2.4 Transmission Control Protocol (TCP)

The service provided by TCP to an application is different from the service provided by UDP. TCP is described in RFC 793 [Postel 1981c], and updated by RFC 1323 [Jacobson, Braden, and Borman 1992], RFC 2581 [Allman, Paxson, and Stevens 1999], RFC 2988 [Paxson and Allman 2000], and RFC 3390 [Allman, Floyd, and Partridge 2002]. First, TCP provides *connections* between clients and servers. A TCP client establishes a connection with a given server, exchanges data with that server across the connection, and then terminates the connection.

TCP also provides *reliability*. When TCP sends data to the other end, it requires an acknowledgment in return. If an acknowledgment is not received, TCP automatically retransmits the data and waits a longer amount of time. After some number of retransmissions, TCP will give up, with the total amount of time spent trying to send data typically between 4 and 10 minutes (depending on the implementation).

> Note that TCP does not guarantee that the data will be received by the other endpoint, as this is impossible. It delivers data to the other endpoint if possible, and notifies the user (by giving up on retransmissions and breaking the connection) if it is not possible. Therefore, TCP cannot be described as a 100% reliable protocol; it provides reliable delivery of data *or* reliable notification of failure.

TCP contains algorithms to estimate the *round-trip time* (RTT) between a client and server dynamically so that it knows how long to wait for an acknowledgment. For example, the RTT on a LAN can be milliseconds while across a WAN, it can be seconds. Furthermore, TCP continuously estimates the RTT of a given connection, because the RTT is affected by variations in the network traffic.

TCP also *sequences* the data by associating a sequence number with every byte that it sends. For example, assume an application writes 2,048 bytes to a TCP socket, causing TCP to send two segments, the first containing the data with sequence numbers 1–1,024 and the second containing the data with sequence numbers 1,025–2,048. (A *segment* is the unit of data that TCP passes to IP.) If the segments arrive out of order, the receiving TCP will reorder the two segments based on their sequence numbers before passing the data to the receiving application. If TCP receives duplicate data from its peer (say the peer thought a segment was lost and retransmitted it, when it wasn't really lost, the network was just overloaded), it can detect that the data has been duplicated (from the sequence numbers), and discard the duplicate data.

> There is no reliability provided by UDP. UDP itself does not provide anything like acknowledgments, sequence numbers, RTT estimation, timeouts, or retransmissions. If a UDP datagram is duplicated in the network, two copies can be delivered to the receiving host. Also, if a UDP client sends two datagrams to the same destination, they can be reordered by the network and arrive out of order. UDP applications must handle all these cases, as we will show in Section 22.5.

TCP provides *flow control*. 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 the advertised *window*. At any time, the window is the amount of room currently available in the receive buffer, guaranteeing that the sender cannot overflow the receive buffer. The window changes dynamically over time: As data is received from the sender, the window size decreases, but as the receiving application reads data from the buffer, the window size increases. It is possible for the window to reach 0: when TCP's receive buffer for a socket is full and it must wait for the application to read data from the buffer before it can take any more data from the peer.

> UDP provides no flow control. It is easy for a fast UDP sender to transmit datagrams at a rate that the UDP receiver cannot keep up with, as we will show in Section 8.13.

Finally, a TCP connection is *full-duplex*. This means that an application can send and receive data in both directions on a given connection at any time. This means that TCP must keep track of state information such as sequence numbers and window sizes for each direction of data flow: sending and receiving. After a full-duplex connection is established, it can be turned into a simplex connection if desired (see Section 6.6).

> UDP can be full-duplex.