Computer Networks Lab

Instructors:
Ahmed Nawaz, Shehr Bano & Sara Afzal
Email id:
ahmed.nawaz@nu.edu.pk
shehr.bano@nu.edu.pk
sara.afzal@nu.edu.pk

Lab 2:

Socket Programming Implementation:

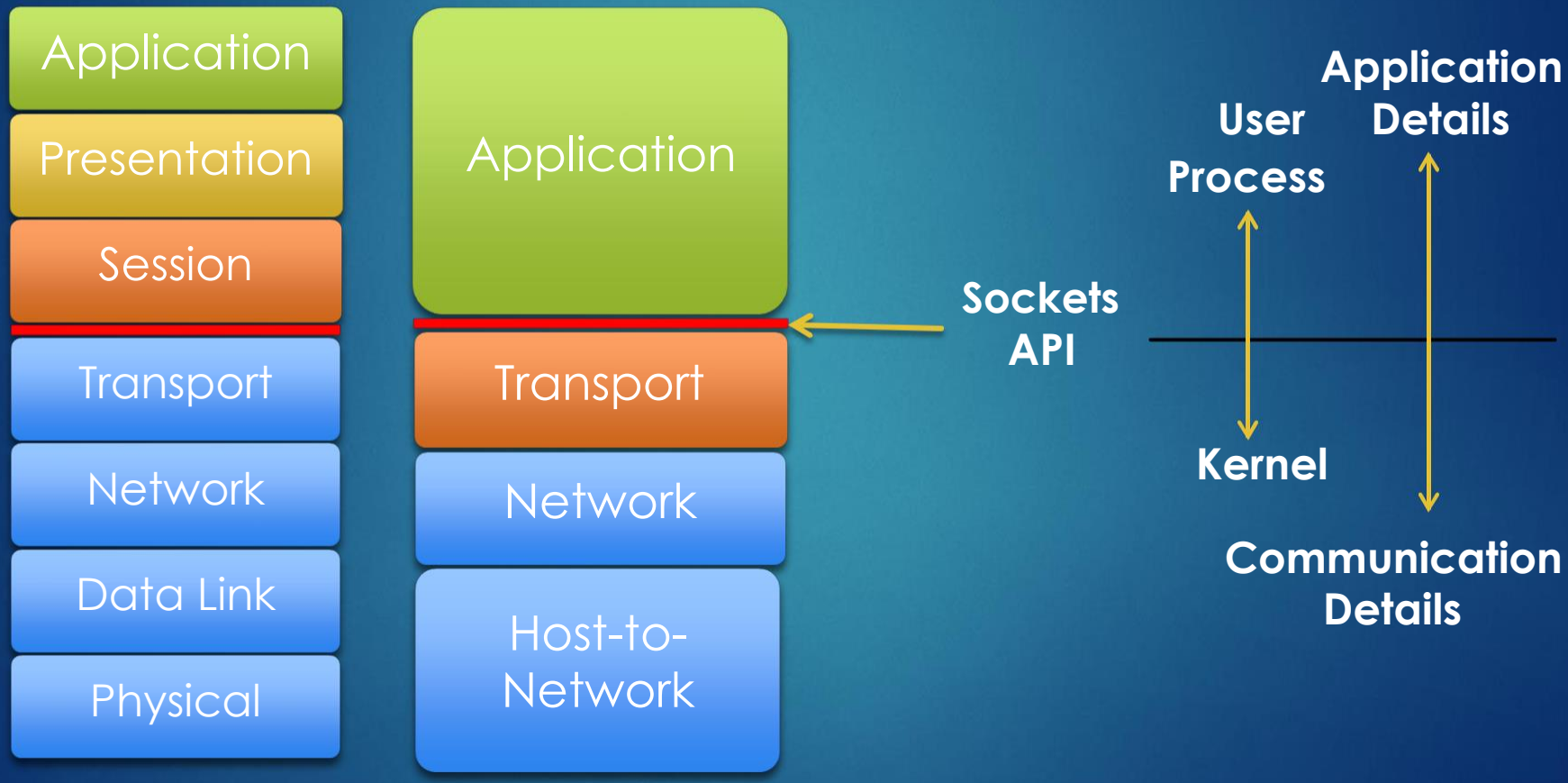
TCP Server – Client Interaction

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.

