ELE548 Essay2

TCP/IP

Yirong Meng
Yirong@ele.uri.edu
March 20th,1999

ISO OSI Reference Model

7 layers in this protocol:

1.application, 2.presentation, 3.session, 4. transport, 5.network link, 6.data link, 7.physical

The International Organization for Standardization (known as ISO, has devised a model for the design of communication protocols (known as the open systems interconnection (OSI) model). In this model each communicating entity has seven layers of protocol. The bottom layer (1) is known as the physical layer, and it essentially corresponds to the wire. The next layer (2) is known as the data link layer. It provides the means for putting data on the wire (and for taking it off). An example is ethernet. The next layer (3) is known as the network layer. Its primary responsibility is to see that the data travels to the intended destination (perhaps via a number of intermediate points). The next layer is known as the transport layer. Its job is to see to it that the data, which is transferred between machines by the network layer, reaches the desired party at the destination machine. The notion of a "connection" is maintained by this layer. The next layer is known as the session layer. This layer doesn't really do very much (if anything); it is responsible for maintaining the notion of a "session." Sessions might be in one-to-one correspondence with transport connections; there might be two successive sessions on the same connection; or one session might span multiple connections (e.g., the first connection was terminated due to a communication failure, the session continues as soon as communication is reestablished). The next layer is known as the presentation layer. Its job is to deal with the fact that different machines have different representations for data (i.e. it must somehow translate between data representations) and to deal with such concerns as compression or encryption of data. Finally, the application layer is where all other software resides. However, it has been discovered that there is "system software" that logically fits above the presentation layer. The only place to put it is in the application layer, so "real" application software sits on top of the application layer. The bottom three layers are sometimes known as the communications subnet.

Internet Architecture used today:

4 parts:

1. Application(OSI Layer 5-7) 2.End-to-End(OSI Layer 4) 3.Internet(OSI Layer 3) 4.Net Interface(OSI Layers 1-2)

The OSI model is a classic example of something designed by committee. The model it describes does not fit exactly with the model of TCP/IP, but its terminology has become so commonplace that its exact requirements are conveniently stretched so that TCP/IP fits in. Originally it was thought (by the designers of the model) that protocols designed to fit in the OSI model would be not only competitors of TCP/IP, but would be replacing TCP/IP. The rumors of TCP/IP's death were greatly exaggerated; one rarely hears about its competitors these days. Today one hears very little of the OSI protocols (though the OSI seven-layer terminology is much used); whatever competition there was between the OSI protocols and the internet protocols was definitely won by the latter. The protocols used on the Internet are, strangely enough, known as the internet protocols (also known as TCP/IP). They don't fit precisely into the OSI model (there is no analog of the session and presentation layers), but the rough correspondence is shown in the picture. (fig2)

Introduction:

TCP/IP is a family of protocols that makes internetworking possible.

Applications use TCP (transmission control protocol) to send data reliably to other applications it is known as an end-to-end protocol in that it used only at the endpoints of the communication path.

To handle the transmission of data between endpoints and gateways and between gateways, IP (internet protocol) is used. It is responsible, among other things, for finding a route to the ultimate destination.

The job of actually communicating data to the next machine is handled by the network interface. It, for example, might implement the ethernet protocol.

IP(network-layer protocol):

The purpose of the IP protocol is to take a packet from machine to machine, starting at the source machine, through a number of gateways, and finally delivering the packet to the destination machine.

The protocol is fairly simple it makes no promises of reliability: if a packet is lost for any reason, the assumption is that a higher-level protocol will deal with the problem, if necessary. (It is also implicitly assumed that packets aren't lost all that often.) The IP protocol, however, must deal with the issue of routing: each outgoing packet is marked with the address of the ultimate destination. Since the final destination might not be directly connected to the sending machine, each machine must look up the destination in an internal routing table to determine the location of the first gateway in the best known path to the final destination. (It is not IP's job to maintain this routing table it is maintained via other, higher-level protocols.) Thus each non-final machine forwards the message to a machine that is (or should be) closer to the final destination.

IP's job is to transfer data from one machine to another. It is of no concern to it which users or processes are sending or receiving this data this is a concern of the next-level protocol (UDP or TCP).

