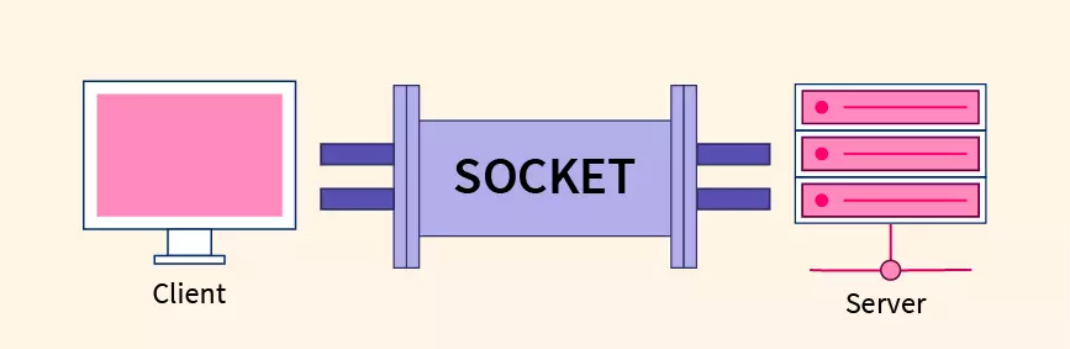
Introduction to Distributed and Parallel Computing
CS-401

Dr. Sanjay Saxena
Visiting Faculty, CSE, IIIT Vadodara
Assistant Professor, CSE, IIIT Bhubaneswar
Post doc – University of Pennsylvania, USA
PhD – Indian Institute of Technology(BHU), Varanasi

Socket Programming

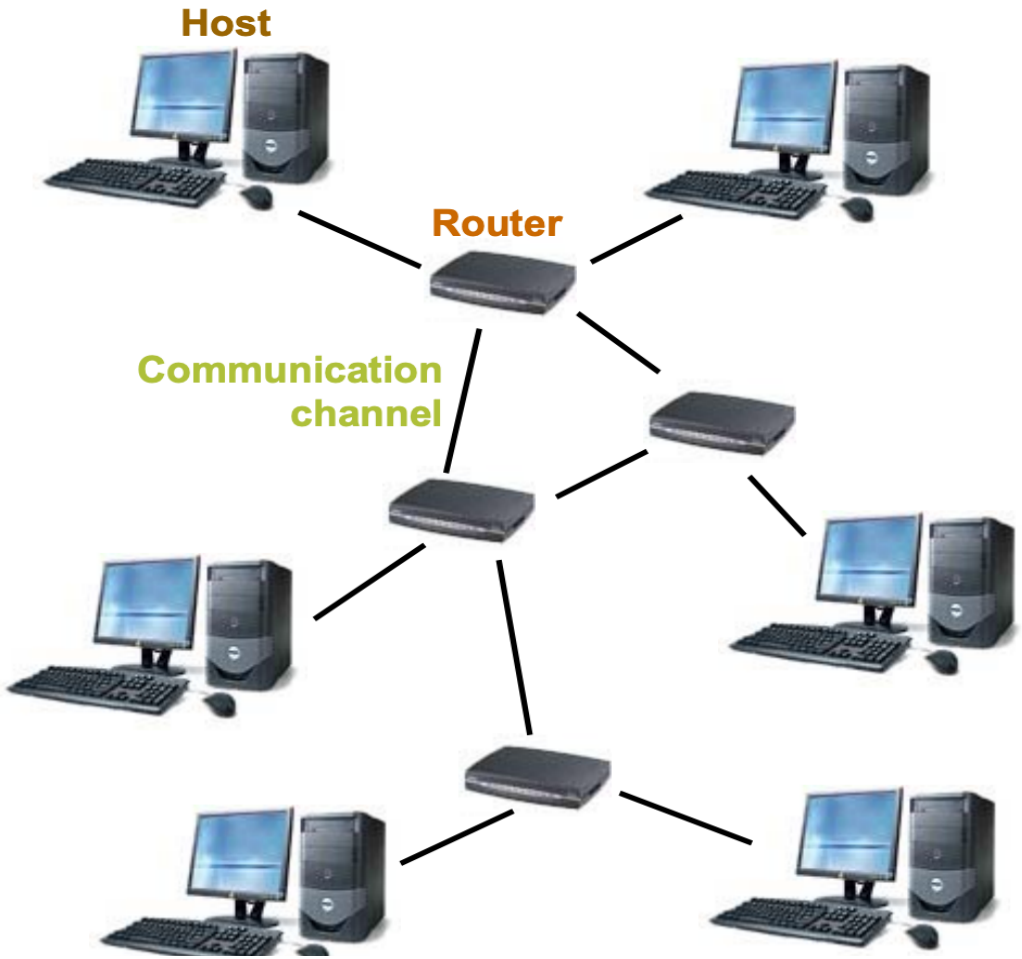
- A socket is a type of medium that provides a connection between two devices.
- The socket can be either a phone charger that provides the connection between the socket and the phone or the phone and that laptop. With the help of a socket, different applications are attached to the local network with different ports.
- Every time when the socket is created, the server specifies the program, and that program specifies the socket and the domain address.
- Socket Programming is a method to connect two nodes over a network to establish a means of communication between those two nodes. A node represents a computer or a physical device with an internet connection.
- A socket is the endpoint used for connecting to a node. The signals required to implement the connection between two nodes are sent and received using the sockets on each node respectively.



Contd..

- Computer Network
  - hosts, routers, communication channels
- Hosts run applications
- Routers forward information
- Packets: sequence of bytes
  - contain control information
  - e.g. destination host
- Protocol is an agreement
  - meaning of packets
  - structure and size of packets

  e.g. Hypertext Transfer Protocol (HTTP)

