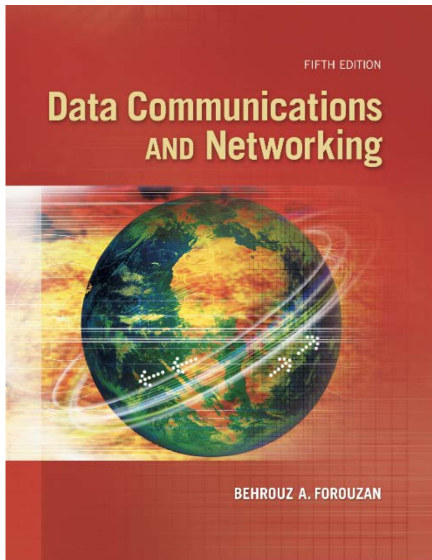
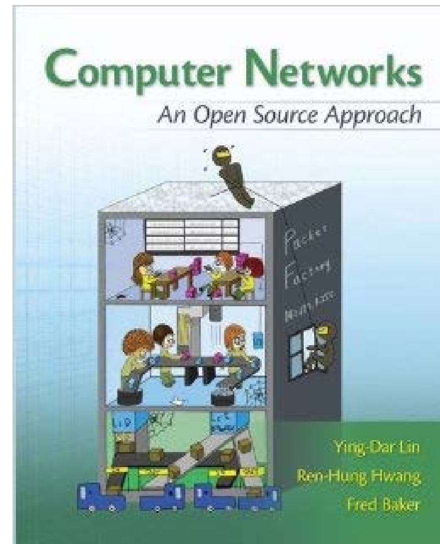
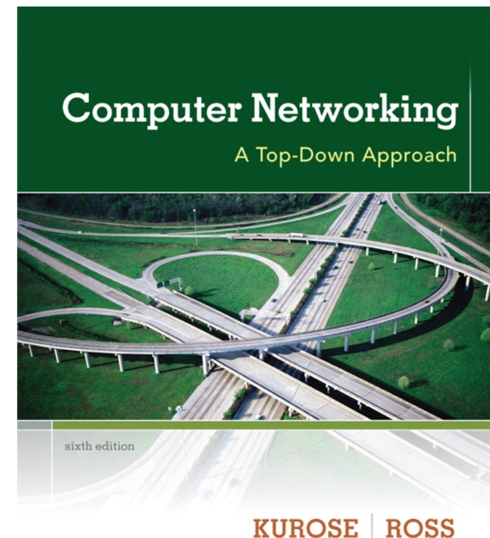
Introduction to Socket Programming





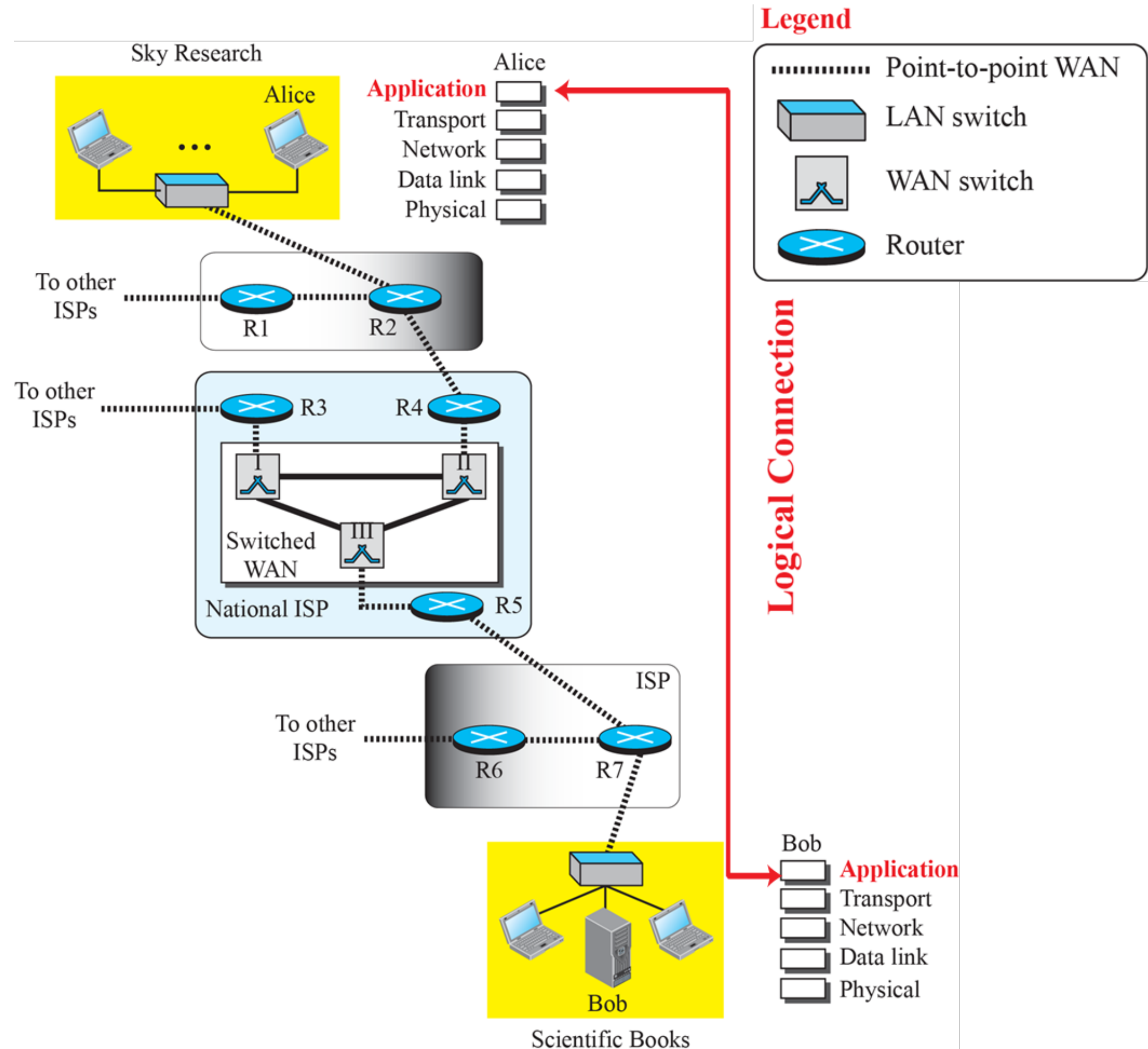
Disclaimer: These slides are inspired and the content is borrowed from the following textbooks



