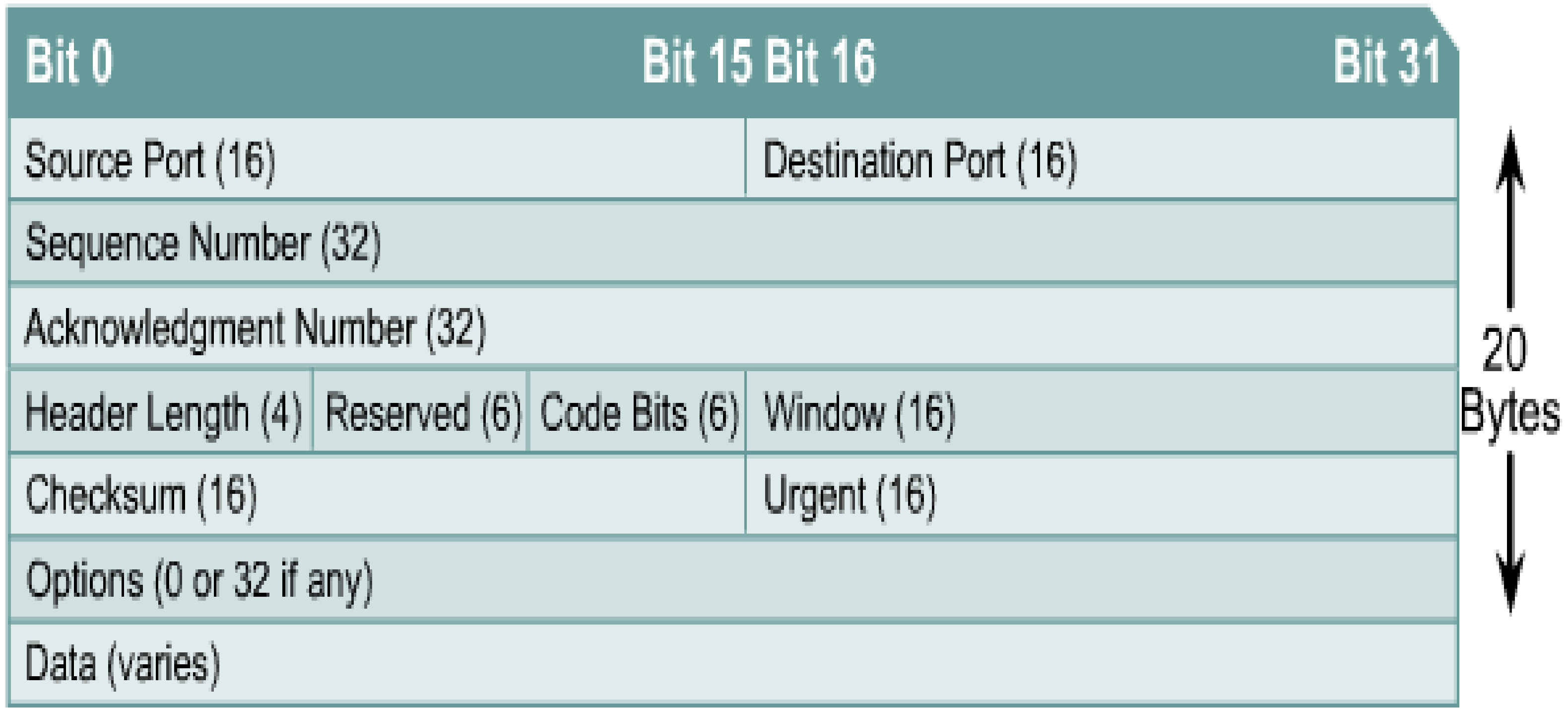
Transport Layer : User Datagram Protocol(UDP)

- UDP does not use windows or ACKs.
- Reliability is provided by application layer protocols.
- UDP is designed for applications that do not need to put sequences of segments together.
- Example:- TFTP, SNMP, DHCP, DNS.