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

Socket() … Get the file descriptor

int sd=Socket(int domain,int type,int protocol);

Domain: AF_INET, PF_INET

Type: SOCK_STREAM, SOCK_DGRAM

Protocol: Set to "0" for appropriate protocol selection, IPPROTO_TCP, IPPROTO_UDP

Return: Socket descriptor on success and -1 on error

Example:

```
int U_s=socket(AF_INET, SOCK_STREAM, 0);
int T_s=socket(AF_INET, SOCK_DGRAM, 0);
```

bind()... what port am I on?

Associate a socket id with an address to which other process can connect

int status=bind(int sd, struct sockaddr* addrptr, int size);

Status: 0 on success and on error -1

sd: socket file decriptor created return by socket()

addrptr: pointer to Struct sockaddr type parameter, contains current socket IP and port

size: size of *addrptr*.

connect()... Request for connection

int status = connect(int sd, struct sockaddr *serv_addr, int addrlen);

status: error -1

sd: socket file descriptor

serv_addr: is a pointer to struct sockaddr that contains destination IP address and port

addrlen: size of serv_addr

listen()

Waits for incoming connections

int status = listen(int sd,int backlog);

sd: socket on which the server is listening

backlog: maximum no of connections pending in a queue

status: return -1 on error

accept()

Blocking System Call Waits for an incoming request, and when received creates a socket for it.

int sid = accept(int sd, struct sockaddr *cli_addr, int *addrlen);

sid: socket file descriptor for communication

sd: socket file descriptor used for listening

addr: pointer to struct sockaddr containing client address IP and Port

addrlen: sizeof struct sockaddr

send()

int sb = send(int sd, const char *msg, int len, int flags);

Sb: return No of bytes send or -1 on error

sd: socket file descriptor

msg: is a pointer to data buffer

len: no of bytes we want to send

flag: set it to 0 default

recv()

int rb = recv(int sd, char *buf, int len, int flags);

rb: No of bytes received or -1 on error **0** if connection is closed at other side

sd: socket file descriptor

buf: is a pointer to data buffer

len: receive up to len bytes in buffer pointer

flag: set it to 0 defalut

close() , Shutdown()

Close connection on given socket and frees the socket descriptor

```
int close(fd);
```

Acts as a partial close, disables sending (how=1) or receiving (how=0). Returns -1 on failure.

```
int shutdown(int sd, int how);
```

Sockaddr, Sockaddr_in

struct sockaddr: Generic Holds socket address information for many types of sockets

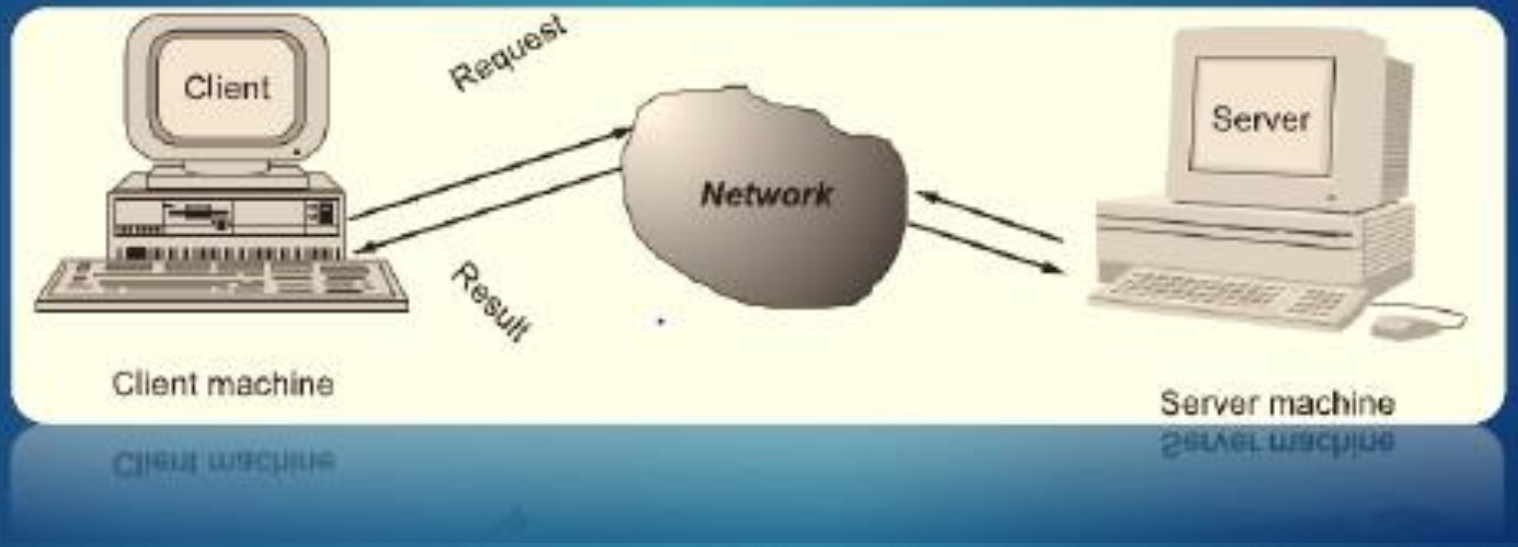
```
struct sockaddr {
    unsigned short sa_family;       //address family AF_xxx
    unsigned short sa_data[14];     //14 bytes of protocol addr
}
```