Thoshitha Gamage, Ph.D.
SIUE

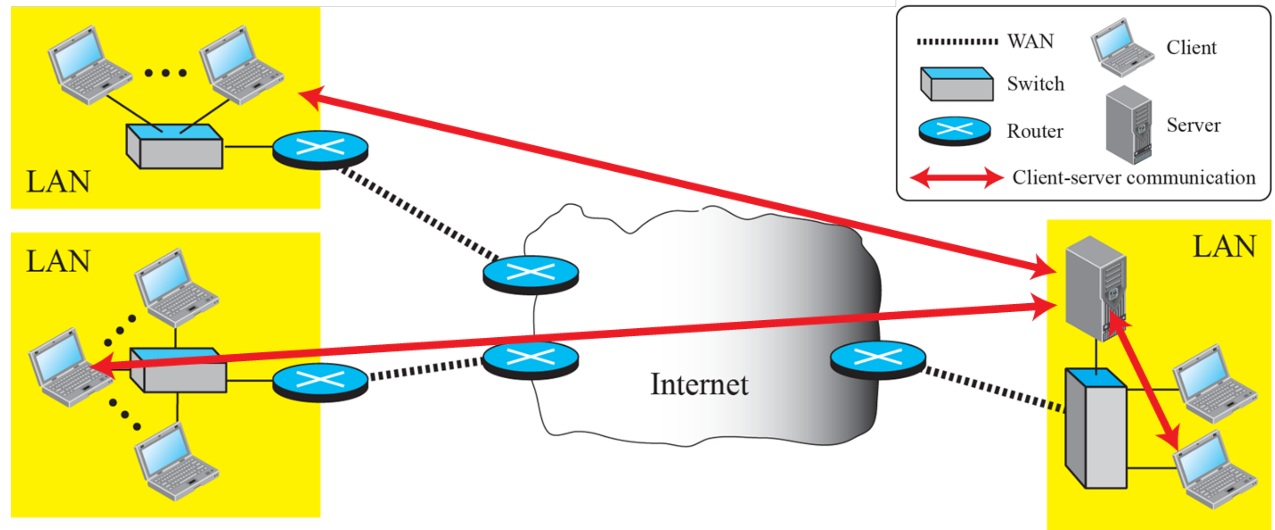
Logical Connection at the Application Layer

- Communication at the application layer is logical, not physical
- End points assume the existence of a two-way logical connection between them for communication
- Actual communication takes place through several devices
  e.g. Alice, R2, R4, R5, R6, and bob

