CS3103: Computer Networks Practice

Network Applications Development

Socket API – A short introduction

Client/Server Applications Demo

Anand Bhojan

COM2-04-26, School of Computing

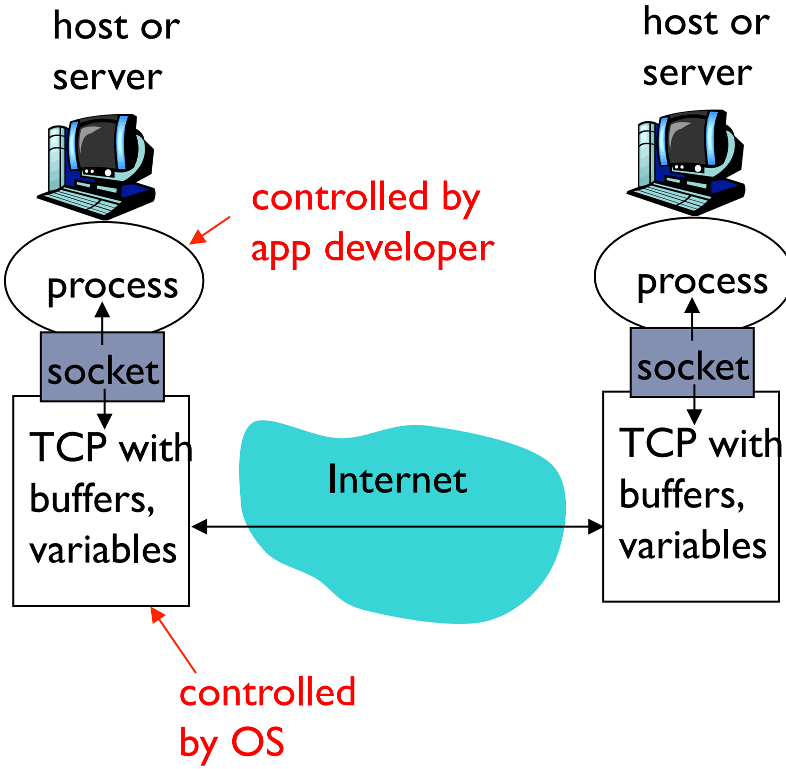
banand@comp.nus.edu.sg ph: 651-67351

Socket

- Socket
  - Interface between the application layer and transport layer
  - OS-controlled interface (a "door") into which application process can both send and receive messages
- Addressing
  - Host address + process identifier
  - Eg. IP address + port number
    - Eg HTTP – port 80, SMTP – port 25, ftp – port 21

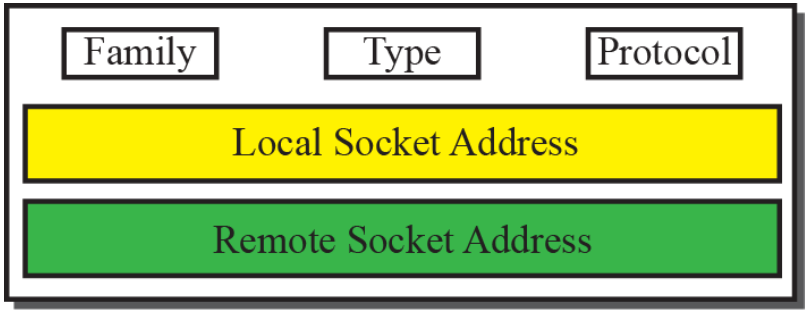
Tool:

- Use ifconfig (ipconfig in windows) to see edit your laptop's IP address.
- Use 'cat /etc/services' to see all well known port numbers


host or server
host or server
controlled by app developer
process
process
socket
socket
TCP with buffers, variables
TCP with buffers, variables
Internet
controlled by OS

Socket: endpoint for comm. A combination of host IP address and port number.

