CS3103: Computer Networks Practice

Network Applications Development

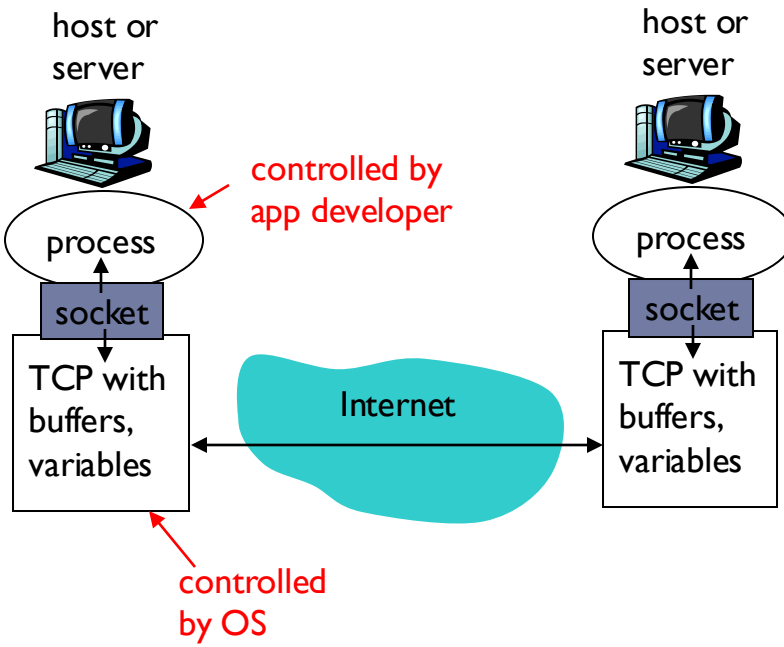
Socket API – A short introduction

Client/Server Applications Demo

Anand Bhojan
COM3-02-49, School of Computing
banand@comp.nus.edu.sg ph: 651-67351

Socket

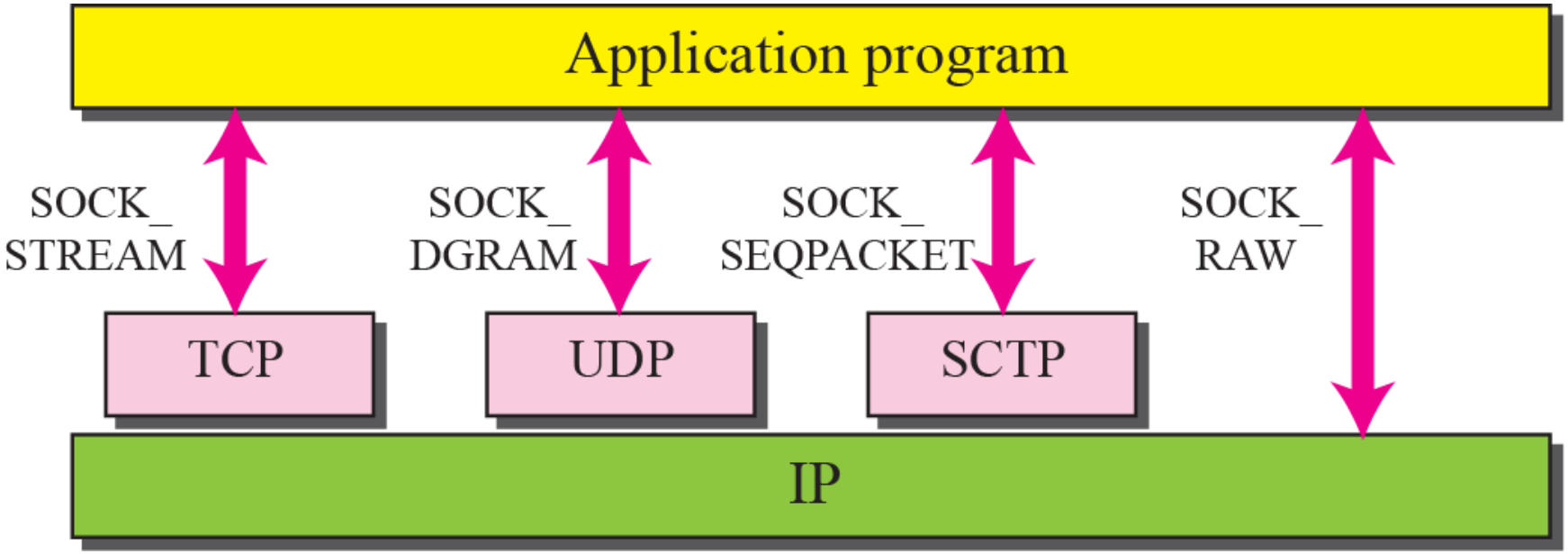
- Socket
  - Interface between the application layer and transport layer
  - OS-controlled interface (a "door") into which application process can both send and receive messages
- Addressing
  - Host address + process identifier
  - Eg. IP address + port number
    - Eg HTTP – port 80, SMTP – port 25, 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.
- Use '*netstat –a*' to show all active connections.
- Use '*lsof -i –n*' to show the COMMAND, PID (process ID), and USER (user ID) of open Internet sockets.

