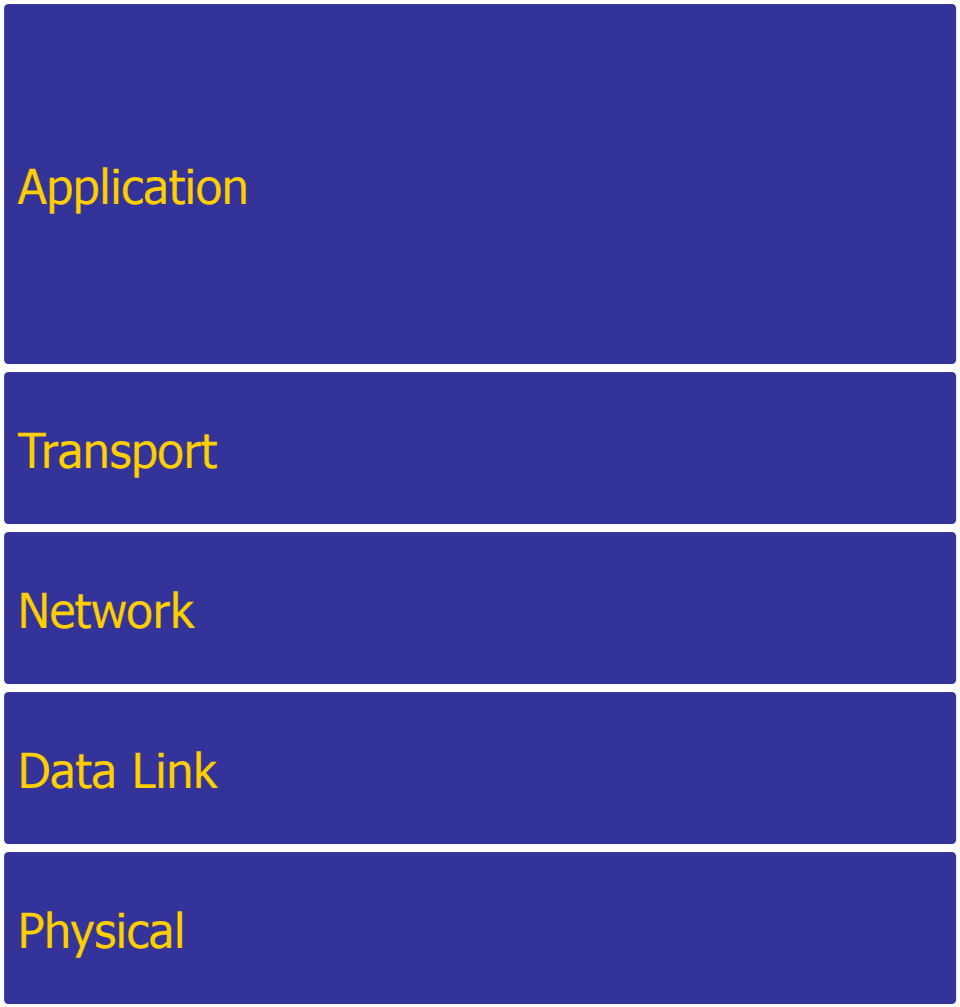
Socket and Network Programming

by Mohamad H. Danesh

Network Layers



End-to-End Connection

Routing

Framing

Physical topology

Physical Layer

- It sends bits and receives bits.