Socket Data Structure

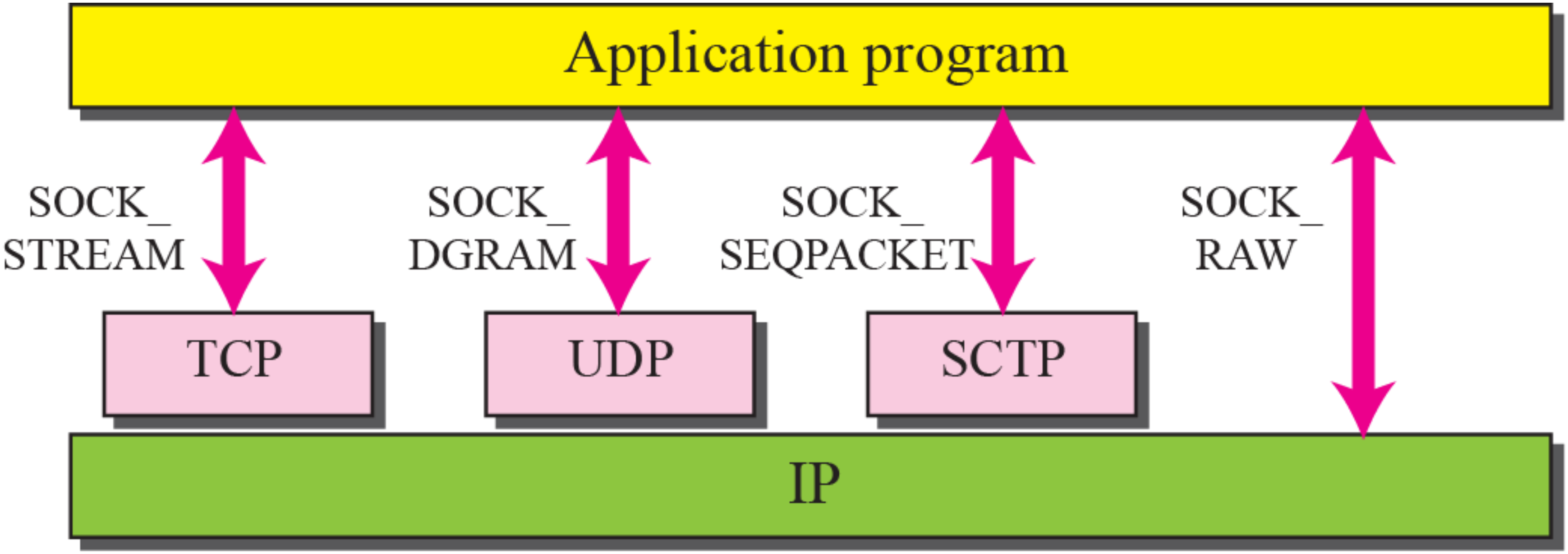


Fields

```
struct  socket
{
     int family;
     int type;
     int protocol;
     socketaddr local;
     socketaddr remote;
 };
```