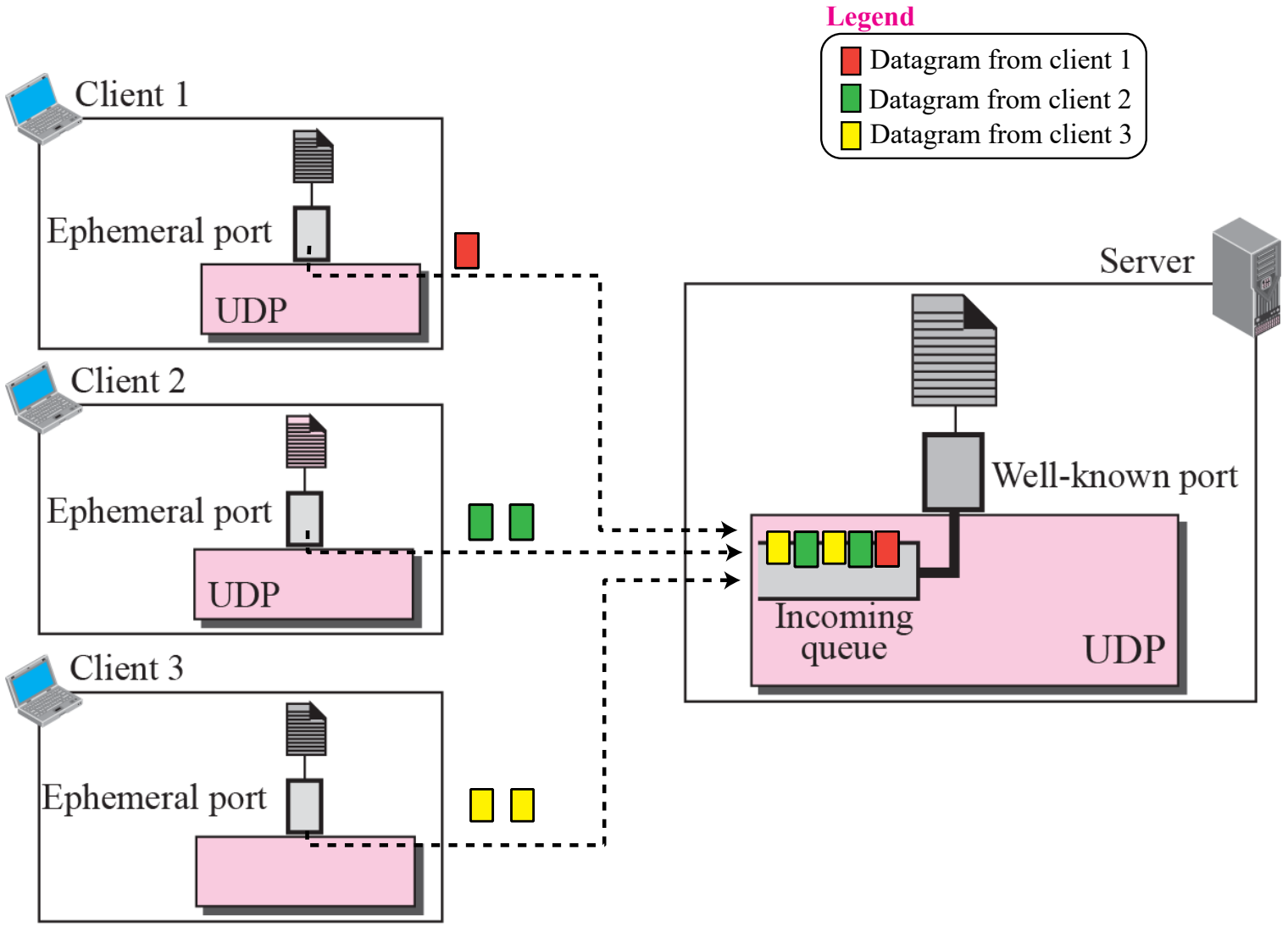
Socket: endpoint for comm. A combination of host IP address and port number.

