TRANSPORT LAYER PROTOCOLS

Duty of transport layer is process-to-process communication.

These protocols use port numbers to accomplish this. Port numbers provide end-to end addresses at the transport layer and allow multiplexing and demultiplexing at this layer.

Stop-and-Wait Protocol

This is a connection-oriented protocol called the Stop-and-Wait protocol, which uses both flow and error control. Both the sender and the receiver use a sliding window of size 1. The sender sends one packet at a time and waits for an acknowledgment before sending the next one. To detect corrupted packets, we need to add a checksum to each data packet. When a packet arrives at the receiver site, it is checked. If its checksum is incorrect, the packet is corrupted and silently discarded.

Every time the sender sends a packet, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next packet (if it has one to send). If the timer expires, the sender resends the previous packet, assuming that the packet was either lost or corrupted. This means that the sender needs to keep a copy of the packet until its acknowledgment arrives Figure 4.1.1 shows the outline for the Stop-and-Wait protocol.

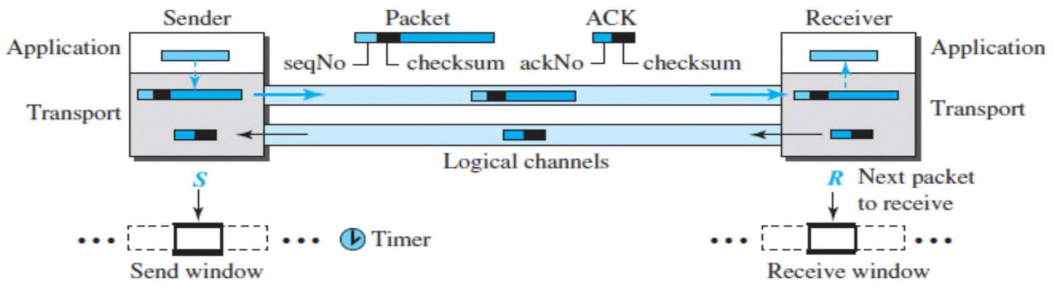

Fig4.1.1:Stop and Wait Protocol

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-709]

Sequence Numbers

To prevent duplicate packets, the protocol uses sequence numbers and acknowledgment numbers.

A field is added to the packet header to hold the sequence number of that packet.

Assume that the sender has sent the packet with sequence number x. Three things can happen.

1. The packet arrives safe and sound at the receiver site; the receiver sends an acknowledgment. The acknowledgment arrives at the sender site, causing the sender to send the next packet numbered x + 1.

2. The packet is corrupted or never arrives at the receiver site; the sender resends the packet (numbered x) after the time-out. The receiver returns an acknowledgment.

3. The packet arrives safe and sound at the receiver site; the receiver sends an acknowledgment, but the acknowledgment is corrupted or lost. The sender resends the packet (numbered x) after the time-out. Note that the packet here is a duplicate. The receiver can recognize this fact because it expects packet x + 1 but packet x was received.

We can see that there is a need for sequence numbers x and $x + 1$ because the receiver needs to distinguish between case 1 and case 3. But there is no need for a packet to be numbered $x + 2$. In case 1, the packet can be numbered x again because packets x and $x + 1$ are acknowledged and there is no ambiguity at either site. In cases 2 and 3, the new packet is $x + 1$, not $x + 2$. If only x and $x + 1$ are needed, we can let $x = 0$ and $x + 1 = 1$. This means that the sequence is 0, 1, 0, 1, 0, and so on. This is referred to as modulo 2 arithmetic.

Acknowledgment Numbers

The acknowledgment numbers always announce the sequence number of the next packet expected by the receiver. For example, if packet 0 has arrived safe and sound, the receiver sends an ACK with acknowledgment 1 (meaning packet 1 is expected next). If packet 1 has arrived safe and sound, the receiver sends an ACK with acknowledgment 0 (meaning packet 0 is expected).

Go-Back-N Protocol (GBN)

To improve the efficiency of transmission (to fill the pipe), multiple packets must be in transition while the sender is waiting for acknowledgment. In other words, we need tolet more than one packet be outstanding to keep the channel busy while the sender is waiting for acknowledgment. In this section, we discuss one protocol that can achieve this goal; in the next section, we discuss a second. The first is called Go-Back-N (GBN) (the rationale for the name will become clear later).