Generic definition

Socket Types

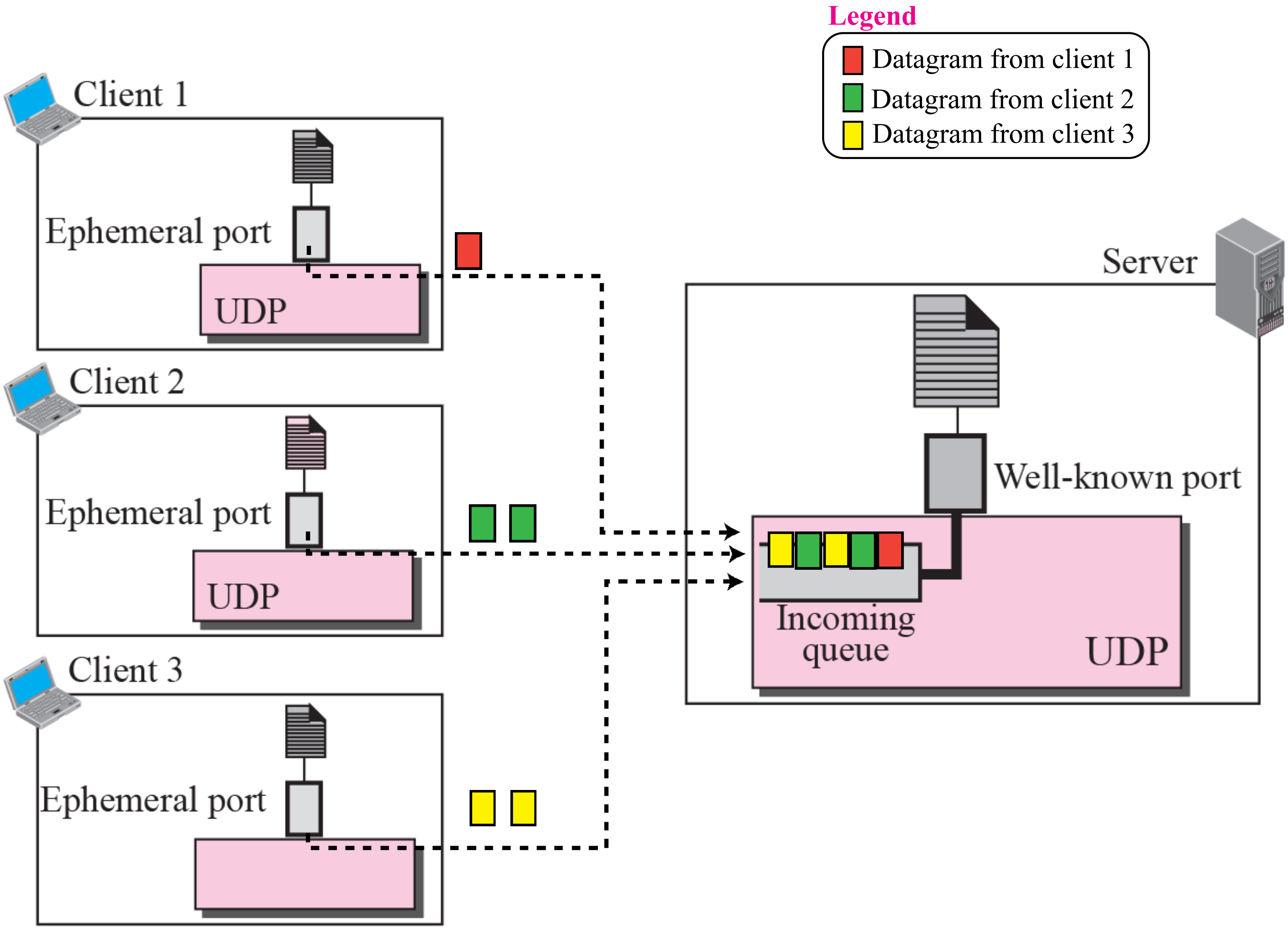


- Types of transport service via socket:
  - unreliable datagram
  - reliable, byte stream-oriented
- Lower level protocols and network interfaces can be accessed through 'RAW Socket'. (not in Java)
- The new SCTP socket provides multiple types of service

Client-Server Applications Development - UDP

- UDP service: unreliable transfer of datagram from one process to another
  - server process must be running first
  - server must have created socket that looks for packets from client
  - no “connection” between client and server
    - no handshaking
    - sender explicitly attaches IP address and port of destination to each packet
    - To reply, the server must extract IP address, port of sender from received packet
  - transmitted data may be received out of order, or lost

UDP Server: Connectionless (Iterative)

