Introduction

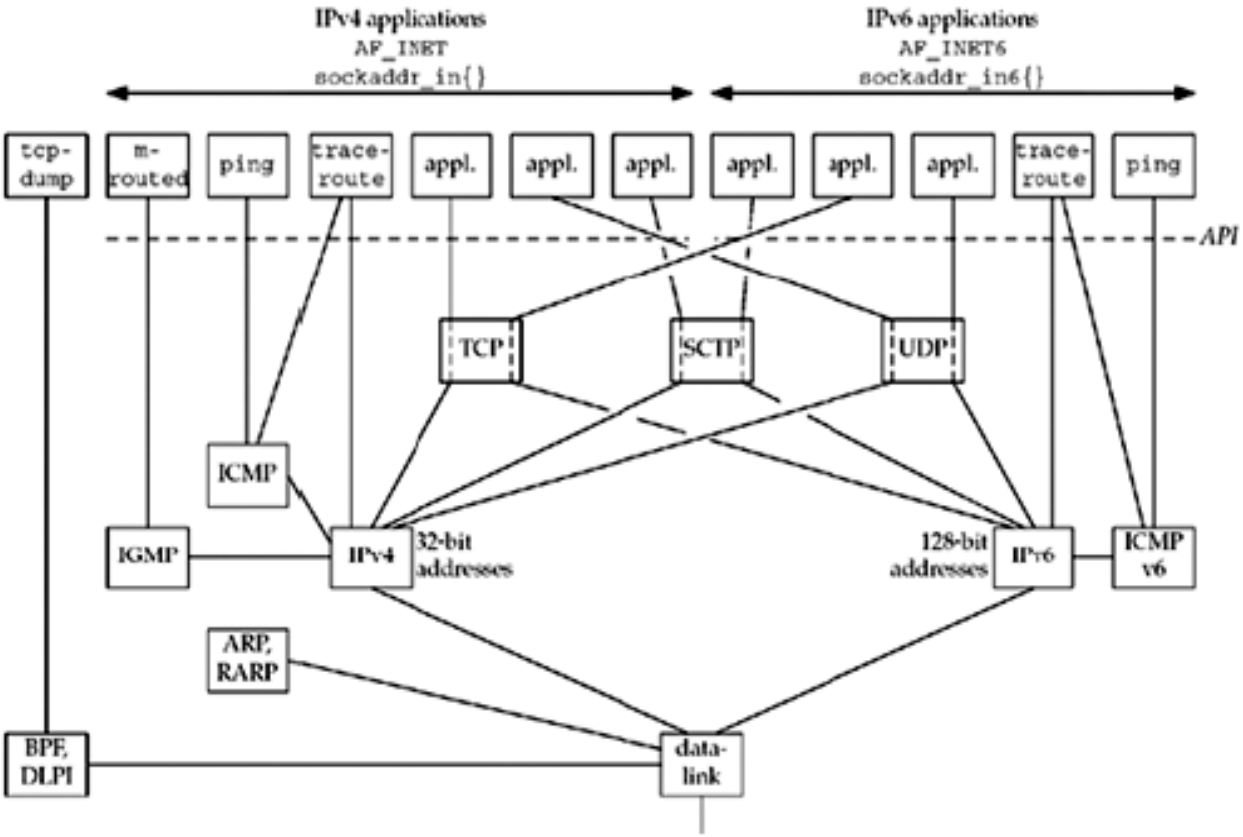
TCP/IP is a protocol that defines the details of how data is sent and received through network adapters, hubs, switches, routers and other network communications hardware. It's the primary protocol used on the Internet, and it has become the most popular networking protocol throughout the world and is therefore well supported by almost all computer systems and networking hardware.

The TCP/IP protocol is designed such that each computer or device in a network has a unique "IP Address" (Internet Protocol Address) and each IP address can open and communicate over up to 65535 different "ports" for sending and receiving data to or from any other network device.

The IP Address uniquely identifies the computer or device on the network and a "Port Number" identifies a specific connection between one computer or device and another. A TCP/IP "port" can be thought of as a private two-way communications line where the port number is used to identify a unique connection between two devices.