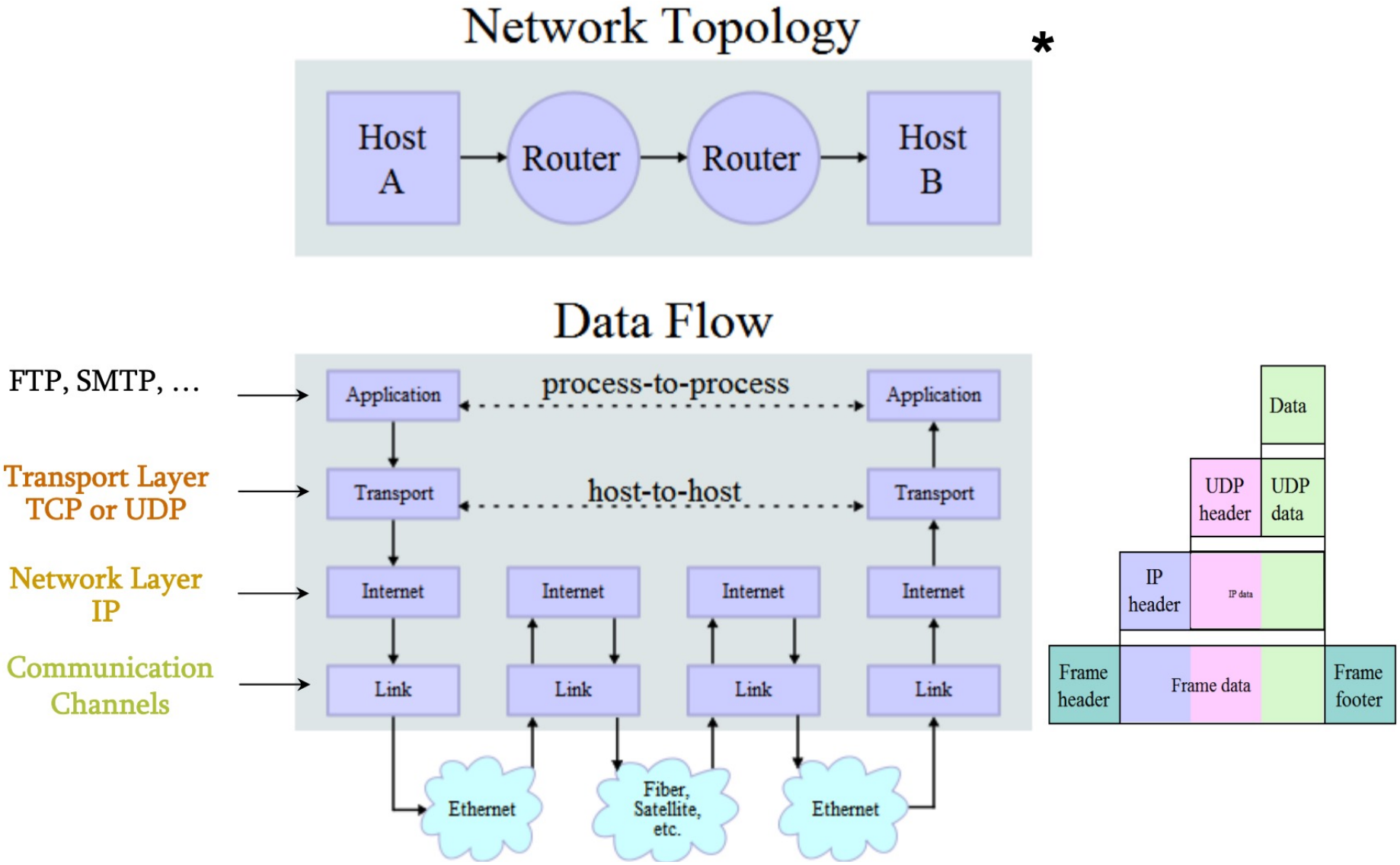


Protocol Families: TCP/IP

- Several protocols for different problems
- ☞ Protocol Suites or Protocol Families: TCP/IP

- TCP/IP provides end-to-end connectivity specifying how data should be
  - formatted,
  - addressed,
  - transmitted,
  - routed, and
  - received at the destination
- can be used in the internet and in stand-alone private networks
- it is organized into layers

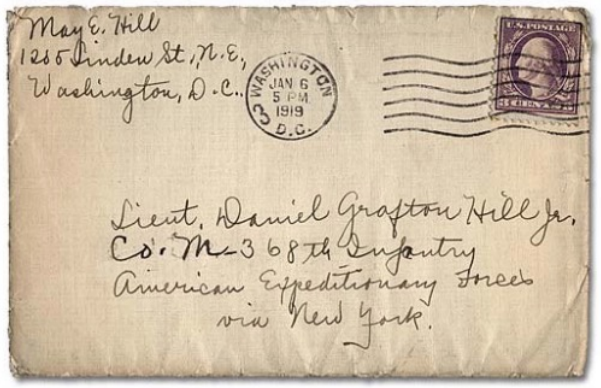
TCP/IP


Network Topology
*
Host A
Router
Router
Host B
Data Flow
FTP, SMTP, …
Transport Layer
TCP or UDP
Network Layer
IP
Communication
Channels
Application
process-to-process
Application
Transport
host-to-host
Transport
Internet
Internet
Internet
Internet
Link
Link
Link
Link
Ethernet
Fiber,
Satellite,
etc.
Ethernet
Data
UDP
header
UDP
data
IP
header
IP data
Frame
header
Frame data
Frame
footer

IP (Internet Protocol)

