Understanding TCP/IP

TCP/IP, the ubiquitous network protocol, is actually a four-layer suite of protocols and is well worth gaining an understanding of. This month we explain UDP and TCP, the two protocols used by applications. Continuing our four-part article.

By Julian Moss

The link layer and network layer protocols of the TCP/IP suite, which are concerned with the basic mechanics of transferring blocks of data across and between networks, are the foundations of TCP/IP. They are used by the protocol stack itself, but they are not used directly by applications that run over TCP/IP.

Now we'll look at the two protocols that are used by applications: User Datagram Protocol (UDP) and Transmission Control Protocol (TCP).

User Datagram Protocol

The User Datagram Protocol is a very simple protocol. It adds little to the basic functionality of IP. Like IP, it is an unreliable, connectionless protocol. You do not need to establish a connection with a host before exchanging data with it using UDP, and there is no mechanism for ensuring that data sent is received.

A unit of data sent using UDP is called a datagram. UDP adds four 16-bit header fields (8 bytes) to whatever data is sent. These fields are: a length field, a checksum field, and source and destination port numbers. "Port number", in this context, represents a software port, not a hardware port.

The concept of port numbers is common to both UDP and TCP. The port numbers identify which protocol module sent (or is to receive) the data. Most protocols have standard ports that are generally used for this. For example, the Telnet protocol generally uses port 23. The Simple Mail Transfer Protocol (SMTP) uses port 25. The use of standard port numbers makes it possible for clients to communicate with a server without first having to establish which port to use.

The port number and the protocol field in the IP header duplicate each other to some extent, though the protocol field is not available to the higher-level protocols. IP uses the protocol field to determine whether data should be passed to the UDP or TCP module. UDP or TCP use the port number to determine which application-layer protocol should receive the data.

Although UDP isn't reliable, it is still an appropriate choice for many applications. It is used in real-time applications like Net audio and video where, if data is lost, it's better to do without it than send it again out of sequence. It is also used by protocols like the Simple Network Management Protocol (SNMP).

Broadcasting

UDP is suitable for broadcasting information, since it doesn't require a connection to be open before communication can take place. On a network, receiving a broadcast is something over which you have no choice. The targets of a broadcast message are determined by the sender, and specified in the destination IP address. A UDP datagram with a destination IP address of all binary ones (255.255.255.255) will be received by every host on the local network. Note the word local: a datagram with this address will not be passed by a router on to the Internet.

Broadcasts can be targeted at specific networks. A UDP datagram with the host and subnet part of the IP address set to all binary ones is broadcast to all the hosts on all the subnets of the network which matches the net part of the IP address. If only the host part (in other words, all the bits that are zero in the subnet mask) is set to binary ones, then the broadcast is restricted to all the hosts on the subnet that matches the rest of the address.

Multicasting is used to send data to a group of hosts that choose to receive it. A multicast UDP datagram has a destination IP address in which the first four bits are 1110, giving addresses in the range 224.x.x.x to 239.x.x.x. The remaining bits of the address are used to designate a multicast group. This is rather like a radio or television channel. For example, 224.0.1.1 is used for the Network Time Protocol. If a TCP/IP application wants to receive multicast messages, it must join the appropriate multicast group, which it does by passing the address of the group to the protocol stack.

Multicasts are, in effect, filtered broadcasts. The multicaster does not address individual messages to each

"Once a connection has been made, data can be sent. TCP is a sliding window protocol, so there is no need to wait for one segment to be acknowledged before another can be sent."

"TCP includes mechanisms for ensuring that data which arrives out of sequence is put back into the order it was sent. It also implements flow control, so a sender cannot overwhelm a receiver with data."

host that joins the group. Instead, the messages are broadcast, and the drivers on each host decide whether to ignore them or pass the contents up the protocol stack.

This implies that multicast messages must be broadcast throughout the entire Internet, since the multicaster does not know which hosts want to receive the messages. Fortunately this is unnecessary. IP uses a protocol called Internet Group Management Protocol (IGMP) to inform routers which hosts wish to receive which multicast group messages, so that the messages are only sent where they are needed.

TCP

Transmission Control Protocol is the transport layer protocol used by most Internet applications, like Telnet, FTP and HTTP. It is a connection-oriented protocol. This means that two hosts - one a client, the other a server - must establish a connection before any data can be transferred between them.

TCP provides reliability. An application that uses TCP knows that data it sends is received at the other end, and that it is received correctly. TCP uses checksums on both headers and data. When data is received, TCP sends an acknowledgement back to the sender. If the sender does not receive an acknowledgement within a certain timeframe the data is re-sent.

TCP includes mechanisms for ensuring that data which arrives out of sequence is put back into the order it was sent. It also implements flow control, so a sender cannot overwhelm a receiver with data.

TCP sends data using IP, in blocks which are called segments. The length of a segment is decided by the protocol. Each segment contains 20 bytes of header information in addition to the IP header. The TCP header starts with 16-bit source and destination port number fields. As with UDP, these fields specify the application layers that have sent and are to receive the data. An IP address and a port number taken together uniquely identify a service running on a host, and the pair is known as a socket.