TCP/IP can be used on many data-link layers (can support many network hardware implementations). It is considered to be a *de facto* standard when it comes to connecting devices (computing devices) to a network. Some of the reasons that have led to the popularity of TCP/IP include: It is non-proprietary, It is easy to implement and It can be configured on almost any device (computing)

The following is an overview of TCP/IP protocols**.** Although, it is known as TCP/IP suite, there are more members to the family as shown below.



Internet Protocol:

The Internet Protocol (IP) is the principal communications protocol in the Internet protocol suite for relaying datagrams across network boundaries. Its routing function enables internetworking, and essentially establishes the Internet.

IP is a network layer protocol (Internet layer of the Internet protocol suite) and hence it must be capable of providing communication between hosts on different kinds of networks.

IP provides connectionless, unreliable delivery of IP datagrams.

- *Connectionless*: each datagram is independent of all others.
- *Unreliable*: there is no guarantee that datagrams are delivered correctly (in the order they were sent) or at all.

IP Addresses

In the TCP/IP protocol, the unique identifier for a computer is called its IP address. There are two standards for IP addresses: IP Version 4 (IPv4) and IP Version 6 (IPv6).

- **IPv4**: It provides the *packet delivery service* for TCP, UDP, SCTP, ICMP and IGMP. It uses 32 bit address used to create a single unique address on the network. An IPv4 address is expressed by four numbers separated by dots. Each number is the decimal (base-10) representation for an eight-digit binary (base-2) number, also called an octet. E.g. *216.27.62.138*
- **IPv6**: It is designed as replacement for IPv4 and uses 128 binary bits to create a single unique address on the network. An IPv6 address is expressed by eight groups of hexadecimal (base-16) numbers separated by colons, e.g., *2001:cdba:0000:0000:0000:0000:3257:9652*. Groups of numbers that contain all zeros are often omitted to save space, leaving a colon separator to mark the gap (2001:cdba::3257:9652).It has larger address made up of 128 bits. It provides packet delivery service for TCP, UDP, SCTP and ICMPv6.

The IP addresses are not the same as the underlying data-link (MAC) addresses. They are considered to be logical and can be changed by a Network Engineer/Administrator or Systems Administrator of an organization. The address must include two pieces of information: information about the *network* the host is on and the unique address of the *host*. Both the Network ID and the Host ID are used for routing. The network ID is assigned to an organization by a global authority while host IDs are assigned locally by a system administrator.

IP Classes

IPv4 addresses represent four eight-digit binary numbers and each number could be 00000000 to 11111111 in binary, or 0 to 255 in decimal (base-10) i.e. 0.0.0.0 to 255.255.255.255.
However, some numbers in that range are reserved for specific purposes on TCP/IP networks. These reservations are recognized by the authority on TCP/IP addressing, the Internet Assigned Numbers Authority (IANA). Four specific reservations include the following:

i). **0.0.0.0** -- This represents the default network, which is the abstract concept of just being connected to a TCP/IP network.
ii). **255.255.255.255** -- This address is reserved for network broadcasts, or messages that should go to all computers on the network.
iii). **127.0.0.1** -- This is called the loopback address, meaning your computer's way of identifying itself, whether or not it has an assigned IP address.
iv). 169.254.0.1 to 169.254.255.254 -- This is the Automatic Private IP Addressing (APIPA) range of addresses assigned automatically when a computer's unsuccessful getting an address from a DHCP server.

The other IP address reservations are for subnet classes. A subnet is a smaller network of computers connected to a larger network through a router. The subnet can have its own address system so computers on the same subnet can communicate quickly without sending data across the larger network. A router on a TCP/IP network, including the Internet, is configured to recognize one or more subnets and route network traffic appropriately _

Services provided by IP

- Connectionless Delivery (each datagram is treated individually).
- Unreliable (delivery is not guaranteed).
- Fragmentation / Reassembly (based on hardware MTU- maximum transmission unit).
- Routing.
- Error detection.

ICMP (*Internet Control Message Protocol*)**:**

It handles errors and control information between router and hosts. These are generated and processed by TCP /IP networking software itself.