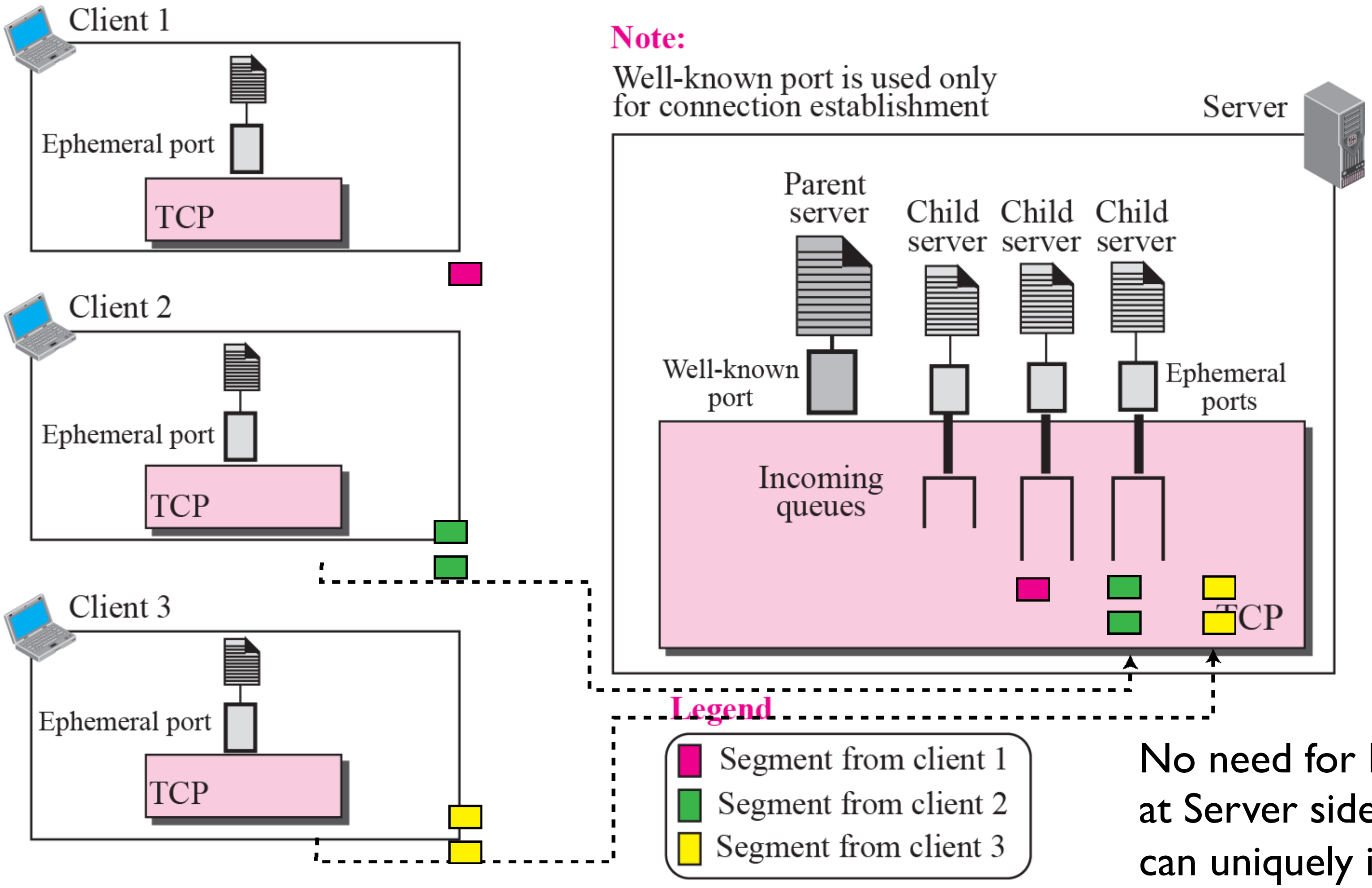


Can be concurrent!

Client-Server Applications Development - TCP

- TCP service: reliable transfer of bytes from one process to another
  - server process must be running first
  - server must have created socket (door) that welcomes client's contact
  - The client creates client-side local TCP socket specifying IP address, port number of server process to bind to the server.
  - When client creates socket: client TCP establishes connection to server TCP
  - Because TCP identifies a connection by a pair of endpoints (sockets), a given TCP port number can be shared by multiple connections on the same machine.

TCP Server: Connection-oriented (Concurrent)



No need for Ephemeral ports at Server side as most systems can uniquely identify a connection through 5-tuple: (local-IP, local-port, remote-IP, remote-port, protocol).

