prevented from being accepted and interpreted by the new connection. For example, the duplicate could be the command to terminate a connection. To deal with these and other problems, we show in the next section that every implementation of TCP assumes a certain value for the **maximum segment lifetime (MSL)**, which is the maximum time that an IP packet can live in the network.

8.5.2 TCP Protocol

We now discuss the structure of TCP segments and the setting up of a TCP connection, the data transfer phase, and the closing of the connection. Detailed information about TCP can be found in RFC 793 and RFC 1122.

TCP SEGMENT

Figure 8.27 shows the format of the TCP segment. The header consists of a 20-byte fixed part plus a variable-size options field.

The description of each field in the TCP segment is given below. The term *sender* refers to the host that sends the segment, and *receiver* refers to the host that receives the segment.

Source port and **destination port:** The source and destination ports identify the sending and receiving applications, respectively. Recall from Section 2.3.2 that the pair of ports and IP addresses identify a process-to-process connection.

Sequence number: The 32-bit sequence number field identifies the position of the first data byte of this segment in the sender's byte stream during data transfer (when SYN bit is not set). The sequence number wraps back to 0 after $2^{32} - 1$. Note that TCP identifies the sequence number for each byte (rather than for each

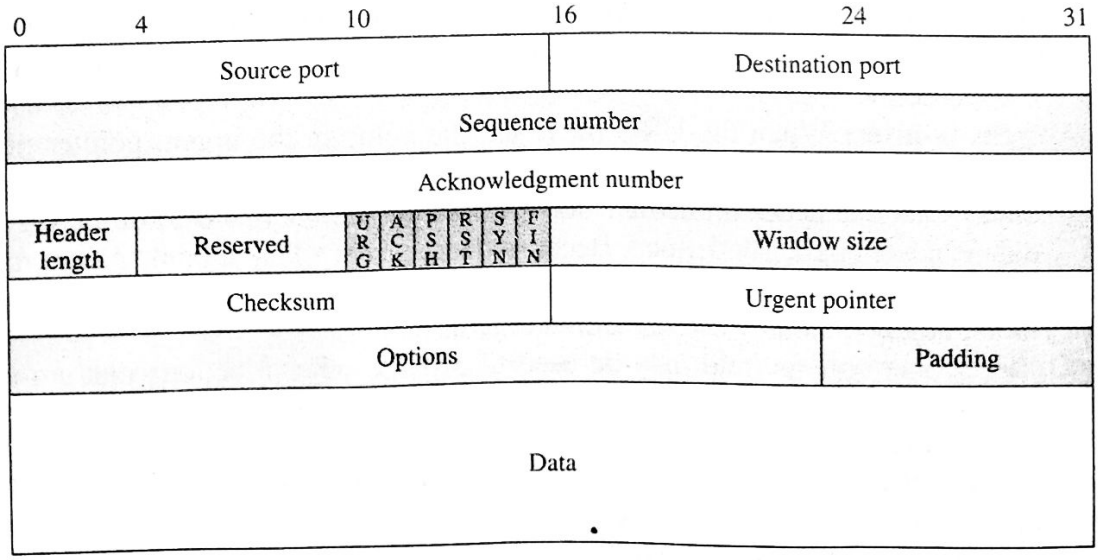

FIGURE 8.27 TCP segment.

segment). For example, if the value of the sequence number is 100 and the data area contains five bytes, then the next time this TCP module sends a segment, the sequence number will be 105. If the SYN bit is set to 1 (during connection establishment), the sequence number indicates the **initial sequence number (ISN)** to be used in the sender's byte stream. The sequence number of the first byte of data for this byte stream will be ISN + 1. It is important to note that a TCP connection is full duplex so that each end point independently maintains its own sequence number.

Acknowledgment number: This field identifies the sequence number of the next data byte that the sender expects to receive if the ACK bit is set. This field also indicates that the sender has successfully received all data up to but not including this value. If the ACK bit is not set (during connection establishment), this field is meaningless. Once a connection is established, the ACK bit must be set.

Header length: This field specifies the length of the TCP header in 32-bit words. This information allows the receiver to know the beginning of the data area because the options field is variable length.

Reserved: As the name implies, this field is reserved for future use and must be set to 0.

URG: If this bit is set, the urgent pointer is valid (discussed shortly).

ACK: If this bit is set, the acknowledgment number is valid.