- Error reports carried by ICMP usually directed to source.

- ICMP uses IP to deliver messages.
- ICMP messages are usually generated and processed by the IP software.
- ICMP Message Types: Echo Request and Response, destination Unreachable, Time Exceeded, and Timestamp request and reply

IGMP *(Internet Group Management Protocol)*: It is used with multicasting (transmission of data from a single source to multiple recipients). It is optional with IPv4.

Address Resolution Protocol (ARP)
This is a protocol used by the Internet Protocol (IP), specifically IPv4, to map IP network addresses to the hardware addresses (such as an Ethernet address) used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer.

- The Address Resolution Protocol is used by a sending host when it knows the IP address of the destination but needs the Ethernet address.
- ARP is a broadcast protocol - every host on the network receives the request.
- Each host checks the request against it's IP address - the right one responds.

RARP *(Reverse ARP)*:
This maps a hardware address into an IPv4 address. It is sometimes used when a diskless node such as X terminal is booting.

ICMPv6: It combines the functionality of ICMPv4, IGMP and ARP.

BPF *(BSD Packet Filter)*: This interface provides the access to the datalink for a process. It is found in Berkley derived kernels.

DLPI *(Data Link Provider Interface):* This provides access to the datalink and is normally provided with SVR4 (System V Release 4).

Note: BPF (BSD Packet Filter) and DLPI (Data Link Provider Interface) does not use sockets or XTI.

Transport Layer
These transport protocols use the network-layer protocol IP (IPv4 or IPv6). The following is a description of transport protocols:

- TCP (Transmission Control Protocol): is a *connection oriented protocol* that provides a reliable, full duplex, byte stream for a user process. It takes care of details such as acknowledgment, timeouts, retransmissions etc.
  The chunk of data that TCP asks IP to deliver is called a TCP segment. Each segment contains: data bytes from the byte stream and control information that identifies the data bytes
- UDP (User Datagram Protocol): is a *connectionless protocol* and UDP sockets are example of *datagram sockets*. There is no guarantee that UDP datagram ever reach their intended destination.
- SCTP (*Stream Control Transmission Protocol*): It is *a connection-oriented protocol* that provides a reliable full-duplex association. SCTP provides a message service, which maintains record boundaries. As with TCP and UDP, SCTP can use either IPv4 or IPv6, but it can also use both IPv4 and IPv6 simultaneously on the same association.

You need to understand the services provided by these transport layer to the application since you need to know what is handled by the protocol and what must be handled in the application. Also, when you understand these features, it becomes easier to debug clients and servers using commonly provided tools such as *netstat*.

Note: All the above protocols are defined in the *RFC (Request for Comments),* which are supported by their formal specification.

Comparisons between TCP and UDP

TCP	UDP
Reliable, guaranteed i.e. data is sent on a TCP socket, it waits for an ACK for a duration equal or more than the RTT	Unreliable. Instead, prompt delivery of packets (efficiency over fast networks with short latency).
connection-oriented byte stream	Connectionless datagram delivery service
Used in applications that require safety guarantee. (e.g. file applications.)	Used in media applications (e.g. video or voice transmissions.)
Flow control (Receiver advertises the size of data which it can accept to its peer), sequencing of packets (sequences data by associating a sequence number with every byte that it sends.), error-control.	No flow or sequence control, user must handle these manually.
Uses byte stream as unit of transfer. (stream sockets)	Uses datagrams as unit of transfer. (datagram sockets)
Allows two-way data exchange, once the connection is established. (full-duplex)	Allows data to be transferred in one direction at once. (half-duplex)
Example is Telnet that uses stream sockets. (everything you write on one side appears exactly in same order on the other side) and File Transfer Protocol (FTP)	Examples is TFTP (trivial file transfer protocol) that uses datagram

Addressing in TCP/IP

Addressing at the Data Link Layer is physical (hardware) while at the network layer it is logical (IP Addressing). At the Transport Layer addressing pays more focus on how applications or processes communicate. For this reason, each TCP/IP address includes: Internet Address, Protocol (UDP or TCP) and Port Number