Operation of IP:

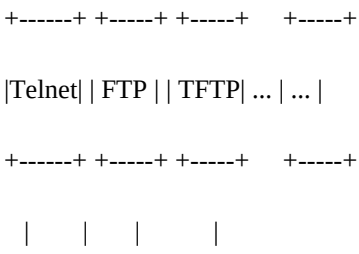
The internet protocol implements two basic functions: addressing and fragmentation. The internet modules use the addresses carried in the internet header to transmit internet datagrams toward their destinations. The selection of a path for transmission is called routing. The internet modules use fields in the internet header to fragment andreassemble internet datagrams when necessary for transmission through "small packet" networks. The model of operation is that an internet module resides in each host engaged in internet communication and in each gateway that interconnects networks. These modules share common rules for interpreting address fields and for fragmenting and assembling internet datagrams. In addition, these modules (especially in gateways) have procedures for making routing decisions and other functions. The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise). The internet protocol uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

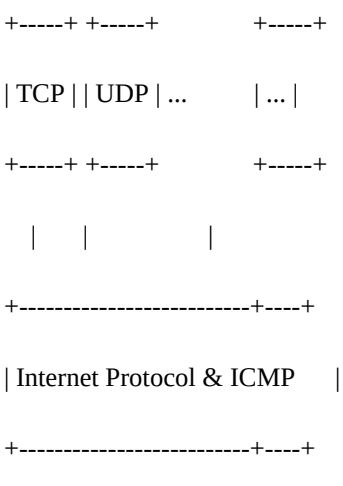
The Type of Service is used to indicate the quality of the service desired. The type of service is an abstract or generalized set of parameters which characterize the service choices provided in the networks that make up the internet. This type of service indication is to be used by gateways to select the actual transmission parameters for a particular network, the network to be used for the next hop, or the next gateway when routing an internet datagram. The Time to Live is an indication of an upper bound on the lifetime of an internet datagram. It is set by the sender of the datagram and reduced at the points along the route where it is processed. If the time to live reaches zero before the internet datagram reaches its destination, the internet datagram is destroyed. The time to live can be thought of as a self destruct time limit. The Options provide for control functions needed or useful in some situations but unnecessary for the most common communications. The options include provisions for timestamps, security, and special routing.

The Header Checksum provides a verification that the information used in processing internet datagram has been transmitted correctly. The data may contain errors. If the header checksum fails, the internet datagram is discarded at once by the entity which detects the error. The internet protocol does not provide a reliable communication facility. There are no acknowledgments either end-to-end or hop-by-hop. There is no error control for data, only a header checksum. There are no retransmissions. There is no flow control. Errors detected may be reported via the Internet Control Message Protocol (ICMP) [3] which is implemented in the internet protocol module.

The following diagram illustrates the place of the internet protocol

in the protocol hierarchy:





Protocol Relationships

Figure 1.

IP Function Description

The function or purpose of Internet Protocol is to move datagrams through an interconnected set of networks. This is done by passing the datagrams from one internet module to another until the destination is reached. The internet modules reside in hosts and gateways in the internet system. The datagrams are routed from one internet module to another through individual networks based on the interpretation of an internet address. Thus, one important mechanism of the internet protocol is the internet address. In the routing of messages from one internet module to another, datagrams may need to traverse a network whose maximum packet size is smaller than the size of the datagram. To overcome this difficulty, a fragmentation mechanism is provided in the internet protocol.

TCP Introduction:

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. Very few assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols. In principle, the TCP should be able to

operate above wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks. TCP fits into a layered protocol architecture just above a basic Internet Protocol [2] which provides a way for the TCP to send and receive variable-length segments of information enclosed in internet datagram "envelopes". The internet datagram provides a means for addressing source and destination TCPs in different networks. The
 nternet protocol also deals with any fragmentation or reassembly of the TCP segments required to achieve transport and delivery through multiple networks and interconnecting gateways. The internet protocol also carries information on the precedence, security classification and compartmentation of the TCP segments, so this information can be communicated end-to-end across multiple networks.