Transport Layer: Transmission Control Protocol(TCP)

- TCP is a connection-oriented transport layer protocol that provides reliable full-duplex data transmission.
- TCP is part of the TCP/IP protocol stack.
- In a connection-oriented environment, a connection is established between both ends before the transfer of information can begin.
- TCP breaks messages into segments, reassembles them at the destination, and resends anything that is not received.
- TCP supplies a virtual circuit between end-user applications.
- TCP uses only a single type of protocol data unit, called a TCP segment.

Transport Layer: Transmission Control Protocol(TCP)

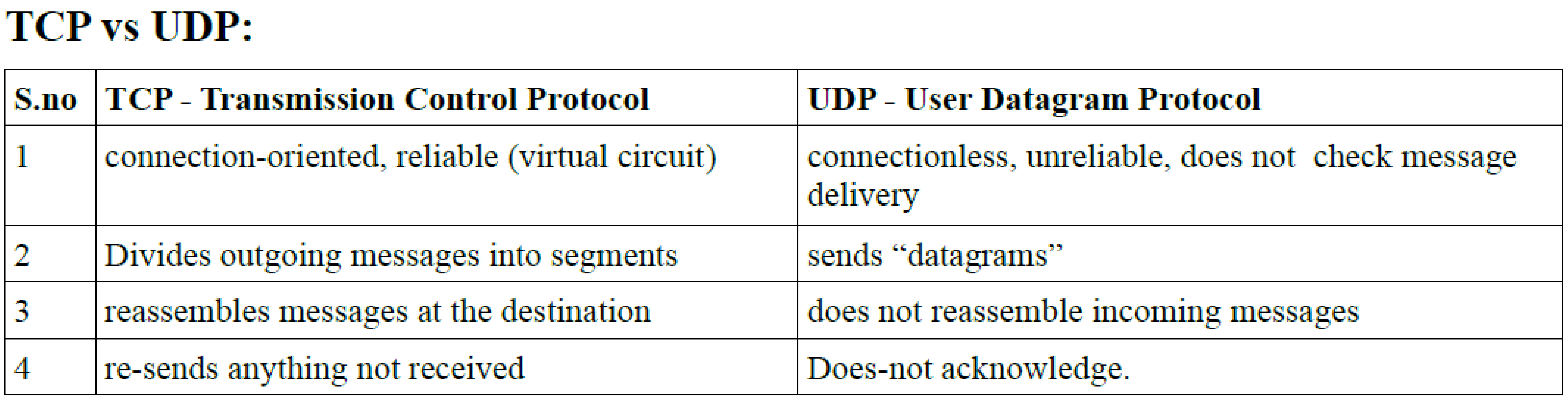
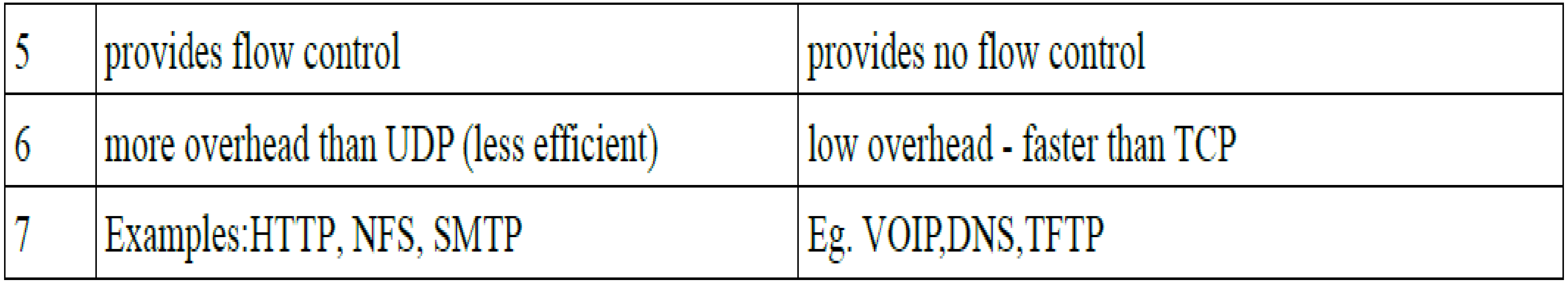