The key to Go-back-N is that we can send several packets before receiving acknowledgments, but the receiver can only buffer one packet. We keep a copy of the sent packets until the acknowledgments arrive. Figure 4.1.2 shows the outline of the protocol. Note that several data packets and acknowledgments can be in the channel at the same time.

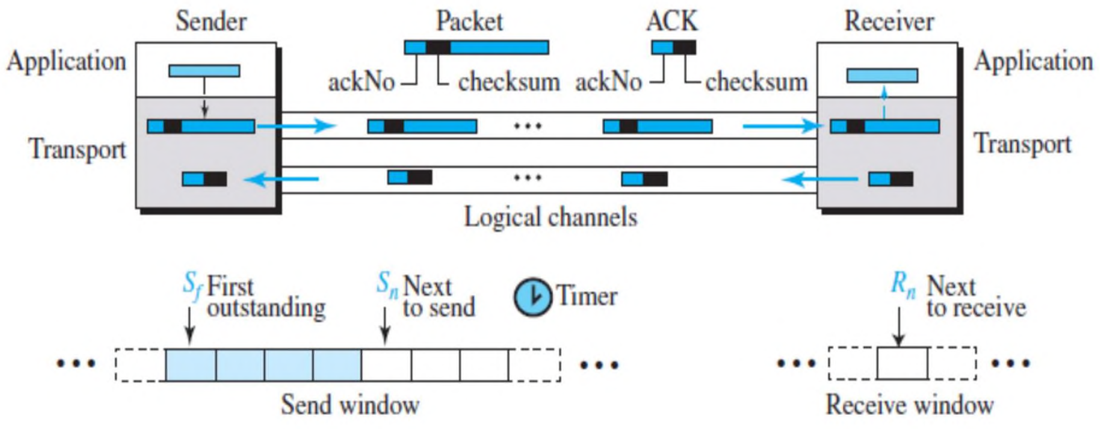

Fig4.1.2:Go-Back- N Protocol

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-713]

Sequence Numbers

The sequence numbers are modulo 2m, where m is the size of the sequence number field in bits.

Acknowledgment Numbers

An acknowledgment number in this protocol is cumulative and defines the sequence number of the next packet expected. For example, if the acknowledgment number (ackNo) is 7, it means all packets with sequence number up to 6 have arrived, safe and sound, and the receiver is expecting the packet with sequence number 7.

USER DATAGRAM PROTOCOL

The User Datagram Protocol (UDP) is a connectionless, unreliable transport protocol as shown in figure 4.1.3. UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP.

User Datagram

UDP packets are called user datagrams. The first two fields define the source and destination port numbers. The third field defines the total length of the user datagram, header plus data. The 16 bits can define a total length of 0 to 65,535 bytes.

Connectionless Services

UDP provides a connectionless service.This means that each user datagram sent by UDP is an independent datagram. There is no relationship between the different user datagrams even if they are coming from the same source process and going to the same destination program.

The user datagrams travel on a different path on the way to destination.

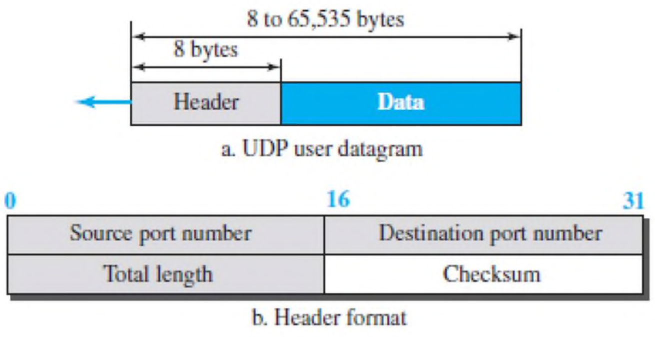

Fig4.1.3: UDP format.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-738]

Flow Control

UDP is a very simple protocol. There is no flow control.The receiver may overflow with incoming messages. The lack of flow control means that the process using UDP should provide for this service, if needed.

Error Control

There is no error control mechanism in UDP except for the checksum. This means that the sender does not know if a message has been lost or duplicated.When the receiver detects an error through the checksum, the user datagram is silently discarded.

