UNIT4 TRANSPORT LAYER

4.1 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.

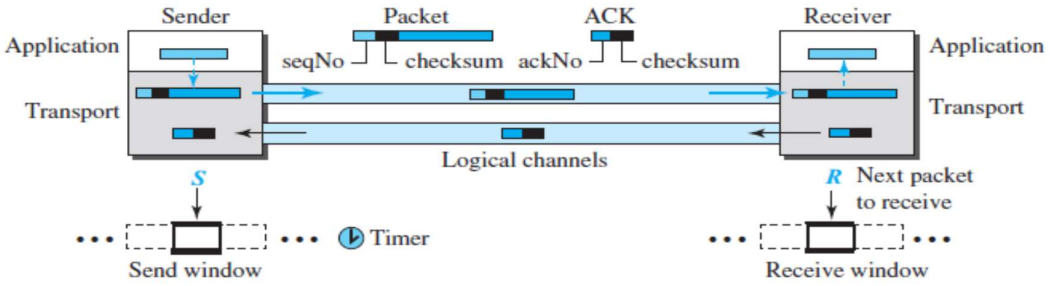

Fig4.1.1:Stop and Wait Protocol

[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.

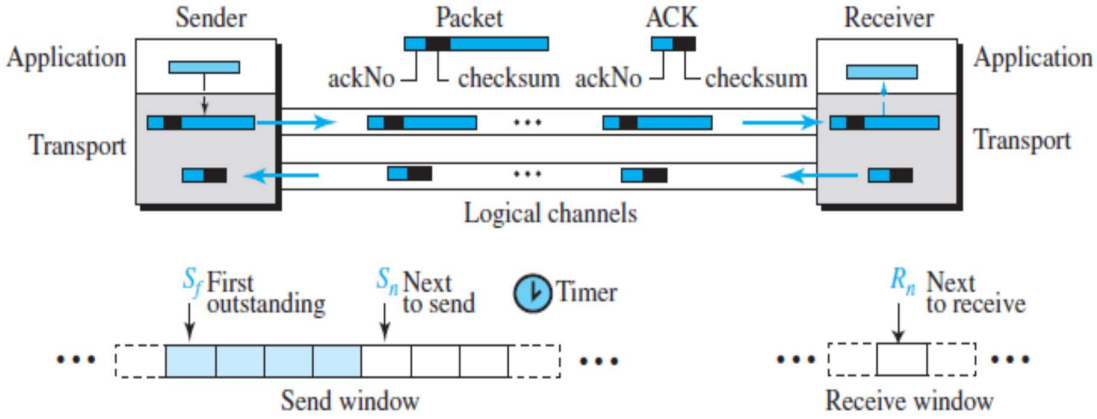

Fig4.1.2:Go-Back- N Protocol

[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.

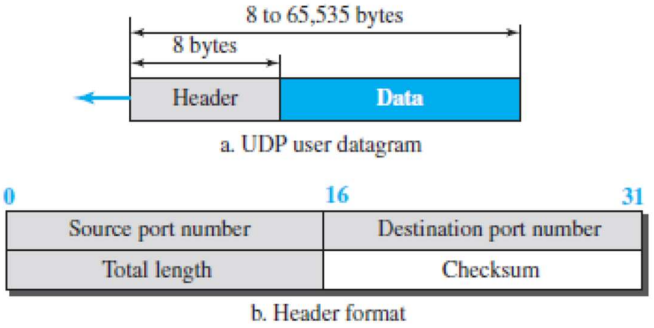

Fig4.1.3: UDP format.

[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.

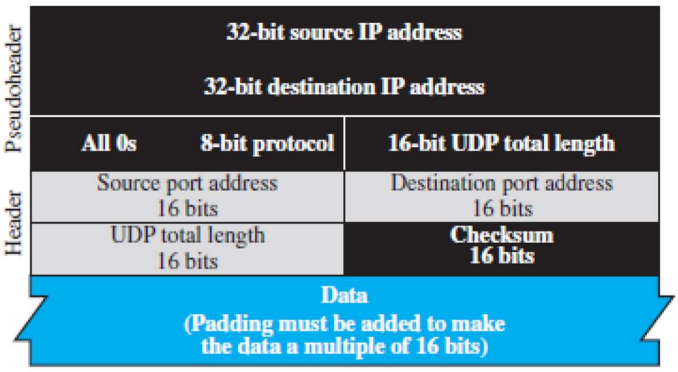

Fig4.1.4:Pseudo header for checksum.

[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).