Socket Types



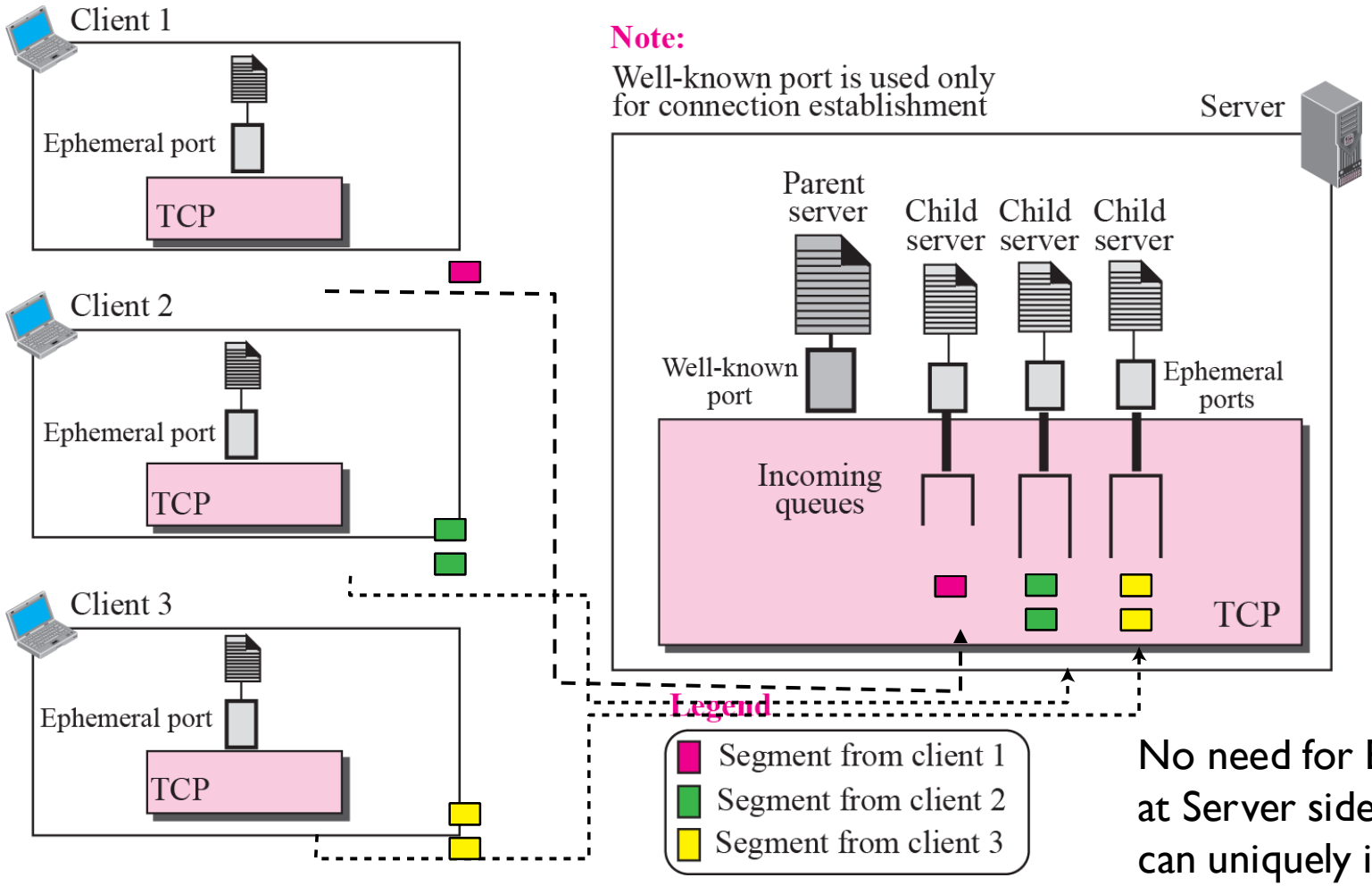
- Types of transport service via socket:
  - unreliable datagram
  - reliable, byte stream-oriented
- Lower level protocols and network interfaces can be accessed through 'RAW Socket'. (c, c++, python, not in Java)
- The new SCTP socket provides multiple types of service

