# 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, if only to ensure that you set it up in the most efficient way on your network.*

*By Julian Moss*

Everyone knows that TCP/IP is a network protocol used on LANs, WANs and the Internet, but not everyone who uses it understands how it works. It's possible to use TCP/IP with little more than a knowledge of how to configure the protocol stack, but a better understanding will give you a clearer picture of what is going on in your network and why the protocol needs to be set up in a particular way.

The aim of this multi-part article is to explain the key concepts behind TCP/IP.

TCP/IP stands for Transmission Control Protocol/Internet Protocol. If this leads you to think that it is not just one protocol, you're right. In fact, it is not just two protocols, either. TCP/IP is a suite of protocols. We'll cover the most important ones in the course of this article.

## Layered Protocol

Like most network protocols, TCP/IP is a layered protocol. Each layer builds upon the layer below it, adding new functionality. The lowest-level protocol is concerned purely with the business of sending and receiving data - any data - using specific network hardware. At the top are protocols designed specifically for tasks like transferring files or delivering email. In between are levels concerned with things like routing and reliability.

The benefit that the layered protocol stack gives you is that, if you invent a new network application or a new type of hardware, you only need to create a protocol for that application or that hardware: you don't have to re-write the whole stack.

## Link Layer

TCP/IP is a four-layer protocol, as illustrated in Figure 1. The lowest level, the link layer, is implemented within the network adapter and its device driver. Like all the TCP/IP protocols, it is defined by standards. The standards for generic Ethernet-type networks are defined by the IEEE 802 Committee: for example, IEEE 802.3 for Ethernet networks, or IEEE 802.5 for Token Ring networks.

Other link layer protocols that could be used include Serial Line IP (SLIP) or Point-to-Point Protocol (PPP), which are used when connecting to a network over an asynchronous dial-up link.

Since Ethernet is the most common type of network, we will look at it in a bit more detail. The Ethernet protocol is designed for carrying blocks of data called frames. A frame consists of a header containing 48-bit hardware destination and source addresses (which identify specific network adapters), a 2-byte length field, and some control fields. There follows the data, and then a trailer which is simply a 32-bit cyclic redundancy check (CRC) field. The data portion of an Ethernet frame must be at least 38 bytes long, so filler bytes are inserted if necessary.

All this means that frames are at least 64 bytes long, even if they carry only one byte of user data: a significant overhead in some types of application.

Frames also have a maximum size. Less headers, the maximum size for an Ethernet frame is 1492 bytes, which is the maximum transmission unit (MTU) for Ethernet. All link layer protocols have an MTU. It is one hardware characteristic that the higher-level protocol needs to be aware of, because larger blocks of data must be fragmented into chunks that fit within the MTU and then reassembled on arrival at their destination.

## Network Layer

The next layer up from the link layer is called the network layer. The most important protocol at this level is IP, the Internet Protocol. Its job is to send packets or datagrams - a term which basically means "blocks of data" - from one point to another. It uses the link layer protocol to achieve this.

Both the network layer and the link layer are concerned with getting data from point A to point B. However, whilst the network layer works in the world of TCP/IP, the link layer has to deal with the real world. Everything it does is geared towards the network hardware it uses.

An IP address is a "soft" address. It is a bit like calling your office block "Pan-Galactic House" instead of its real address, 2326 Western Boulevard. The former is no use to the postman

*"A router examines every packet, and compares the destination address with a table of addresses that it holds in memory."*

who has to deliver the letters, unless he can use it to find out the latter. The link-layer Ethernet protocol needs to know the unique hardware address of the specific network adapter it has to deliver the message to and, in case of an error, the address of the one it came from.

To make this possible, the TCP/IP protocol suite includes link-layer protocols which convert between IP and hardware addresses. The Address Resolution Protocol (ARP) finds out the physical address corresponding to an IP address. It does this by broadcasting an ARP request on the network. When a host recognises an ARP request containing its own IP address, it sends an ARP reply containing its hardware address. There is also a Reverse ARP (RARP) protocol. This is used by a host to find out its own IP address if it has no way of doing this except via the network.

## Internet Protocol

IP is the bedrock protocol of TCP/IP. Every message and every piece of data sent over any TCP/IP network is sent as an IP packet.

IP's job is to enable data to be transmitted across and between networks. Hence the name: inter-net protocol. In a small LAN, it adds little to what could be achieved if the network applications talked directly to Ethernet. If every computer is connected to the same Ethernet cable, every message could be sent directly to the destination computer.

Once you start connecting networks together, however, direct Ethernet communication becomes impractical. At the application level you may address a message to a computer on the far side of the world, but your Ethernet card can't communicate with the Ethernet card on that computer. Physical Ethernet limitations would prevent it, for a start. It would, in any case, be undesirable for every computer in the world to be connected to one big network. Every message sent would have to be heard by every computer, which would be bedlam.

Instead, inter-net communications take place using one or more "hops". Your Ethernet card will communicate with another Ethernet device on the route to the final destination. Routing is the important capability that IP adds to a hardware network protocol. Before we come to it, we will look at some other features of IP.

*"The TTL field is a safety mechanism which prevents packets from travelling the Internet forever in routing loops. It is exploited in a novel way by the Traceroute diagnostic tool."*

## Features Of IP

IP is a connectionless protocol. This means that it has no concept of a job or a session. Each packet is treated as an entity in itself. IP is rather like a postal worker sorting letters. He is not concerned with whether a packet is one of a batch. He simply routes packets, one at a time, to the next location on the delivery route.