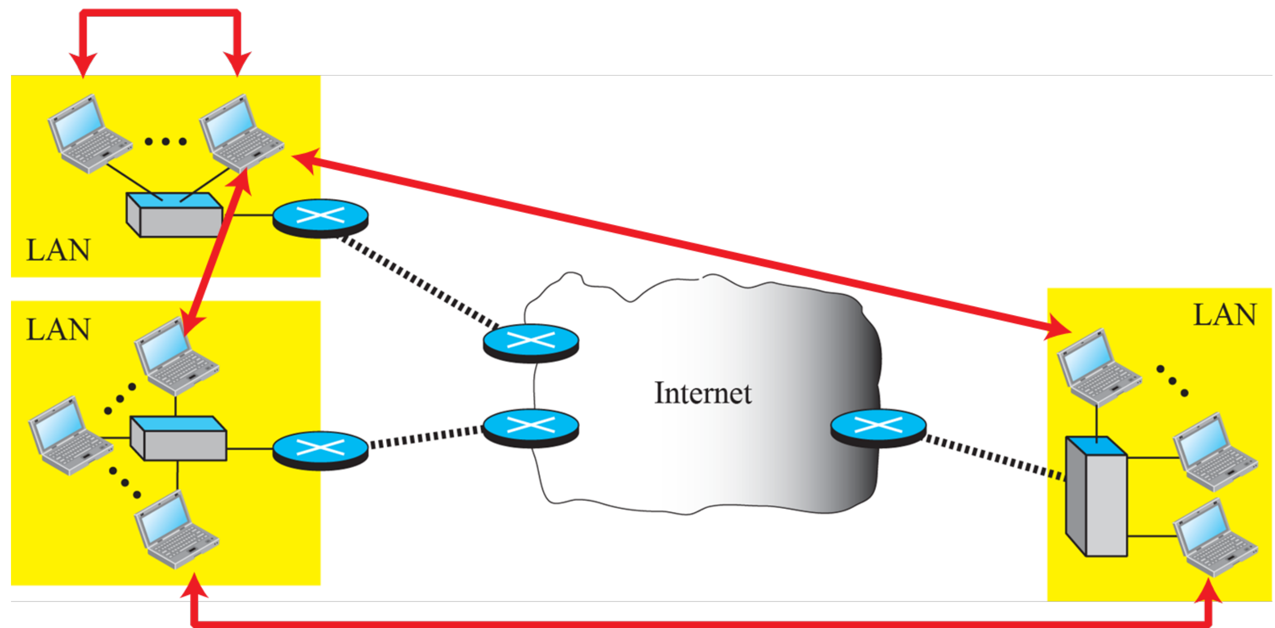


Client-Server vs. P2P Paradigm

Client-Server

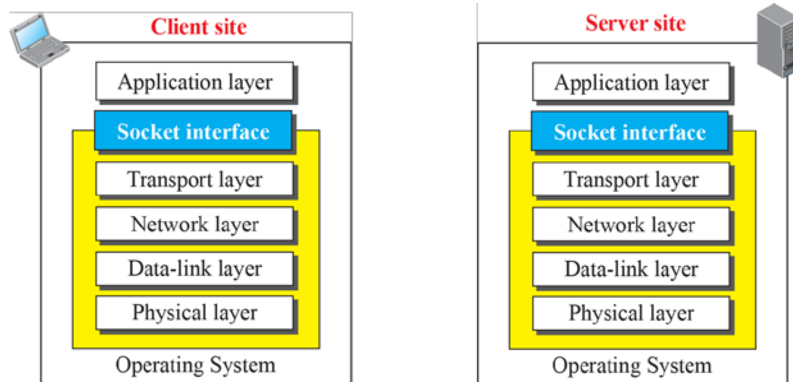


P2P



Client-Server Programming

- Communication occurs between two *processes* – programs in execution
    - A *client* and a *server*
- Client program initializes the communication; send the *request*
- Server program *waits* for client requests, *process*, and *responds* to client program
    - Server program must be running before client makes the request

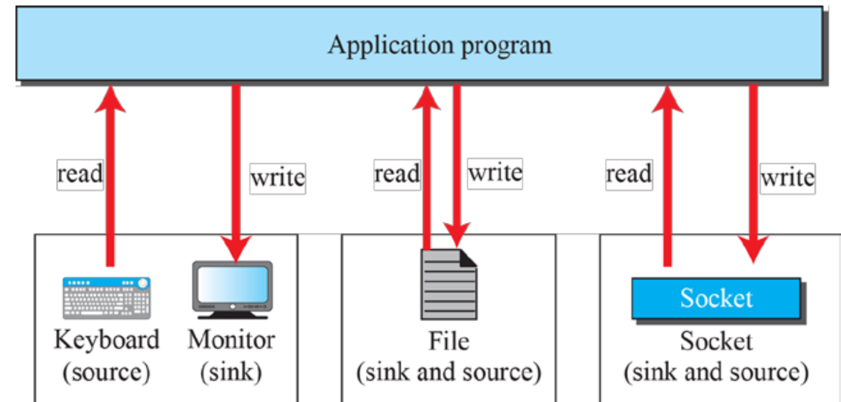


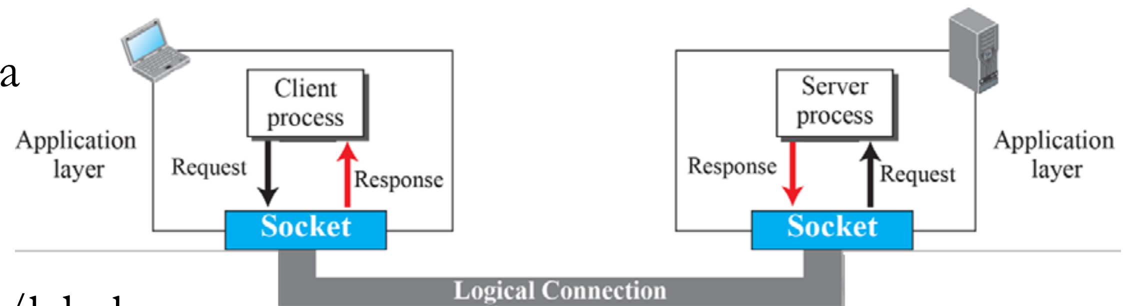
Application Programming Interface (API)

- The end-to-end processes must tell the lower four TCP/IP suite layers to
    - *Open* the connection
    - *Send* and *Receive* data from the other end
    - *Close* the connection
- The lower four layers are built into the OS with an API presented to the *Application Layer*
    - e.g. Socket Interface, TLI, STREAM