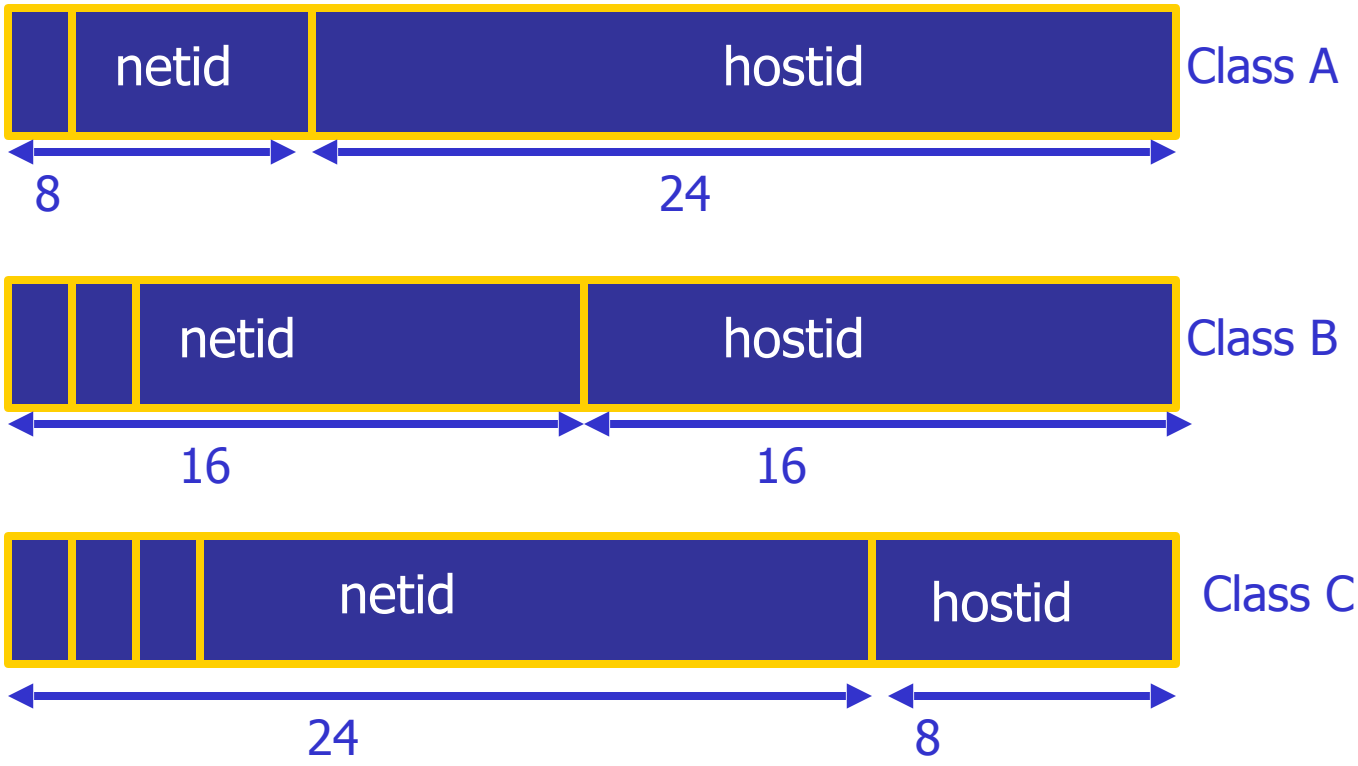
Data Link Layer

- It ensures that messages are delivered to the proper device.
- It translates messages from the Network layer into bits for physical layer to transmit.
  - It formats the message into data frames.
  - It adds a header containing the hardware destination and source address.

Network Layer

- It is responsible for routing through an internetwork and for network addressing.
  - It is responsible for transporting traffic between devices that are not locally attached.
- It uses software address.