Bit 0			Bit 15 Bit 16	Bit 31
Source Port (16)			Destination Port (16)	
Sequence Number (32)				
Acknowledgment Number (32)				
Header Length (4)	Reserved (6)	Code Bits (6)	Window (16)	
Checksum (16)			Urgent (16)	
Options (0 or 32 if any)				
Data (varies)				

Transport Layer: Transmission Control Protocol(TCP)

- Source port – Number of the port that sends data.
- Destination port – Number of the port that receives data.
- Sequence number – Number used to ensure the data arrives in the correct order.
- Acknowledgment number – Next expected TCP octet.
- HLEN – Number of 32-bit words in the header.
- Reserved – Set to zero.
- Code bits – Control functions, such as setup and termination of a session..
- Window – Number of octets that the sender will accept.
- HLEN – Number of 32-bit words in the header.
- Reserved – Set to zero.

Transport Layer: TCP vs UDP

TCP vs UDP:

S.no	TCP - Transmission Control Protocol	UDP - User Datagram Protocol
1	connection-oriented, reliable (virtual circuit)	connectionless, unreliable, does not check message delivery
2	Divides outgoing messages into segments	sends “datagrams”
3	reassembles messages at the destination	does not reassemble incoming messages
4	re-sends anything not received	Does-not acknowledge.
5	provides flow control	provides no flow control
6	more overhead than UDP (less efficient)	low overhead - faster than TCP
7	Examples:HTTP, NFS, SMTP	Eg. VOIP,DNS,TFTP