IP is also unconcerned with whether a packet reaches its eventual destination, or whether packets arrive in the original order. There is no information in a packet to identify it as part of a sequence or as belonging to a particular job. Consequently, IP cannot tell if packets were lost or whether they were received out of order.

IP is an unreliable protocol. Any mechanisms for ensuring that data sent arrives correct and intact are provided by the higher-level protocols in the suite.

## Packets

An IP packet consists of the IP header and data. The header includes a 4-bit protocol version number, a header length, a 16-bit total length, some control fields, a header checksum and the 32-bit source and destination IP addresses. This totals 20 bytes in all.

We won't go into the detail of all the IP control fields. However, the protocol field is important. It identifies which higher-level TCP/IP protocol sent the data. When data arrives at its destination (either the packet's destination address equals the host's own IP address, or it is a broadcast address) this field tells IP which protocol module to pass it on to.

One control field, the time-to-live (TTL) field, is interesting. It is initialised by the sender to a particular value, usually 64, and decremented by one (or the number of seconds it is held on to) by every router that the packet passes through. When it reaches zero the packet is discarded and the sender notified using the Internet Control Message Protocol (ICMP), a network-layer protocol for sending network-related messages.

The TTL field is a safety mechanism which prevents packets from travelling the Internet forever in routing loops. It is exploited in a novel way by the Traceroute diagnostic tool (see box).

| | |
|---|---|
| Application layer: | FTP, SMTP, SNMP |
| Transport layer: | TCP, UDP |
| Network layer: | IP |
| Link layer: | IEEE 802.x, PPP, SLIP |

*Figure 1 - TCP/IP is a four-layer protocol, of which the link layer is the lowest layer.*

# TCP/IP

*"Like most network protocols, TCP/IP is a layered protocol. Each layer builds upon the layer below it, adding new functionality."*

Although the total field length in the IP protocol header is 16 bits, IP packets are usually much smaller than the 64 KB maximum this implies. For one thing, the link layer will have to split this into smaller chunks anyway, so most of the efficiency advantages of sending data in large blocks is lost. For another, IP standards did not historically require a host to accept a packet of more than 576 bytes in length. Many TCP/IP applications limit themselves to using 512-byte blocks for this reason, though today most implementations of the protocol aren't so restricted.

## Internet Addressing

Internet protocol addresses, or IP addresses, uniquely identify every network or host on the Internet. To make sure they are unique, one body, called InterNIC, is responsible for issuing them.

If your network is connected to the Internet and the computers need to be addressable from the Internet you must use IP addresses issued by InterNIC. If you don't use InterNIC-issued addresses, you must set up the gateway between your network and the Internet so that packets containing the made-up addresses will never pass through it in either direction.

Internet addresses are 32 bits long, written as four bytes separated by periods (full stops). They can range from 1.0.0.1 to 223.255.255.255. It's worth noting that IP addresses are stored in big-endian format, with the most significant byte first, read left to right. This contrasts with the little-endian format used on Intel-based systems for storing 32-bit numbers. This minor point can cause a lot of trouble for PC programmers and others working with raw IP data if they forget.

IP addresses comprise two parts, the network ID and the host ID. An IP address can identify a network (if the host part is all zero) or an individual host. The dividing line between the network ID and the host ID is not constant. Instead, IP addresses are split into three classes which allow for a small number of very large networks, a medium number of medium-sized networks and a large number of small networks.

Class A addresses have a first byte in the range 1 to 126. The remaining three bytes can be used for unique host addresses. This allows for 126 networks each with up to 16m hosts.

Class B addresses can be distinguished by first byte values in the range 128.0.x.x to 191.255.x.x. In these addresses, the first two bytes are used for the net ID, and the last two for the host ID, giving addresses for 16,000 networks, each with up to 16,000 hosts.

Class C addresses are in the range 224.0.0.x to 239.255.255.x. Here, the first three bytes identify the network, leaving just one byte for the individual hosts. This provides for 2 million networks of up to 254 hosts each.

Although these addresses make it possible to uniquely identify quite a lot of networks and hosts, the number is not that large in relation to the current rate of expansion of the Internet. Consequently, a new addressing system has been devised which is part of Internet Protocol version 6 (IPv6). IPv6 won't come into use for a couple of years, and understanding it isn't essential to understanding how IP works in general, so we won't cover it here. [*For a full description of IPv6, see article C0655 in PCNA 83 - Ed.*]

IP addresses can be further divided to obtain a subnet ID. The main net ID identifies a network of networks. The subnet ID lets you address a specific network within that network. This system of addressing more accurately reflects how real-world large networks are connected together.

You decide how the subnet ID is arrived at by defining a 32-bit value called the subnet mask. This is logically ANDed with the IP address to obtain the subnet address. For example, if a subnet mask was 255.255.255.0 and an IP address was 128.124.14.5, 128.124 would identify the Class B network, 128.124.14 would identify the

### Traceroute - How It Works

Traceroute, if you haven't used it before, is a diagnostic tool that lets you find out the route Internet traffic takes between you and any given destination. It exploits the fact that traffic between two points will usually follow the same route at any given time, and that a router will notify the sender using an ICMP message whenever it receives an IP packet containing a time-to-live (TTL) field of one.