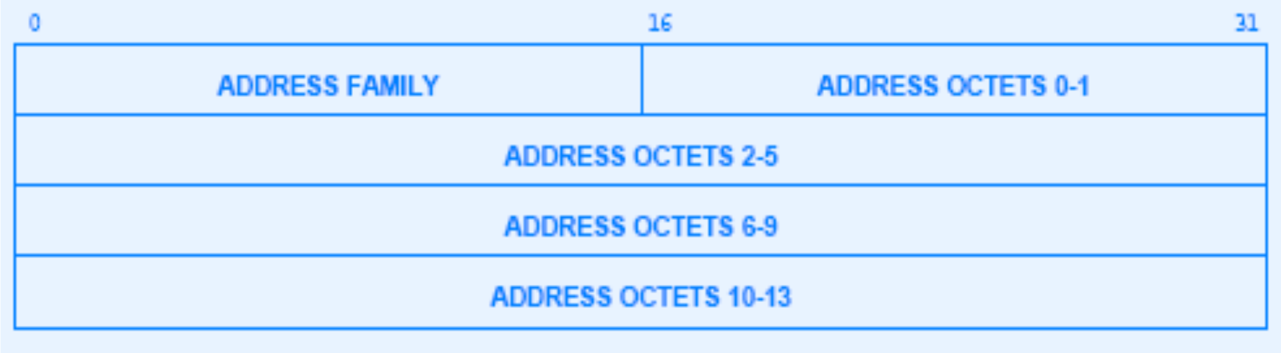
Can be Iterative!

UNIX/LINUX Socket API (C/C++)

- Creating a Socket
  - **sockDes = new socket(pf, type, protocol)** - creates a socket
    - pf - protocol family (PF_INET for IPv4, PF_INET6 for IPv6…)
    - type – SOCK_STREAM for TCP(reliable), SOCK_DGRAM for UDP(unreliable), SOC_SEQPACKET for sctp, SOCK_RAW only for privileged programs to access low-level protocols and network interfaces
    - protocol – if many protocols in the PF provides same TYPE of service, this field is used to select the Protocol, otherwise, it is set to 0. (Eg. Only TCP provides reliable service in PF_INET family, hence this third argument can be set to 0)
    - NOTE: All other functions take sockDes as input parameter. (similar to Unix fileDescriptor!)

UNIX/LINUX Socket API (C/C++)

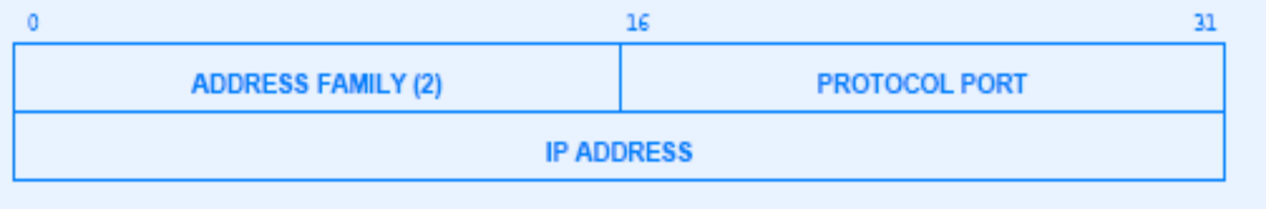
- Binding the Socket to local Internet Address (IP and PORT)
  - **bind(sockDes, localaddr, addrlen)** - *[socket will have wild-card foreign destination; foreign addr is yet to be assigned]*
    - *localaddr is pointer to a structure (sockaddr, sockaddr_in are structures defined in sys/socket.h) which specifies the local socket address*
    - *addrlen is length of the socket address*



0 16 31

ADDRESS FAMILY	ADDRESS OCTETS 0-1
ADDRESS OCTETS 2-5	
ADDRESS OCTETS 6-9	
ADDRESS OCTETS 10-13	

sockaddr structure -> generic



0 16 31

ADDRESS FAMILY (2)	PROTOCOL PORT
IP ADDRESS	

sockaddr_in structure -> for TCP/IP address

* *Address Family = 2 is for TCP/IP address*

UDP Server

- UDP server (5 steps)
  1. sockDes = new **socket**(pf, type, protocol) - creates a socket
  2. **bind**(sockDes, localaddr, addrlen) - *[socket will have wild-card foreign destination; foreign addr is yet to be assigned]*
  3. **recvfrom**(sockDes, buffer, length, flags, fromaddr, addrlen)
     - *OS will record the fromaddr and addrlen based on the datagram received*
  4. **sendto**(sockDes, message, length, flags, destaddr, addrlen)
  5. **close**(sockDes)