Next in the header comes a 32-bit sequence number. This number identifies the position in the data stream that the first byte of data in the segment should occupy. The sequence number enables TCP to maintain the data stream in the correct order even though segments may be received out of sequence.

The next field is a 32-bit acknowledgement field, which is used to convey back to the sender that data has been received correctly. If the ACK flag is set, which it normally is, this field contains the position of the next byte of data that the sender of the segment expects to receive.

In TCP there is no need for every segment of data to be acknowledged. The value in the acknowledgement field is interpreted as "all data up to this point received OK". This saves bandwidth when data is all being sent one way by reducing the need for acknowledgement segments. If data is being sent in both directions simultaneously, as in a full duplex connection, then acknowledgements involve no overhead, as a segment carrying data one way can contain an acknowledgement for data sent the other way.

Next in the header is a 16-bit field containing a header length and flags. TCP headers can include optional fields, so the length can vary from 20 to 60 bytes. The flags are: URG, ACK (which we have already mentioned), PSH, RST, SYN and FIN. We shall look at some of the other flags later.

The header contains a field called the window size, which gives the number of bytes the receiver can accept. Then there is a 16-bit checksum, covering both header and data. Finally (before the optional data) there is a field called the "urgent pointer". When the URG flag is set, this value is treated as an offset to the sequence number. It identifies the start of data in the stream that must be processed urgently. This data is often called "out-of-band" data. An example of its use is when a user presses the break key to interrupt the output from a program during a Telnet session.

Connection

Before any data can be sent between two hosts using TCP, a connection must be established. One host, called the server, listens out for connection requests. The host requesting a connection is called the client.

To request a connection, a client sends a TCP segment specifying its own port number and the port that it

"If a name isn't found in the HOSTS file, the software contacts one of the local name servers whose IP address is in the TCP/IP configuration, to see if it knows the address."

TCP/IP

wants to connect to. The SYN (synchronise sequence numbers) flag is set, and the client's initial data sequence number is specified.

To grant the connection, the server responds with a segment in which the header contains its own initial data sequence number. The SYN and ACK flags are set. To acknowledge receipt of the client's data sequence number the acknowledgement field contains that value plus one.

To complete the connection establishment protocol, the client acknowledges the server's data sequence number by sending back a segment with the ACK flag set and the acknowledgement field containing the server's data sequence number plus one.

Using TCP, segments are only sent between client and server if there is data to flow. No status polling takes place. If the communication line goes down, neither end will be aware of the failure until data needs to be sent.

In practice, an application timeout would usually terminate the connection if a certain interval elapsed without any activity occurring. However, as many dial-up Internet users have found, it is possible to continue a failed session as if nothing has happened if you can bring the connection up again. Note that this is only true if your ISP gives you a fixed IP address. If IP addresses are allocated dynamically when you log on, you won't be able to resume the connection because your socket (which, as we mentioned earlier, is comprised of your IP address and port number) would be different.

Data Transmission

Once a connection has been made, data can be sent. TCP is a sliding window protocol, so there is no need to wait for one segment to be acknowledged before another can be sent. Acknowledgements are sent only if required immediately, or after a certain interval has elapsed. This makes TCP an efficient protocol for bulk data transfers.

One example of when an acknowledgement is sent immediately is when the sender has filled the receiver's input buffer. Flow control is implemented using the window size field in the TCP header. In the segment containing the acknowledgement the window size would be set to zero. When the receiver is once more able to accept data, a second acknowledgement is sent, specifying the new window size. Such an acknowledgement is called a window update.

When an interactive Telnet session is taking place, a single character typed in at the keyboard could be sent in its own TCP segment. Each character could then be acknowledged by a segment coming the other way. If the characters typed are echoed by the remote host then a further pair of segments could be generated, the first by the remote host and the second, its acknowledgement, by the Telnet client. Thus, a single typed character could result in four IP packets, each containing 20 bytes of IP header, 20 bytes of TCP header and just one byte of data being transmitted over the Internet.

How The Domain Name System Works

IP addresses are easy for computers to work with, but hard for humans to remember. The Domain Name System (DNS) solves that problem by allowing us to refer to hosts by names like "mail.compulink.co.uk" instead of "153.158.14.1". A computer called a name server lets Internet applications look up the IP address of any known host, and conversely get the hostname associated with a given IP address.

Domain names are organised hierarchically. At the right is the top-level domain, which may indicate a class of organisation such as .com or .gov, or a country, such as .au or .uk. The top-level domains are divided into second-level domains, such as .co.uk. Second-level domains can be further subdivided, and so on.

The organisations which manage the top-level domains maintain name servers, called the root name servers, which know the IP addresses of the name servers for the second-level domains. The managers of the second-level domains must maintain servers which know the addresses of the third-level name servers, and so on. A lower-level domain such as "ibm.com" or "compulink.co.uk" can represent an entire network. The name servers at that level must supply the IP addresses of all the hosts within it.

