

TCP And UDP

By Steve Steinke, Network Magazine

Feb 5, 2001 (10:03 AM)

URL: <http://www.networkmagazine.com/article/NMG20010126S0005>

The two Transport layer protocols in the TCP/IP family, TCP and UDP, provide network services for applications and Application layer protocols (including HTTP, SMTP, SNMP, FTP, and Telnet.) These two protocols perform those services by employing the IP to route packets to their destination networks. TCP provides connection-oriented, reliable, byte-stream packet delivery, while UDP provides connectionless, unreliable, byte-stream packet delivery. These terms need explanation.

Connection-oriented protocols establish an end-to-end link before any data moves. ATM and frame relay are connection-oriented protocols, but they operate at the Data-link layer rather than the Transport layer. Placing an ordinary voice phone call is also connection-oriented.

Reliable protocols safeguard against several forms of transmission mishaps. You can compare checksums included with a packet's data payload with a recalculation of the checksum algorithm at the destination to detect corrupted data. You must retransmit corrupted or lost data, so the protocol must provide methods for the destination to signal the source when retransmission is needed. Packetized data can arrive out of sequence, so the protocol must have a way to detect out-of-sequence packets, buffer them, and pass them to the Application layer in the correct order. It must also detect and discard duplicate transmissions. A collection of timers enables limiting the wait for various acknowledgements, so you can initiate retransmissions or link re-establishment.

Byte-stream protocols don't specifically support data units other than bytes. TCP can't structure bytes of the data payload in a packet, nor can it cope with individual bits. As far as TCP is concerned, it's responsible for transporting an unstructured string of 8-bit bytes.

A connectionless protocol doesn't establish paths across the network before data can flow. Instead, the protocol routes connectionless packets or datagrams individually at each intermediate node. Without an end-to-end link, a connectionless protocol such as UDP isn't reliable. When a UDP packet moves into the network, the sending process can't know whether the packet arrives at its destination unless the Application layer acknowledges this fact. Nor can the protocol detect duplicate or out-of-sequence packets. The standard jargon describes UDP as "unreliable," though a more descriptive term might be "nonreliable." On modern networks, UDP traffic isn't prone to disruption, but you can't really call it "reliable," either.

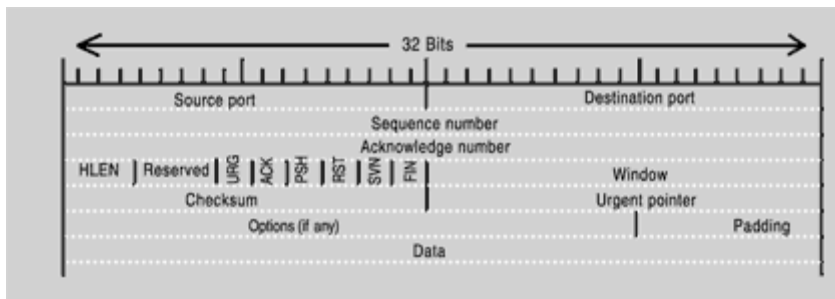


Figure 1: This diagram illustrates the fields of a TCP segment.

Figure 1 shows the fields of a TCP segment, the part of an IP packet that follows the IP header information. The first 16 bits identify the source port, and the second 16 bits identify the destination port. Port numbers provide a way for IP hosts to multiplex numerous types of concurrent connections at a single IP address. The combination of a 32-bit IP address and a 16-bit port address identifies a socket in most modern operating systems. The combination of a source socket and a destination socket defines a TCP connection. There are 2^{16} or 65,536 possible ports. The lowest 1,024 ports are called well-known ports; these are set aside by default for particular Application layer protocols. For example, HTTP uses port 80 by default, while POP3 uses port 110. Other applications can use the higher port numbers.

The next two fields, the sequence number and the acknowledgement number, are the keys to TCP's reliability functions. When a TCP connection is established, the initiating host sends an arbitrary initial sequence number to the initiatee. The initiatee adds 1 to the sequence number and returns it to the initiator in the acknowledgement field, thereby indicating the next byte that should be sent. Once data begins to flow, the sequence and acknowledgement numbers keep track of which data bytes have been sent and which data bytes have been acknowledged. Because each field is 32 bits, it can have 2^{32} values, so each field ranges from 0 to 4,294,967,295 and wraps around to 0 when it passes the upper limit.

The 4-bit data offset field simply indicates how many 32-bit words the TCP header has. This information is necessary because there are optional header fields, and the data offset marks where the header ends and the data begins.

TCP designers set aside the next 6 bits, just in case they might be needed for future development. Since RFC793 (Transmission Control Protocol) dates to 1981, and no one since has established a good reason to use these reserved bits, Jon Postel and his colleagues must have been overly cautious.

Each of the following 6 bits is a flag. An URG (urgent) flag with a value of 1 indicates that the data in the urgent pointer field, farther along in the header, is significant. An ACK (acknowledgement) flag with a value of 1 indicates that the data in the acknowledgement number field is significant. (Note that an initial setup or SYN packet has a meaningful sequence number, but not a meaningful acknowledgement number, since it isn't acknowledging anything.) The PSH (push) flag prevents data from waiting to be sent and from waiting to be processed by the receiving process. The RST (reset) flag shuts down a connection. The SYN (synchronization) flag indicates that the sequence number is significant. The FIN (finish) flag indicates the sender has no more data to send.