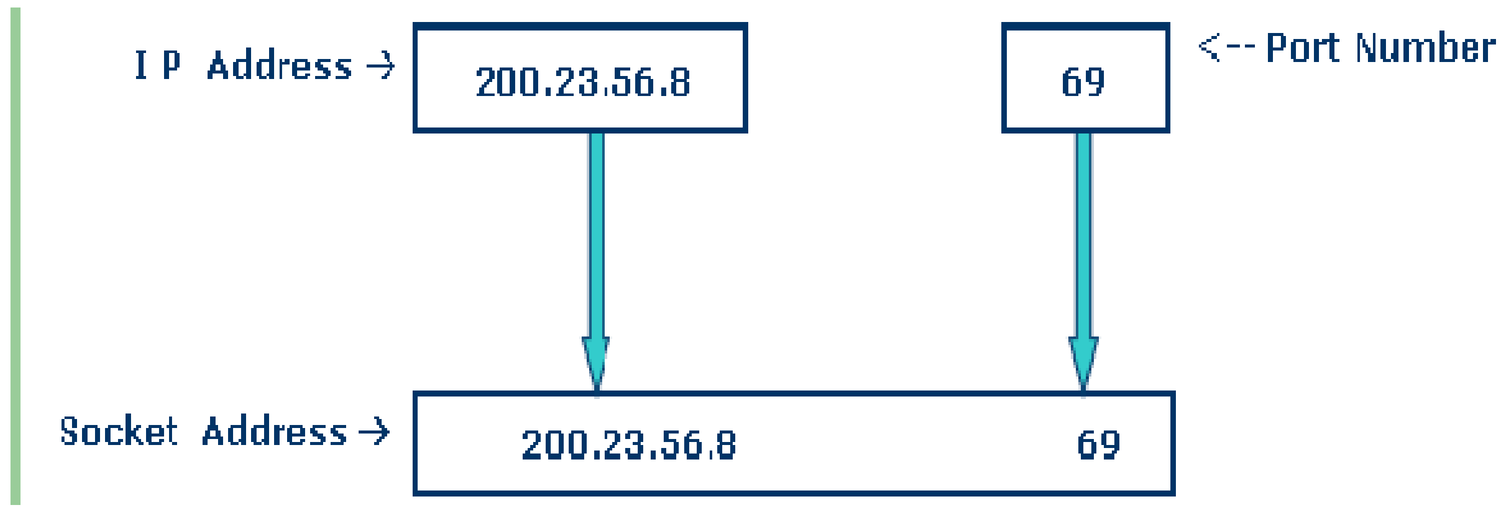
Concept of Socket Programming

- **Socket**
- Allow communication between two different process on the same or different machine.
- Socket is end point of two way communication link.
- Socket provides interface for programming network at the transport layer.
- Every TCP connection is uniquely identified by its two endpoint.
- Multiple connection between host and server.

Socket Address

- Process to Process delivery of data needs two identifiers, IP Address and Port Number at each endpoint .
- Socket Address → combination of IP address and a Port number.

Example



Socket Address

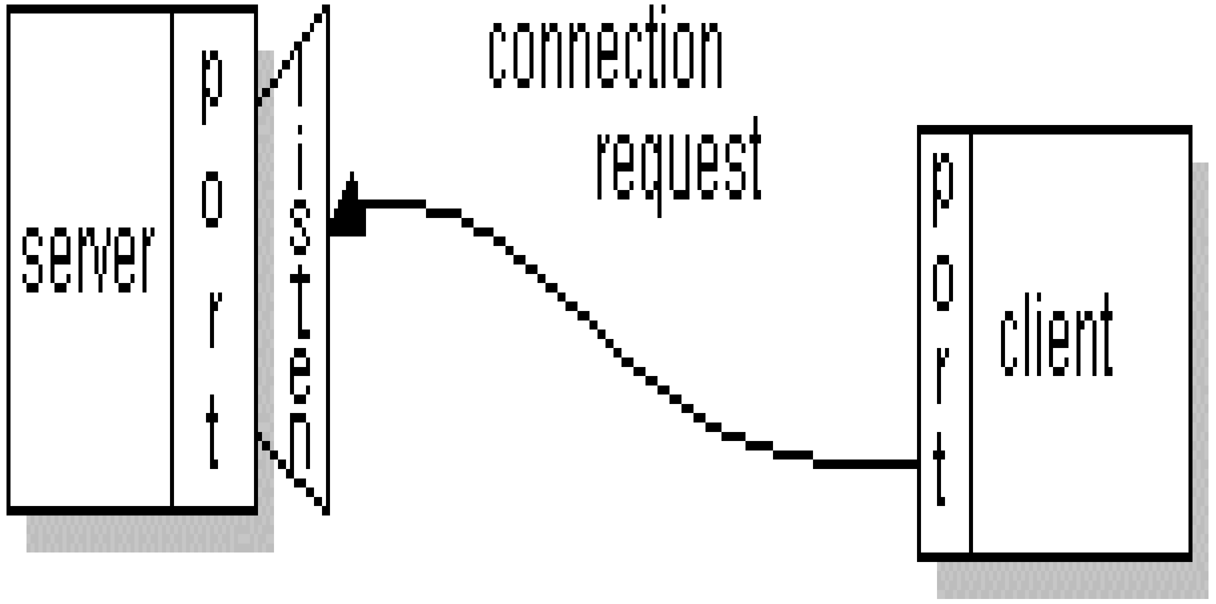
- Transport Layer Protocol needs a pair of Socket addresses
  - Client Socket Address
    - Uniquely defines the Client Process
  - Server Socket Address
    - Uniquely defines Server Process

- Both Socket Addresses contain IP Header and Transport Layer Protocol Header
    - IP Header contains IP Addresses
    - TCP & UDP Header contains the Port Numbers