UDP Client

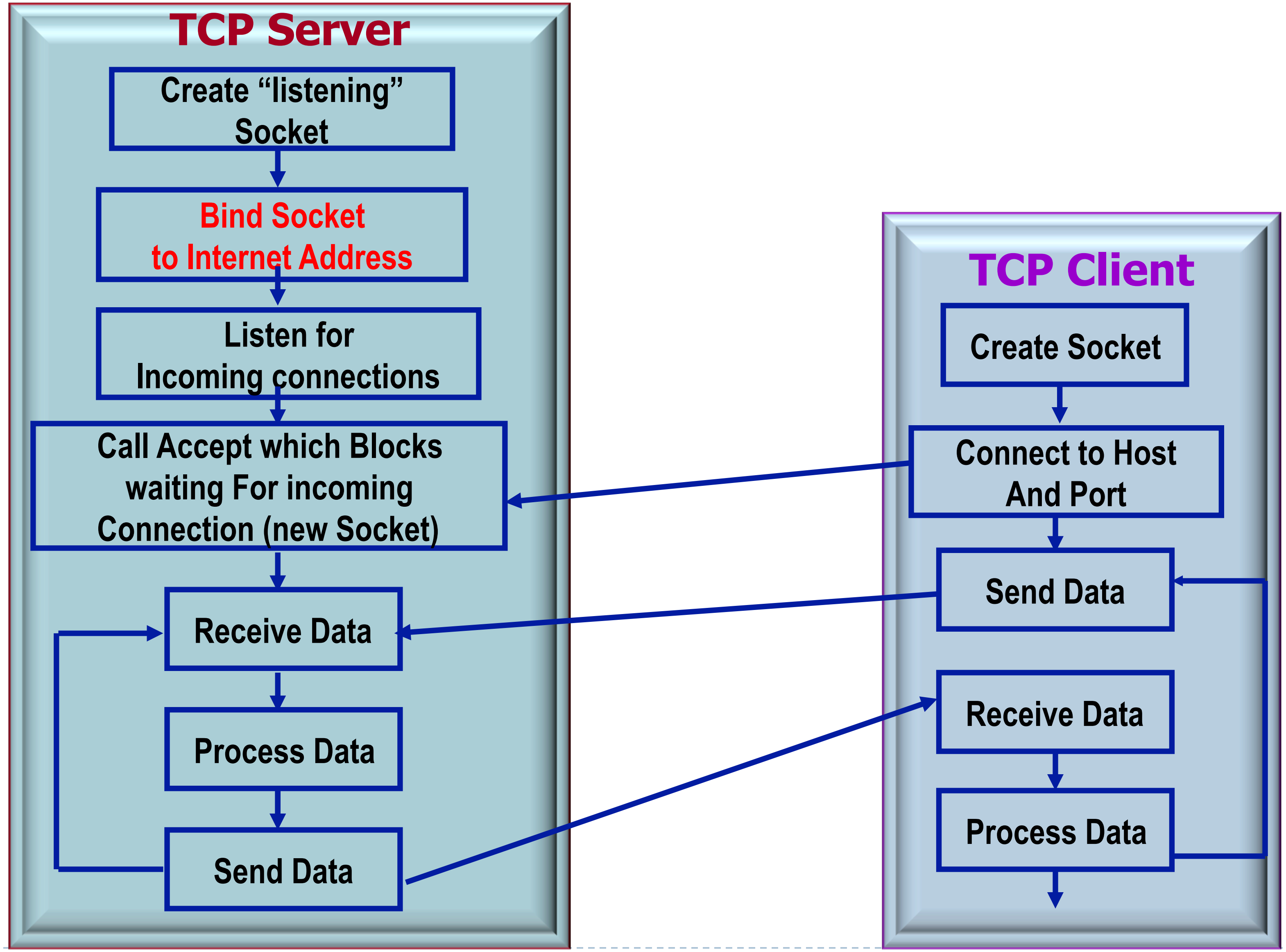
- UDP Client (4 steps)

1. sockDes = new **socket**(pf, type, protocol) - creates a socket
2. **sendto**(sockDes, message, length, flags, destaddr, addrlen)
3. **recvfrom**(sockDes, buffer, length, flags, fromaddr, addrlen)
    - *OS will record the fromaddr and addrlen based on the datagram received*
4. **close**(sockDes)

DEMO!