Normally, the TTL field of an IP packet is set to the value 64. Traceroute starts by sending a UDP datagram to the destination you specify, setting the TTL field to 1. The first router that receives it discards it, and sends an ICMP "time-to-live equals 0" notification back. In the header of the ICMP message is the router's IP address, from which its name can be determined. Next, Traceroute sends the datagram with a TTL of 2. This gets as far as the second router before being discarded. Again, an ICMP message comes back.

This process is repeated with ever-increasing TTLs until the datagram reaches the destination. To create an error when the destination is reached, the UDP datagram is addressed to a non-existent port on the destination host. This causes the host to respond with an ICMP "destination port unreachable" message. Thus, Traceroute knows that the route has been completed.

*"If you don't use InterNIC-issued addresses, you must set up the gateway between your network and the Internet so that packets containing the made-up addresses will never pass through it in either direction."*

subnetwork, and 5 would identify the host on that subnetwork. [*An article which covers subnet masks and related topics in more detail is currently in preparation - Ed.*]

## Special Meanings

A few IP addresses have special meanings. A network ID of 0 in an address means "this network", so for local communication only the host ID need be specified. A host ID of 0 means "this host".

A network ID of 127 denotes the loopback interface, which is another way of specifying "this host". The host ID part of the address can be anything in this case, though the address 127.0.0.1 is normally used. Packets sent to the loopback address will never appear on the network. It can be used by TCP/IP applications that run on the same machine and want to communicate with one another.

Addresses in the range 224.x.x.x to 239.x.x.x are Class D addresses, which are used for multi-casting. Addresses 240.x.x.x to 247.x.x.x are reserved for experimental purposes.

Net, subnet and host IDs of all binary ones (byte value 255) are used when an IP packet is to be broadcast. Mercifully, an address of 255.255.-255.255 does not result in a broadcast to the entire Internet.

Three sets of addresses are reserved for private address space - networks of computers that do not need to be addressed from the Internet. There is one class A address (10.x.x.x), sixteen class B addresses (172.16.x.x to 172.31.x.x), and 256 class C addresses (192.168.0.x to 192.168.255.x). If you have equipment which uses IP addresses that have not been allocated by InterNIC then the addresses used should be within one of these ranges, as an extra precaution in case router misconfiguration allows packets to "leak" onto the Internet.

## IP Routing

So how does an IP packet addressed to a computer on the other side of the world find its way to its destination? The basic mechanism is very simple.

On a LAN, every host sees every packet that is sent by every other host on that LAN. Normally, it will only do something with that packet if it is addressed to itself, or if the destination is a broadcast address.

A router is different. A router examines every packet, and compares the destination address with a table of addresses that it holds in memory. If it finds an exact match, it forwards the packet to an address associated with that entry in the table. This associated address may be the address of another network in a point-to-point link, or it may be the address of the next-hop router.

If the router doesn't find a match, it runs through the table again, this time looking for a match on just the network ID part of the address. Again, if a match is found, the packet is sent on to the address associated with that entry.

If a match still isn't found, the router looks to see if a default next-hop address is present. If so, the packet is sent there. If no default address is present, the router sends an ICMP "host unreachable" or "network unreachable" message back to the sender. If you see this message, it usually indicates a router failure at some point in the network.

The difficult part of a router's job is not how it routes packets, but how it builds up its table. In the simplest case, the router table is static: it is read in from a file at start-up. This is adequate for simple networks. You don't even need a dedicated piece of kit for this, because routing functionality is built into IP.

Dynamic routing is more complicated. A router builds up its table by broadcasting ICMP router solicitation messages, to which other routers respond. Routing protocols are used to discover the shortest path to a location. Routes are updated periodically in response to traffic conditions and availability of a route. However, the details of how this all works is beyond the scope of this article.

Click here for the second part of this article

PCNA


## The Author

Julian Moss is a freelance IT writer and programmer, and developer of Visual DialogScript, a scripting and automation tool for Windows. He can be contacted as jmoss@cix.co.uk.

# 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.

*"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."*

### 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**

*"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.

# 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. The third instalment of our four-part article.*

*By Julian Moss*

In the previous instalment of this article [*PCNA 88, File C04100*] we looked at the transport layer protocols of the TCP/IP suite: User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). We saw that UDP is an unreliable, connectionless protocol suitable for transferring small amounts of data and for broadcast and multicast applications, and we saw that TCP implements reliability mechanisms and requires clients to establish a connection with a server before data can be transferred. This month we will examine some of the application-layer protocols, how they work, and how they exploit the characteristics of UDP and TCP.

## Time

A network time service is one of the simplest possible Internet applications. It tells you the time as a 32-bit value, giving the number of seconds that have elapsed since midnight on 1st January 1900.

Time servers use the well-known port number 37. When your time client opens UDP port 37 on the server, the server responds by sending the four bytes of time information.

For such a simple transaction UDP is perfectly adequate, though as it happens many time servers do support connections using TCP as well. TCP's built in reliability is of little use in this application, because by the time the protocol decides that the message may have been lost and re-sends it, the information it contained will be out of date. UDP is the most suitable protocol for real-time applications like this, and others like audio, video and network gaming.

## SNMP

A slightly more complex UDP application is Simple Network Management Protocol (SNMP). It allows applications to glean information about how various elements of the network are performing, and to control the network by means of commands sent over it rather than by physical configuration of equipment.