struct sockaddr_in: IPV4 specific

```
struct sockaddr_in {
    short int sin_family;           // set to AF_INET
    unsigned short int sin_port;    // Port number
    struct in_addr sin_addr;        // Internet address
    unsigned char sin_zero[8];      //set to all zeros
}
```

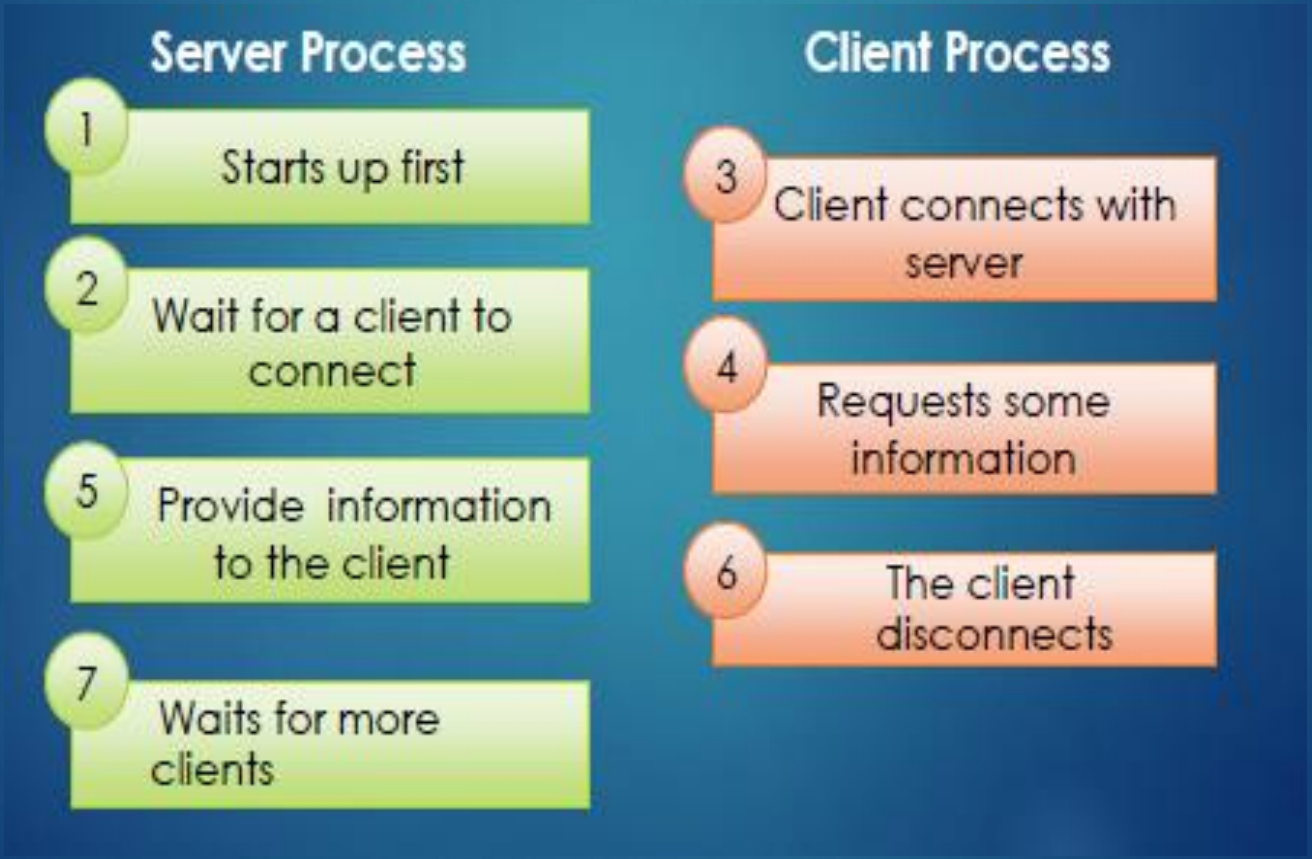
The Client – Server model

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

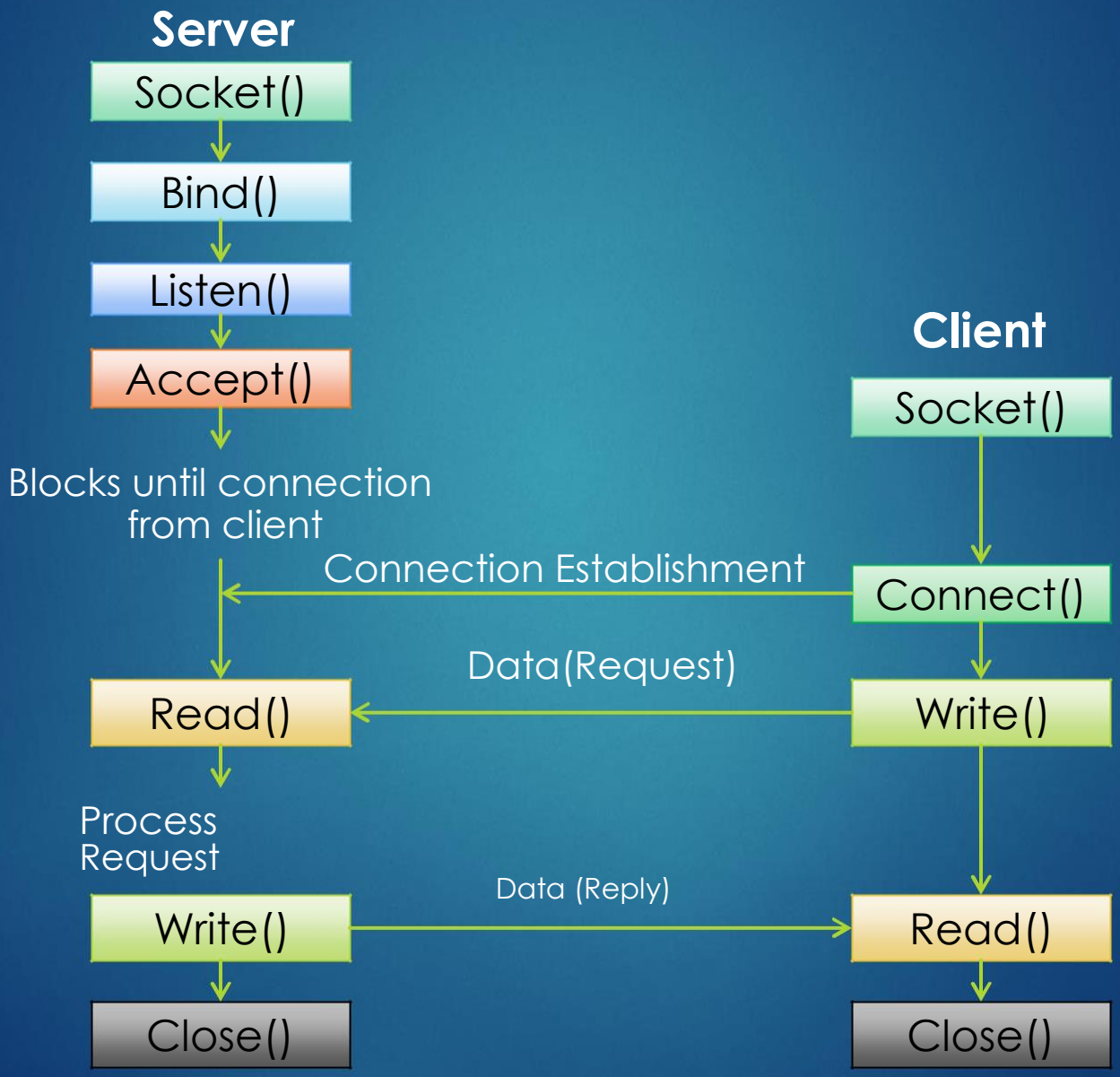


The Client – Server model