Unix Socket API (C/C++) – TCP

TCP Client/Server


TCP Server
Create “listening” Socket
Bind Socket to Internet Address
Listen for Incoming connections
Call Accept which Blocks waiting For incoming Connection (new Socket)
Receive Data
Process Data
Send Data
TCP Client
Create Socket
Connect to Host And Port
Send Data
Receive Data
Process Data

TCP Server

- TCP Server (6 steps)

1. sockDes = new **socket**(pf, type, protocol) - creates a socket ["Listen Socket" or "Server Socket"]
2. **bind**(sockDes, localaddr, addrlen) - *[socket will have wild-card foreign destination; foreign addr is yet to be assigned]*
3. **listen**(sockDes, qlength)
4. newSockDes = **accept**(sockDes,addr,addrlen) *[new socket will have requesting client as destination and returns newSockDes to server, connection is established tospecific client] ["Connect Socket"].*
5. **read/write** (newSockDes, buffer, length)
6. **close**(newSockDes), **close**(sockDes)

TCP Client

- TCP Client (4 steps)
  1. sockDes = new **socket**(pf, type, protocol) **-** creates a socket ["Client Socket"]
  2. **connect**(socket, destaddr, addrlen) *[socket will have local and foreign destination, connection establised]*
  3. **read/write** (sockDes, buffer, length)
  4. **close**(sockDes)

DEMO!

RAW Socket

- Receiving DATA through RAW SOCKET (4 steps)
  1. sockDes = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP); - creates a socket
  2. **Bind** – binding to local port, or ethernet interface is optional. If not bound, it will capture all packets.
  3. **recvfrom**(sockDes, buffer, length, flags, fromaddr, addrlen)
     - *OS will record the fromaddr and addrlen based on the datagram received*
     - Eg. recvfrom(sockDes,buf,10000,0,(struct sockaddr *)&cliaddr, &clilen);
  4. **close**(sockDes)

- Sending DATA through RAW SOCKET (4 steps)
  - Figure out the method to send raw IP packets. (try it yourself!)

DEMO!

UNIX/LINUX Sockets - Commands

- More functions
  - **getpeername**(sockDes, destaddr, addrlen)
    - sockDes – for which the address is required
    - OS will fill the destaddr and addrlen
  - **getsockname**(sockDes, localaddr, addrlen)
    - sockDes – for which the local address is required
    - OS will fill the localaddr and addrlen
  - Network Byte Order conversion (for integers)
    - □ ntohs - network to host short,
    - □ htons - host to network short
    - localshort = **ntohs**(netshort)
    - locallong = **ntohl**(netlong)
    - netshort= **htons**(localshort)
    - netlong = **htonl**(locallong)

UNIX/LINUX Sockets - Commands

- More functions
  - Translation between 32-bit integer IP addr and the corresponding dotted decimal notation string

error-code = inet_aton(string_address)	forms 32 bit host adddr
address = inet_network(string)	Forms network address, with zeros for host part
str=inet_ntoa(internetaddr)	Forms dotted decimal notation
internetadr=inet_makeaddr(net, local)	Combines net and host address.
net=inet_netof(internetaddr)	Separates and returns network part of the addr
local=inet_lnaof(internetaddr)	Separates and returns the host part of the addr

UNIX/LINUX Sockets - Commands

- More Functions
  - Obtaining information about Hosts (official host name, aliases, host address type, address length ….)

```
ptr = gthostbyname(namestr)
ptr = gethostbyaddr(addr, len, type)
```

ptr - points to the structure containing the details

Windows Sockets (Winsock, Winsock2) - A modified version of BSD socket API used in windows systems.

Sample Codes and Demonstrations

PEER-TO-PEER Paradigm

- Although most of the applications available in the Internet today use the client-server paradigm, the idea of using peer-to-peer (P2P) paradigm recently has attracted some attention. In this paradigm, two peer computers can communicate with each other to exchange services. This paradigm is interesting in some areas such file as transfer in which the client-server paradigm may put a lot of the load on the server machine. However, we need to mention that the P2P paradigm does not ignore the client-server paradigm; it is based on this paradigm.

END