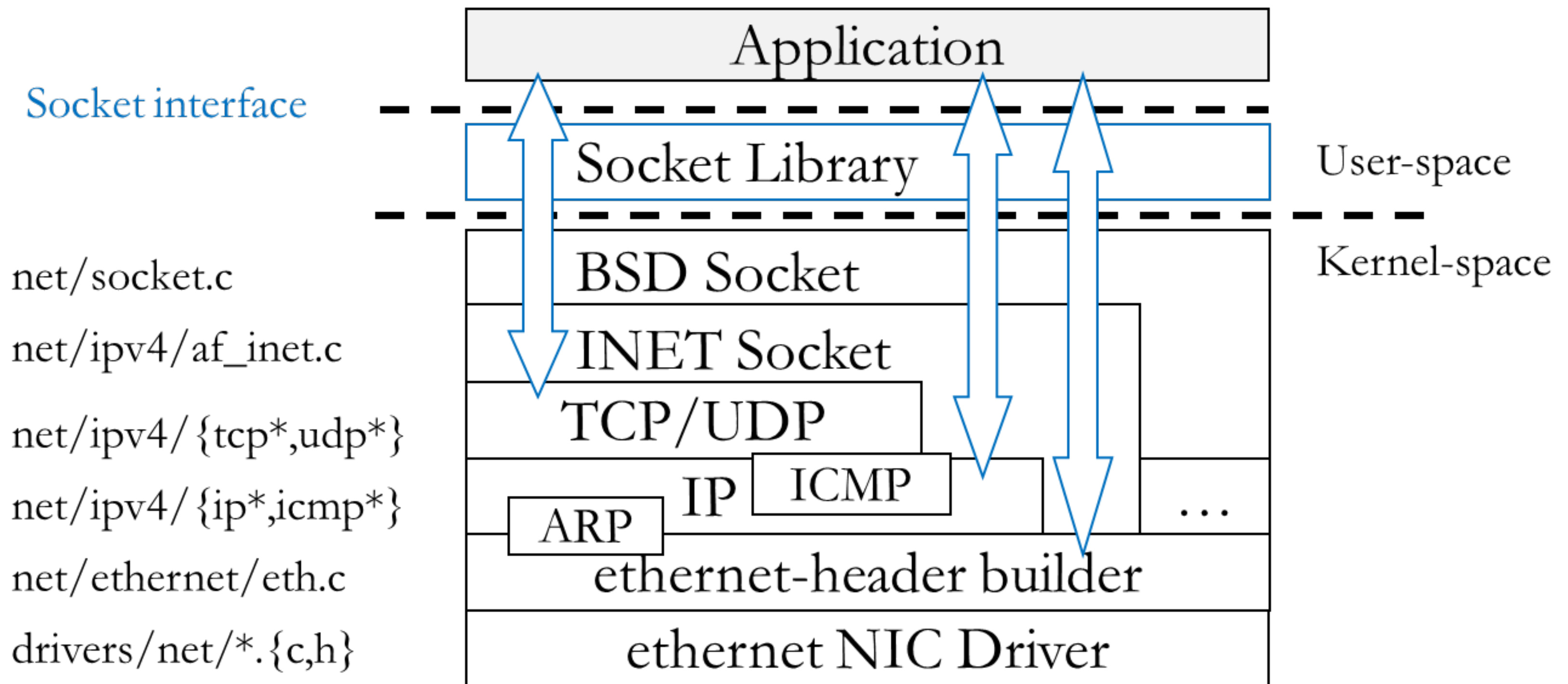
Sockets

- Core Idea: Use instructions already designed for *sources* and *sinks* in programming languages
    - E.g. file I/O – *source*: keyboard, *sink*: terminal
- Add ONLY new sources and sinks (for communication) w/o changing *read/write* instructions
- NOT a physical entity like a file; an abstraction
- End-to-end communication is between two sockets
- Has NO buffer to store data to be sent or received
- Not capable of sending or receiving data
    - Acts as just a reference/label





Socket Interface in Linux (kernel 2.7.11)



Socket Addressing

- Communication is between two (end-point) sockets
- Need a pair of *socket addresses*
  - Local (sender)
  - Remote (receiver)
- The local address one way is the remote address the other way
- A socket address
  - Defines the computer which runs the client or the server
    - IP address (32-bit)
  - Also defines the application running on the computer
    - Port (16-bit)
- Local address provided by the OS
  - OS knows the IP address
  - Port either assigned if a standard port or defined if otherwise
- Remote address
  - Server: Finds on the client connection request
  - Client: Should know server address before establishing connectivity
    - Manually
    - Explore and find (through DNS)

Socket Address

Transport Layer Services

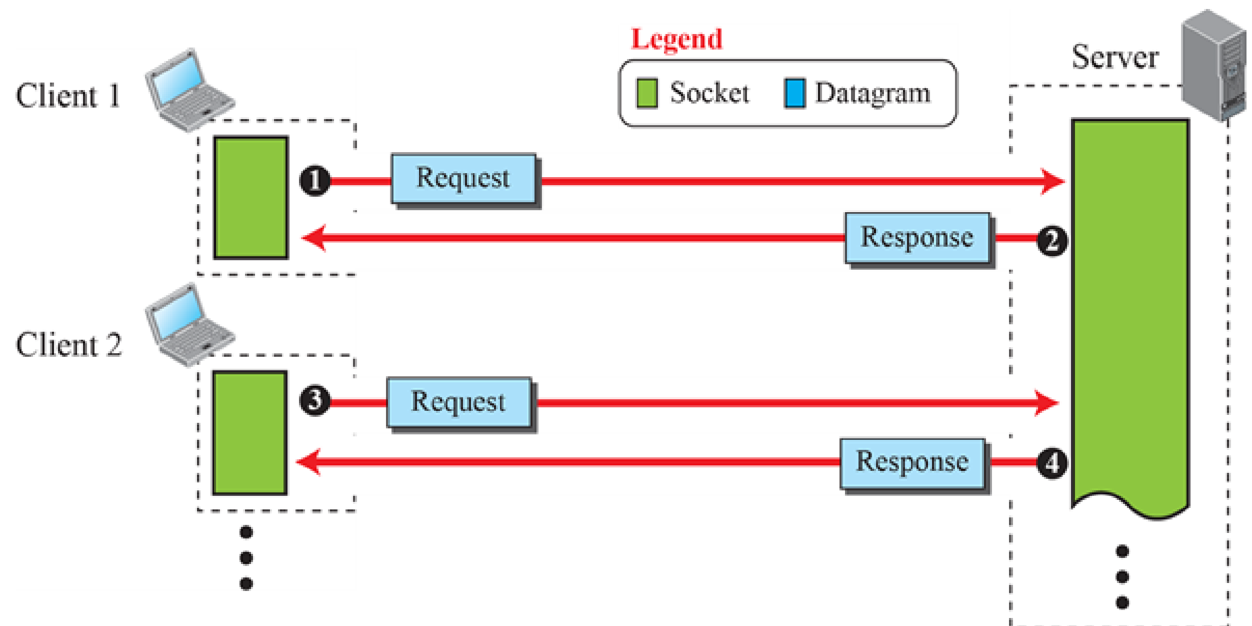
- Applications depend on services provided by the transport layer for communication
  - No physical communication at the application layer
- Common TCP/IP suite transport layer protocols
  - TCP
    - *Connection-oriented* – two endpoints establish a logical connection before communication begins
      - *Handshake*
    - Data exchanged in segments; resends allowed – *reliable*
    - Byte stream service
  - UDP
    - Each message an independent entity encapsulated in a datagram – *connectionless*
    - No resends for corrupt or lost datagrams – *unreliable*
    - *Message-oriented*; promotes *speed* over *reliability*
    - Datagram service
  - SCTP
    - Combination of TCP and UDP – connection-oriented, reliable, message-oriented