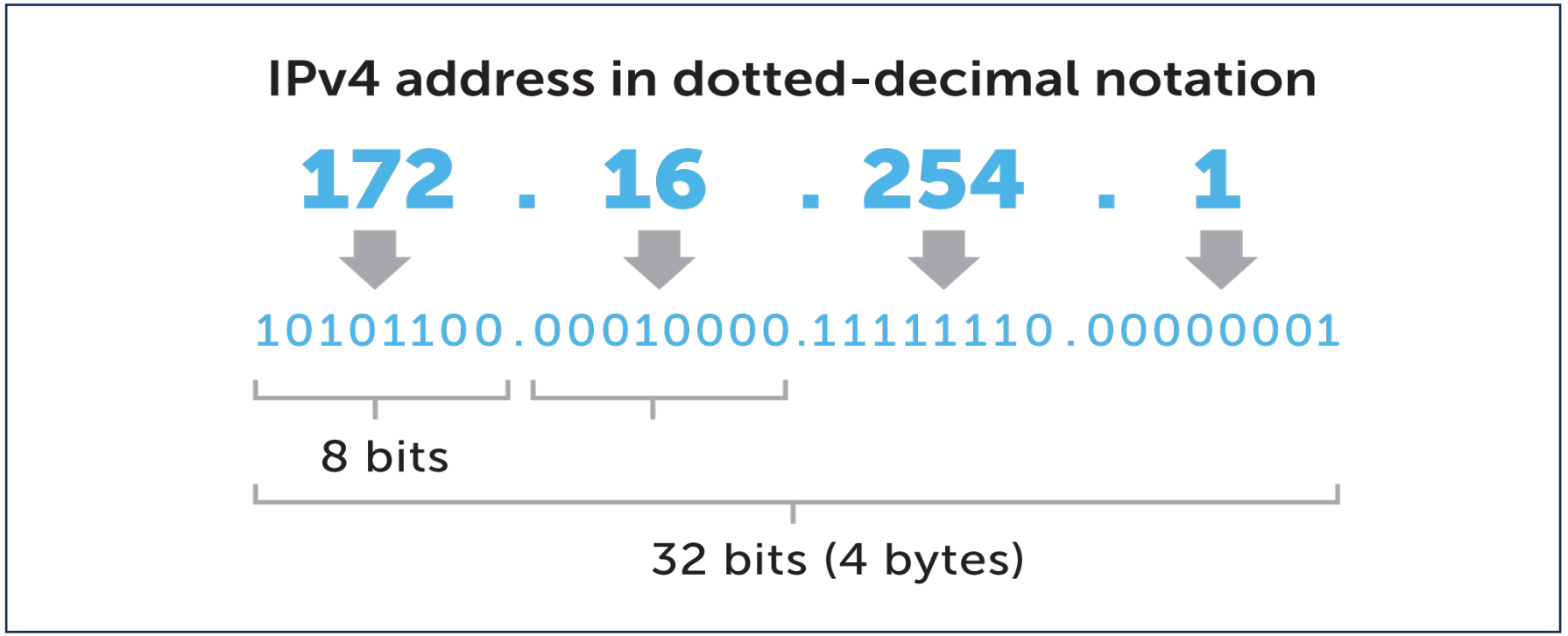
- provides a datagram service
  - packets are handled and delivered independently
- best-effort protocol
  - may loose, reorder or duplicate packets

- each packet must contain an IP address of its destination

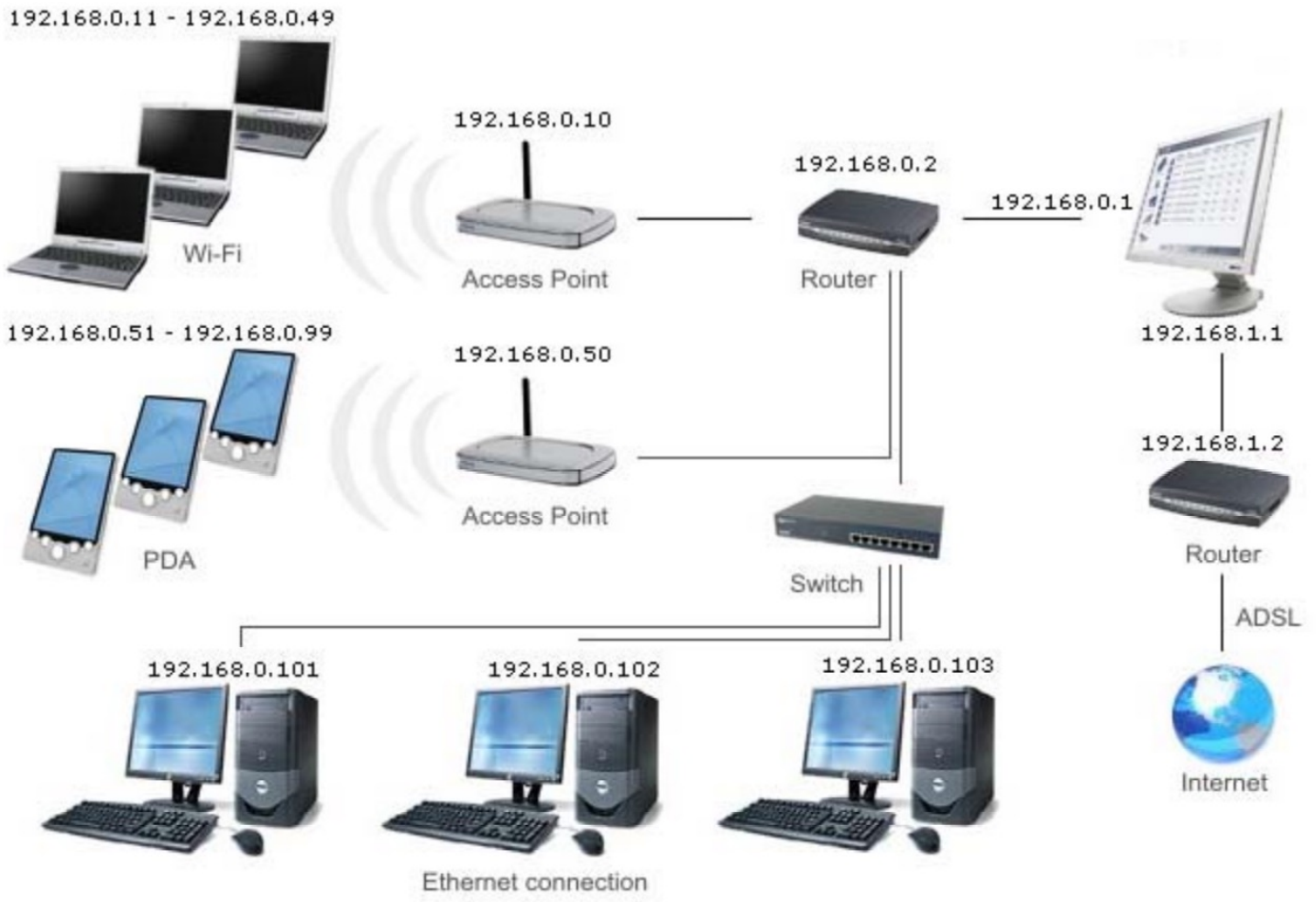


Addresses: IPv4

- The 32 bits of an IPv4 address are broken into 4 octets, or 8 bit fields (0-255 value in decimal notation).
- For networks of different size,
  - the first one (for large networks) to three (for small networks) octets can be used to identify the network, while
  - the rest of the octets can be used to identify the node on the network.



Local Area Network Addresses-IPv4



TCP vs UDP

- Both use port numbers
  - application-specific construct serving as a communication endpoint
  - 16-bit unsigned integer, thus ranging from 0 to 65535

  ☞ to provide end-to-end transport
- UDP: User Datagram Protocol
  - no acknowledgements
  - no retransmissions
  - out of order, duplicates possible
  - connectionless, i.e., app indicates destination for each packet
- TCP: Transmission Control Protocol
  - reliable byte-stream channel (in order, all arrive, no duplicates)
    - similar to file I/O
  - flow control
  - connection-oriented
  - bidirectional

Contd..

