TCP/IP-2

Transmission control protocol:

TCP and IP are the workhorses in the Internet. In this section we first discuss how TCP provides reliable, connection-oriented stream service over IP. To do so, TCP implements a version of Selective Repeat ARQ. In addition, TCP implements congestion control through an algorithm that identifies congestion through packet loss and that controls the rate at which information enters the network through a congestion window.

TCP Reliable Stream Service

The Transmission Control Protocol (TCP) provides a logical full-duplex (two- way) connection between two application layer processes across a datagram network. TCP provides these application processes with a connection-oriented,reliable, in-sequence, byte-stream service. TCP also provides flow control that allows receivers to control the rate at which the sender transmits information so that buffers do not overflow. TCP can also support multiple application processes in the same end system.

TCP does not preserve message boundaries and treats the data it gets from the application layer as a byte stream. Thus when a source sends a 1000-byte message in a single chunk (one write), the destination may receive the message in two chunks of 500 bytes each (two reads), in three chunks of 400 bytes, 300 bytes and 300 bytes (three reads), or in any other combination. In other words, TCP may split or combine the application information in the way it finds most appropriate for the underlying network.

TCP Operation

We now consider the adaptation functions that TCP uses to provide a connection-oriented, reliable, stream service. We are interested in delivering the user information so that it is error free, without duplication, and in the same order that it was produced by the sender. We assume that the user information consists of a stream of bytes as shown in Figure 8.18. For example, in the transfer of a long file the sender is viewed as inserting a byte stream into the transmitter's send buffer. The task of TCP is to ensure the transfer of the byte stream to the receiver and the orderly delivery of the stream to the destination application. TCP was designed to deliver a connection-oriented service in an internet environment, which itself offers connectionless packet transfer service, so different packets can traverse a different path from the same source to the same destination and can therefore arrive out of order. Therefore, in the internet old messages from previous connections may arrive at a receiver, thus potentially complicating the task of eliminating duplicate messages. TCP deals with this problem by using long (32 bit) sequence numbers and by establishing randomly selected initial sequence numbers during connection setup. At any given time the receiver is accepting sequence numbers from a much smaller window, so the likelihood of accepting a very old message is very low. In addition, TCP enforces a time-out period at the end of each connection to allow the network to clear old segments from the network.

TCP uses a sliding-window mechanism, as shown in Figure 4.19. The send window contains three pointers: S_{last} points to the oldest byte that has not yet been acknowledged.

S_{recent} points to the last byte that has been transmitted but not yet acknowledged.

$S_{last} + W_S$ 1 indicates the highest numbered byte that the transmitter is willing to accept from its application.

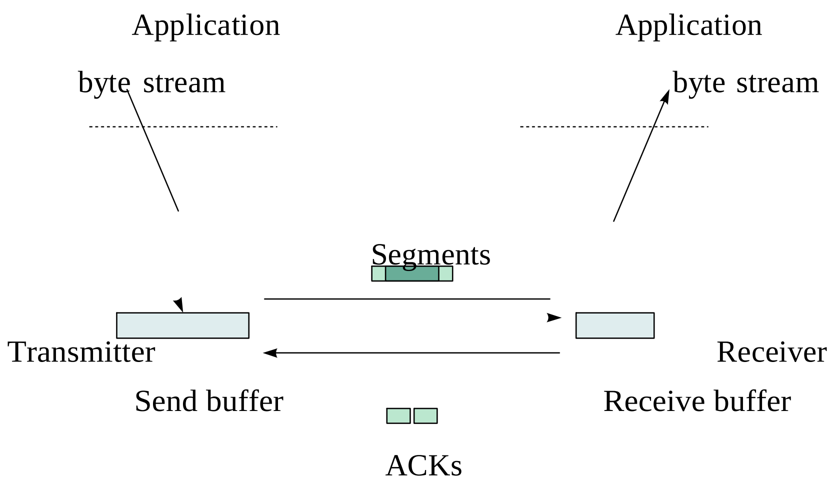

FIGURE 4.18 TCP preview

Enables the network to direct the segment to its destination application process. The segment also contains a sequence number which corresponds to the number of the first byte in the string that is being transmitted. Note that this differs significantly from conventional ARQ. The transmitter decides to transmit a segment when the number of bytes in the send buffer exceeds some specified threshold or when a timer that is set periodically expires. The sending application can also use a push command that forces the transmitter to send a segment.

When a segment arrives, the receiver performs an error check to detect transmission errors. If the segment is error free and is not a duplicate segment, then the bytes are inserted into the appropriate locations in the receive buffer if the bytes fall within the receive window. Note that the receiver will accept out-of-order but error-free segments. If the received segment contains the byte corresponding to R_{next}, then the R_{next} pointer is moved forward to the location of the next byte that has not yet been received. An acknowledgment with the sequence number R_{next} is sent in a segment that is transmitted in the reverse direction. R_{next} acknowledges the correct receipt of all bytes up to R_{next} 1. This acknowledgment, when received by the transmitter, enables the transmitter to update its parameter S_{last} to R_{next}, thus moving the send window forward. Later in the section we give an example of data transfer in TCP.

TCP separates the flow control function from the acknowledgment function.The flow control function is implemented through an advertised window field in the segment header. Segments that travel in the reverse direction contain the advertised window size that informs the transmitter of the number of buffers currently available at the receiver. The advertised window size is given by

$$W_A \;\hat{}\; W_R \quad \ldots R_{new} \quad R_{last}\dagger$$

The transmitter is obliged to keep the number of outstanding bytes below the advertised window size, that is

$$S_{recent} \quad S_{last} \quad W_A$$

To see how the flow control takes effect, suppose that the application at the receiver's side stops reading the bytes from the buffer. Then R_{new} will increase while R_{last} remains fixed, thus leading to a smaller advertised window size.
Eventually the receive window will be exhausted when R_{new} R_{last} ˆ W_R, and the advertised window size will be zero. This condition will cause the transmitter to stop sending. The transmitter can continue accepting bytes from its application until its buffer

contains W_S bytes. At this point the transmitter blocks its application from inserting any more bytes into the buffer.

Finally, we consider the retransmission procedure that is used in TCP. The transmitter sets a timer each time a segment is transmitted. If the timer expires before any of the bytes in the segment are acknowledged, then the segment is retransmitted.