Computer Networks Lab

Week 4:
Socket Programming

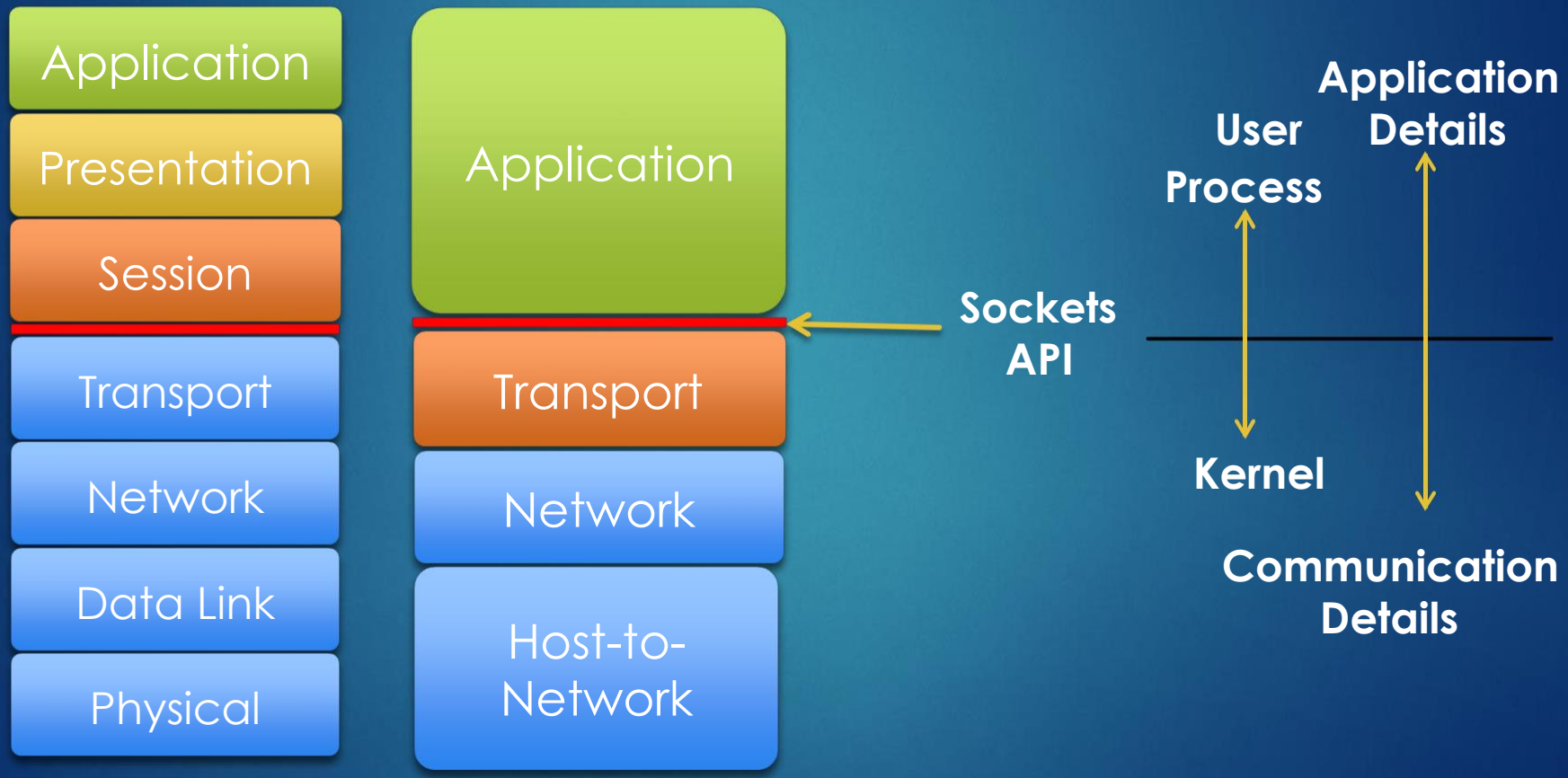
UDP Client-Server

Socket Programming

- Why Socket Programming?
  - To build any Network Application
  - Web browsers (Internet Explorer , Firefox)
  - Web Apps (Chat, Mail, File Transfer Apps)

What is the Socket?

