TCP & UDP

Adam M. Costello <amc@cs.berkeley.edu> 1999-Sep-25-Sat

IP (Internet Protocol, RFC 791):

Finds paths between hosts across multiple heterogeneous networks.
Forwards datagrams (packets) along the paths.
Datagrams may be lost, altered, duplicated, and reordered.
Normally no feedback to sender.

TCP (Transmission Control Protocol, RFC 793) provides:

Multiplexing: Port numbers allow multiple communicating endpoints per host.
Reliability: Lost/altered data is retransmitted until it arrives correctly.
Ordering: Data is delivered to the receiver in the order it was sent.
Flow control: The sender is blocked if the receiver cannot keep up.
Congestion control: Senders avoid congesting the network.

TCP segment (payload of IP datagram):

32 bits wide

source port number								destination port number
sequence number								
acknowledgement number								
data offset	reserved			URG	ACK	PSH	RST	SYN	FIN	window
checksum								urgent pointer
options								
data								

A TCP endpoint is identified by an IP address (in the IP header!) plus a port number.

A TCP connection is identified by both endpoints, so one endpoint can be involved in more than one connection (for example, an HTTP server talking to two browsers). A connection is a pair of reliable bytestreams, one in each direction. A TCP segment can contain both data for one bytestream and an acknowledgement for the other.

Setting up a connection:

A process on host S tells TCP to put a well-known port into listen mode (for example, 23 for Telnet, 25 for SMTP) and waits for connection requests.
A process on host C selects any local port (or lets TCP choose one) and requests that it be connected to the well-known port on S.
The process on host S is informed of the new connection, and typically spawns a new process to handle it, then goes back to waiting for connection requests.

Transfering data:

A sending process passes data to TCP, which sends it in segments to the other endpoint and saves a copy until it gets an acknowledgement for the data. If too much time elapses, the data is retransmitted. The receiver's window and estimates of available network resources limit how far beyond the ack number the sender may send.

When TCP receives data, it sends a cumulative acknowledgement (possibly piggybacked with outgoing data). Out-of-order data is buffered, because data is passed to the receiving process in order. Segment boundaries are not visible to the application.

Closing a connection:

Each bytestream is closed seperately. The source sends a FIN, then continues to retransmit data as necessary until all sent data has been acknowledged.

UDP (User Datagram Protocol, RFC 768) only multiplexes and detects corruption.

source port number	destination port number
length	checksum
data	

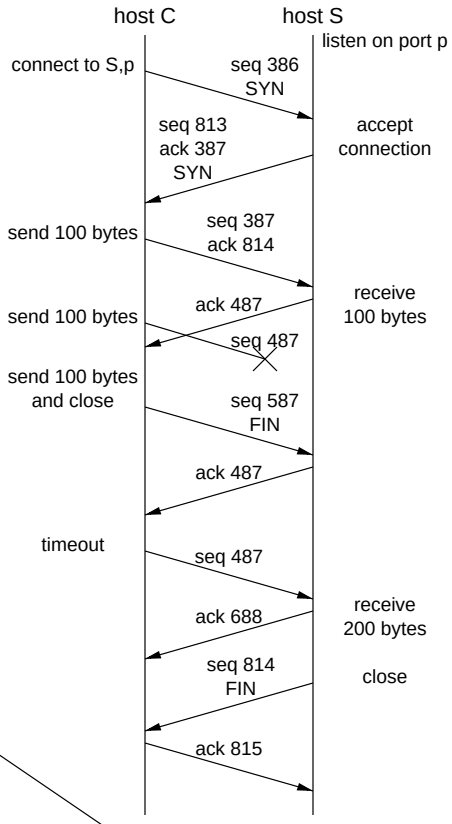
Sending process must specify destination IP address and (well-known) port number.
Source IP address can be specified to choose interface, or left up to UDP to choose.
Source port can be specified or left up to UDP to choose.

Receiving process tells UDP which combinations of <srcaddr, srcport, destaddr, destport> it is interested in, and matching datagrams are delivered to it.
<srcaddr, srcport> can be ANY, destaddr can be used to select incoming interface, or ANY.
UDP passes <srcaddr, srcport> to receiving process so it can reply.

application layer (e.g. Telnet, SMTP, HTTP)
transport layer (TCP, UDP)
network layer (IP)
data link layer (e.g. Ethernet, FDDI, PPP)

The transport layer is not quite independent of the network layer. The other layers have their own addresses for locating endpoints (like Ethernet addresses, IP addresses, email addresses, URLs), but transport endpoints are identified by an IP address and port number.

Sequence number tells where the first data byte goes in the stream.
Acknowledgement number tells where the first unreceived byte is in the reverse stream.
Data offset implies the size of the options.
ACK flag indicates that ack number is valid.
SYN flag means please start receiving.
FIN flag means I am done sending.
Window is available space for reverse stream.
Checksum detects corruption of header, data.
Some options are timestamp, selective ack.



Notice that SYN and FIN get their own sequence numbers.