UDP Server: Connectionless (Iterative)



Can be concurrent!

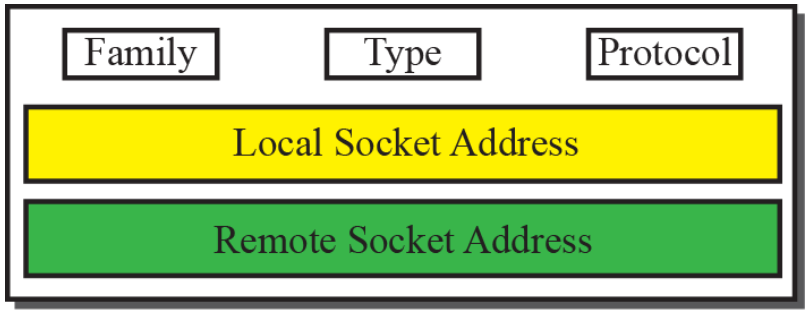
TCP Server: Connection-oriented (Concurrent)



Can be Iterative!

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).

Socket Data Structure



Fields

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

Generic definition

UNIX/LINUX Socket API (Python)

- Creating a Socket

socket.socket(*family, type, proto, fileno*)

- **family**: Protocol Family (PF)/ Address Family(AF) - socket.**AF_INET,** socket.**AF_INET6**
- **type**: socket.**SOCK_STREAM,** socket.**SOCK_DGRAM,** socket.**SOCK_RAW,** socket.**SOCK_SEQPACKET**
- **proto**: protocol - if many protocols in the PF / AF 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)
- **fileno**: If *fileno* is specified, the values for *family*, *type*, and *proto* are auto-detected from the specified file descriptor.

UDP Server (Python)

- UDP server (5 steps)
  1. OPEN: **s** = socket.**socket**(socket.AF_INET, socket.SOCK_DGRAM) - creates a socket
  2. BIND: **s.bind**(('127.0.0.1', 2080)) – bind socket to local host and given port number 2080. If IP address field is empty string, it will assign Ephemeral IP
  3. RECEIVE: **s.recvfrom**(bufferSize) - Returns a list containing [*message, client socket address*]
  4. SEND: **s.sendto**(messageToClient, destaddr)
  5. CLOSE: **s.close**()