IP Addresses



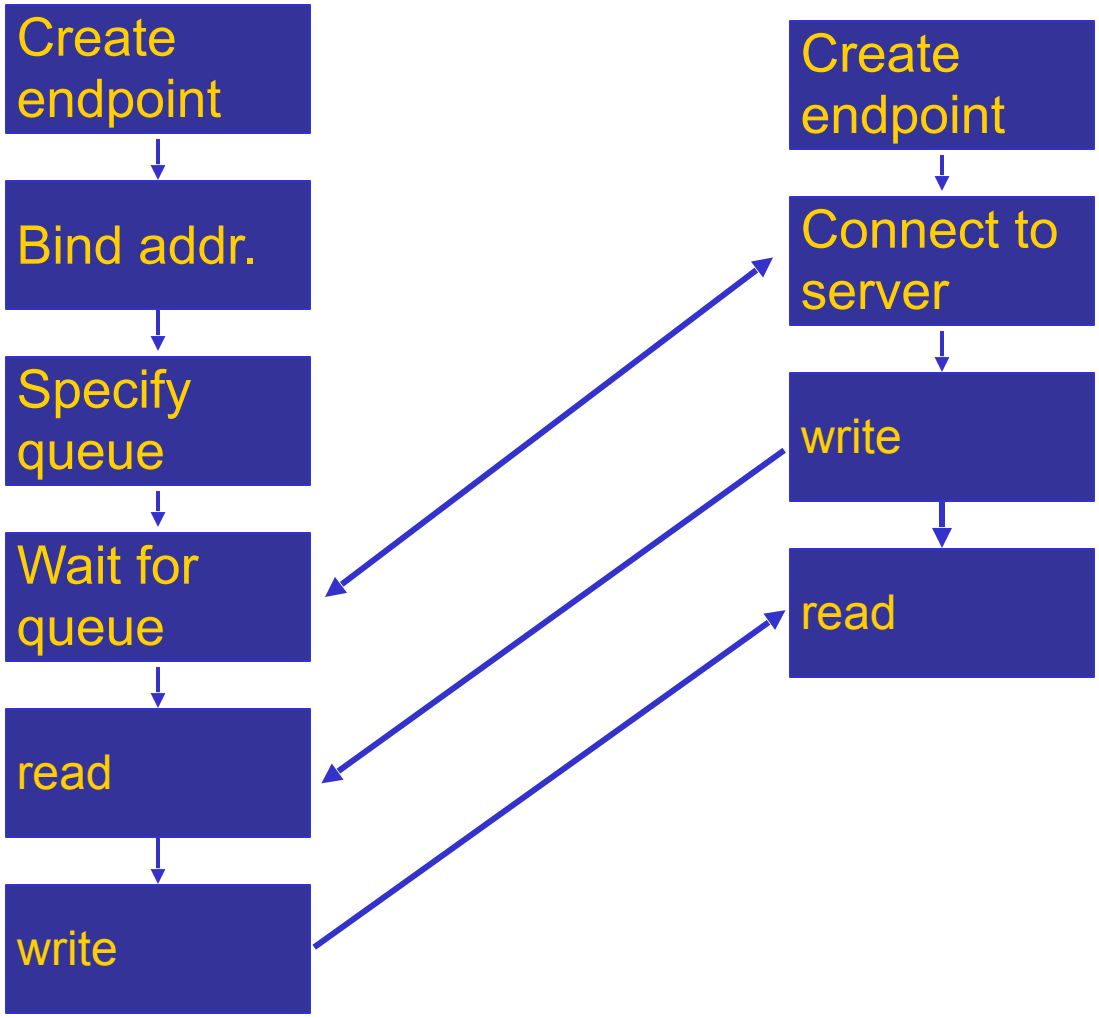
Transport Layer

- Flow control
  - It prevents a sending host on one side of connection from overflowing the buffers in the receiving host.
- Acknowledgment
  - It guarantees the data won't be duplicated or lost.
- Windowing
  - It controls how much information is transferred from one end to the other.