TCP Socket Connections

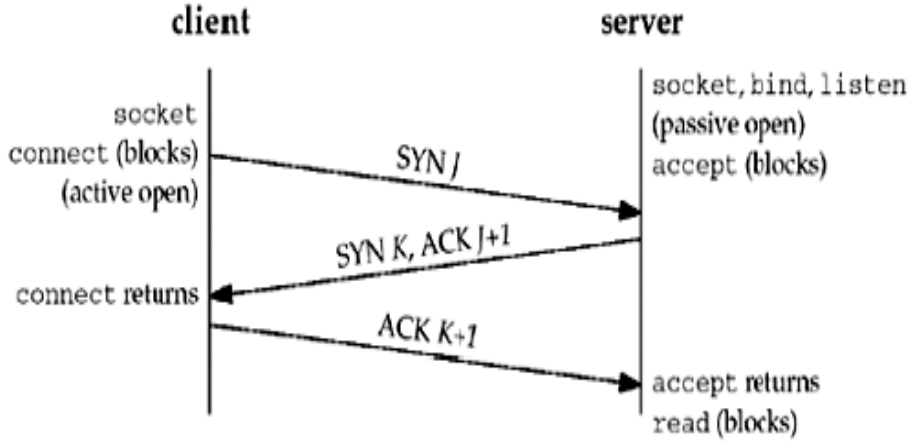
TCP socket connections which are a fundamental part of socket programming since they provide a connection oriented service with both flow and congestion control. What this means to the programmer is that a TCP connection provides a reliable connection over which data can be transferred with little effort required on the programmers part i.e. TCP takes care of the reliability, flow control, congestion control etc.

TCP Connection Establishment

It takes three segments (*three way handshake)* to establish a connection, which takes place as follows:

i) The server must be prepared to accept an incoming connection. This is normally done by calling `socket, bind` and `listen` functions and is called *passive open*.
ii) The client issues an *active open* by calling `connect`. This causes the client TCP to send "synchronize" (SYN) segment to tell the server the client‘s initial sequence number for the data that the client will send on the connection. Normally, there is no data sent with the SYN; it just contains an IP header, a TCP header, and possible TCP options
iii) The server acknowledge (ACK) the client‘s SYN and sends its own SYN and the ACK of the client‘s SYN in a single segment.
iv) The client must acknowledge (ACK) the server‘s SYN.

As the minimum number of packets required is three, it is called three way handshake. This is shown in the following figure.



J is the initial sequence number of client and K is that of Server. The acknowledgment number in ACK is the next expected sequence number for the end sending the ACK. Since a SYN occupies one byte of the sequence number space, the acknowledgment number in the ACK of each SYN is the initial sequence number plus one. When the `accept` is called, the server actually waits (i.e., blocks) until a client becomes available (i.e., an incoming client request arrives).

Note: `accept` returns the client's identity only after the connection has been established

TCP Options:
TCP SYN can contain TCP options. Common options are: MSS (Maximum Segment Size), Window Scale Option and Time Stamp Option.

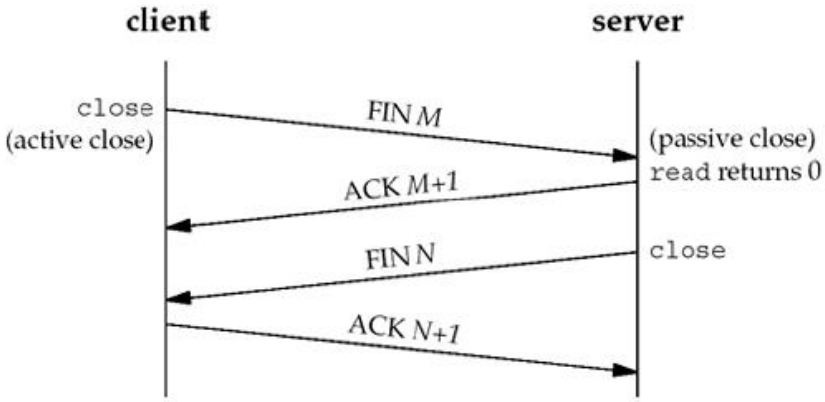
- **MSS Option**: With this option the TCP sending SYN announces its *maximum segment size*, the maximum amount of data that it is willing to accept in each TCP segment, on this connection. This option is set by `TCP_ MAXSEG` socket option.
- **Window scale option**: The maximum window that either TCP can advertise to the other TCP is 65,535 as the corresponding field in the TCP header occupies 16 bits. But high speed connections (45 Mbps/sec) or long delay paths require larger window which can be set by left shifting (scaling) by 0-14 bits giving rise to one gigabyte. This is effected with `SO_RCVBUF` socket option.
- **Timestamp option**: This option is needed for high speed connections to prevent possible data corruption caused by lost packets that reappears.