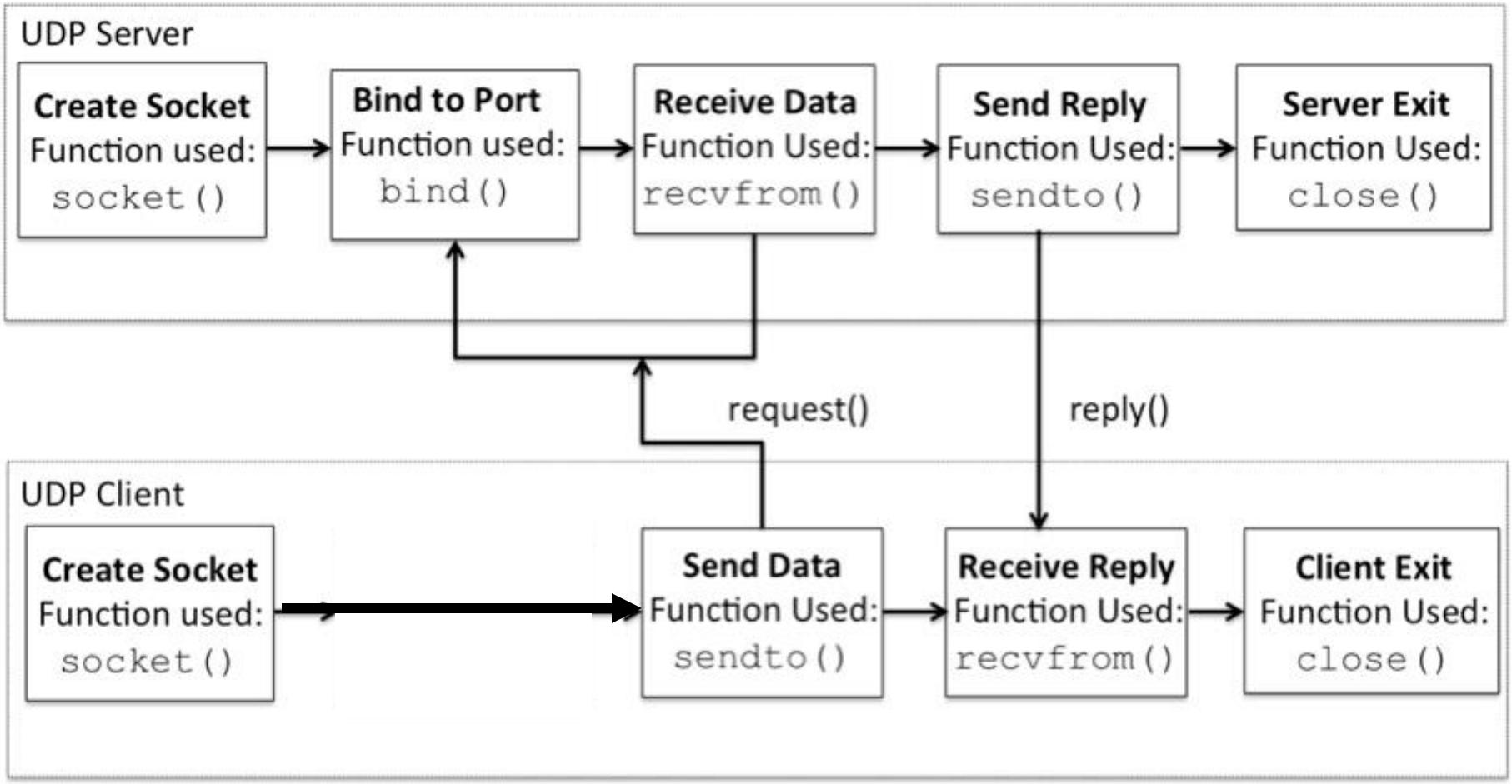
Note: source or destination addresses are Python **tuples** with **IP address string** and **port number**. Eg. ('198.2.3.90', 8080)

UDP Client (Python)

- UDP Client (4 steps)
  1. OPEN: **s** = socket.**socket**(socket.AF_INET, socket.SOCK_DGRAM) - creates a socket
  2. SEND: **s.sendto** (bytesToSend, destaddr)
  3. RECEIVE: **s.recvfrom**(bufferSize)
  4. CLOSE: **close**(sockDes)

Note: source or destination addresses are Python **tuples** with **IP address string** and **port number**. Eg. ('198.2.3.90', 8080)

UDP Client/Server (Summary)



DEMO!