Checksum

UDP checksum calculation has three sections: a pseudo header as shown in figure 4.1.4, the UDP header, and the data coming from the application layer.The pseudo header is the part of the header of the IP packet in which the user datagram is to be encapsulated with some fields filled with 0s.If the checksum does not include the pseudo header, a user datagram may arrive safe and sound. If the IP header is corrupted, it may be delivered to the wrong host.

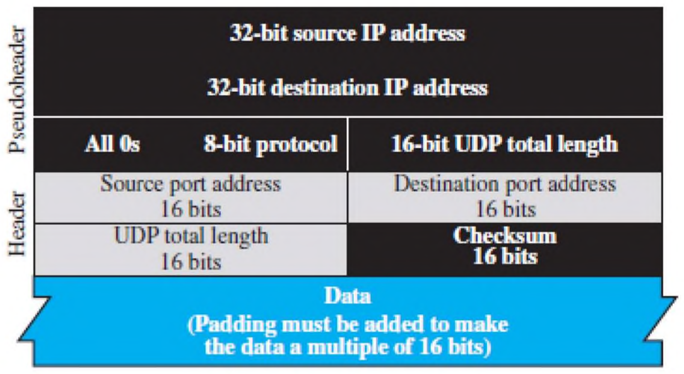
Pseudoheader
Header
32-bit source IP address
32-bit destination IP address
All 0s
8-bit protocol
16-bit UDP total length
Source port address 16 bits
Destination port address 16 bits
UDP total length 16 bits
Checksum 16 bits
Data (Padding must be added to make the data a multiple of 16 bits)

Fig4.1.4:Pseudo header for checksum.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-739]

Congestion Control

Since UDP is a connectionless protocol, it does not provide congestion control.

Encapsulation and Decapsulation

To send a message from one process to another, the UDP protocol encapsulates and decapsulates messages.

Multiplexing and Demultiplexing

In a TCP/IP protocol suite, there is only one UDP but several processes that may want to use the services of UDP. To handle this situation, UDP multiplexes and demultiplexes.

UDP features

Connectionless Service

UDP is a connectionless protocol. Each UDP packet is independent from other packets sent by the same application program. It is an advantage.

For example, a client application needs to send a short request to a server and to receive a short response. If the request and response can each fit in a single user datagram, a connectionless service may be preferable.In the connection oriented service, to send and receive short message , at least 9 packets are exchanged between the client and the server; in connectionless service only 2 packets are exchanged.

UDP Applications

UDP is suitable for a process that requires simple request-response communication with little concern for flow and error control.UDP is suitable for a process with internal flow- and error-control mechanisms. For example, the Trivial File Transfer Protocol (TFTP) process includes flow and error control. It can easily use UDP.UDP is a suitable transport protocol for multicasting.

Multicasting capability is embedded in the UDP software but not in the TCP software. UDP is used for management processes such as SNMP. UDP is used for some route updating protocols such as Routing Information Protocol(RIP).

4.2 TRANSMISSION CONTROL PROTOCOL

Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service.

TCP uses a combination of GBN and SR protocols to provide reliability. To achieve this ,TCP uses checksum(for error detection), retransmission of lost or corrupted packets, cumulative and selective acknowledgments, and timers.

TCP Services

Process-to-Process Communication

TCP provides process-to-process communication using port numbers.

Stream Delivery Service

TCP is a stream-oriented protocol.

TCP allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their bytes across the Internet. The sending process produces (writes to) the stream and the receiving process consumes (reads from) it.

Segments

At the transport layer, TCP groups a number of bytes together into a packet called a segment.

TCP adds a header to each segment (for control purposes) and delivers the segment to the network layer for transmission. The segments are encapsulated in an IP datagram and transmitted.

Full-Duplex Communication

TCP offers full-duplex service, where data can flow in both directions at the same time.

Multiplexing and Demultiplexing

Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver.

Connection-Oriented Service

When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:

The two TCP's establish a logical connection between them.

1.Data are exchanged in both directions.

2.The connection is terminated.

Reliable Service

TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe arrival of data.

TCP Features

The sequence number and the acknowledgment number is used in TCP.