- TCP is used for services with a large data capacity, and a persistent connection
- UDP is more commonly used for quick lookups, and single use query-reply actions.

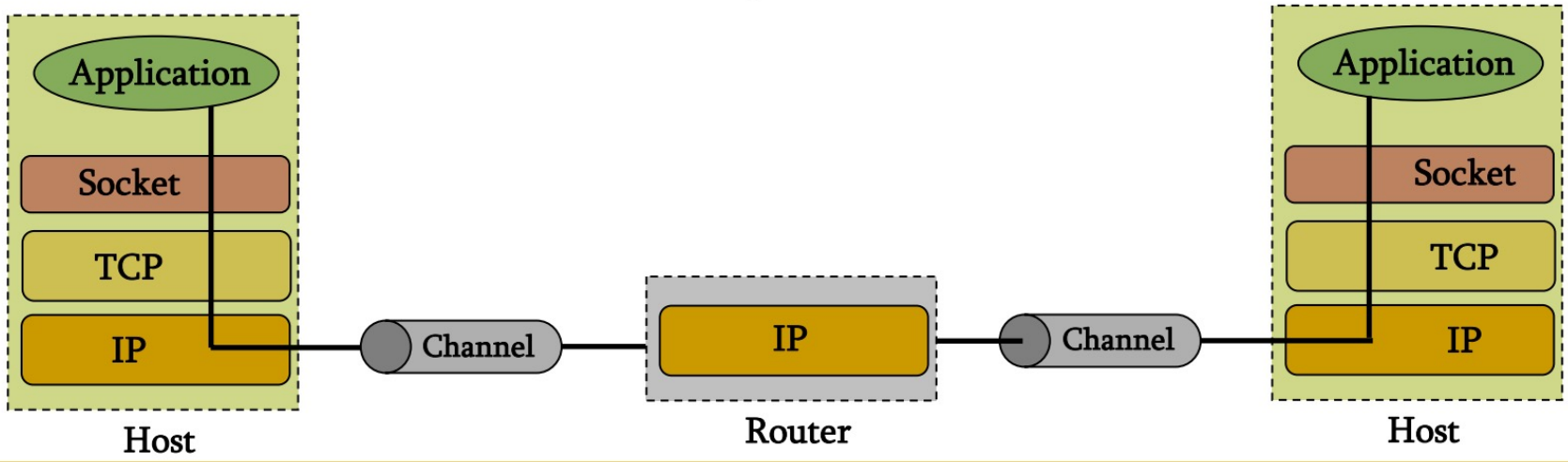
- Some common examples of TCP and UDP with their default ports:

DNS lookup	UDP	53
FTP	TCP	21
HTTP	TCP	80
POP3	TCP	110
Telnet	TCP	23

imp

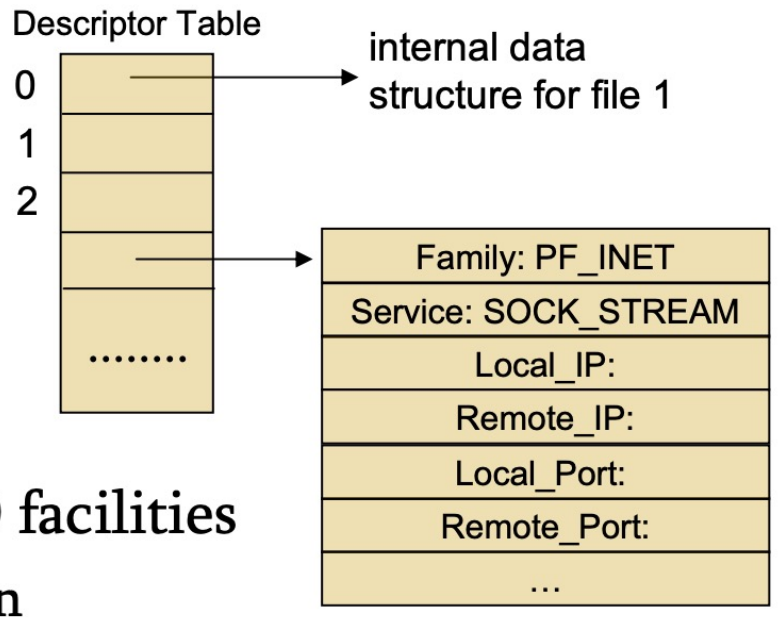
Berkley Socket

- Universally known as Sockets
- It is an abstraction through which an application may send and receive data
- Provide generic access to interprocess communication services
  - e.g. IPX/SPX, Appletalk, TCP/IP
- Standard API for networking