Socket Communication

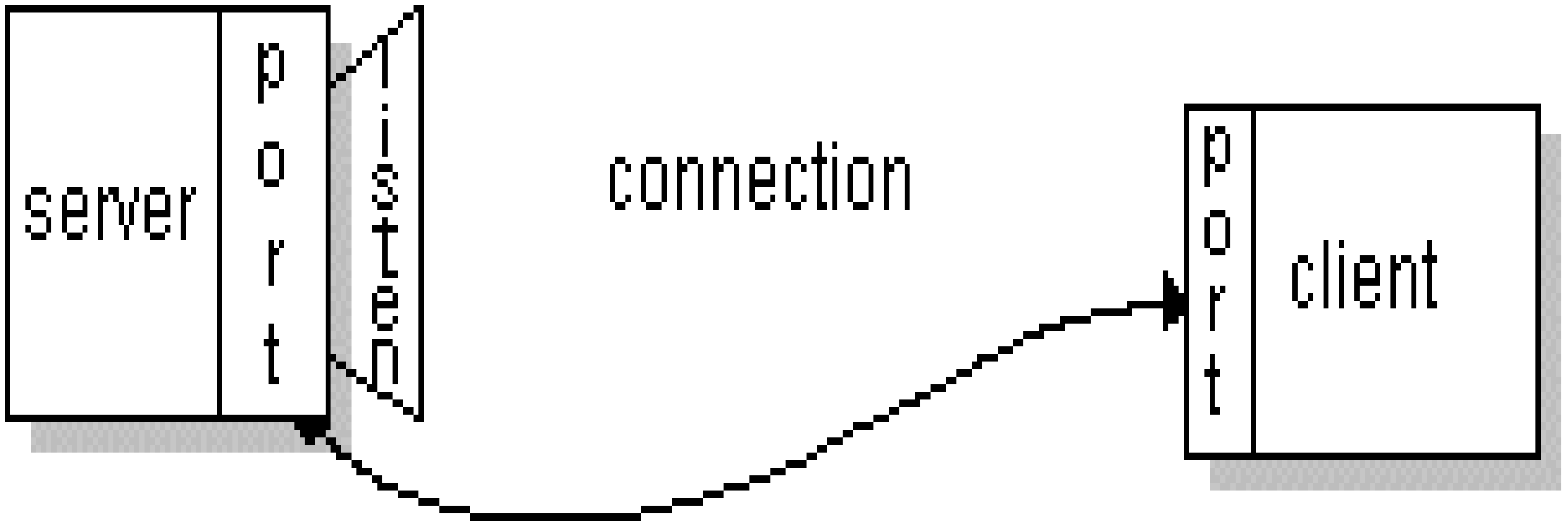
- A server (program) runs on a specific computer and has a socket that is bound to a specific port.
- The server waits and listens to the socket for a client to make a connection request
- The client makes a connection request knowing the hostname and port Number on which the server is listening.
- The client binds to its local port number that it will use during this connection.

Socket Communication



Socket Communication

- 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.
- If connection is established, 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.