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

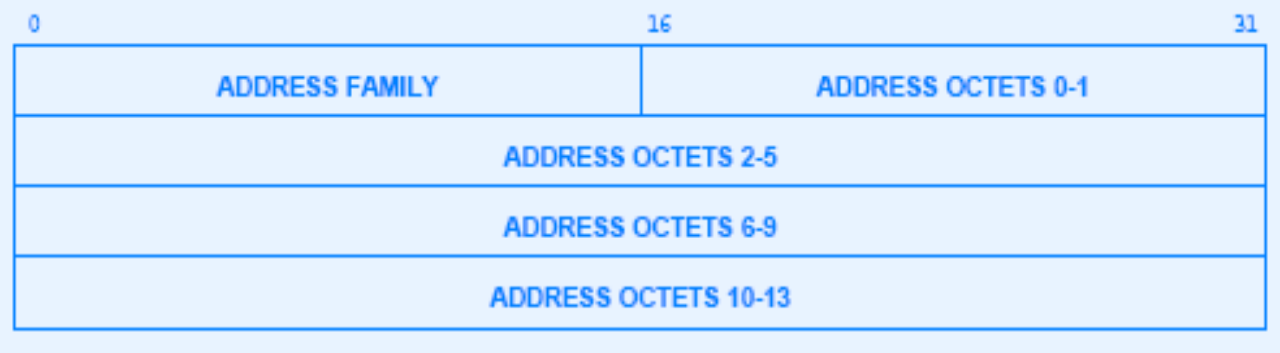
- Creating a Socket

sockDes = new socket(pf, type, protocol)

- **family**: Protocol Family (PF)/ Address Family(AF) - socket.**AF_INET,** socket.**AF_INET6**
- **type**: socket.**SOCK_STREAM,** socket.**SOCK_DGRAM,** socket.**SOCK_RAW,** socket.**SOCK_SEQPACKET**
- **protocol**: protocol - if many protocols in the PF / AF 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)

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

- Binding the Socket (server side) 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*



sockaddr structure -> generic

sockaddr_in structure -> for TCP/IP address

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

UDP Server (C/C++)

- 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)

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

Generic definition

UDP Client (C/C++)

- 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)

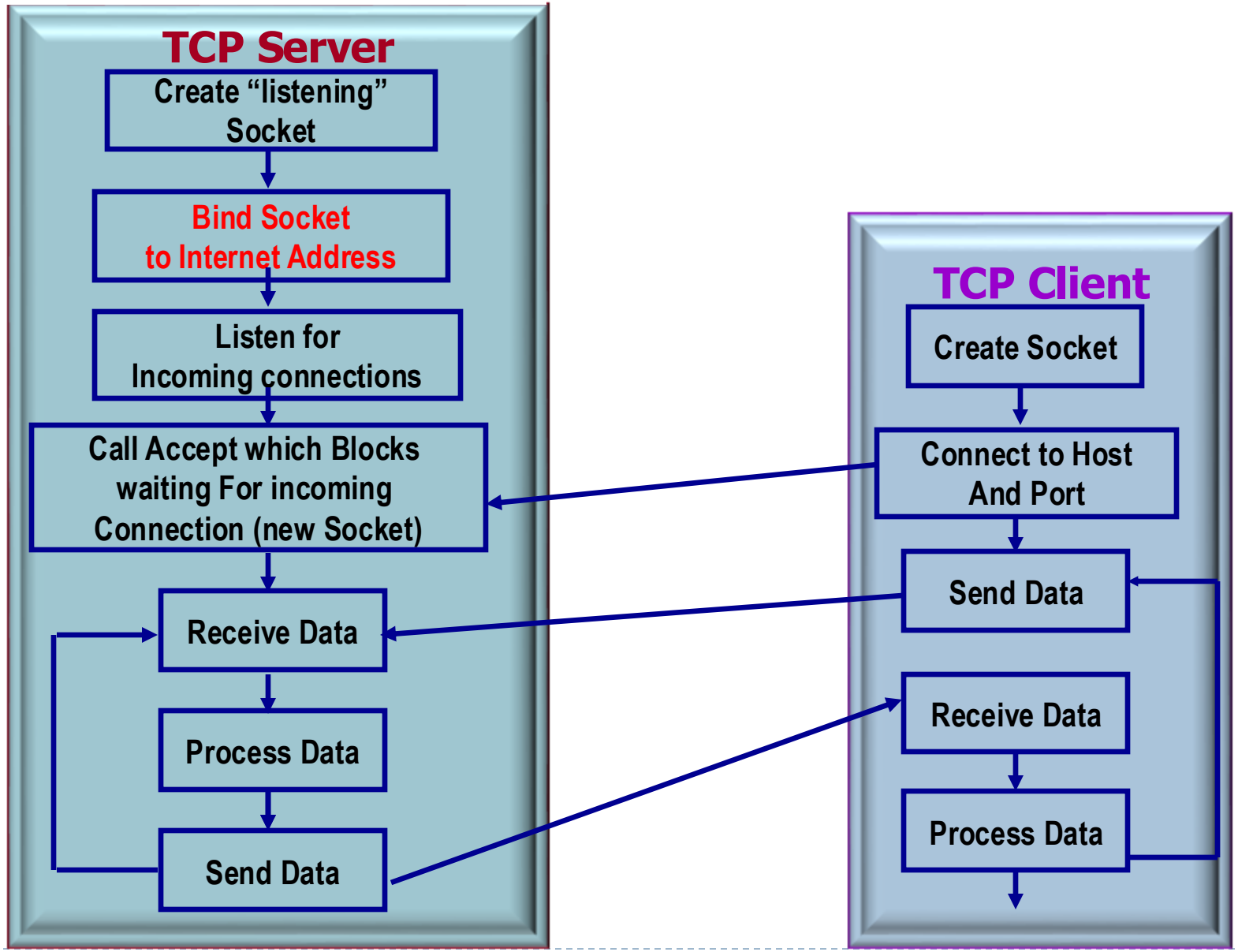
```
struct  socket
{
      int family;
      int type;
      int protocol;
     socketaddr local;
     socketaddr remote;
};
```