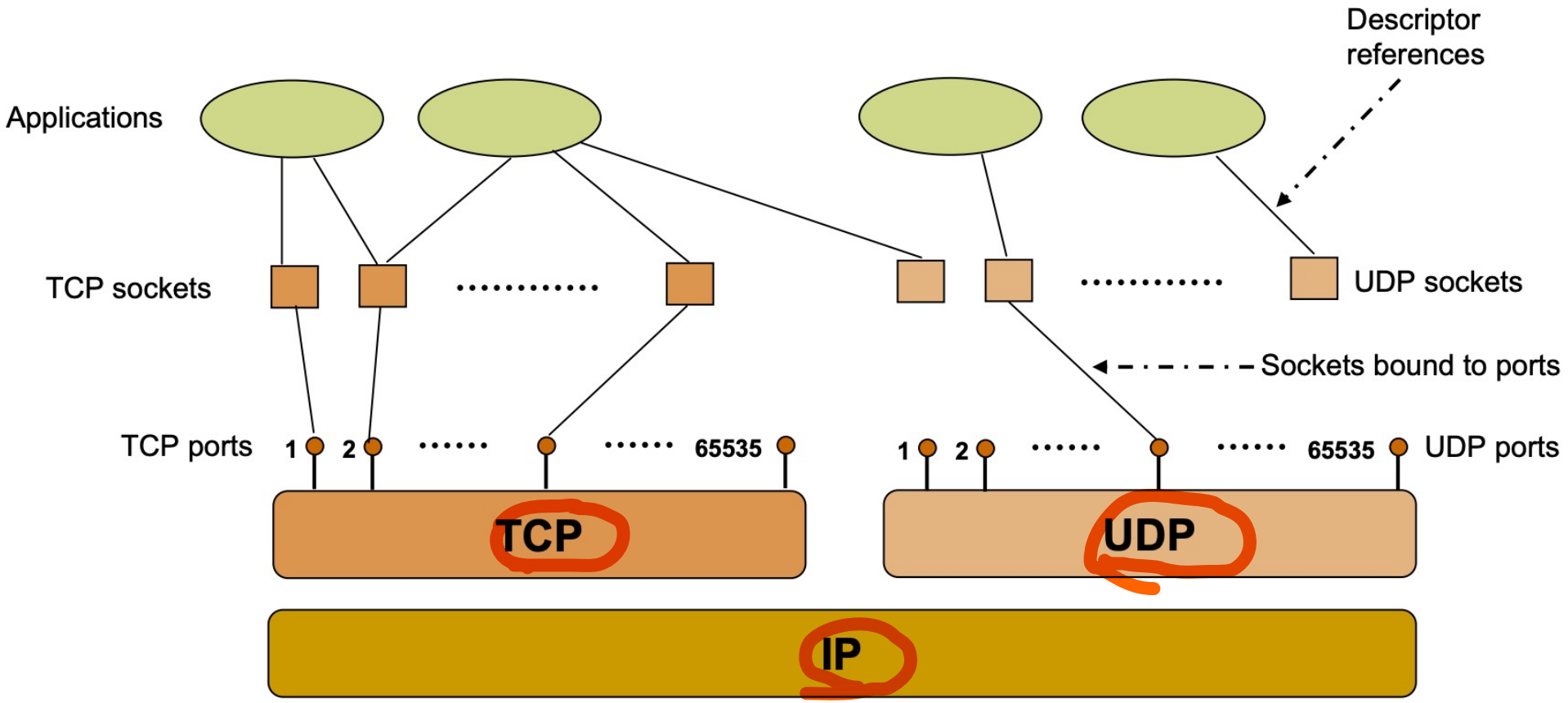


Contd..

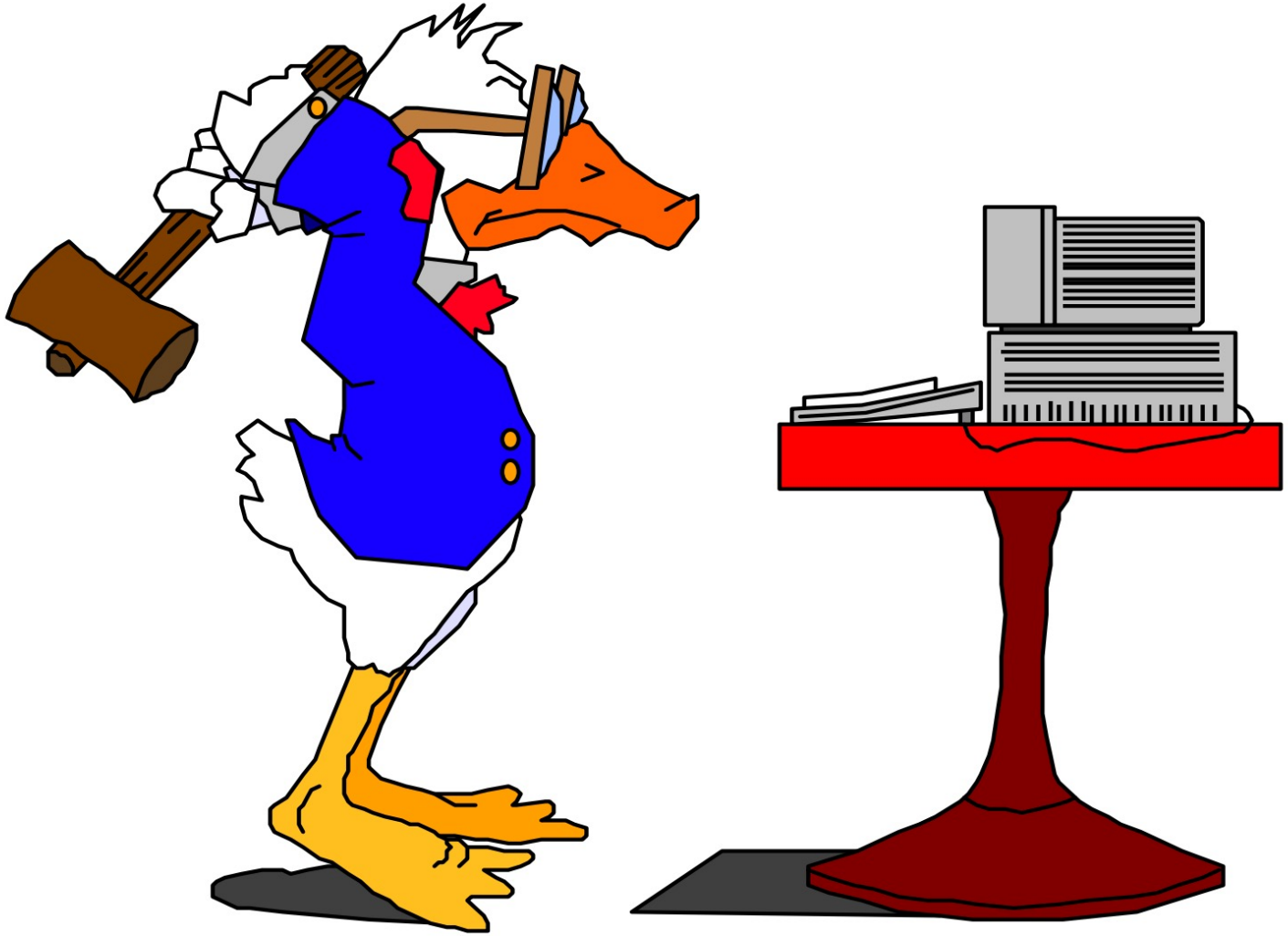
- Uniquely identified by
  - an internet address
  - an end-to-end protocol (e.g. TCP or UDP)
  - a port number
- Two types of (TCP/IP) sockets
  - Stream sockets (e.g. uses TCP)
    - provide reliable byte-stream service
  - Datagram sockets (e.g. uses UDP)
    - provide best-effort datagram service
    - messages up to 65.500 bytes
- Socket extend the convectional UNIX I/O facilities
  - file descriptors for network communication
  - extended the read and write system calls



Contd..