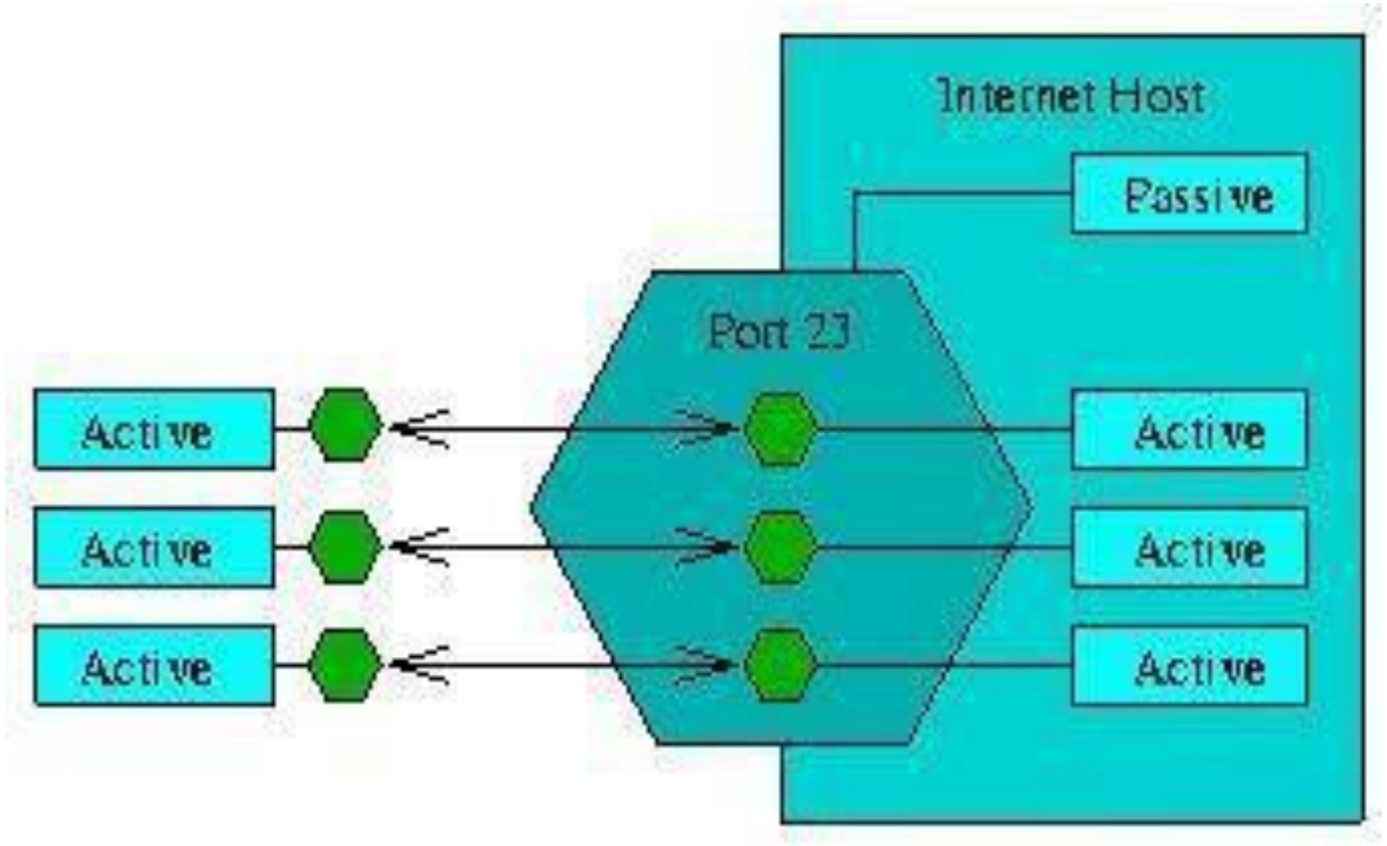


Socket Communication

- If the connection is accepted by the Client, a socket is successfully created and the client can use the socket to communicate with the server.
- The client and server can now communicate by using their sockets.
- **Types of Socket**
- Active Socket
    - Connected to a remote active socket via an open data connection.
    - Closing the connection, destroys the active sockets at each point.

- Passive Socket
    - Connected, but awaits an incoming connection, which will generate new active socket.