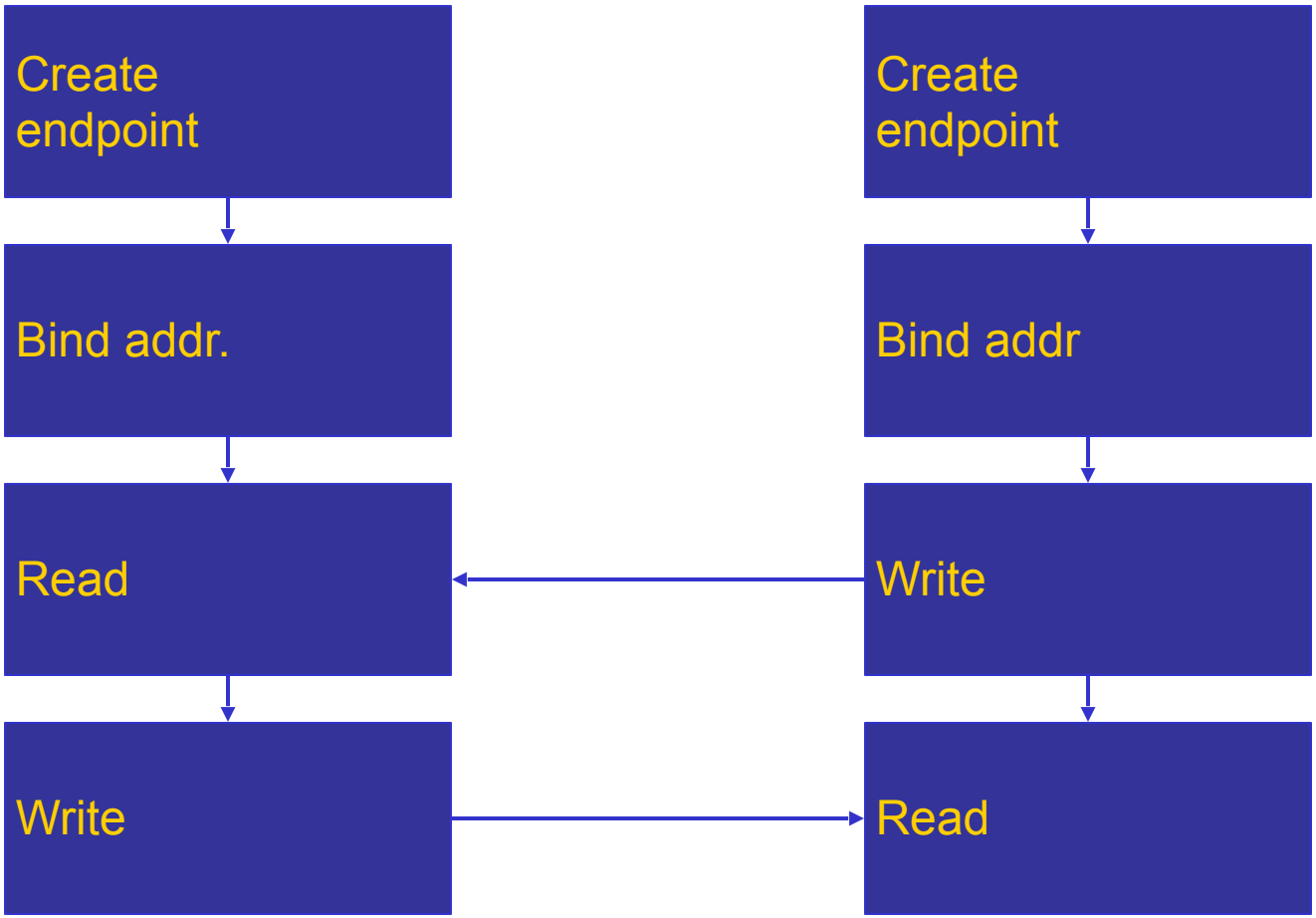
Network Connections

- The Transport layer provide two types of connection:
- Connection-less (UDP)
  - It is an unreliable connection.
- Connection-oriented (TCP)
  - It handshakes before transfers information.

Connection-oriented



Connection-less



Port Numbers

- It is possible for more than one user process at a time to be using either TCP or UDP.
- This requires some method for identifying the data associated with each user process.

Sockets

- What is a socket?
  - To an application, a socket is a file descriptor that lets the application read/write from/to the network.
    - Remember: All Unix I/O devices, including networks, are modeled as files.
- Clients and servers communicate with each by reading from and writing to socket descriptors.
- The main distinction between regular file I/O and socket I/O is how the application "opens" the socket descriptors.

Socket

- It is an interface between the application layer and other layers.

```
main()
{
    FILE *fd;
    fd = fopen (…);
    process (fd);
    fclose (fd);
}
```

```
main()
{
    int sockfd;
    sockfd = socket (…);
    process (sockfd);
    close (sockfd);
}
```

Type of Sockets

- Stream Socket
  - Provide a reliable, sequenced, two-way connection.
  - This is use TCP Socket.
- Datagram Socket
  - Unreliable connection.
  - This is use UDP Socket.
- Raw Socket
  - Used for internal network protocols.
  - a raw socket is a socket that allows direct sending and receiving of network packets by applications, bypassing all encapsulation in the networking software of the operating system.

socket System Call

- int socket (int family, int type, int protocol);
- It creates the end point.
- Family:
  - AF_INET, AF_UNIX, …
- Type:
  - SOCK_STREAM
  - SOCK_DGRAM
  - SOCK_RAW
- Protocol:
  - protocol of sockets

bind System Call

- int bind (int sockfd, struct sockaddr *addr, int addrlen);
- It assigns a name to an unnamed socket.

connect System Call

- int connect (int sockfd, struct sockaddr *addr, int addrlen);
- A client use it to establish a connection with a server.

listen System Call

- int listen (int sockfd, int backlog);
- This system call is used to indicate that it is willing to receive connections.