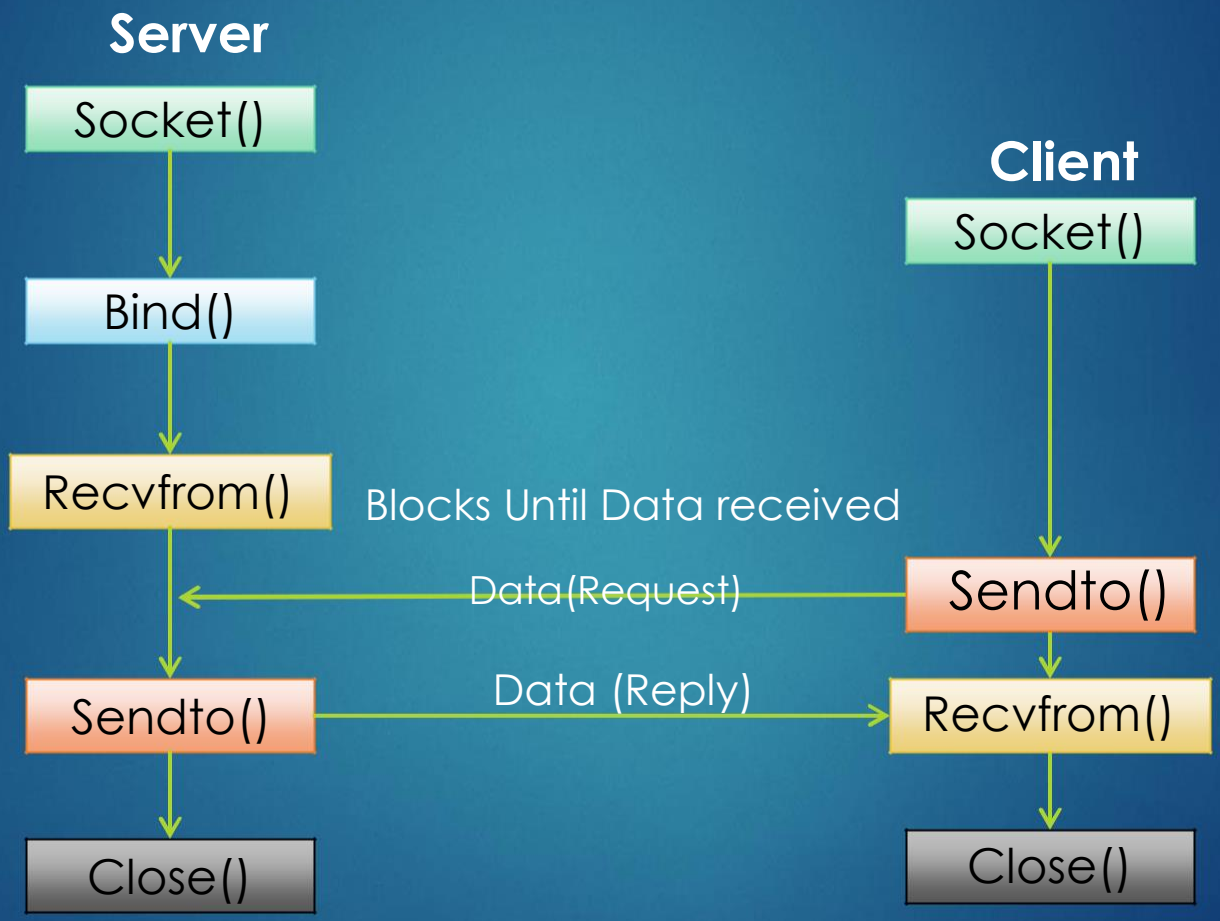
- In the socket programming world almost all communication is based on the Client-Server model.



TCP Server – Client Interaction



UDP Server – Client Interaction



Lab Task 2:

Question 1:

You are required to write a TCP client server application program in which

a. Server is waiting for connections
b. Client connects with the server
c. Client send a message to the server
d. Server echoes the same message to the client

Question 2:

Write a pair of TCP client/server program which involves exchange of messages between client and server. Server should display the message received from client as long as client does not enter ‘GoodBye’?

Question 3:

Write a pair of TCP client/server chatting programs, where client/server send messages to one another and respond back accordingly (2-way communication). Server and client should display the messages received as long as anyone of them does not enter ‘exit’?

Thank You