

[Previous topic](#) | [Next topic](#) | [Contents](#) | [Glossary](#) | [Contact z/OS](#) | [PDF](#)

Transmission Control Protocol (TCP)

Networking on z/OS

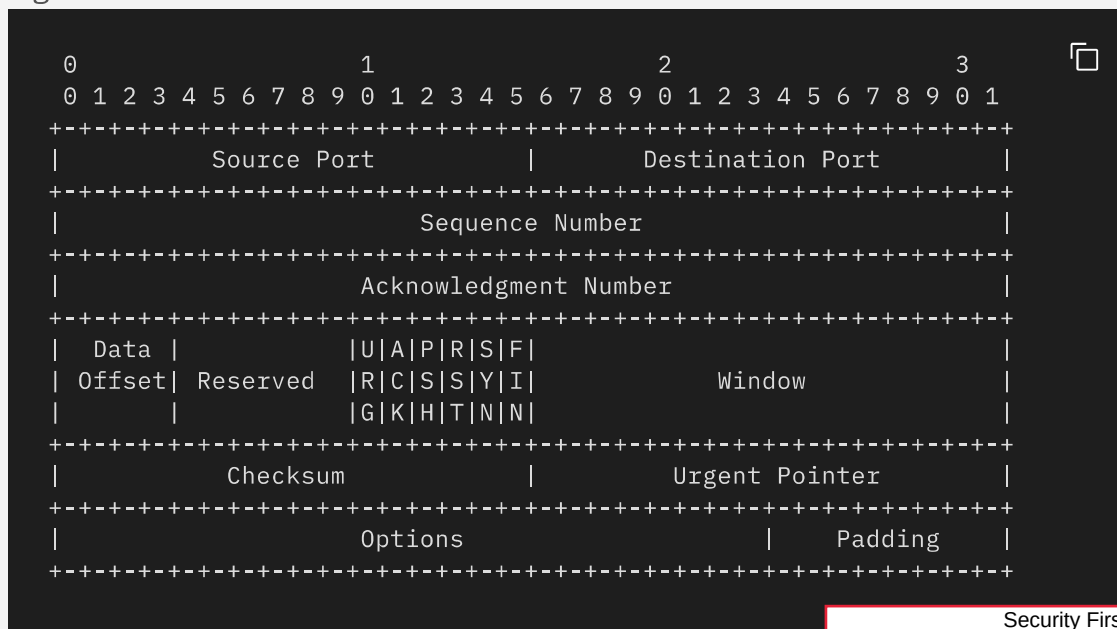
TCP is the "registered mail" protocol of internets.

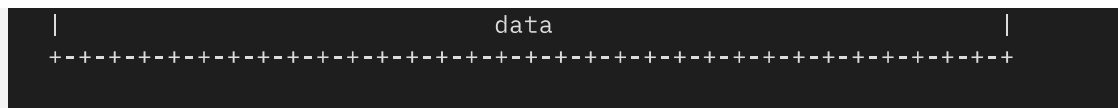
The standard way of ensuring the delivery of postal mail is to register the mail with the mail carrier. When the mail is received at the other end, an acknowledgement in the form of a signature is required. This signature is the sender's assurance that the mail has been received successfully at the remote end.

Acknowledged data

When a host requires assurance that the remote end has actually received the data it sends. But instead of requesting a signature at the remote end (computers have messy handwriting anyway), TCP requires an acknowledgement be returned. To get into details on how this is done, we'll begin by having a look at [Figure 1](#).

Figure 1. TCP header





At offset 32 into the TCP header is the sequence number. The *sequence number* is a counter used to keep track of every byte sent outward by a host. If a TCP packet contains 1400 bytes of data, then the sequence number will be increased by 1400 after the packet is transmitted.

At offset 64 is the *acknowledgement number*. This number is a counter to keep track of every byte that has been received. If 1000 bytes are received by a host, it increases the acknowledgement number by 1000 when it sends out a packet in response.

As mentioned, receiving data from a remote host causes the acknowledgement number at the local host to be increased by the number of bytes received. When the local host sends out its next packet, it will send this updated acknowledgement number, and it will also turn on the ACK flag (offset 107) to indicate to the other end that it is acknowledging the receipt of data. This is the nearest thing to a signature that TCP can do. The result is that TCP is capable of ensuring reliable delivery of data.



Note: The local host's sequence number usually matches the remote host's acknowledgement number, and the local host's acknowledgement number usually matches the remote host's sequence number.

Because a transmitted packet might not reach its destination, for whatever reason, and a transmitted packet might take some time to cross the network to its destination, the difference between a sender's sequence number and the remote host's acknowledgement number represents any outstanding, unacknowledged data.

Ports

The other fields of significant note in the TCP header are the source port number and the destination port number. A TCP-capable host, and particularly z/OS, is capable of running more than one TCP application.

For example, a web server and a FTP server might both be running on the same host, using the same IP address. After a packet has been delivered (via the

NIC, and up to the IP layer), how does the host know which application should receive the packet? The answer is by using the port number.

Port numbers are TCP's method of knowing which application should receive a packet. Returning to our postal mail envelope, we did not mention that the address placed onto that envelope included an apartment number. Sure, the IP address gets us to the correct host, but the port number tells us *which* application on the host is the final recipient.

In order to facilitate communication, many applications are assigned specific ports. Such ports are called *well-known ports*. For example, a web server normally listens on port 80. An FTP server normally listens on port 21.

Connection-oriented and state-aware

TCP is always referred to as a *connection-oriented protocol*. What this entails is that prior to any communication occurring between two endpoints, a connection must be established. During the communications (which can last for seconds or for days) the state of the connection is continually tracked. And, when the connection is no longer needed, the connection must be ended.

Because TCP forms the backbone of so much activity over an internet, a summary of the possible states for a TCP connection is appropriate:

LISTEN

An application (such as a web server) is awaiting an inbound connection request (from a browser, for example).

SYN-SENT

A connection request has been sent but no acknowledgement has been received from the remote host.

SYN-RECEIVED

A connection request has been received and a connection request and acknowledgement have been sent in response. Awaiting an acknowledgement of the connection request sent out as a response to the original connection request.

ESTABLISHED

All connection requests and acknowledgements have been sent and received. Data can move freely over the connection.

FIN_WAIT_1

An end connection request has been sent, but no acknowledgement has been received.

FIN_WAIT_2

An end connection request has been acknowledged by the remote host, but no corresponding end connection has been received from the remote host.

CLOSING

An end connection request has been sent out and an end connection response has been received and acknowledged. However, the remote end has not yet acknowledged the original end connection request.

CLOSE_WAIT

An end connection request was received and acknowledged but no corresponding end connection has been sent out yet.

TIME_WAIT

Waiting a reasonable amount of time to ensure that the final acknowledgement of a received end connection has been received at the remote end.

LAST_ACK

Awaiting a final acknowledgment after sending an end of connection in response to having received an end of connection request.

Parent topic:

→ [Transport layer, layer 4](#)



|



[Notices](#) | [Terms of use](#) | [Support](#) | [Contact z/OS](#)

©Copyright IBM Corporation 1990, 2010