Socket (An application programming interface(API) for inter process communication)



What is the Socket?

- Socket(Communication End Point)
- Working with Sockets is similar to working with files

File I/O	Socket I/O
Open File	Open Socket
	Name the Socket
	Associate with another Socket
Read and write	Send and Receive between Sockets
Close the File	Close the Socket

- Socket has always an address (**IP and Port**)
- Functionality (Communication)

  One application process can communicate with another application process (local or remote) using a socket.

TCP & UDP

- Difference between UDP and TCP
- Where to use what?
- Applications of UDP
- Applications of TCP

Socket Types

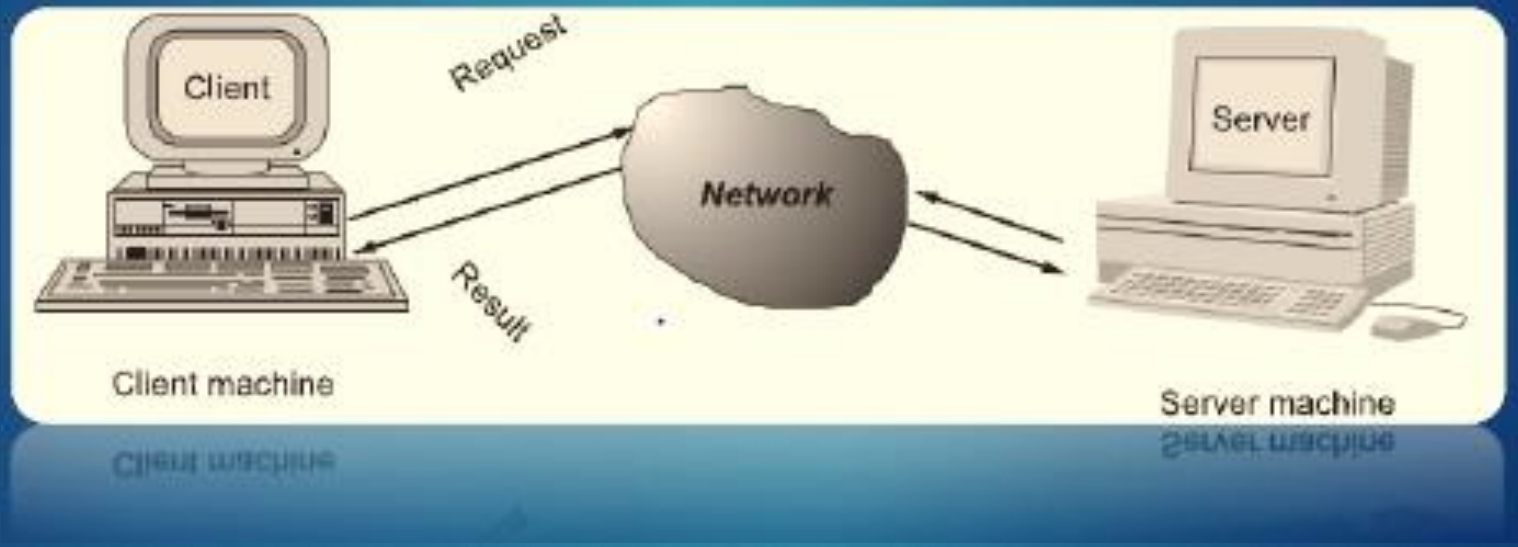
- Stream Sockets (SOCK_STREAM)
  - Connection oriented
  - Rely on TCP to provide reliable two-way connected communication
- Datagram Sockets (SOCK_DGRAM)
  - Rely on UDP
  - Connection is unreliable

Functions used in Socket Programming

Socket()	Endpoint for communication
Bind()	Assign a unique telephone number
Listen()	Wait for a caller
Connect()	Dial a number
Accept()	Receive a call
Send(), Recv()	Talk
Close()	Hang up

