Finishing up the Transport Layer

CS158a

Chris Pollett

May 7, 2007.

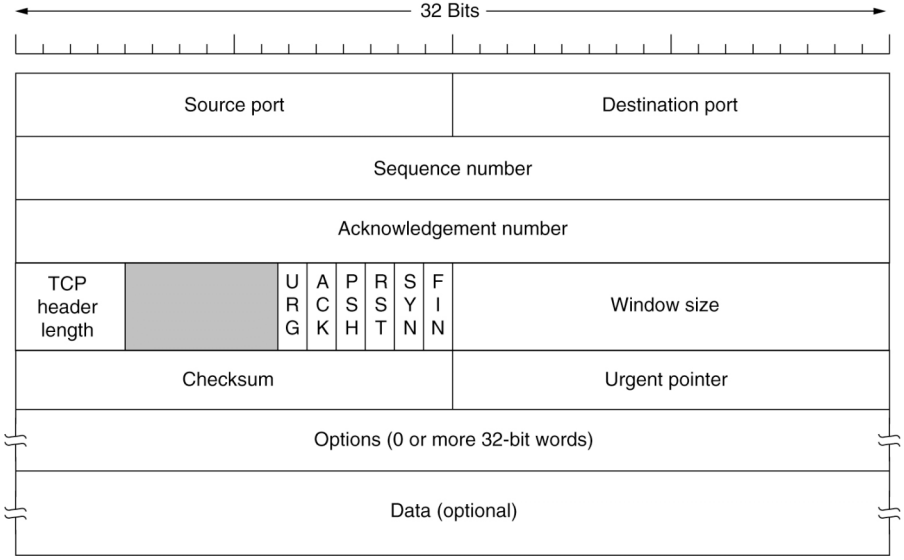
Outline

- The TCP Protocol
- The TCP Segment Header
- Connection Establishment and Release
- TCP Connection Management Modeling
- TCP Transmission Policy
- TCP Congestion Control
- TCP Timer Management

The TCP Protocol

- Sending and receiving TCP entities exchange data in the form of **segments.**
- A **TCP segment** consists of a 20 byte header plus an optional part followed by zero or more bytes of data.
- The TCP software determines how big a segment should be: it can accumulate data from several writes into one segment or split data from one write into several segments.
- Because the network layer has a maximum size of 65515, a TCP segment must be less than this.
- The basic protocol used by TCP entities is the sliding window protocol: When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiver sends back a segment with an acknowledgement. If the sender's timer goes off before it gets the acknowledgement it resends.
- Last day we said some differences between this set-up and the data link layer as window sizes can be varied depending on congestion.
- Another complication is that in TCP retransmitted data can be bundled with parts of other segments. This requires more book keeping. In particular in TCP each byte range has its own sequence number.

The TCP Segment Header



- We have already discussed what the source and destination port are for. The sequence number is as in the usual sliding window. Window size is how large this window is.
- The acknowledgement number is the next byte that the receiver is waiting to receive.
- The header length allows one to say how many optional directives are being used.
- URG is 1 if the urgent pointer is being used. The urgent pointer is used to indicate a byte offset from the current sequence number at which urgent data (like CTRL-C etc) is located.
- ACK is used to indicate the Acknowledgement number should be used.
- PSH -indicates the receiver should deliver the data without buffering.
- RST is used to reset the connection or prevent a connection from opening.
- SYN is used to open a connection. SYN=1 ACK=0 is a CONNECTION REQUEST, SYN=1 ACK=1 is a CONNECTION ACCEPTED.
- FIN bit is used to indicate the sender is finished sending.