In SNMP there are two distinct components, the SNMP manager and SNMP agents. A manager can communicate with many agents. Typically, the SNMP manager would be an application running on the network manager's console, and agents will run on user workstations, in hubs, routers and other pieces of network hardware.

All communication is between the manager and an agent. Agents don't communicate with each other. Communication may be infrequent and sporadic, and the amount of information exchanged small. Usually a command sent by the manager will generate just a single response.

SNMP uses UDP. This avoids the overhead of having to maintain connections between the SNMP manager and each agent. Because the communication protocol consists essentially of a request for data and a reply containing the data requested, UDP's lack of reliability is not a problem. Reliability is easily implemented within the SNMP manager by re-sending a request if no response is received within a certain period.

The main function of SNMP is to allow the manager to get information from tables maintained by the agents. The tables are known as the Management Information Base (MIB). The MIB is divided into groups, each containing information about a different aspect of the network. Examples of the information that the MIB may contain include the name, type and speed of a network interface, a component's physical location and the contact person for it, and statistics such as the number of packets sent and the number that were undeliverable.

## Object IDs

Data is addressed using object IDs. These are written as sequences of numbers separated by periods, rather like long IP addresses. Each number going from left to right represents a node in a tree structure, with related information being grouped in one branch of the tree. There are standardised object IDs for commonly used items of information, and also a section for vendor-specific information. The assignment of object IDs is controlled by the Internet Assigned Numbers Authority (IANA).

*"A network time service is one of the simplest possible Internet applications. It tells you the time as a 32-bit value, giving the number of seconds that have elapsed since midnight on 1st January 1900."*

*"Most SNMP messages have a fixed format. In a typical transaction, an SNMP manager will send a UDP datagram to port 161 on a host running an SNMP agent."*

Most SNMP messages have a fixed format. In a typical transaction, an SNMP manager will send a UDP datagram to port 161 on a host running an SNMP agent. The datagram has fields for the type of message (in this case a get-request message), the transaction ID (which will be echoed in the response so that the manager can match up requests with the data received), and a list of object ID/value pairs. In the get-request message the object IDs specify the information requested and the value fields are empty.

The agent will respond with a datagram in which the message type field is get-response. An error status field will indicate whether the request has been fulfilled, or whether an error such as a request for a non-existent object ID occurred. The same list of object ID / value pairs as in the get-request message will be returned, but with the value fields filled in.

There are five types of message in SNMP version 1. Apart from get-request and get-response there is set-request, used by the SNMP manager to initialise a value, and get-next-request. The latter is a bit like listing a directory with a wildcard file spec, in that it returns a list of all the available object IDs in a particular group.

The fifth message type, trap, is used by SNMP agents to signal events to the SNMP manager. These messages are sent to UDP port 162. Trap messages have a format of their own. This includes a trap type field which indicates the type of event being signalled: for example, the agent initialising itself or the network device being turned off. There is a vendor-specific trap type which allows vendors to define traps for events of their own choosing.

### *Message Types*

One problem with SNMP version 1 is that the maximum size of a message is 512 bytes. This limit was chosen so that the UDP datagram in which it is sent falls within the limit (576 bytes) that all TCP/IP transports are guaranteed to pass. The error status value will indicate if the information requested is too big. Typically, this can occur when asking for text-based information, which is returned as strings of up to 255 characters in length.

SNMP version 2 adds two new message types. Get-bulk-request provides a way to retrieve larger amounts of data than version 1 can handle, and inform-request allows SNMP managers to communicate with one another. SNMP 2 also adds security features which can be used to help ensure that information is passed only to agents authorised to receive it.

### *Telnet*

Telnet is a terminal emulation application that enables a workstation to connect to a host using a TCP/IP link and interact with it as if it was a directly connected terminal. It is a client/server application. The server runs on a host on which applications are running, and passes information between the applications and the Telnet clients. The well-known port number for Telnet servers is TCP port 23.

Telnet clients must convert the user data between the form in which it is transmitted and the form in which it is displayed. This is the difficult part of the application, the terminal emulation, and has little to do with the Telnet protocol itself. Telnet protocol commands are principally used to allow the client and server to negotiate the display options, because Telnet clients and servers don't make assumptions about each other's capabilities.

TCP provides the reliability for Telnet, so neither the client nor the server need be concerned about re-sending data that is lost, nor about error checking. This makes the Telnet protocol very simple. There is no special format for TCP segments that contain commands - they simply form part of the data stream.

Data is sent, usually as 7-bit ASCII, in TCP packets (which you may recall are called segments). A byte value of 255, "interpret as command" (IAC), means that the bytes which follow are to be treated as Telnet commands and not user data. This is immediately followed by a byte that identifies the command itself, and then a value. Many commands are fixed length, so the byte after that, if not another IAC, would be treated as user data. To send the byte 255 as data, two consecutive bytes of value 255 are used.

Some commands, such as those that include text values, are variable length. These are implemented using the sub-

*"Telnet clients must convert the user data between the form in which it is transmitted and the form in which it is displayed. This is the difficult part of the application."*

# TCP/IP

*"Telnet allows you to interact with an application running on a remote computer, but it has no facility for enabling you to copy a file from that computer's hard disk to yours."*

option begin (SB) and sub-option end (SE) command bytes. These command bytes enclose the variable length data like parentheses.