Last two options being new, may not be supported. These are also known as "long fat pipe" option, since a network with either a high bandwidth or a long delay is called a *long fat pipe*.

Note: TCP always tells its peer exactly how many bytes of data it is willing to accept from the peer at any one time. This is called advertised *window*

TCP connection Termination:
While it takes three segments to establish a connection, it takes *four segment* to terminate a connection a TCP connection as shown below:

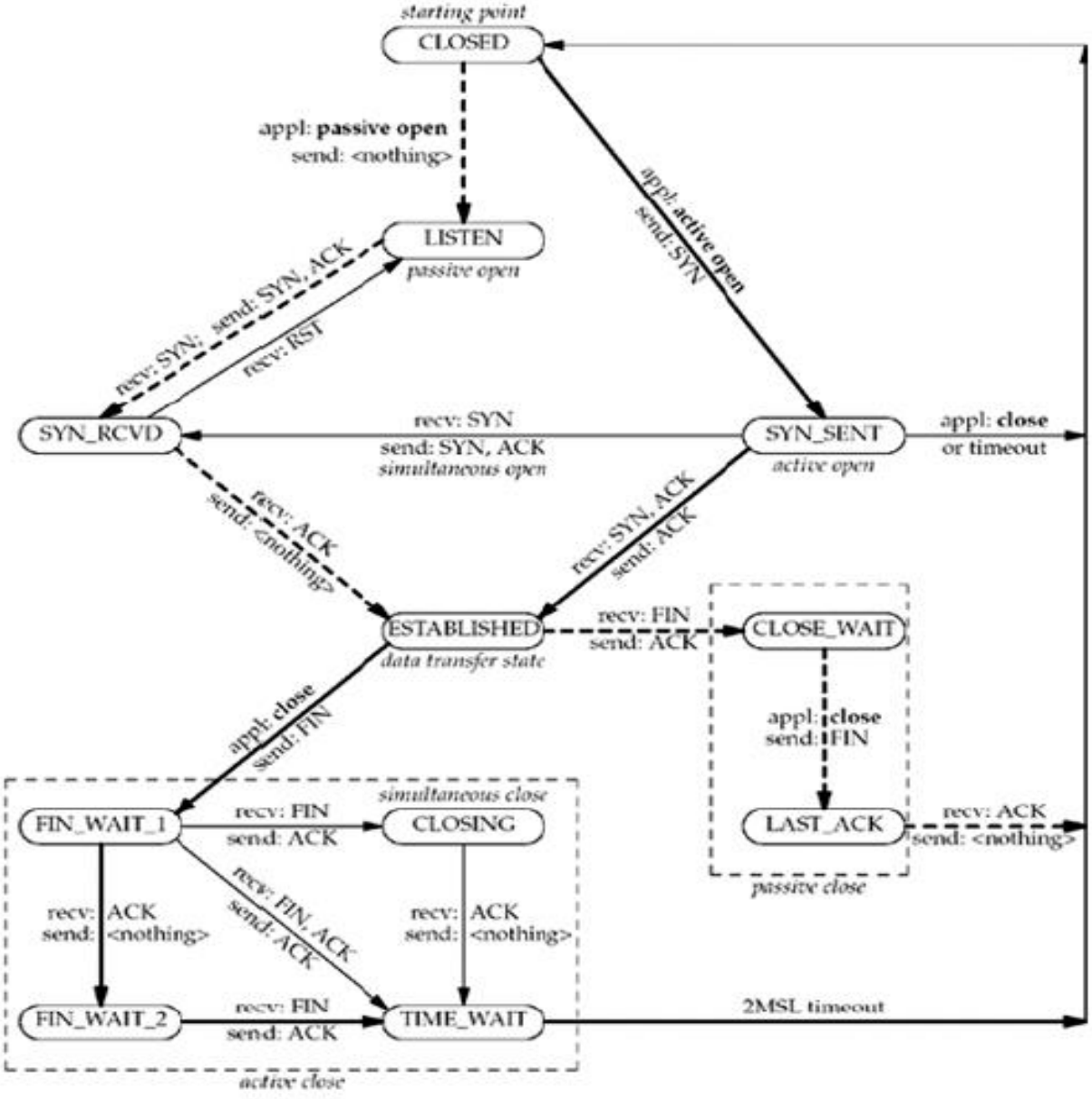
1. One application calls `close`**,** and we say that this end performs the *active close*. This end's TCP sends FIN segment, which means it is finished sending data.
2. The other end that receives the FIN performs the *passive close*. The received FIN is acknowledged by TCP. The FIN is passed to the application as an end of file (after any data that may already be queued for the application to receive) and the receiver will not receive any further data from the sender.
3. When the received application closes its socket, the TCP sends FIN.
4. The TCP on the system that receives the FIN acknowledges the FIN.