Byte Number

TCP numbers all data bytes (octets) that are transmitted in a connection. Numbering is independent in each direction. When TCP receives bytes of data from a process, TCP stores them in the sending buffer and numbers them.TCP chooses an arbitrary number between 0 and $2^{32} - 1$ for the number of the first byte. For example, if the number happens to be 1057 and the total data to be sent is 6000 bytes, the bytes are numbered from 1057 to 7056.

Sequence Number

After the bytes have been numbered, TCP allot a sequence number to each segment that is being sent.The sequence number of the first segment is the ISN (initial sequence number), which is a random number.The sequence number of any other segment is the sequence number of the previous segment plus the number of bytes (real or imaginary) carried by the previous segment.

Acknowledgment Number

Communication in TCP is full duplex; when a connection is established, both parties can send and receive data at the same time. Each party numbers the bytes, usually with a different starting byte number.The sequence number in each direction shows the number of the first byte carried by the segment.Each party uses an acknowledgment number to confirm the bytes it has received.

TCP Segment

A packet in TCP is called a segment. The format of a segment is shown in Figure 4.2.1.

The segment consists of a header of 20 to 60 bytes, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options.

Source port address. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.

Destination port address. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.

Sequence number. This 32-bit field defines the number assigned to the first byte of data contained in this segment.

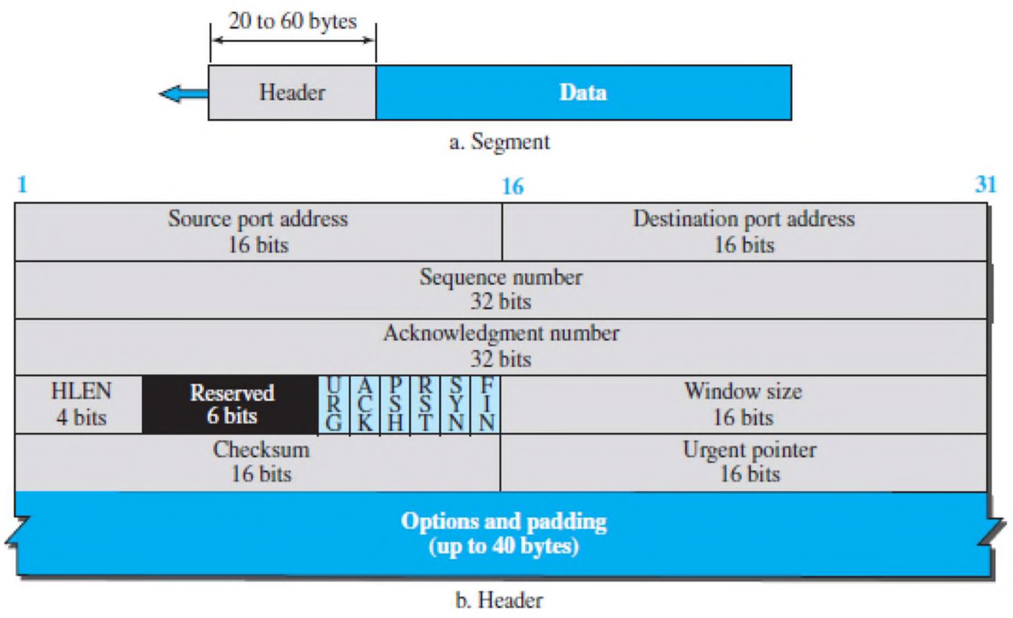

Fig4.2.1: TCP segment.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-748]

TCP is a stream transport protocol.

To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment. During connection establishment each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.

Acknowledgment number. This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte number x from the other party, it returns x +1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.

Header length. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes.

Control. This field defines 6 different control bits or flags.These bits enable flow control, connection establishment and termination, and the mode of data transfer in TCP.

Window size. This field defines the window size of the sending TCP in bytes. The length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes.

This value is referred to as the receiving window(rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

Checksum. This 16-bit field contains the checksum (error detection).The calculation of the checksum for TCP is important. The same pseudo header, serving the same purpose, is added to the segment.

Urgent pointer. This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

Options. There can be up to 40 bytes of optional information in the TCP header.

Encapsulation

A TCP segment encapsulates the data received from the application layer.