The principal Telnet commands used to negotiate the display options when a client connects to a server are WILL (sender wants to enable this option), WONT (sender wants to disable this option), DO (sender wants the receiver to enable this option) and DONT (sender wants the receiver to disable this option).

To see how this works, consider an example. You start your Telnet client, which is configured to emulate a VT 220 terminal, and connect to a host. The client sends WILL <terminal-type> (where <terminal-type> is the byte value representing the terminal type display option) to say that it wants to control what terminal type to use. The server will respond with DO <terminal-type> to show that it is happy for the client to control this option.

Next the server will send SB <terminal-type> <send> SE. This is an invitation to the client to tell the server what its terminal type is: <send> is a byte that means "send the information". The client responds with SB <terminal-type> <is> VT 220 SE (<is> is a byte that indicates that the requested information follows) and so the server is informed of the terminal emulation that the client will be using.

Client and server will negotiate various other options at the start of a connection. Certain options may also be changed during the Telnet session. The echo option determines whether or not characters that are sent by the client are echoed on the display, and by which end. If characters that are typed at the terminal are to be echoed back by the host application the Telnet server will send WILL <echo> to the client, which will agree to this by sending DO <echo>. This option can be changed during a session to suppress the display of password characters.

Another Telnet option to be negotiated is the transmission mode. The usual mode is character-at-a-time mode, where each character typed at the terminal is echoed back by the server unless the host application specifically turns echoing off. You can tell when character-at-a-time mode is being used because there is a delay between a key being pressed and a character appearing in the terminal window.

The main alternative to character-at-a-time mode is line mode. In this mode, the client displays the characters typed and provides line editing capabilities for the user. Only completed lines are sent to the server. Line mode is used by some mainframe terminal emulations. Again, it is possible to switch modes during a Telnet session if it is required to interact with an application running on the host that responds to single keystrokes rather than whole lines of input.

The urgent flag and urgent pointer in a TCP segment come into use when a Telnet terminal user presses the Break key to interrupt a process on the host. Break is converted by the Telnet client into two Telnet commands which are sent to the server: IP (interrupt process) followed by DO <timing mark> (again, we use angle brackets to indicate a byte representing an option). The server responds to the latter with WILL <timing mark> followed by a DM (data mark) command. The urgent pointer is set to point to the DM command byte, so even if flow control has halted the transmission of normal data this command will still be received. Data mark is a synchronisation marker which causes any queued data up to that point to be discarded.

Most of the data that passes between client and server during a Telnet session is user input and application data. The important thing to realise is that Telnet does not package up this data with additional headers or control information: it is simply passed directly to TCP. One side effect of this is that you can use a Telnet client to talk to other TCP applications that use ASCII-based protocols simply by connecting to the appropriate port. Though it might not normally be sensible to do this, it can be a useful troubleshooting tool.

### *Finger*

Finger is a simple example of a TCP/IP application that uses an ASCII-based protocol. A Finger server is

*"The well-known Finger port is TCP port 79. A Finger client opens this port and then sends a request, which is either a null string or a user name. The server responds by sending some text and closing the connection."*

a program that supplies information to a requesting client. The information supplied usually relates to the user accounts on a host, though many ISPs use Finger servers to provide status information.

The well-known Finger port is TCP port 79. A Finger client opens this port and then sends a request, which is either a null string or a user name. The server responds by sending some text and closing the connection. If a null string was sent you may receive information about all users known to the system; a user name will return information about that specific user.

For security reasons many organisations do not run Finger servers, or have them reply with a standard message whatever the request. From our perspective the point of interest is that the protocol is pure ASCII text, as you can verify by connecting to a Finger server using a Telnet client.

## File Transfer Protocol

Telnet allows you to interact with an application running on a remote computer, but it has no facility for enabling you to copy a file from that computer's hard disk to yours, nor for you to upload files to the remote system. That function is carried out using File Transfer Protocol (FTP).

The FTP specification caters for several different file types, structures and transfer modes, but in practice FTP implementations recognise either text files or binary files. Text files are converted from their native format to 7-bit ASCII with each line terminated by a carriage-return, line-feed pair for transmission. They are converted back to the native text file format by the FTP client. FTP therefore provides a cross-platform transfer mechanism for text files. Binary files are transmitted exactly as-is.

Data is transferred as a continuous stream of bytes. The TCP transport protocol provides all the reliability, making sure that data that is lost is re-sent and checking that it is received correctly. It is worth noting that error detection uses a simple 16-bit checksum so the probability of undetected errors is high compared to a file transfer protocol like Zmodem which uses a 32-bit CRC.

FTP is unusual compared to other TCP applications in that it uses two TCP connections. A control connection is made to the well-known FTP port number 21, and this is used to send FTP commands and receive replies. A separate data connection is established whenever a file or other information is to be transferred, and closed when the data transfer has finished. Keeping data and commands separate makes life easier for the client software, and means that the control connection is always free to send an ABOR (abort) command to terminate a lengthy data transfer.

FTP commands are sent in plain 7-bit ASCII, and consist of a command of up to 4 characters followed by zero or more parameters (those familiar with text mode FTP clients like that supplied with Microsoft TCP/IP may find it curious that FTP commands are not the same as the commands given to the FTP client). The replies consist of a three digit number followed by an optional text explanation, for example, "250 CWD command successful". The numbers are for easy interpretation by FTP client software, the explanations are for the benefit of the user.