Descriptor references
Applications
TCP sockets
UDP sockets
Sockets bound to ports
TCP ports
1
2
65535
1
2
65535
UDP ports
TCP
UDP
IP

Socket Programming



Client Server Communication

- Server
  - passively waits for and responds to clients
  - passive socket
- Client
  - initiates the communication
  - must know the address and the port of the server
  - active socket

Procedure in Client-Server Communication

There are some procedures that we have to follow to establish client-server communication. These are as follows.

1. Socket: With the help of a socket, we can create a new communication.

2. Bind: With the help of this we can attach the local address with the socket.

3. Listen: With this help; we can accept the connection.

4. Accept: With this help; we can block the incoming connection until the request arrives.

5. Connect: With this help; we can attempt to establish the connection.

6. Send: With the help of this; we can send the data over the network.

7. Receive: With this help; we can receive the data over the network.

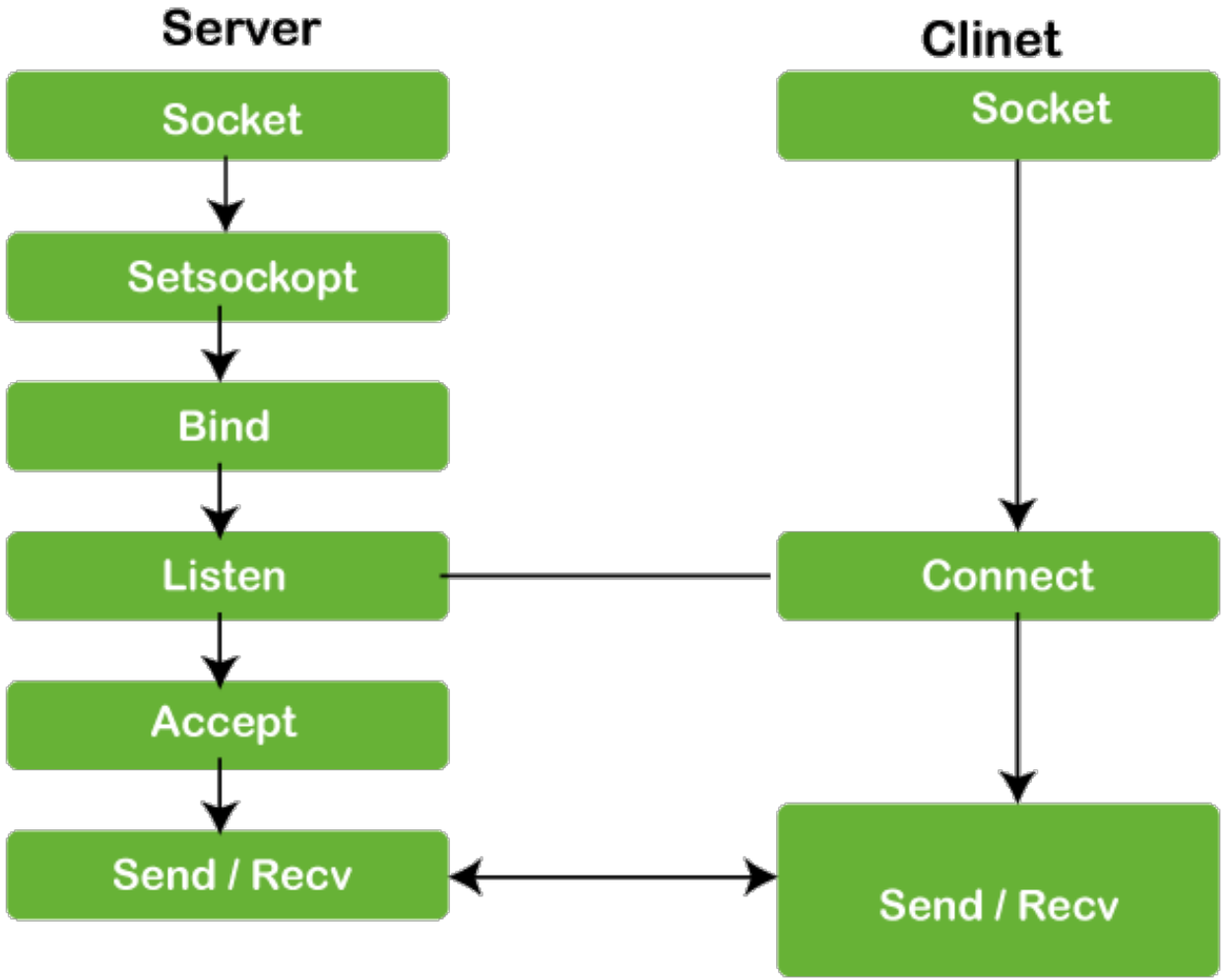
8. Close: With the help of this, we can release the connection from the network.

Socket Procedure

Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Contd..

- The socket is a type of mechanism that is used to exchange data between different processes. Here these processes are either present in different devices or the same device which are connected over a network. Once the connection for the socket is created, then the data can be sent in both directions and continues until one of the endpoints closes the connection.



Stages for Server Socket Creation

There are some stages by which we can create the socket for the server. These are as follows.

1. int socketcr: Socket(domain, type, protocol)

2. Socketcr: It is an integer type, and it is like a file handler.

3. Domain: It is a communication domain and it is an integer type.

4. Type: It is a communication type.

5. SOCK_DGRAM: It is a type of UDP which is unreliable and connectionless.

6. Protocol: It is used to assign the protocol value for the IP address, which is 0. The protocol value is similar to the value appearing in the protocol field of the IP header of the pocket.

What is a Connection?

- A connection is a type of relationship between two machines where the two software are known about each other.
- These two software know how to establish a connection with each other; in other words, we can say that this two software know how to send the bits over the network.
- A connection of the socket means the two machines should know all the information between each other, like the phone number, IP address, and the TCP port.
- A socket is a type of object which is similar to the file that allows the program to accept the incoming connection and allow them to send or receive the incoming connection. Also, it is a type of resource assigned to the server's process.
- The server can create the socket with the help of the socket(). This socket can not be shared with any other processor.

Contd..

•**Setsockopt:** With the help of Setsockopt, we can manipulate the various option of the socket, which are referred to by the socket's file descriptor. This process is completely optional. With the help of Setsockopt, we can reuse the port and address of the client and the server. When the server gives the error " address already in use," we can prevent it with the help of Setsockopt.

•**Bind:** We can bind the socket with the address and the port with the help of the bind function. This operation is done after the creation of the socket. For example, if we try to bind the server with the local host, then we use INADDR_ANY to define the server's IP address.

Contd..