Generic definition

DEMO!

Unix Socket API (C/C++) – TCP

TCP Client/Server Overview



TCP Server (Python)

- TCP Server (6 steps)
  1. OPEN LISTEN SOCKET: sListen = socket.**socket**(socket.AF_INET, socket.SOCK_STREAM) - creates a socket ["Listen Socket" or "Server Socket"]
  2. BIND: sListen.**bind**(('127.0.0.1', 60080)) – bind socket to local host and given port number 60080.
  3. LISTEN: sListen.**listen(5)** - listen for incoming connections. max 5 concurrent connections
  4. CONNECT SOCKET: sConn, addr = s.**accept()** *[new connection socket is created] ["Connect Socket"].*
  5. RECEIVE/SEND DATA:
     a) data = sConn.**recv**(1024) - The bufsize argument of 1024 used above is the maximum amount of data to be received at once.
     b) sConn.**sendall**(data)
  6. CLOSE SOCKETS: conn.**close**(), sListen.**close**()

TCP Client (Python)

- TCP Client (4 steps)
  1. OPEN: **s** = socket.**socket**(socket.AF_INET, socket.SOCK_STREAM) - creates a socket
  2. SEND: s.**sendall**(bytesToSend)
  3. RECEIVE: recvdata = s.**recv**(bufferSize)
  4. CLOSE: s.**close**()

DEMO!

TCP Server (C/C++)

- 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 (C/C++)

- 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 established]*
  3. **read/write** (sockDes, buffer, length)
  4. **close**(sockDes)

DEMO!

RAW Socket (Python)

- Receiving DATA (ICMP message) through RAW SOCKET (4 steps)
  1. OPEN: s = socket.**socket**(socket.AF_INET,socket.SOCK_RAW,socket.IPPROTO_ICMP)
  2. s.**setsockopt**(socket.SOL_IP, socket.IP_HDRINCL, 1)
     - Level, Option Name, Value. [IP Level (NW later), Option is whether to Include IP4 headers, Option value =1. That means the IPv4 header should be included in the messages. (NW layer will not autogenerate header)
  3. RECEIVE: recPacket, addr = s.**recvfrom**(1024)
  4. CLOSE: s.**close**()

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

DEMO!

RAW Socket (C/C++)

- Receiving DATA through RAW SOCKET (4 steps)
  1. sockDes = socket(PF_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!

HTTP/3.0 (Python)

- There are third-party libraries to create HTTP/3.0 Server and Client.
- You can try yourself using the library - **AIOQUIC**
  - https://github.com/aiortc/aioquic
  - Example Client/Server codes are available in the same repo.

UNIX/LINUX Sockets –UNIX C API [C/C++/Python]

- More functions of UNIC Socket Interface. [*python function names are similar and they call underlying UNIX C API]
  - **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)

https://docs.python.org/3/library/socket.html

UNIX/LINUX Sockets –UNIX C API [C/C++/Python]

- 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 –UNIX C API [C/C++/Python]

- 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