Standard Sockets

Port	*Protocol*	*UDP*	*TCP*	*Description*
7	Echo	√		Echoes back a received datagram
9	Discard	√		Discards any datagram that is received
11	Users	√	√	Active users
13	Daytime	√	√	Returns the date and the time
17	Quote	√	√	Returns a quote of the day
19	Chargen	√	√	Returns a string of characters
20, 21	FTP		√	File Transfer Protocol
23	TELNET		√	Terminal Network
25	SMTP		√	Simple Mail Transfer Protocol
53	DNS	√	√	Domain Name Service
67	DHCP	√	√	Dynamic Host Configuration Protocol
69	TFTP	√		Trivial File Transfer Protocol
80	HTTP		√	Hypertext Transfer Protocol
111	RPC	√	√	Remote Procedure Call
123	NTP	√	√	Network Time Protocol
161, 162	SNMP		√	Simple Network Management Protocol

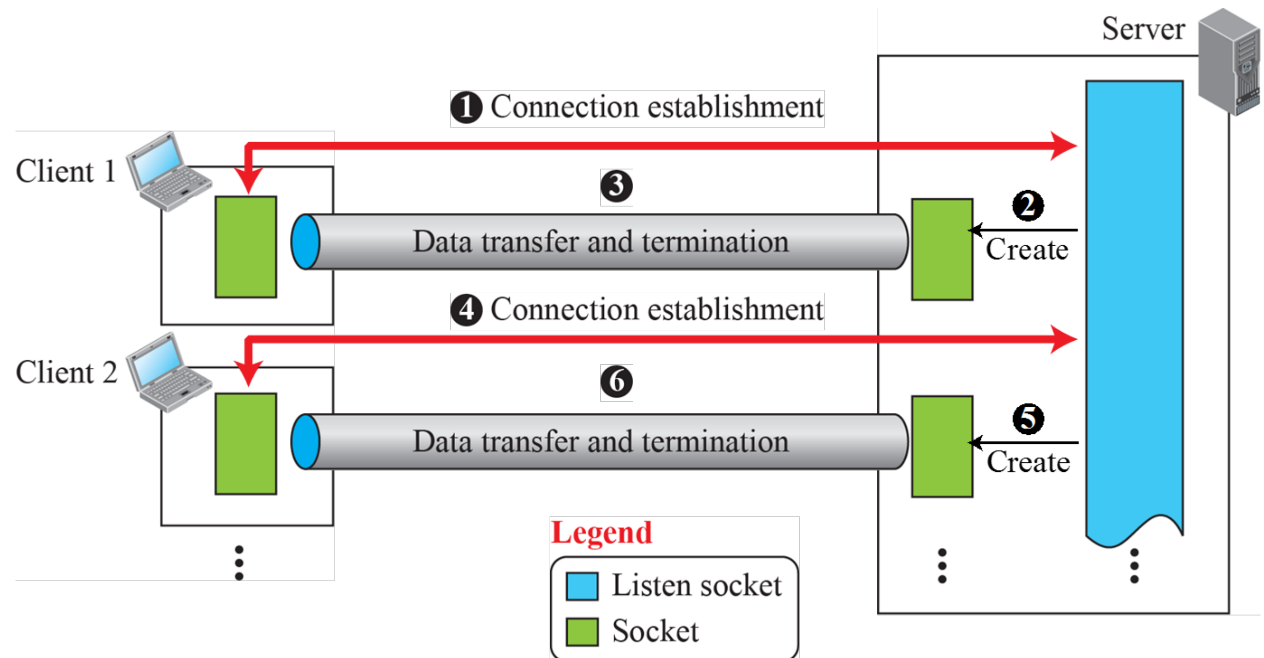
Iterative Communication : UDP

- Client and Server ONLY use one socket each
  - Server socket lasts forever
  - Client socket is closed when the client process terminates
- Different clients use different sockets
- Server creates only one socket. Changes remote socket address for each new client connection



Iterative Communication : TCP

- Server uses two different sockets
  - One used to establish connection – *listen socket*
    - Used to listen for incoming connection requests from clients
  - One for data transfer – *socket*
- Connection establishment separated from exchange



UDP Client/Server Socket Interaction

Server

create socket, port= x:

serverSocket = socket(AF_INET,SOCK_DGRAM)

read datagram from serverSocket

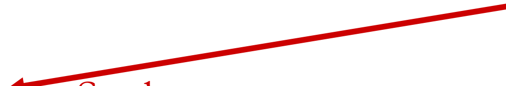
write reply to serverSocket specifying client address, port number

Client

create socket:

clientSocket = socket(AF_INET,SOCK_DGRAM)