Note: The sending of each FIN occurs when a socket is closed. Also, either end the client or the server can perform the active close. Often the client performs the active close

TCP State Transition Diagram

To understand the `connect, accept and close` functions and to debug TCP application using *netstat*, we need to understand the *state transition diagram*, which specifies the operation of TCP with regard to connection establishment and connection termination



indicate normal transactions for client

indicate normal transactions for server

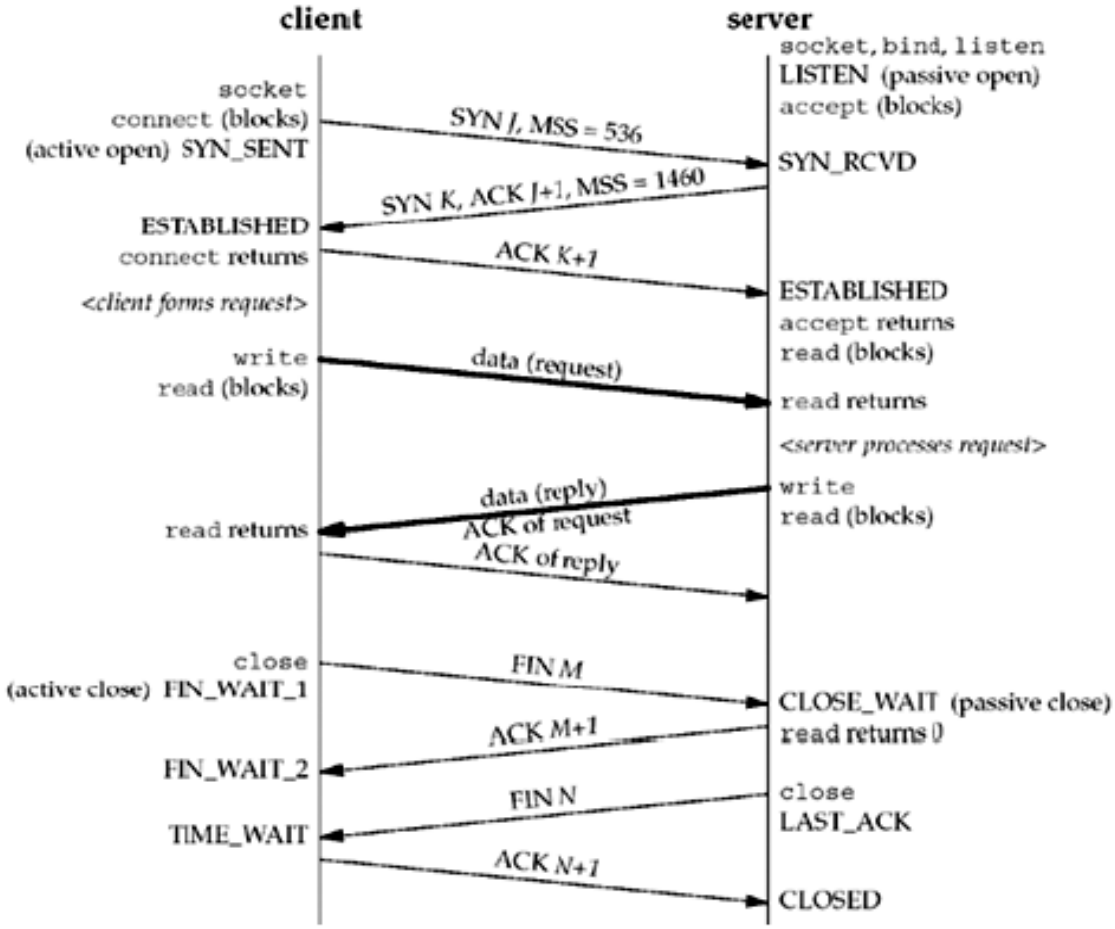
appl: *indicate state transitions taken when application issues operation*