Protocol Layering

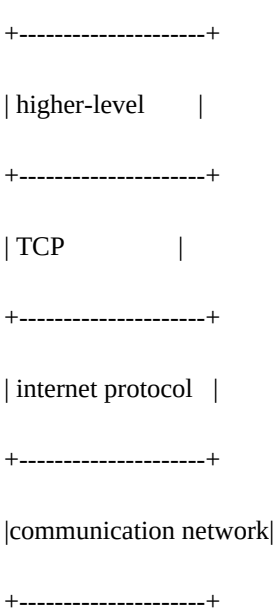


Figure 1

Much of this document is written in the context of TCP implementations which are co-resident with higher level protocols in the host computer. Some computer systems will be connected to networks via] front-end computers which house the TCP and internet protocol layers, as well as network specific software. The TCP specification describes an interface to the higher level protocols which appears to be implementable even for the front-end case, as long as a suitable host-to-front end protocol is implemented.

Interfaces

The TCP interfaces on one side to user or application processes and on the other side to a lower level protocol such as Internet Protocol. This interface consists of a set of calls much like the calls an operating system provides to an application process for manipulating files. For example, there are
 alls to open and close connections and to send and receive data on established connections. It is also expected that the TCP can asynchronously communicate with application programs.

 The interface between TCP and lower level protocol is essentially unspecified except that it is assumed

there is a mechanism whereby the two levels can asynchronously pass information to each other. Typically, one expects the lower level protocol to specify this interface. TCP is designed to work in a very general environment of interconnected networks. The lower level protocol which is assumed throughout this document is the Internet Protocol [2].

TCP Operation

As noted above, the primary purpose of the TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable internet communication system requires facilities in the following areas:

Basic Data Transfer

Reliability

Flow Control

Multiplexing

Connections

Precedence and Security

Basic Data Transfer:

The TCP is able to transfer a continuous stream of octets in each direction between its users by packaging some number of octets into segments for transmission through the internet system. In general, the TCPs decide when to block and forward data at their own convenience. Sometimes users need to be sure that all the data they have submitted to the TCP has been transmitted. For this purpose a push function is defined. To assure that data submitted to a TCP is actually transmitted the sending user indicates that it should be pushed through to the receiving user. A push causes the TCPs to promptly forward and deliver data up to that point to the receiver. The exact push point might not be visible to the receiving user and the push function does not supply a record boundary marker.

Reliability:

The TCP must recover from data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments. As long as the TCPs continue to function properly and the internet system does not

become completely partitioned, no transmission errors will affect the correct delivery of data. TCP recovers from internet communication system errors.

Flow Control:

TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a "window" with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received. The window indicates an allowed number of octets that the sender may transmit before receiving further permission.

Multiplexing:

To allow for many processes within a single Host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenated with the network and host addresses from the internet communication layer, this forms a socket. A pair of sockets uniquely identifies each connection. That is, a socket may be simultaneously used in multiple connections. The binding of ports to processes is handled independently by each Host. However, it proves useful to attach frequently used processes (e.g., a "logger" or timesharing service) to fixed sockets which are made known to the public. These services can then be accessed through the known addresses. Establishing and learning the port addresses of other processes may involve more dynamic mechanisms.

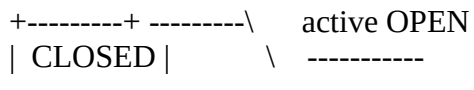
Connections:

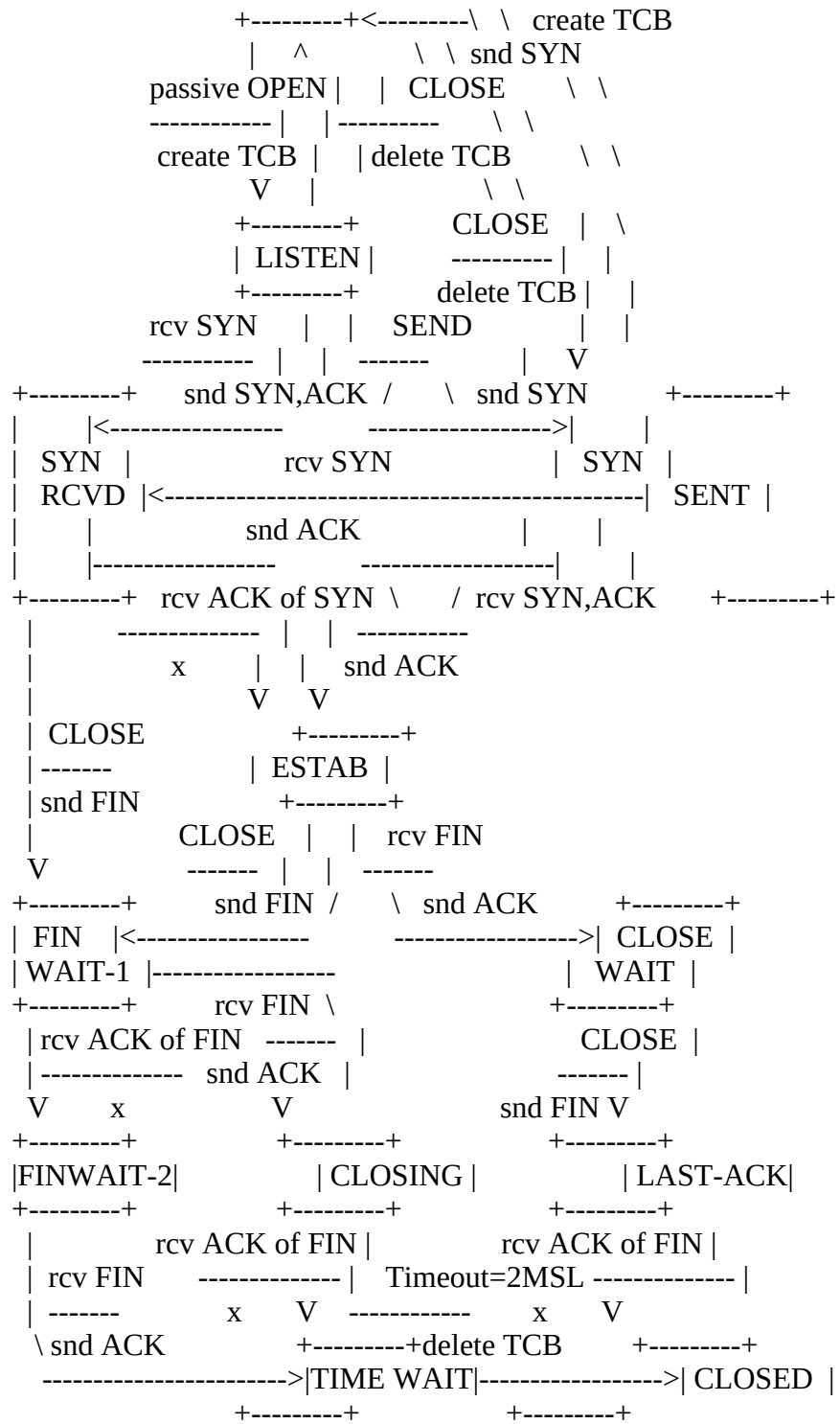
The reliability and flow control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is uniquely specified by a pair of sockets identifying its two sides. When two processes wish to communicate, their TCP's must first establish a connection (initialize the status information on each side). When their communication is complete, the connection is terminated or closed to free the resources for other uses. Since connections must be established between unreliable hosts and over the unreliable internet communication system, a handshake mechanism with clock-based sequence numbers is used to avoid erroneous initialization of connections.

Precedence and Security:

The users of TCP may indicate the security and precedence of their communication. Provision is made for default values to be used when these features are not neede

TCP Connection State Diagram





TCP Connection State Diagram
Figure 6.

A connection progresses through a series of states during its lifetime. The states are: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and the fictional state CLOSED. CLOSED is fictional because it represents the state when there is no TCB, and therefore, no connection. Briefly the meanings of the states are:

LISTEN - represents waiting for a connection request from any remote TCP and port.

SYN-SENT - represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

FIN-WAIT-1 - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2 - represents waiting for a connection termination request from the remote TCP.

CLOSE-WAIT - represents waiting for a connection termination request from the local user.

CLOSING - represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

TIME-WAIT - represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.

CLOSED - represents no connection state at all.

A TCP connection progresses from one state to another in response to events. The events are the user calls, OPEN, SEND, RECEIVE, CLOSE, ABORT, and STATUS; the incoming segments, particularly those containing the SYN, ACK, RST and FIN flags; and timeouts.

The state diagram in figure 6 illustrates only state changes, together with the causing events and resulting actions, but addresses neither error conditions nor actions which are not connected with state changes. In a later section, more detail is offered with respect to the reaction of the TCP to events.

NOTE BENE: this diagram is only a summary and must not be taken as the total specification.