In a fully-qualified domain name, the host name is the name on the left. Thus, in order for "www.ibm.com" to take you to IBM's Web site, IBM must name its Web server "www" and have an entry on its name servers linking this name with the server's IP address.

When an application tries to contact a host by name, the TCP/IP stack runs a module called the resolver. First, this tries to look up the IP address locally. On a Windows PC, it looks in the file C:\WINDOWS\HOSTS, which is a text file containing a list of entries in the format <IP address> <host name>. This is the way all look-ups were done in the days before name servers were invented.

If the name isn't found in the HOSTS file, the software contacts one of the local name servers whose IP address is in the TCP/IP configuration, to see if it knows the address. If the host you are after isn't in the local zone it probably won't, unless that host has been contacted recently and its address is cached. Name servers cache IP addresses so they don't have to find out the addresses of popular hosts every time they are contacted.

If the local name server doesn't know the address for the host you want, it contacts the root name server for that host's top-level domain, whose address it does know. The root-level name server gives the local name server the address of the appropriate second-level server. The second-level server gives it the third-level server's address and so on, until eventually a server

TCP has some features to try to make things a bit more efficient. An acknowledgement delay of anything up to 500 ms can be specified in the hope that within that time some data will need to be sent the other way, and the acknowledgement can piggyback along with it.

The inefficiency of sending many very small segments is reduced by something called the Nagle algorithm. This states that a TCP segment containing less data than the receiver's advertised window size can only be sent if the previous segment has been acknowledged. Small amounts of data are aggregated until either they equal the window size, or the acknowledgement for the previous segment is received. The slower the connection, the longer will be the period during which data can be aggregated, and thus fewer separate TCP segments will be sent over the busy link.

Error Correction

An important advantage of TCP over UDP is that it is a reliable data transport protocol. It can detect whether data has been successfully received at the other end and, if it hasn't been, TCP can take steps to rectify the situation. If all else fails, it can inform the sending application of the problem so that it knows that the transmission failed.

The most common problem is that a TCP segment is lost or corrupted. TCP deals with this by keeping track of the acknowledgements for the data it sends. If an acknowledgement is not received within an interval determined by the protocol, the data is sent again.

The interval that TCP will wait before retransmitting data is dependent on the speed of the connection. The protocol monitors the time it normally takes to receive an acknowledgement and uses this information to calculate the period for the retransmission timer. If an acknowledgement is not received after re-sending the data once, it is sent repeatedly, at ever-increasing intervals, until either a response is received or (usually) an application timeout value is exceeded.

As already mentioned, TCP implements flow control using the window size field in the header. A potential deadlock situation arises if a receiver stops the data flow by setting its window size to zero and the window update segment that is meant to start data flowing again is lost. Each end of the connection would then be stalled, waiting for the other to do something.

Acknowledgements are not themselves ACKed, so the retransmission strategy would not resolve the problem in this case. To prevent deadlock from occurring, TCP sends out window probe messages at regular intervals to query the receiver about its window size.

Closing A Connection

When the time comes to close a TCP connection, each direction of data flow must be closed down separately. One end of the connection sends a segment in which the FIN (finished sending data) flag is set. The receipt of this segment is acknowledged, and the receiving end notifies its application that the other end has closed that half of the connection.

The receiver can, if it wishes, continue to send data in the other direction. Normally, however, the receiving application would instruct TCP to close the other half of the connection using an identical procedure.

Click here for the third part of this article

"The concept of port numbers is common to both UDP and TCP. The port numbers identify which protocol module sent (or is to receive) the data. Most protocols have standard ports that are generally used for this."

"An acknowledgement delay of anything up to 500 ms can be specified in the hope that within that time some data will need to be sent the other way, and the acknowledgement can piggyback along with it."

PCNA

The Author

Julian Moss is a freelance IT writer and software developer. He can be contacted as jmoss@cix.co.uk.

Recent Reviews from Tech Support Alert

Reviews of the Best Windows Backup Software

In this detailed comparative review, we checked out eighteen backup software utilities designed for home or SOHO use. Many of the products reviewed were disappointing. However 6 products passed our tests with flying colors and 2 of these were so impressive, they were awarded our "Editor's Choice."

Suppliers of Cheap Inkjet Printer Cartridges Reviewed and Rated

With hundreds of companies all claiming to have the *"cheapest and best inkjet printer cartridges*," our editors decided to put their claims to the test. Not unexpectedly, many suppliers flunked but we did manage to come up with a number of web sites that sell good quality inkjet printer cartridges at heavily discounted prices.

The Best Anti Trojan Software

Our editors took a close look at the 6 leading anti-trojan/trojan remover software utilities. Unfortunately, they found only 2 products that were effective in their ability to detect and remove dangerous modern polymorphic and process injecting trojans.

The 46 Best Ever Freeware Utilities

This is our Editor, Ian "Gizmo" Richards, personal selection of the best freeware utilities. He's hunted down some real gems, many of which perform better than expensive commercial products.