recv: *indicate state transitions taken when segment received*

send: *indicate what is sent for this transition*

There are eleven (11) different states defined for a connection and the rules of TCP dictate the transitions from one state to another, based on the current state and the segment received in that state. E.g. if an application performs an active open in the CLOSED state, TCP sends a SYN and the new state is SYN_SENT. If TCP next receives a SYN with an ACK, it sends an ACK and the new state is ESTABLISHED. This final state is where most data transfer occurs.
The two arrows leading from the ESTABLISHED state deal with the termination of a connection. If an application calls `close` before receiving a FIN (an active close), the transition is to the FIN_WAIT_1 state. But if an application receives a FIN while in the ESTABLISHED state (a passive close), the transition is to the CLOSE_WAIT state.

Packet exchange for TCP connection:
The following diagram shows the actual packet exchange that takes place for a complete TCP connection: the *connection establishment*, *data transfer*, and *connection termination*. The TCP states through which each endpoint passes are included.



The client in the example announces an MSS (Maximum Segment Size) of 536 and the server announces an MSS of 1,460. It is okay for the MSS to be different in each direction.
Once the connection is established, the client forms a request and sends it to the server (fits in one segment and less than 1,460 bytes). The server processes the request and sends a reply. The data segments transfer

is shown in bold arrows. Notice that the acknowledgment of the client's request is sent with the server's reply and this is called *piggybacking*
The four other remaining segment shown terminate the connection. As it can be seen that the end that performs the active close enters the TIME_WAIT state.

Port Numbers
At any given time, multiple processes can be using any given transport: UDP or TCP and both use 16-bit integer *port numbers* to differentiate between these processes i.e. ports are software objects used to multiplex data between different applications.
Some of these are port 21 for the FTP server and Trivial File Transfer Protocol (TFTP) is assigned the UDP port of 69 etc. Clients use *ephemeral ports* that is short lived ports. These port numbers are manually assigned by TCP or UDP to the client. IANA *(Internet Assigned Numbers Authority*) maintains a list of port number assignments. The port numbers are divided into following three categories:

i) **Well known ports** (0 – 1023): These port numbers are controlled and assigned by the IANA. When possible, same port number is assigned for both TCP and UDP as in the case of web server, which is assigned port 80.
ii) **The Registered Ports** (1024 – 49151): These are not controlled by IANA but it registers and list the uses of these ports as a convenience to the community
iii) **Dynamic or private ports (**49152 – 65535): These are what we call as ephemeral ports.

Socket Pair
The *socket pair* for a TCP connection is the *four-tuple* that defines the two endpoints of the connection: the local IP address, local port, foreign IP address, and foreign port. A socket pair uniquely identifies every TCP connection on a network. The two values that identify each endpoint, an IP address and a port number, are called a socket.
Notice that when we describe the socket functions (`bind, connect`, etc.), it enables you to specify values in the socket pair. E.g. bind lets the application specify the local IP address and local port for TCP and UDP sockets.