The Window field, 16 bits long, indicates the size of a so-called sliding window, which tells the sender how many bytes of data it's prepared to accept. TCP controls flow and congestion by adjusting the window size. A window equal to 0 tells a sender the receiver is overwhelmed and can't accept anything more without further notice. Large window sizes enable as many as 65,536 unacknowledged bytes to be in transit at any given time, but congestion—indicated when the retransmission timer expires without an acknowledgement—cuts the window size in half, effectively slowing the transmission rate.

The 16-bit checksum field protects the data payload's integrity, the TCP header, and certain fields of the IP header. The sender calculates the checksum value and inserts it in this field, and the receiver recalculates the value based on the received packet and compares the two. If they match, the data is probably intact.

The urgent pointer is a 16-bit offset value indicating the last byte that must be expedited when the urgent flag is set. The options field can hold 0 or more 32-bit words, extending TCP's capabilities. The most commonly used option supports window sizes greater than 65,536 bytes, reducing the time spent waiting for acknowledgements, especially at high data rates.

TCP processing entities have multiple timers. The retransmission timer begins when a segment is sent and stops when the acknowledgement is received. If it times out without an acknowledgement, the segment is sent again. One tricky problem is setting the value for the timeout period. If it's too long, unnecessary waiting occurs when the network drops or garbles numerous segments. If it's too short, the network will have too many duplicate segments when response slows down. Modern TCP implementations set the retransmission timer value dynamically in response to conditions.

The persistence timer is necessary to prevent a particular deadlock condition. If the network receives a 0-size window acknowledgement and loses the subsequent acknowledgement that restarts the flow, the persistence timer expires and sends a probe. The response indicates the window size (which may still be zero, in which case the timer starts over.)

The keepalive timer checks whether there is still an active process at the other end of the connection after no activity. The timer shuts down the connection if no response occurs.

A closing connection timer also provides for a period of twice the maximum packet lifetime when a connection is shut down. This timer makes sure the traffic is flushed through the connection before it's closed.

No matter how efficiently the retransmission process is implemented, however, a small number of dropped packets can seriously undermine the throughput of a TCP connection. Each packet, or fragment of a packet, that isn't received will only be missed when the retransmission timer expires. The receiving process must deliver the byte stream in order, so retransmission stops the flow of data until the missing bytes can be replaced. These retransmissions account for the sometimes herky-jerky performance of TCP-based links.

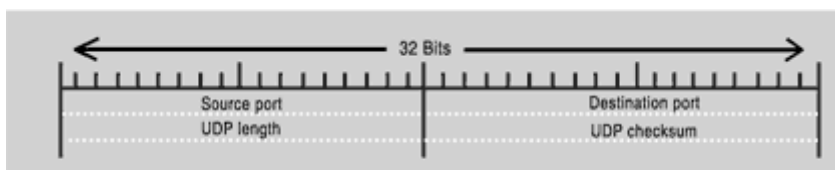


Figure 2: The UDP Leader is much simpler than the TCP Leader.

If you compare a UDP segment's structure (see Figure 2) to that of TCP, it's apparent that UDP doesn't have TCP's complex reliability and control mechanisms. UDP's source and destination port numbers support multiplexed applications on a host, just as TCP's do. The content of a 16-bit UDP length field is equal to the length of the 8-byte header plus the length of data, while the checksum field enables integrity checking. (Many applications commonly employing UDP, such as streaming media, derive no added value from data integrity, and wouldn't retransmit corrupted packets even if they identified them.)

TCP is clearly the protocol of choice for data transactions where performance must give way to integrity, controllability, and reliability. UDP is the best choice when performance matters more than perfect data integrity, as in voice and multimedia applications. UDP is also a good choice for

transactions so short that connection setup overhead is a large fraction of total traffic, for example, in DNS exchanges. The decision to base SNMP on UDP was made partly because designers thought that UDP would have a better chance of delivering management data when networks were distressed or congested because of UDP's lower overhead. TCP's rich functionality sometimes results in unpredictable performance, but reliable end-to-end connections are likely to support most networked applications for the foreseeable future.

Steve Steinke, editor in chief, can be reached at ssteinke@cmp.com.

Resources

The horse's mouth for TCP is RFC793, which dates to 1981. Among the many Internet RFC repositories is www.freesoft.org/CIE/index.htm. This Internet Encyclopedia site also includes further descriptions of TCP operation.

The preeminent discussions of TCP for application developers and others needing implementation details can be found in:

TCP/IP Illustrated, Volume 1: The Protocols by W. Richard Stevens (1994, Addison Wesley, ISBN: 0201633469)

TCP/IP Illustrated, Volume 2: The Implementation by Gary R. Wright and W. Richard Stevens (1995, Addison Wesley, ISBN: 020163354X)

Internetworking with TCP/IP Vol. I: Principles, Protocols, and Architecture by Douglas E. Comer (1995, Prentice Hall, ISBN: 0132169878)

Internetworking with TCP/IP, Vol.II, ANSI C Version, Design, Implementation, and Internals by Douglas E. Comer, David L. Stevens (1998, Prentice Hall, ISBN: 0139738436)