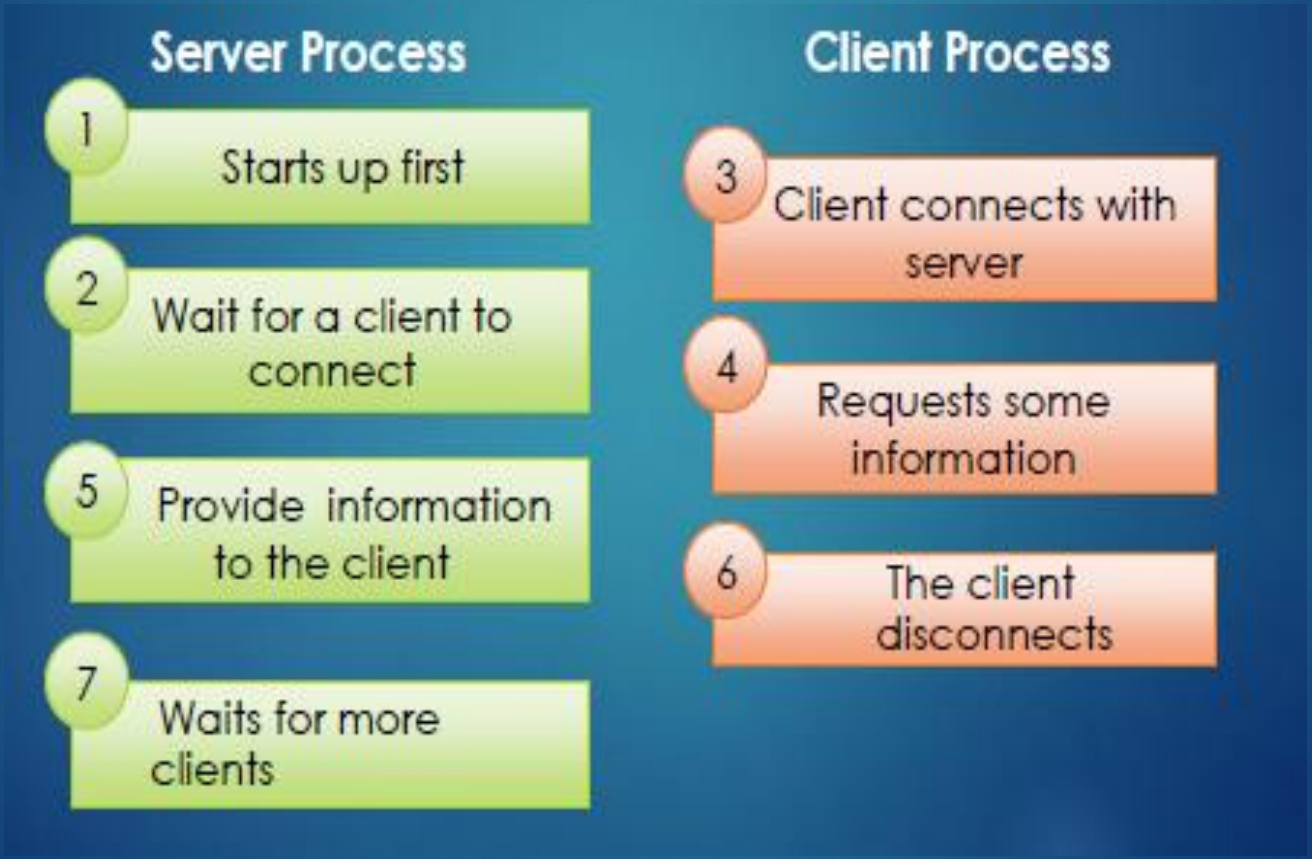
The Client – Server model

- Server – Provider of Services
- Client – Seeker of Services

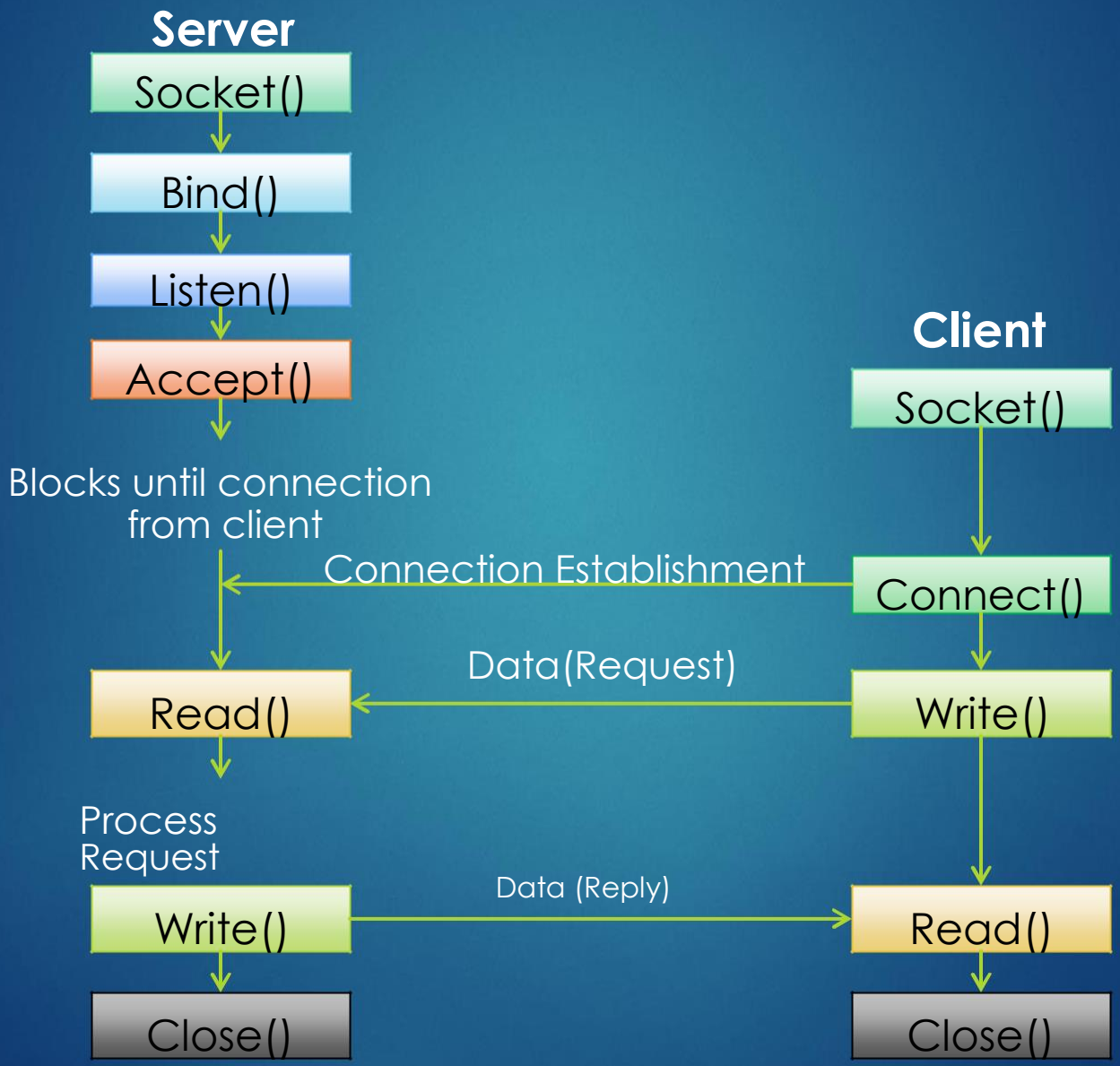


The Client – Server model

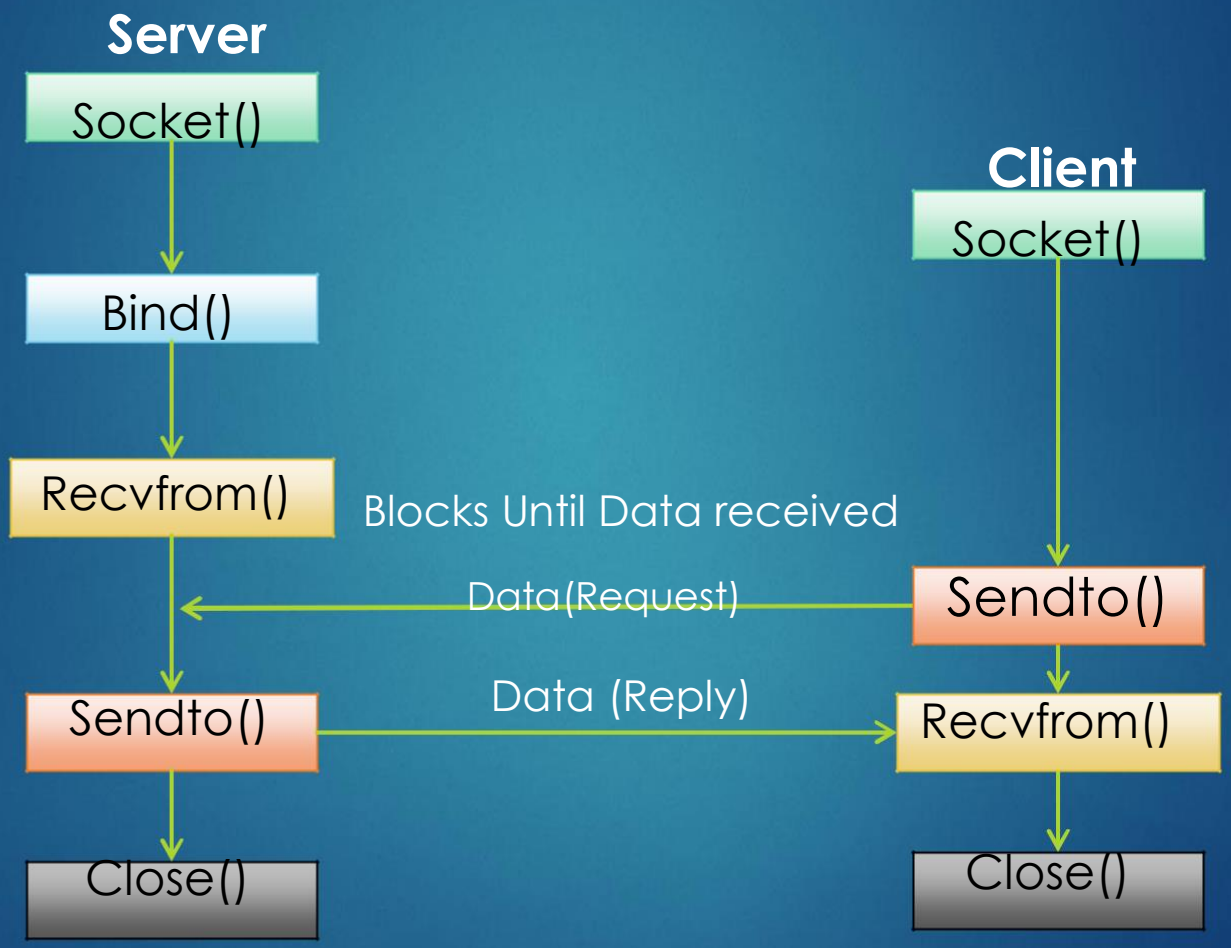
- In the socket programming world almost all communication is based on the Client-Server model.



TCP Server – Client Interaction



UDP Server – Client Interaction



User Datagram Protocol(UDP): An Analogy

UDP

- Single socket to receive messages
- No guarantee of delivery
- Not necessarily in-order delivery
- Datagram - independent packets
- Must address each packet

Postal Mail

- Single mailbox to receive letters Unreliable
- Not necessarily in-order delivery
- Letters sent independently Must address each reply