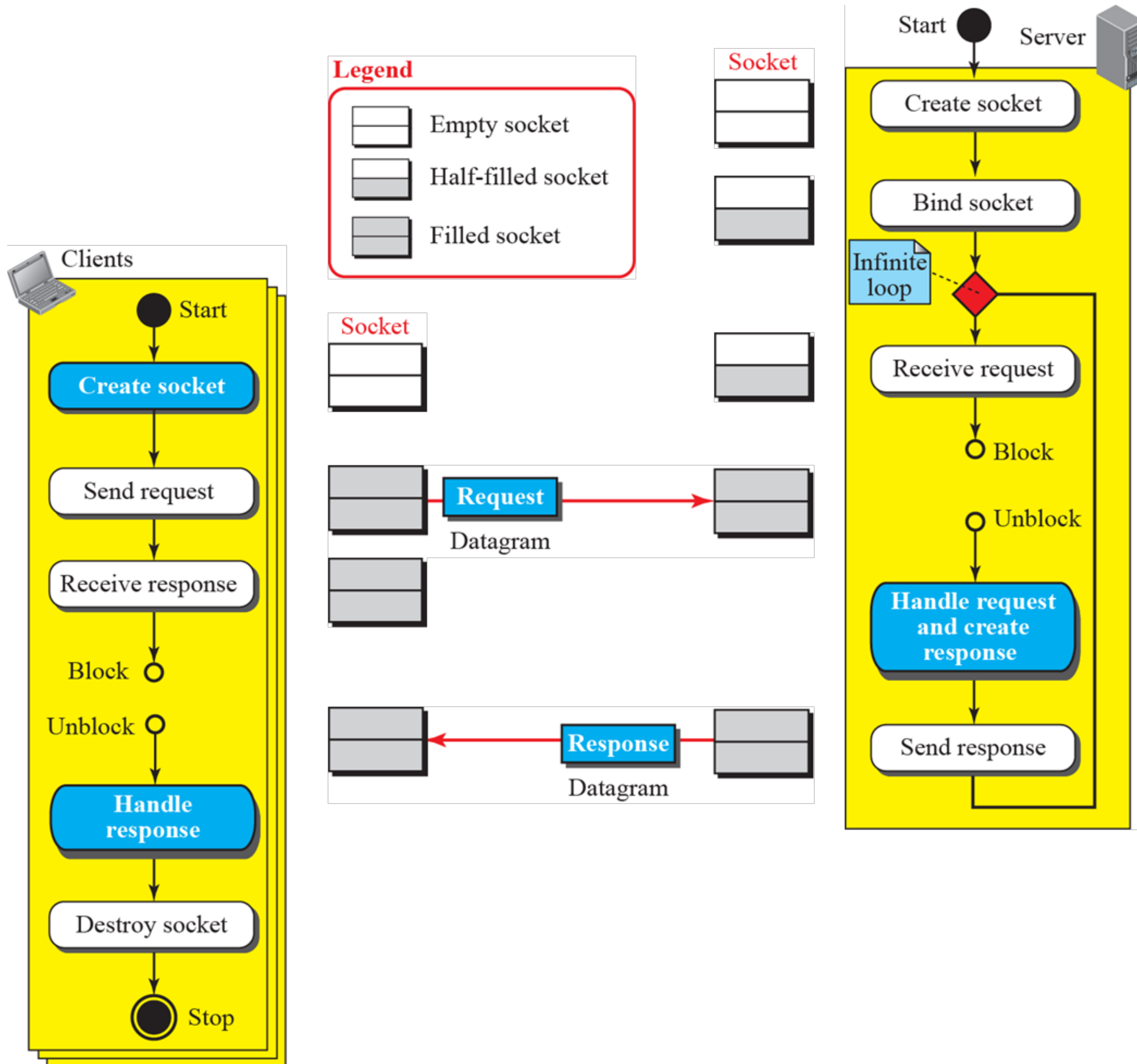
Create datagram with server IP and port=x; send datagram via clientSocket