It is instructive to see what happens during a simple FTP session. When you connect to the FTP server (TCP port 21) it sends its welcome message prefixed by the numeric code 220. The FTP client prompts you for your username, which it then sends using the FTP command "USER username". The server may respond with "331 Need password for username". The client detects this, prompts you for the password and sends this to the server using the command "PASS password". If the password is correct the client will receive the response "230 Access granted".

The next thing you might do is type DIR, to list the current directory on the server. This command to the client results in two FTP commands being issued to the server. The first, "PORT x,x,x,x,y1,y2" tells the server the IP address (x.x.x.x) and port number (y1 * 256 + y2) to use for the data connection. The port number is one in the range 1024 to 4999, a range used for ephemeral connections (those that are used briefly for some specific purpose). The second, LIST, causes the server to open the specified port, send the directory list, and close it again.

The sequence for downloading a file is very similar to that for obtaining a directory list. First, a PORT command is used to specify the data connection port, and then the command "RETR filename" is sent to specify the file to be retrieved. The server opens the data port and sends the data, which the client writes to the hard disk. The server closes the TCP connection to the data port when the file transfer has finished, which is the signal to the client to close the newly-created file.

## Conclusion

Since you are unlikely to be asked to write your own client or server there is little to be gained from looking at these application protocols in more detail. However, it is hoped that some useful insights into the working of Internet applications can be gained from these brief descriptions of how a few of them work.

Perhaps the most striking thing about the protocols that use TCP is how simple they are. Because the lower protocol levels take care of reliability, routing and physical transfer matters, the application protocol need concern itself only with things relating to the application. This, of course, is the whole point of using a layered protocol stack.

**Click here for the final part of this article**

PCNA


## The Author

Julian Moss is a freelance writer and software developer. The URL of his Web site is http://www.jm-tech.com/.

# Understanding TCP/IP

*We conclude our four-part article looking in depth at the TCP/IP protocol. Here, we examine the difference between the SMTP and POP3 email protocols.*

*By Julian Moss*

In this series of articles we have looked at the TCP/IP suite of protocols, beginning with the link layer and progressing by stages to the application layer. We have seen how each layer relies upon the layers below it, so that network applications can be written without needing to take account of considerations such as how the network is constructed or what type of hardware or cabling is used.

A striking point about many of the application layer protocols is how simple they are. The protocols based on TCP mostly use commands and responses in plain ASCII text, making them easier for a user to understand and for a programmer to implement. For further illustration we shall look at the two protocols that you may use every day to send and receive Internet email: SMTP and POP3.

## SMTP

Simple Mail Transfer Protocol (SMTP) is one of the most venerable of the Internet protocols. Designed in the early 1980s, its function is purely and simply to transfer electronic mail across and between networks and other transport systems. As such, its use need not be restricted to systems that use TCP/IP. Any communications system capable of handling lines of up to 1,000 7-bit ASCII characters could be used to carry messages using SMTP. On a TCP/IP network, however, TCP provides the transport mechanism.

In SMTP the sender is the client, but a client may communicate with many different servers. Mail can be sent directly from the sending host to the receiving host, requiring a separate TCP connection to be made for each copy of each message. However, few mail recipients run their own SMTP servers.

It is more usual for the destination of an SMTP message to be a server that serves a group of users such as all those int one domain. The server receives all mail intended for its users and then allows them to collect it using POP3 (Post Office Protocol version 3) or some other mail protocol. Similarly, most SMTP clients send messages to a single server, whose job it is to relay those messages on to their eventual recipients.

An SMTP transaction begins when the sender client opens a TCP connection with the receiver using the well-known port number 25. The server acknowledges the connection by sending back a message of the form "220 SMTP Server Ready". SMTP uses a similar format of replies to ftp, which we looked at previously. The three-digit code is all the client software needs to tell if everything is going OK. The text is there to help the humans who might be troubleshooting a problem by analysing a log of the transaction. The box "Application Protocol Reply Codes" provides more information about message reply codes.

An SMTP relay server might refuse a connection by sending back a message with a "421 Service not available" reply code. For example, an Internet Service Provider's SMTP server provided for use by its subscribers to relay outgoing mail might refuse a connection from a host whose IP address indicates that it is not a subscriber to that ISP. SMTP has no form of access control - the way it can be used to relay messages would make this impractical - so this is about the only way ISPs can prevent non-subscribers such as spammers from using their mail servers to send out messages.

Having received the correct acknowledgement the sender signs on to the server by sending the string "HELO hostname". HELO is the sign-on command and hostname is the name of the host. As we will see, the hostname is used in the Received: header which the server adds to the message when it sends it on its way. This information allows the recipient to trace the path taken by the message.

## Sending

Once the sender gets a "250 OK" acknowledgement it can start sending messages. The protocol is extremely simple. All the sender has to do is say who the message is from, who it is to, and supply the contents of the message.

Who a message is from is specified with the command "MAIL FROM: <address>". This command also tells the receiver that it is about to receive a new message, so it knows to clear out its list of recipients. The address in the angle brackets (which are required) is the return path for the message. The return path is the address that any error report - such as would be generated if the message is undeliverable - is sent to.