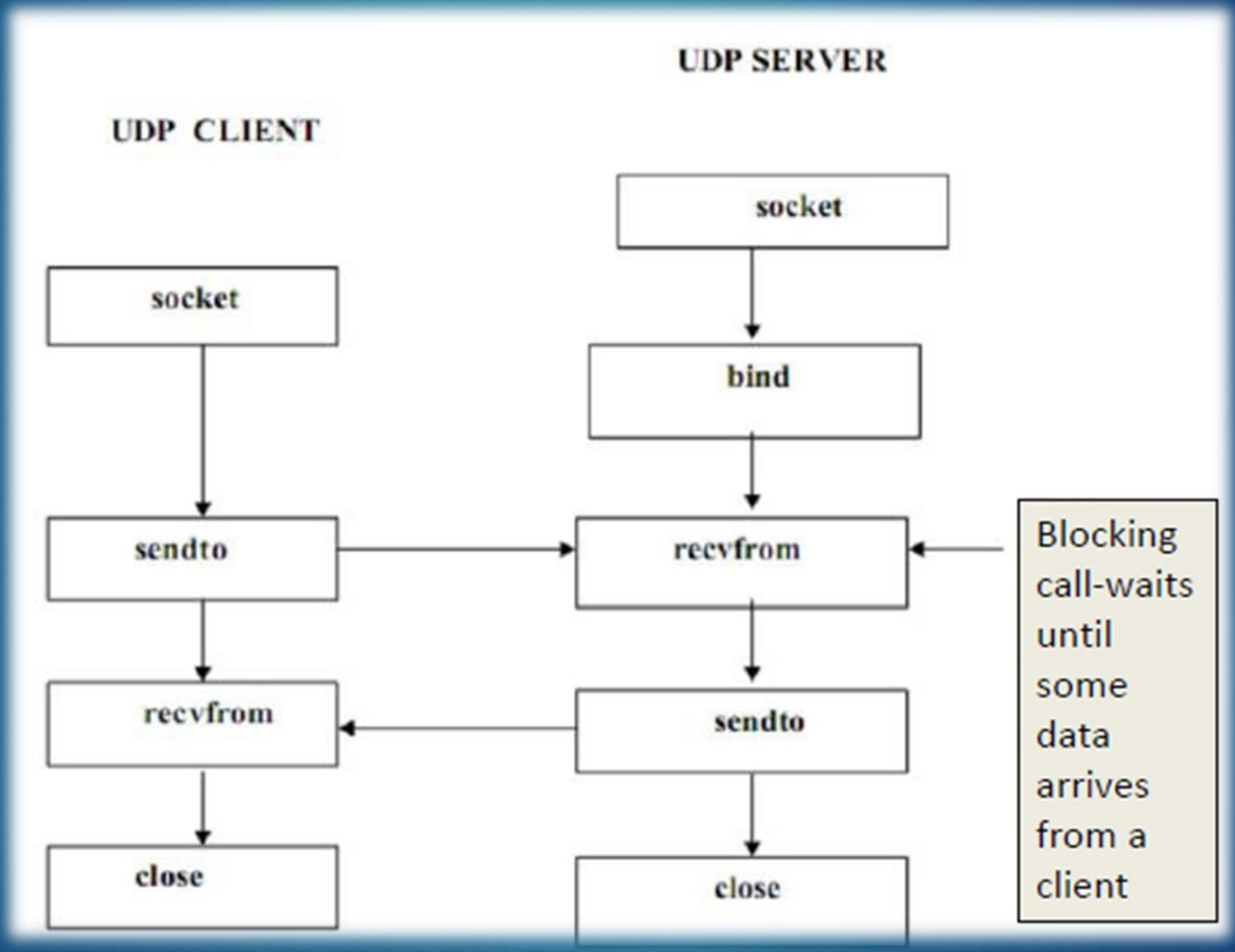
Example UDP applications
Multimedia, voice over IP

TCP VS UDP

- tcp is stream
- tcp is reliable
- tcp is point to point and "connected"
- connect/accept specify addresses at setup time, read/write don't need addresses
- data is checksummed

- udp discrete packets
- udp is unreliable
- udp can broadcast, 1 to N or
- server can receive from many clients
- each read/write specifies address
- data MAY be csum'ed

A UDP Client-Server Interaction



A UDP Client-Server Interaction

- The client does not establish a connection with the server.
- The client just sends a datagram to the server using the sendto function, which requires the address of the destination as a parameter.
- Similarly, the server does not accept a connection from a client.
- Instead, the server just calls the recvfrom function, which waits until data arrives from some client.
- recvfrom gets the address of the client, along with the datagram, so the server can send a response to the correct client.

Sendto() and recvfrom() functions

- int sendto(int sockfd, const void *buff, size_t nbytes, int flags, const struct sockaddr *to, socklen_t addrlen);

- int recvfrom(int sockfd, void *buff, size_t nbytes, int flags, struct sockaddr *from, socklen_t *addrlen);

- The first three arguments, sockfd, buff, and nbytes, are identical to the first three arguments for read and write.
- The to argument for sendto is a socket address structure containing the protocol address (e.g., IP address and port number) of where the data is to be sent. The size of this socket address structure is specified by addrlen.

Sendto() and recvfrom() functions

- The recvfrom function fills in the socket address structure pointed to by from with the protocol address of who sent the datagram
- The number of bytes stored in this socket address structure is also returned to the caller in the integer pointed to by addrlen.

- Both functions return the length of the data that was read or written as the value of the function.

Thank You