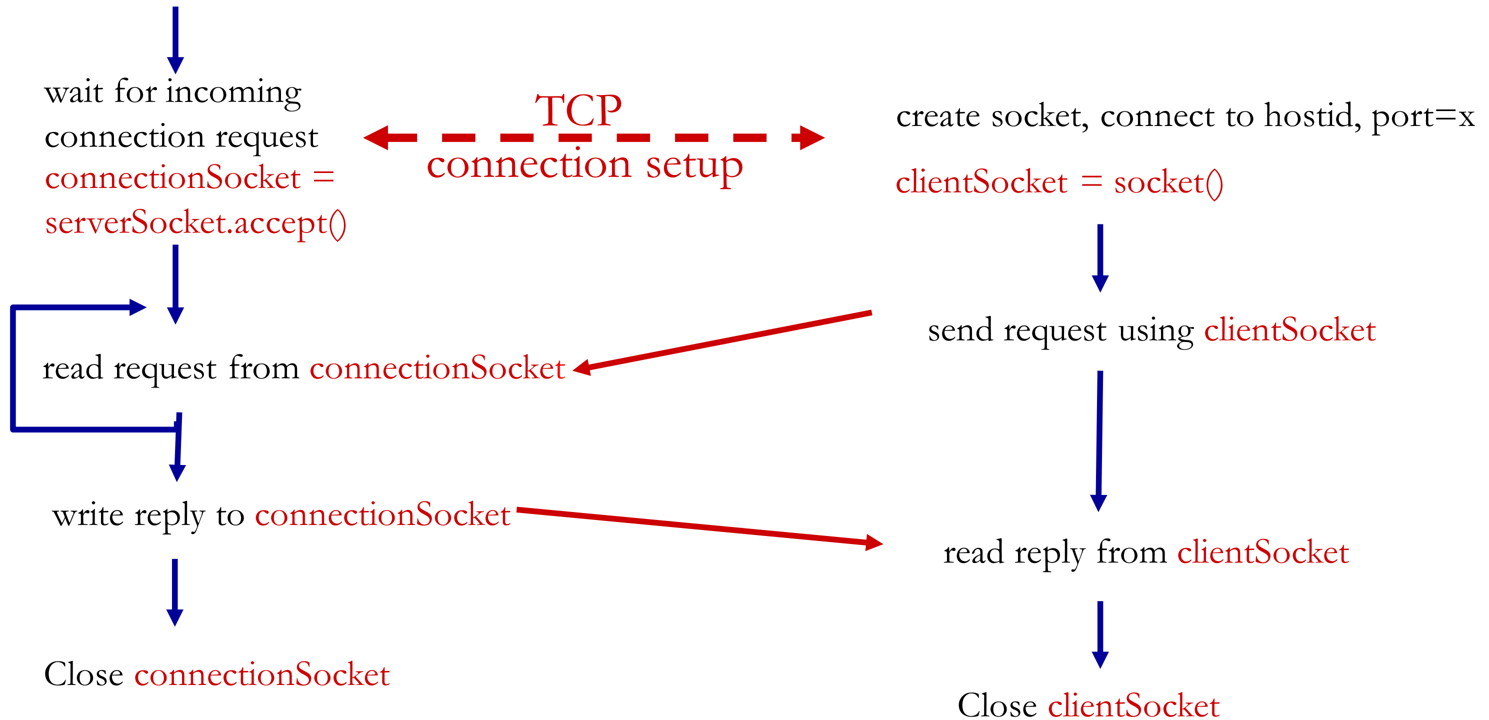
read datagram from clientSocket

Close clientSocket

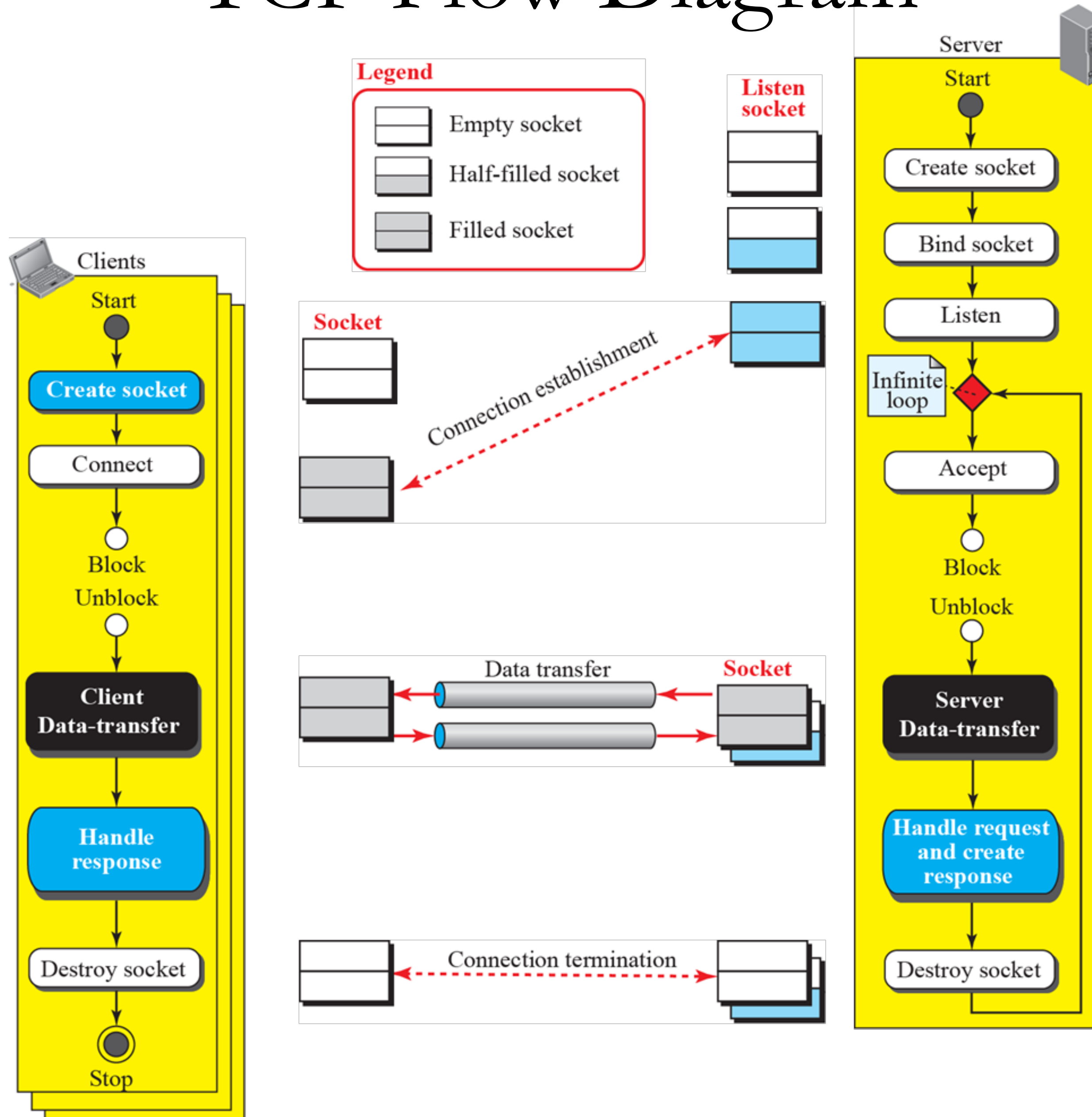
UDP Flow Diagram



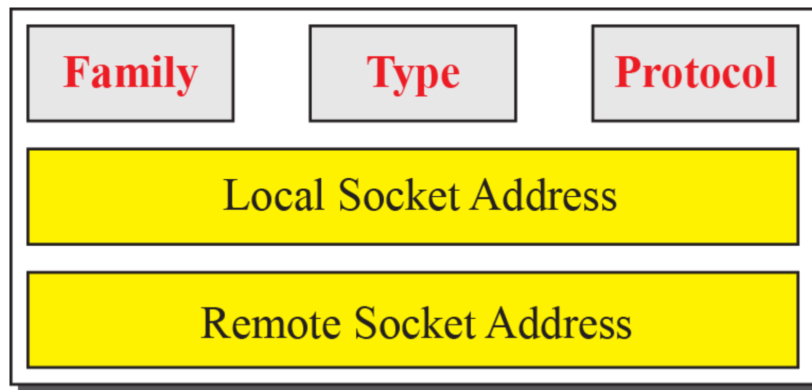
TCP Client/Server Socket Interaction