More on the TCP Header

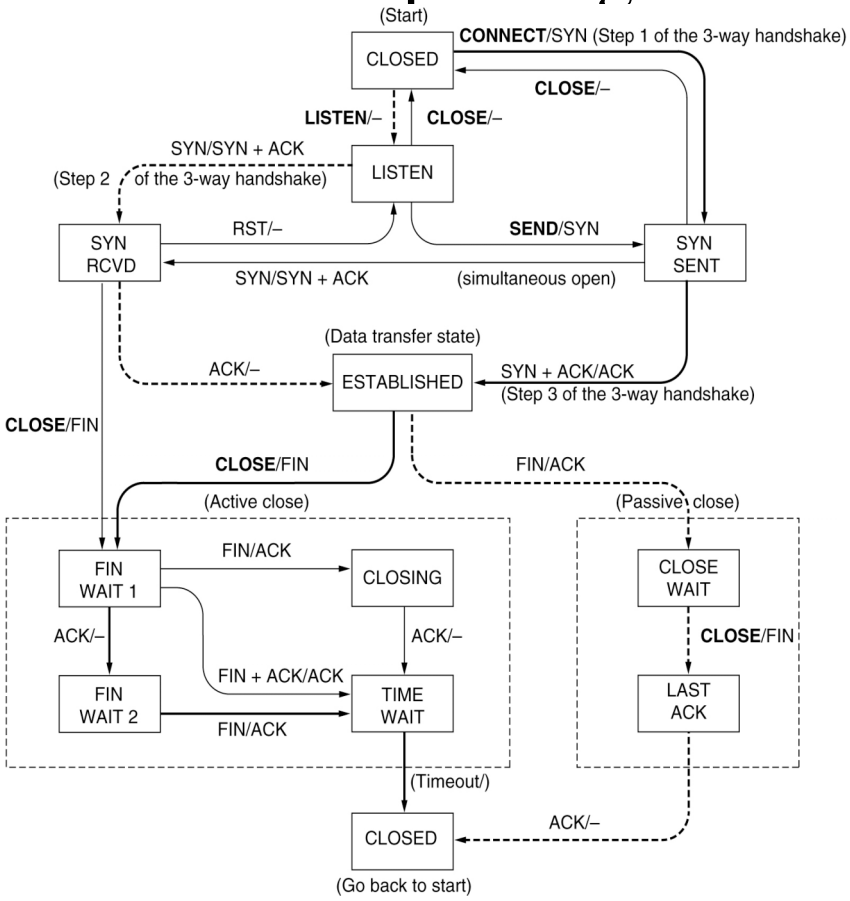
- The Options field can be used to specify larger than usual segments for especially fast connections
- It can also be used to specify larger than usual Window sizes, again for situations where the connection can handle a lot of data but there is a large round trip time delay.
- Another option says to use selective repeat rather than the default go back n.

Connection Establishment and Release

- Connections in TCP are established by means of a three way handshake:
  - The server passively LISTENs for an incoming connection
  - The client sends a TCP segment with SYN=1, ACK=0, SEQ=x as a CONNECTION REQUEST.
  - If it accepts, the server sends a SYN=1, ACK=x+1, SEQ=y segment back (if not RST=1 bit is sent), to do a CONNECTION ACKNOWLEDGEMENT.
  - Finally, the sender sends data with SEQ=x+1, ACK = y+1.
- Connection release is done using the FIN bit and each side must send a segment with FIN=1 to release.
- The release is done after the segment with the FIN is acknowledged or after two maximum packet lifetimes.

TCP Connection Management Modeling

- A connection can be modeled as a finite state machine. This is useful for proving correctness properties.



TCP Transmission Policy

- We now look at how the Window size field in the TCP header is used.
- As an example, if a receiver has a 4096-byte buffer and it correctly receives a 2048 byte buffer, it will send back ACK=2048, WIN=2048. The buffer size is 2048 until the application layer removes some data.
- If another 2048 bytes is received the next WIN would be 0, at which point the sender can no longer continue to send until a larger window is advertised.
- There are two exceptions, urgent data may be sent (say to kill the process on remote machine), and the sender can send a 1-byte header to make the receiver re-announce the next byte expected and window size.

Improving TCP Performance

- In the case of some applications such a telnet, data from the application layer might come in byte by byte.
- This might be sent by sending it with a 20byte TCP header, 20 byte IP header for 41 bytes total. Together with the acknowledgement, this is very wasteful.
- One way to reduce this is to use **Nagle's Algorithm**: When data come into the sender one byte at a time, sender the first byte and buffer all the rest until an acknowledgement comes. Then send all the buffered characters in one TCP segment and start buffering again. If more than half of a maximum segment of data has been buffered one can also send.
- For typing Nagle's algorithm greatly reduces the bandwidth used. For some application the algorithm can't be used such as X Windows.

TCP Congestion Control

- We now look at techniques for dealing with network congestion at the TCP level.
- There are two possible bottlenecks to transmission: network capacity and receiver capacity.
- Each sender maintains two windows the window the receiver has granted and the window the for the network called the **congestion window.**
- The sender can at most send the minimum of the two window sizes.
- To maintain the congestion window a technique called **slow start is** used.
- When a connection is established the sender initializes the congestion window to the size of the maximum segment in use on the connection.
- It then sends one segment, if this segment is ACK'd before the timer goes off the congestion window is increased by 1 one maximum segment size. (Each future ACK within time limit doubles the window size.)
- When the timer goes off before an ACK the congestion window is no longer doubled.
- When a timeout occurs a second parameter called the **threshold** is set to half the current congestion window, and the congestion window is reset to one maximum segment, doubling starts again from there but when the threshold is reach, the window is grown linearly by one max segment to find the correct size.

TCP Timer Management

- TCP uses multiple timers.
- The most important is the **retransmission timer** which is started when a segment is sent and stopped when an acknowledgement arrives.
- If the timer goes off before an ack, then it the segment is resent.
- The question is how long should the timeout interval be?
- This is harder than in the data link layer because there is more variation in the round-trip times.
- TCP maintains a variable RTT that is the current best estimate of the roundtrip time. Acknowledged segments update this value according to:

  $RTT_{new} = a*RTT_{old} +(1-a)M$ where a is usually around 7/8 and M is the round trip time of the last acknowledged segment.

  The timeout period is then set to b*RTT where b is usually 2.