PSH: When this bit is set, it tells the receiving TCP module to pass the data to the application immediately. Otherwise, the receiving TCP module may choose to buffer the segment until enough data accumulates in its buffer.

RST: When this bit is set, it tells the receiving TCP module to abort the connection because of some abnormal condition.

SYN: This bit requests a connection (discussed later).

FIN: When this bit is set, it tells the receiver that the sender does not have any more data to send. The sender can still receive data from the other direction until it receives a segment with the FIN bit set.

Window size: The window size field specifies the number of bytes the sender is willing to accept. This field can be used to control the flow of data and congestion.

Checksum: This field detects errors on the TCP segment. The procedure is discussed below.

Urgent pointer: When the URG bit is set, the value in the urgent pointer field added to that in the sequence number field points to the last byte of the "urgent data" (data that needs immediate delivery). However, the first byte of the urgent data is never explicitly defined. Because the receiver's TCP module passes data to the application in sequence, any data in the receiver's buffer up to the last byte of the urgent data may be considered urgent.

Options: The options field may be used to provide other functions that are not covered by the header. If the length of the options field is not a multiple of 32 bits, extra padding bits will be added. The most important option is used by the sender to indicate the **maximum segment size (MSS)** it can accept. This option is specified during connection setup. Two other options that are negotiated during connection setup are intended to deal with situations that involve large delay-bandwidth products. The **window scale** option allows the use of a larger

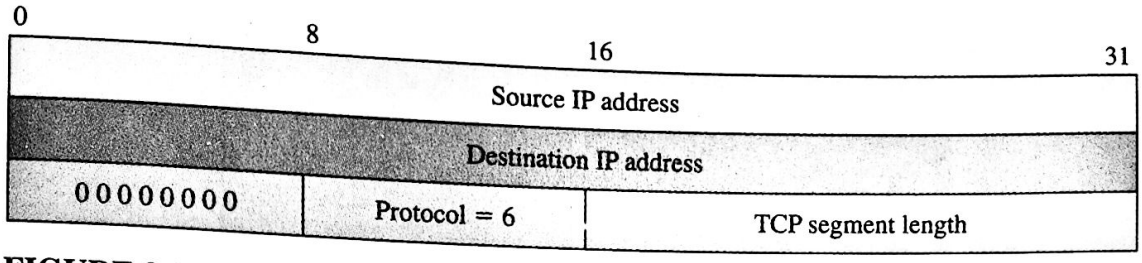

FIGURE 8.28 TCP pseudoheader.

advertised window size. The window can be scaled upward by a factor of up to 2^{14}. Normally the maximum window size is $2^{16} - 1 = 65{,}535$. With scaling the maximum advertised window size is $65{,}535 \times 2^{14} = 1{,}073{,}725{,}440$ bytes. The **timestamp** option is intended for high-speed connections where the sequence numbers may wrap around during the lifetime of the connection. The timestamp option allows the sender to include a timestamp in every segment. This timestamp can also be used in the RTT calculation.

TCP CHECKSUM

The purpose of the TCP checksum field is to detect errors. The checksum computation procedure is similar to that used to compute an IP checksum except for two features. First, if the length of the segment is not a multiple of 16 bits, the segment will be padded with zeros to make it a multiple of 16 bits. In doing so, the TCP length field is not modified. Second, a **pseudoheader** (shown in Figure 8.28) is added to the beginning of the segment when performing the checksum computation. The pseudoheader is created by the source and destination hosts during the checksum computation and is not transmitted. This mechanism ensures the receiver that the segment has indeed reached the correct destination host and port and that the protocol type is TCP (which is assigned the value 6). At the receiver the IP address information in the IP packet that contained the segment is used in the checksum calculation.

CONNECTION ESTABLISHMENT

Before any host can send data, a connection must be established. TCP establishes the connection using a **three-way handshake** procedure shown in Figure 8.29. The handshakes are described in the following steps:

1. Host A sends a connection request to host B by setting the SYN bit. Host A also registers its initial sequence number to use (Seq_no = x).
2. Host B acknowledges the request by setting the ACK bit and indicating the next data byte to receive (Ack_no = $x + 1$). The "plus one" is needed because the SYN bit consumes one sequence number. At the same time, host B also sends a request by setting the SYN bit and registering its initial sequence number to use (Seq_no = y).
3. Host A acknowledges the request from B by setting the ACK bit and confirming the next data byte to receive (Ack_no = $y + 1$). Note that the sequence number is set to $x + 1$. On receipt at B the connection is established.