- **Listen:** We can make a connection mode socket with the help of the listening to () function.
- **Accept:** With the help of accept() system call; we can make the connection-based socket. Some connection-based sockets are SOCK_STREAMand SOCK_SEQPACKET. It extracts all the incoming connections that come in first and allows their request to go to the server. The newly connected list is not able to listen with the help of another argument for the creation of the new socket.

Stages for Client

1.Socket connection: It is exactly the same as the method for the creation of the server.

2.Connect: We can initiate a connection to the socket with the help of connect() system call. If the parameter for the socket is a type of SOCK_DGRAM, then we can define the datagram as permanent with the help of connect(). If the socket is of type SOCK_STREAM, then we can attempt to make another connection for the server. With the help of connect() function, we can also create a connection for the foreign association. If the socket is unbound, then the system assigns the unique value to the local association. When the system successfully calls completed, the socket is ready to send or receive any type of data.

3.Send/Receive: The send() and recv() functions can perform the below operation.

•The socket on which the data can be communicated with each other.

•The storage buffer can store data about the address, like addr_of_data and addr_of_buffer.

•It deals with the size of the buffer, like len_of_data and len_of_buffer.

•It deals with the flag that says how the data will be sent.

Steps to Establish the Connection in the Socket

The steps involved in establishing a socket on the client side are as follows:

•It creates a socket with the help of a socket() system call.

•Then we have to connect with the socket address of the server with the help of a system() call.

•Then we have to send and receive the data. We can do this in various ways. we can do this read() and write() function.

The steps involved in establishing a socket on the server side are as follows:

•It first creates a socket with the help of a socket() system call.

•Then it binds the socket to an address with the help of the bind() system call. An address consists of a port number for the server socket in the host machine.

•Then it listens for the connection with the help of the listening () system call.

•Then the server accepts the incoming connection with the help of accept() system call. It also blocks all incoming commands until a client is connected to a server.

•Then the process of sending and receiving data starts.

Connecting Multiple Clients without Multithreading

- There are various examples in which we see how a single user can connect with the server. In today's programming world, multiple users are connected to the server with different sockets.
- There is various ways to achieve this. One of them is multithreading. With the help of multithreading, we can achieve this. We can implement a multithreading process with the help of by the help of the select() function.

Socket Creation in C

- `int sockid = socket(family, type, protocol);`
  - sockid: socket descriptor, an integer (like a file-handle)
  - family: integer, communication domain, e.g.,
    - PF_INET, IPv4 protocols, Internet addresses (typically used)
    - PF_UNIX, Local communication, File addresses
  - type: communication type
    - SOCK_STREAM - reliable, 2-way, connection-based service
    - SOCK_DGRAM - unreliable, connectionless, messages of maximum length
  - protocol: specifies protocol
    - IPPROTO_TCP IPPROTO_UDP
    - usually set to 0 (i.e., use default protocol)
  - upon failure returns -1

☞ NOTE: socket call does not specify where data will be coming from, nor where it will be going to – it just creates the interface!

Socket Close in C

- When finished using a socket, the socket should be closed
- `status = close(sockid);`
  - sockid: the file descriptor (socket being closed)
  - status: 0 if successful, -1 if error
- Closing a socket
  - closes a connection (for stream socket)
  - frees up the port used by the socket

Specifying Addresses

- Socket API defines a generic data type for addresses:

```
struct sockaddr {
    unsigned short sa_family;  /* Address family (e.g. AF_INET) */
    char sa_data[14];          /* Family-specific address information */
}
```

- Particular form of the sockaddr used for TCP/IP addresses:

```
struct in_addr {
    unsigned long s_addr;          /* Internet address (32 bits) */
}
struct sockaddr_in {
    unsigned short sin_family;     /* Internet protocol (AF_INET) */
    unsigned short sin_port;       /* Address port (16 bits) */
    struct in_addr sin_addr;     /* Internet address (32 bits) */
    char sin_zero[8];              /* Not used */
}
```

☞ Important: sockaddr_in can be casted to a sockaddr

Assign Address to Socket: bind()

- associates and reserves a port for use by the socket

- `int status = bind(sockid, &addrport, size);`
  - sockid: integer, socket descriptor
  - addrport: struct sockaddr, the (IP) address and port of the machine
    - for TCP/IP server, internet address is usually set to INADDR_ANY, i.e., chooses any incoming interface
  - size: the size (in bytes) of the addrport structure
  - status: upon failure -1 is returned

Complete C Program ??

