

In TCP, the underlying IP network is assumed to be unreliable and to deliver messages out of order; TCP uses the sliding window algorithm on an end-to-end basis to provide reliable/ordered delivery. In contrast, X.25 networks use the sliding window protocol within the network, on a hop-by-hop basis. The assumption behind this approach is that if messages are delivered reliably and in order between each pair of nodes along the path between the source host and the destination host, then the end-to-end service also guarantees reliable/ordered delivery.

The problem with this latter approach is that a sequence of hop-by-hop guarantees does not necessarily add up to an end-to-end guarantee. First, if a heterogeneous link (say, an Ethernet) is added to one end of the path, then there is no guarantee that this hop will preserve the same service as the other hops. Second, just because the sliding window protocol guarantees that messages are delivered correctly from node A to node B, and then from node B to node C, it does not guarantee that node B behaves perfectly. For example, network nodes have been known to introduce errors into messages while transferring them from an input buffer to an output buffer. They have also been known to accidentally reorder messages. As a consequence of these small windows of vulnerability, it is still necessary to provide true end-to-end checks to guarantee reliable/ordered service, even though the lower levels of the system also implement that functionality.

► This discussion serves to illustrate one of the most important principles in system design—the *end-to-end argument*. In a nutshell, the end-to-end argument says that a function (in our example, providing reliable/ordered delivery) should not be provided in the lower levels of the system unless it can be completely and correctly implemented at that level. Therefore, this rule argues in favor of the TCP/IP approach. This rule is not absolute, however. It does allow for functions to be incompletely provided at a low level as a performance optimization. This is why it is perfectly consistent with the end-to-end argument to perform error detection (e.g., CRC) on a hop-by-hop basis; detecting and retransmitting a single corrupt packet across one hop is preferable to having to retransmit an entire file end-to-end.

5.2.2 Segment Format

TCP is a byte-oriented protocol, which means that the sender writes bytes into a TCP connection and the receiver reads bytes out of the TCP connection. Although “byte stream” describes the service TCP offers to application processes, TCP does not itself transmit individual bytes over the Internet. Instead, TCP on the source host buffers enough bytes from the sending process to fill a reasonably sized packet and then sends this packet to its peer on the destination host. TCP on the destination host then empties the contents of the packet into a receive buffer, and the receiving process reads from this buffer at its leisure. This situation is illustrated in Figure 5.3, which, for simplicity, shows