*"SMTP uses a similar format of replies to ftp, which we looked at previously. The three- digit code is all the client software needs."*

It is valid for the return path to be null, as in "MAIL FROM: <>". This is typically used when sending an error report. A null return path means that no delivery failure report is required. Its main purpose is to avoid getting into the situation in which delivery failure messages continually shuttle back and forth because both sender and recipient addresses are unreachable.

The recipients of a message are defined using the command "RCPT TO: <address>". Each address is enclosed in angle brackets. A message may have many recipients, and an RCPT TO: command is sent for each one. It is the RCPT TO: command, not anything in the message headers, that results in a message arriving at its destination. In the case of blind carbon copies or list server messages the recipient address will not appear in the headers at all.

*"The return path is the address that any error report - such as would be generated if the message is undeliverable - is sent to."*

Each recipient is acknowledged with a "250 OK" reply. A recipient may also be rejected using a reply with a 550 reply code. This depends on how the server has been configured. Dial-up ISP SMTP relay servers may accept every RCPT TO: command, even if the address specified is invalid, because the server doesn't know that the address is invalid until it does a DNS lookup on it. However, a mail server intended to receive messages for local users only would reject recipients that aren't at that domain.

Other replies may be received in response to RCPT TO: messages as a result of the SMTP server being helpful. If an address is incorrect but the server knows the correct address it could respond with "251 User not local; will forward to <address>" or "551 User not local; please try <address>". Note the different reply codes signifying whether the server has routed the message or not. These replies aren't common, and a mail client may simply treat the 551 response as an error, rather than try to parse the alternative address out of the reply text.

For the sake of completeness it should be pointed out that RCPT TO: commands may specify routes, not merely addresses. A route would be expressed in the form "RCPT TO: <server1,server2:someone@server3>". Today this capability is rarely needed.

## Message Text

Once all the recipients have been specified, all that remains is for the sender to send the message itself. First it sends the command "DATA", and then waits for a reply like: "354 Start mail input; end with <CRLF>.<CRLF>". The message is then sent as a succession of lines of text. No acknowledgement is received for each line, though the sender needs to watch for a reply that indicates an error condition.

The end of the message is, as indicated by the reply shown above, a period (full stop) on a line of its own. Thus, one of the simplest but most essential things that a mail client must do is ensure that a line containing a single period does not appear in the actual text.

### Application Protocol Reply Codes

Many Internet application layer protocols which are based on ASCII text commands use a system of replies in which an initial three-digit code provides the essential status information. Each digit has a particular meaning, as shown below.

*First Digit*

**1xx**: Positive Preliminary Reply. Command accepted but held awaiting a further confirmation command (continue or abort).

**2xx**: Positive Completion Reply. Command completed. Awaiting next command.

**3xx**: Positive Intermediate Reply. Command accepted but held awaiting further information (such as a password).

**4xx**: Transient Negative Completion Reply. Command not accepted due to a temporary error condition (such as an HTTP server busy). The command may be tried again later.

**5xx**: Permanent Negative Completion Reply. Command not accepted due to a permanent error condition. The command is unlikely to be accepted if repeated later.

*Second Digit*

**x0x**: Syntax Error. For example, command unimplemented or valid but incorrect in the circumstances.

**x1x**: Information. The text following the code contains the answer to an information request.

**x2x**: Connections. Message reply relates to the communications channel.

**x5x**: Server. Message reply relates to the state of the server.

*Third Digit*

Used to distinguish individual messages.

# TCP/IP

The end of the message is acknowledged with "250 OK".

It's worth noting that SMTP isn't in the least bit interested in the content of the message. It could be absolutely anything, though strictly speaking it should not contain any characters with ASCII values in the range 128 to 255, and lines of text may not exceed 1,000 characters. There is no requirement for the headers to show the same sender and recipient addresses that were used in the SMTP commands, which makes it easy to make a message appear to have come from someone other than the true sender.

## Tracking

When a message is relayed by the server it inserts a "Received:" header at the start of the message showing the identity of the host that sent the message, its own host name, and a time stamp. Each SMTP server that a message passes through adds its own "Received:" header. Thus it is possible to track the path taken by a message. Although this won't identify the sender it may shed some light on whether or not the address the message is apparently from is in fact the true one.

After the "250 OK" that acknowledges the end of the message, the sender can start again with a new message by sending a new "MAIL FROM:" command or it can sign off from the server using "QUIT". A 221 reply will be received in response to the QUIT command.

SMTP servers should support two further commands for a minimum implementation. NOOP does nothing, but should provoke a "250 OK" reply. RSET aborts the current message transaction. There are other commands such as HELP which are really only of interest to those trying to communicate with SMTP servers interactively and are therefore not really relevant to understanding how the protocol works in day-to-day use.

## POP3

SMTP is capable of delivering mail direct to the recipient's desktop, but in practice it isn't the ideal protocol for this. If an SMTP relay is unable to deliver a message to the next (or final) host in the chain, it will try at ever-lengthening intervals over a period of a few days before giving up and sending a delivery failure notification to the return path address.

*"As with the other text-based application protocols you can connect with a POP3 server using a Telnet terminal emulator and interact with it using POP3 commands."*

SMTP offers no way for the recipient to prompt a server into sending mail that it is trying to deliver. If a recipient connects to the Internet infrequently their server may never be active at the right time. In this case the mail will eventually bounce.