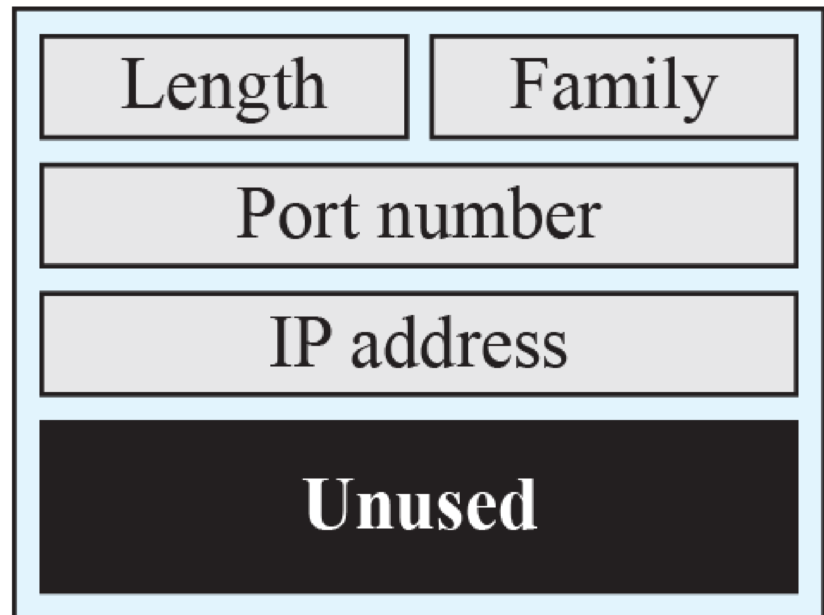


TCP Flow Diagram



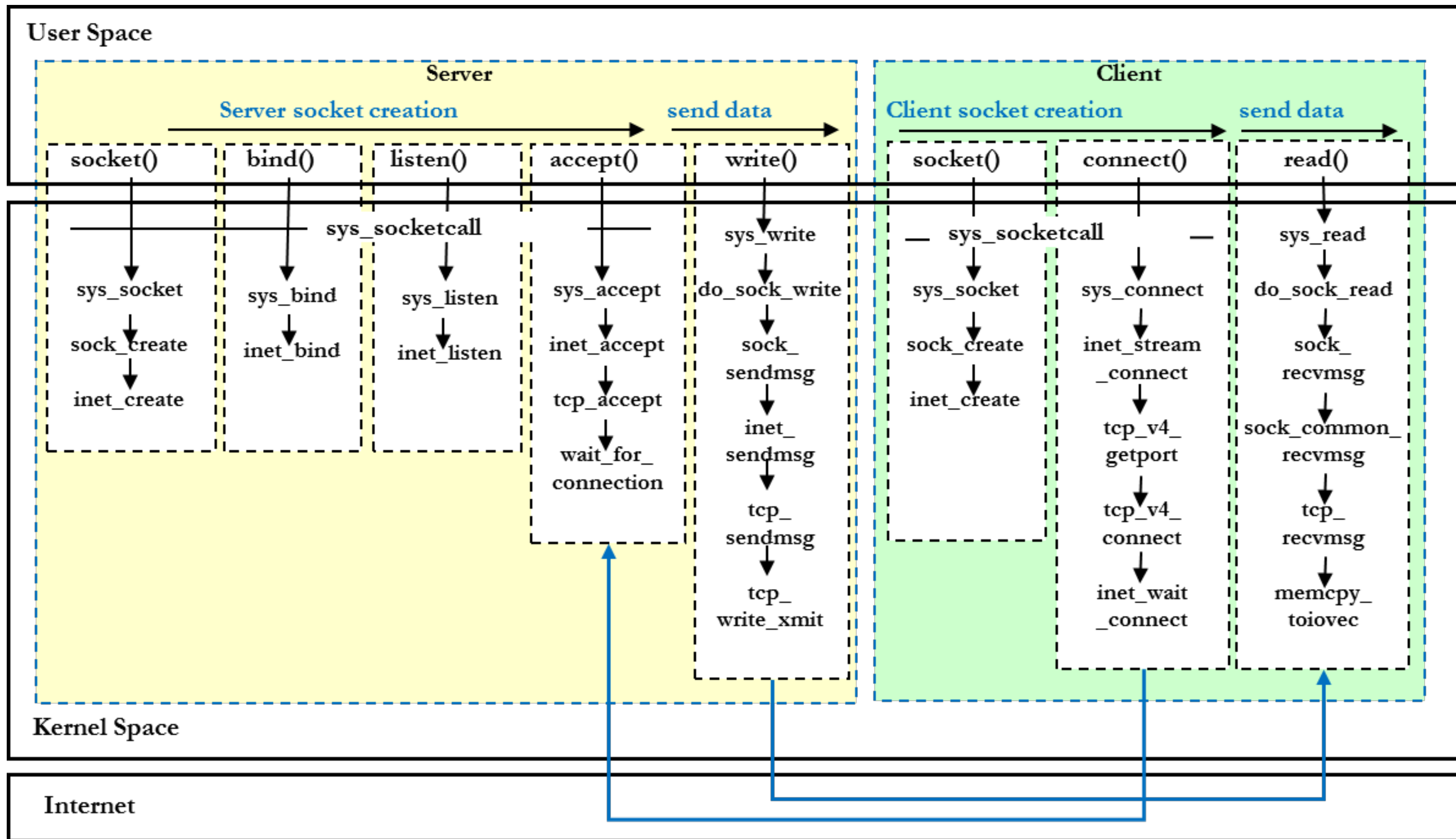
Socket Data Structures





Socket address

Socket Read/Write Inside Out



Socket Read/Write Inside Out