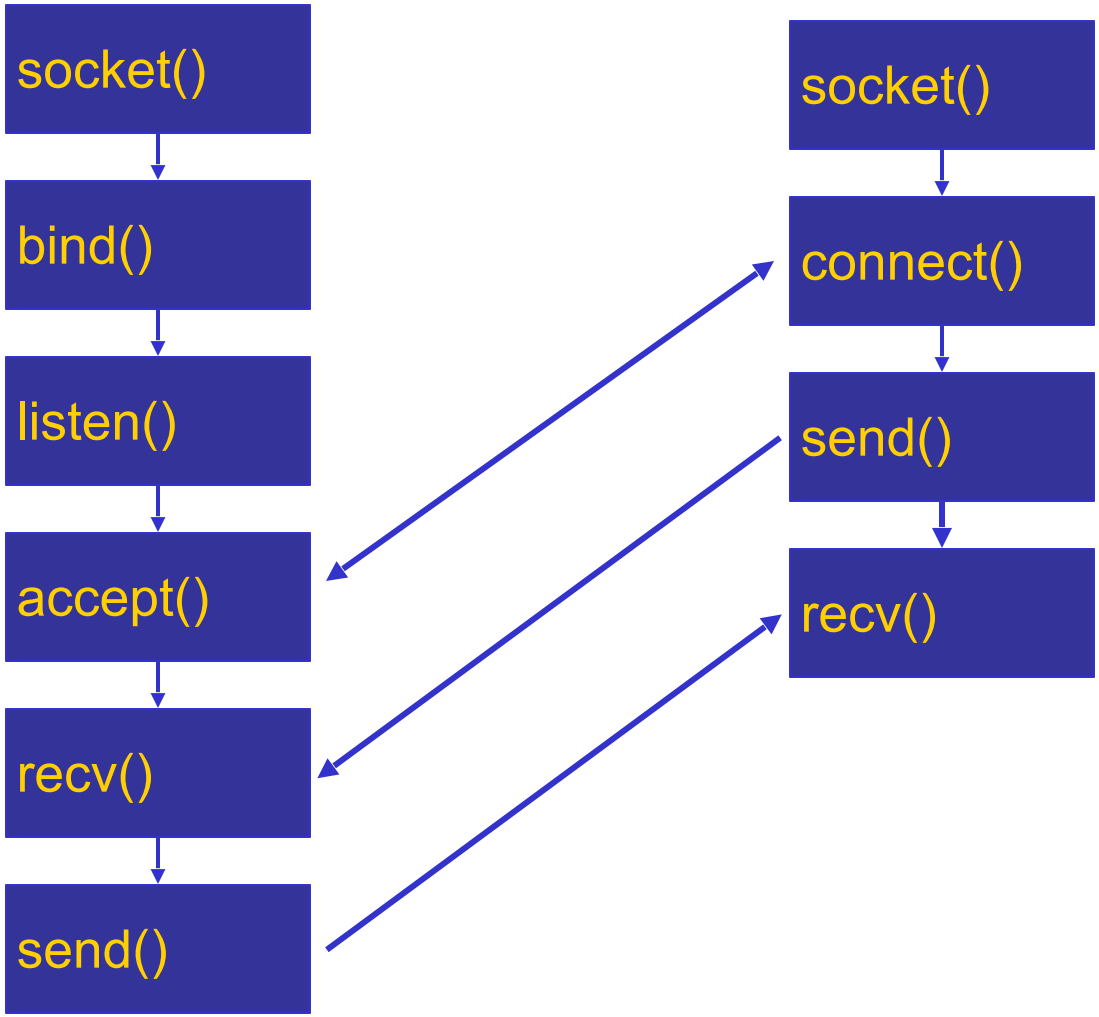
accept System Call

- int accept (int sockfd, struct sockaddr *addr, int *len);
- An incoming calls arrive at a listening socket, they will be queued until the server program ready to process them.

send and recv System Calls

- int send (int sockfd, char *buff, int len., int flag);
- int sendto (int sockfd, char *buff, int len., int flag, struct sockaddr *to, int addrlen);
- int recv (int sockfd, char *buff, int len., int flag);
- int recvfrom (int sockfd, char *buff, int len., int flag, struct sockaddr *from, int *addrlen);

Connection-oriented



Connection-less