SMTP is rather like a courier delivery service. If you aren't in when it calls then, after a couple of re-delivery attempts, the message is returned to the sender. Post Office Protocol version 3 (POP3) - as the name suggests - lets you have your mail held at the post office so you can collect it at a time of your own choosing.

POP3 is another TCP application, and uses the well-known port number 110. As with the other text-based application protocols you can connect with a POP3 server using a Telnet terminal emulator and interact with it using POP3 commands. This can sometimes be useful, as for example to manually delete a corrupt message that crashes a mail client whenever it is downloaded. (However, don't try connecting to your ISP's port 110 and sending random commands without permission. Their automatic hacker detection systems might spring into operation and you may well be asked to explain what you're doing.)

On connecting to the server, the server should respond with the message "+OK POP3 server ready". POP3 uses "+OK" and "-ERR" at the start of replies to indicate acceptance or rejection of commands. This is simpler than the numeric codes used by SMTP and other protocols: software need only check the first character for a plus or a minus. The text that may appear after a "+OK" is a prompt for what to do next. After "-ERR" it is an error description. The exact content of the text may vary between server implementations.

## To Access The Server

A POP3 server holds people's personal mail, so unsurprisingly you need to enter a user name and a matching password before you can gain access to it. To log in you must send "USER username". A "+OK" response shows that the user name is valid. You must then send "PASS password". If the password is correct you will receive another positive acknowledgement in a reply like "+OK username has two message(s) (914 octets)". "-ERR" replies may be received if the user name is not known, the password is incorrect or the server is for some reason unable to open a user's mailbox.

Once a client is successfully logged in it can issue several different commands which allow it to find out how many messages are waiting and how big they are, and to download the messages and delete them from the server.

The "STAT" command returns the number of messages waiting (mw) and their total size in bytes (sb), as a response in the form "+OK mw sb". Note that this is the same information given in the login acknowledgement, but in a form (two numbers separated by a single space) that is easier for the client software to process.

*"SMTP is rather like a courier delivery service. If you aren't in when it calls then, after a couple of re-delivery attempts, the message is returned to the sender."*

The command "LIST" can be used to determine the size of each message. After the "+OK" the server sends, on separate lines, the message numbers (mn) and the message sizes (ms) separated by a space. Waiting messages are numbered sequentially, starting from 1. The command "LIST mn" can be used to find out the size of a specific message. The LIST command is typically used by mail clients that implement a user-defined restriction on the size of messages that will be downloaded, or those that want to display a progress indicator that shows how much of each message has been downloaded.

POP3 provides no commands that enable a client to find out the subject of a message or who it is from. However, the TOP command lets the client download a message's headers and a specified number of lines from the message body, from which this information may be obtained. TOP is an optional POP3 command but its implementation is strongly recommended.

The format of the command is "TOP mn nl" where mn is the message number and nl the number of lines required. The response is "+OK" (if mn is valid) followed by a partial download of the message. The end of the download is indicated by a line containing a single period (full stop).

Some spam filtering software - which kills unwanted messages without downloading them - uses the TOP command to determine whether a message meets the criteria for being killed or not. However, the time taken to get this information for every message may exceed the time it would have taken simply to download the spam and delete it later.

The command "RETR mn" is used to retrieve messages from the server. The command must include a message number (mn). After an "+OK" acknowledgement the server sends the whole message. Again, the end of the message is indicated by a line containing just a period.

## Wiping

The command "DELE mn" is used to delete a message. In fact, the DELE command only marks messages for deletion. Any messages marked for deletion during a session may be undeleted by issuing an "RSET" command. The messages are only deleted once the client has closed the POP3 session by issuing a "QUIT" command. If a client never gets to close a session properly because the connection is lost or timed out then you may find some messages being downloaded again the next time you connect to the server.

In order to avoid downloading messages twice, a POP3 client can use the command "UIDL" or "UIDL mn" to obtain unique, server-generated IDs for each message. By storing the UIDLs of downloaded messages in a file, a client can easily determine whether a message on the server has been previously retrieved or not.

Implementation of the UIDL command is optional, but most POP3 servers seem to support it and most mail clients use it.

## Benefits

SMTP and POP3 are two of the most commonly-used Internet protocols, which is why we have devoted this article to looking at them in some detail. Their text-based nature, which makes it possible to send and receive messages by communicating with a server interactively using a simple Telnet client, also makes it easy to write client software using just about any programming language that can send and receive text using TCP.

This simplicity is in stark contrast to many other network architectures which require the use of proprietary APIs and languages that support complex data structures.

## Conclusion

In this article it has only been possible to give an overview of the most important protocols used on the Internet. The full specifications of these and other Internet protocols can be found in Requests For Comments (RFCs) published by the Network Working Group. RFCs are freely available for download from the Internet. Anyone interested in finding out more about TCP/IP, and particularly in implementing their own TCP/IP applications, should obtain and study the RFCs for the protocols concerned.

However, even if you never have to write your own Internet software it is hoped that this article has piqued your interest, and contributed to a better understanding of how TCP/IP and the Internet really work.

PCNA


## The Author

Julian Moss is a freelance writer and software developer with experience of developing TCP/IP client software. He can be contacted as jmoss@cix.co.uk.

## Additional Resources

- IPv6 Explained
- The OSI 7 Layer Model Explained
- Understanding Frame Relay
- Understanding DHCP
- Virtual Private Networking Explained

**All these articles are available free online now at www.pcnetworkadvisor.com**

**PCNA**