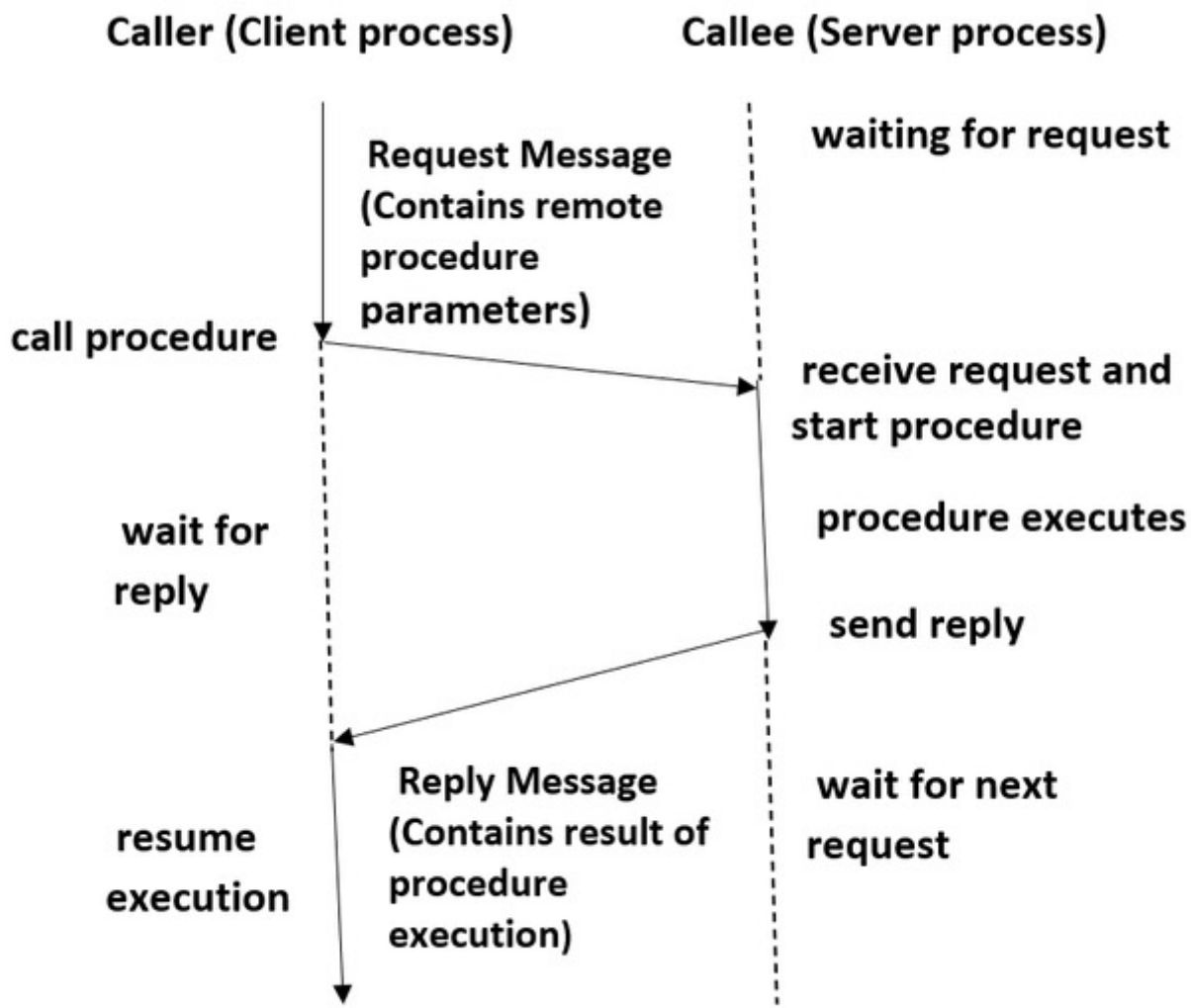
For Client Server Socket Programming

RPC Mechanism in Distributed System

- Remote Procedure Call (RPC) is a communication technology that is used by one program to make a request to another program for utilizing its service on a network without even knowing the network's details. A function call or a subroutine call are other terms for a procedure call.
- It is based on the client-server concept. The client is the program that makes the request, and the server is the program that gives the service.
- An RPC, like a local procedure call, is based on the synchronous operation that requires the requesting application to be stopped until the remote process returns its results.
- Multiple RPCs can be executed concurrently by utilizing lightweight processes or threads that share the same address space.
- Remote Procedure Call program as often as possible utilizes the Interface Definition Language (IDL), a determination language for describing a computer program component's Application Programming Interface (API).

Working Procedure for RPC Model:

- The process arguments are placed in a precise location by the caller when the procedure needs to be called.
- Control at that point passed to the body of the method, which is having a series of instructions.
- The procedure body is run in a recently created execution environment that has duplicates of the calling instruction's arguments.
- At the end, after the completion of the operation, the calling point gets back the control, which returns a result.
  - The call to a procedure is possible only for those procedures that are not within the caller's address space because both processes (caller and callee) have distinct address space and the access is restricted to the caller's environment's data and variables from the remote procedure.
  - The caller and callee processes in the RPC communicate to exchange information via the message-passing scheme.
  - The first task from the server-side is to extract the procedure's parameters when a request message arrives, then the result, send a reply message, and finally wait for the next call message.
  - Only one process is enabled at a certain point in time.
  - The caller is not always required to be blocked.
  - The asynchronous mechanism could be employed in the RPC that permits the client to work even if the server has not responded yet.
  - In order to handle incoming requests, the server might create a thread that frees the server for handling consequent requests


Caller (Client process)
Callee (Server process)
waiting for request
Request Message
(Contains remote
procedure
parameters)
call procedure
receive request and
start procedure
wait for
reply
procedure executes
send reply
Reply Message
(Contains result of
procedure
execution)
resume
execution
wait for next
request

Types of RPC:

Callback RPC: In a Callback RPC, a P2P (Peer-to-Peer)paradigm opts between participating processes. In this way, a process provides both client and server functions which are quite helpful. Callback RPC's features include:

•The problems encountered with interactive applications that are handled remotely
•It provides a server for clients to use.
•Due to the callback mechanism, the client process is delayed.
•Deadlocks need to be managed in callbacks.
•It promotes a Peer-to-Peer (P2P) paradigm among the processes involved.

RPC for Broadcast: A client's request that is broadcast all through the network and handled by all servers that possess the method for handling that request is known as a broadcast RPC. Broadcast RPC's features include:
•You have an option of selecting whether or not the client's request message ought to be broadcast.
•It also gives you the option of declaring broadcast ports.
•It helps in diminishing physical network load.

Contd..

Batch-mode RPC: Batch-mode RPC enables the client to line and separate RPC inquiries in a transmission buffer before sending them to the server in a single batch over the network. Batch-mode RPC's features include:

•It diminishes the overhead of requesting the server by sending them all at once using the network.
•It is used for applications that require low call rates.
•It necessitates the use of a reliable transmission protocol.

Local Procedure Call Vs Remote Procedure Call:

•Remote Procedure Calls have disjoint address space i.e. different address space, unlike Local Procedure Calls.
•Remote Procedure Calls are more prone to failures due to possible processor failure or communication issues of a network than Local Procedure Calls.
•Because of the communication network, remote procedure calls take longer than local procedure calls.

Advantages of Remote Procedure Calls:

- The technique of using procedure calls in RPC permits high-level languages to provide communication between clients and servers.
- This method is like a local procedure call but with the difference that the called procedure is executed on another process and a different computer.
- The thread-oriented model is also supported by RPC in addition to the process model.
- The RPC mechanism is employed to conceal the core message passing method.
- The amount of time and effort required to rewrite and develop the code is minimal.
- The distributed and local environments can both benefit from remote procedure calls.
- To increase performance, it omits several of the protocol layers.
- Abstraction is provided via RPC. To exemplify, the user is not known about the nature of message-passing in network communication.
- RPC empowers the utilization of applications in a distributed environment.

Disadvantages

- In Remote Procedure Calls parameters are only passed by values as pointer values are not allowed.
- It involves a communication system with another machine and another process, so this mechanism is extremely prone to failure.
- The RPC concept can be implemented in a variety of ways, hence there is no standard.
- Due to the interaction-based nature, there is no flexibility for hardware architecture in RPC.
- Due to a remote procedure call, the process's cost has increased.

Thanks & Cheers!!

Small aim is a crime; have great aim.
Bharat-Ratan A. P. J. Abdul Kalam