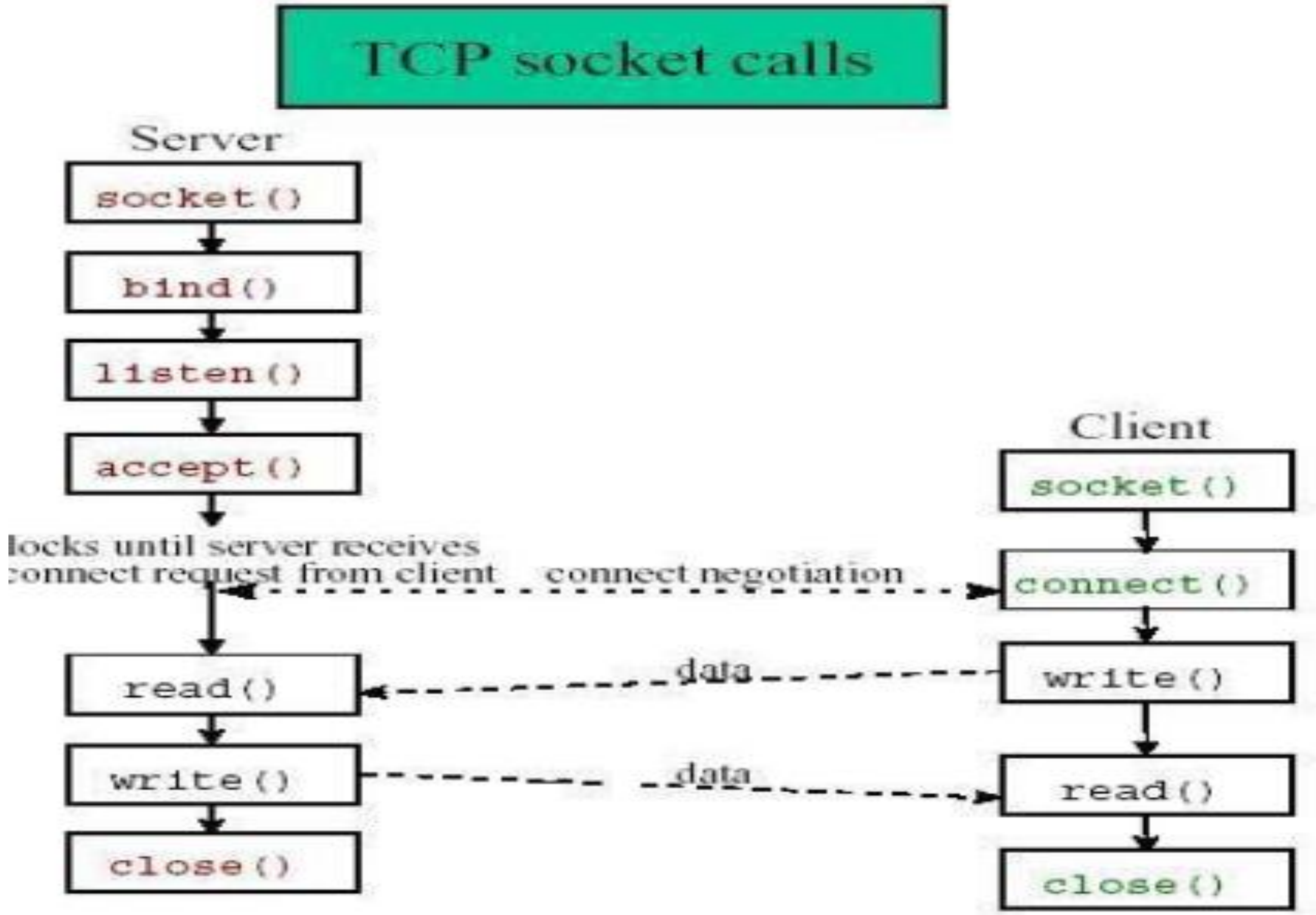
Socket Communication



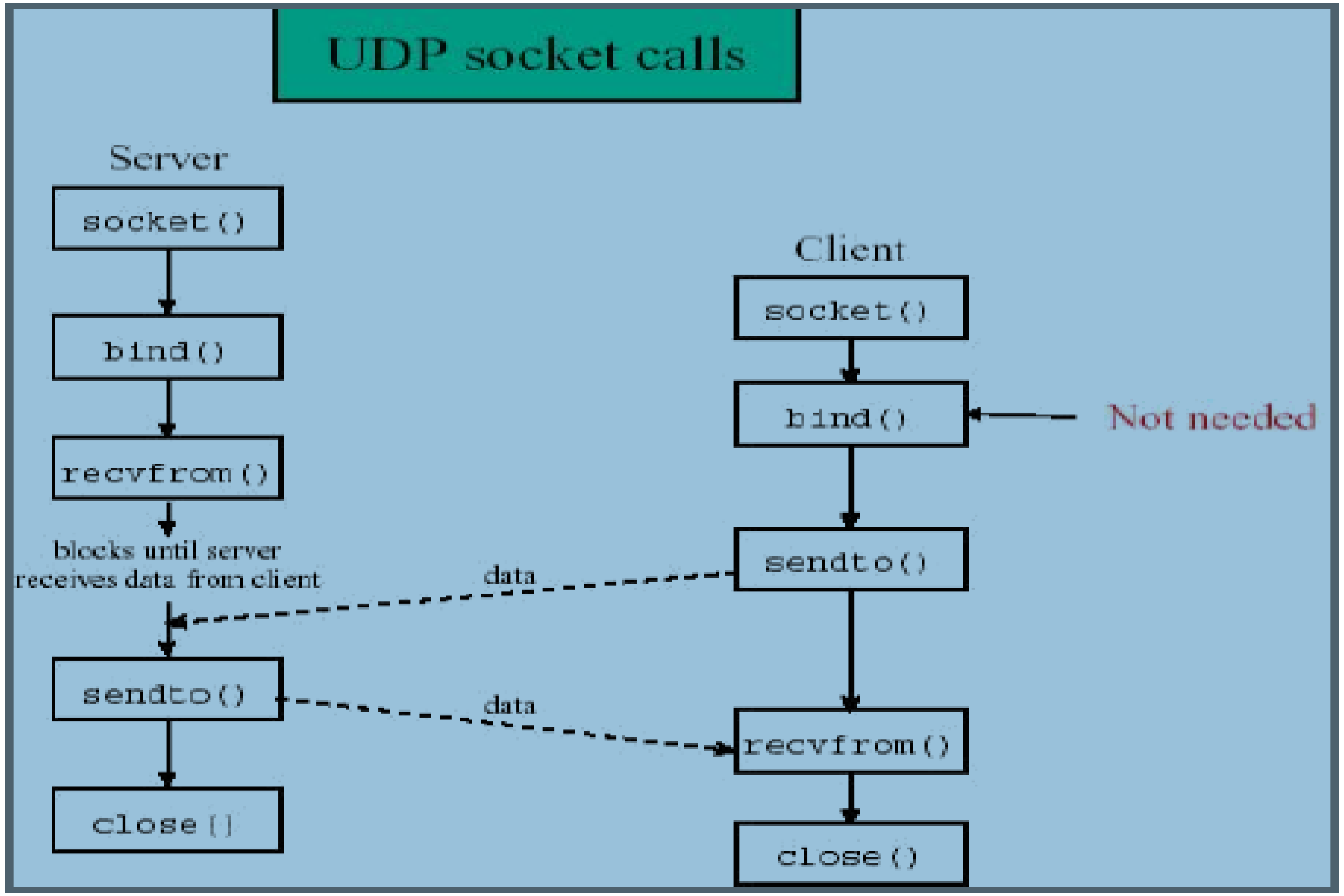
Socket Communication

- **socket()** create a socket
- **bind()** associate a socket with a network address
- **connect()** connect a socket to a remote network address
- **listen()** wait for incoming connection attempts
- **accept()** accept incoming connection attempts

TCP Socket Call



UDP Socket Call


UDP socket calls
Server
socket()
bind()
recvfrom()
blocks until server receives data from client
sendto()
close()
Client
socket()
bind()
Not needed
sendto()
data
data
recvfrom()
close()