CLASSICAL TCP IMPROVEMENTS

TCP Connection Creation

- Programming details later – for now we are concerned with the actual communication.
- A server accepts a connection.
- Must be looking for new connections!
- A client requests a connection.
- Must know where the server is!

Client Starts

- A client starts by sending a SYN segment with the following information:
- Client's ISN (generated pseudo-randomly)
- Maximum Receive Window for client.
- Optionally (but usually) MSS (largest datagram accepted).
- No payload! (Only TCP headers)

Server Response

- When a waiting server sees a new connection request, the server sends back a SYN segment with:
- Server's ISN (generated pseudo-randomly)
- Request Number is Client ISN+1
- Maximum Receive Window for server.
- Optionally (but usually) MSS
- No payload! (Only TCP headers)
- When the Server's SYN is received, the client sends back an ACK with:
- Acknowledgment Number is Server's ISN+1

TCP 3-way handshake

- Client: “I want to talk, and I’m starting with byte number X”.
- Server: “OK, I’m here and I’ll talk. My first byte will be called number Y, and I know your first byte will be number X+1”.
- Client: “Got it – you start at byte number Y+1”.
- Bill: “Monica, I’m afraid I’ll syn and byte your ack”

TCP Data and ACK

- Once the connection is established, data can be sent.
- Each data segment includes a sequence number identifying the first byte in the segment.
- Each segment (data or empty) includes a request number indicating what data has been received

Buffering

- Keep in mind that TCP is part of the Operating System. The O.S. takes care of all these details asynchronously.
- The TCP layer doesn’t know when the application will ask for any received data.
- TCP buffers incoming data so it’s ready when we ask for it.

TCP Buffers

- Both the client and server allocate buffers to hold incoming and outgoing data
- The TCP layer does this.
- Both the client and server announce with every ACK how much buffer space remains (the Window field in a TCP segment).

Send Buffers

- The application gives the TCP layer some data to send.
- The data is put in a send buffer, where it stays until the data is ACK’d.
- The TCP layer won’t accept data from the application unless (or until) there is buffer space.

ACKs

- A receiver doesn't have to ACK every segment (it can ACK many segments with a single ACK segment).
- Each ACK can also contain outgoing data (piggybacking).
- If a sender doesn't get an ACK after some time limit, it resends the data.

TCP Segment Order

- Most TCP implementations will accept out-of-order segments (if there is room in the buffer).
- Once the missing segments arrive, a single ACK can be sent for the whole thing.
- Remember: IP delivers TCP segments, and IP is not reliable – IP datagrams can be lost or arrive out of order.

Termination

- The TCP layer can send a RST segment that terminates a connection if something is wrong.
- Usually the application tells TCP to terminate the connection politely with a FIN segment.

TCP Sockets Programming

- Creating a passive mode (server) socket.
- Establishing an application-level connection.
- Sending/receiving data.
- Terminating a connection.

Establishing a passive mode TCP socket

Passive mode:

- Address already determined.
- Tell the kernel to accept incoming connection requests directed at the socket address.

  3-way handshake
  - Tell the kernel to queue incoming connections for us.

Accepting an incoming connection

- Once we start listening on a socket, the O.S. will queue incoming connections
    - Handles the 3-way handshake
    - Queues up multiple connections.
- When our application is ready to handle a new connection, we need to ask the O.S. for the next connection.

Terminating a TCP connection

- Either end of the connection can call the close() system call.
- If the other end has closed the connection, and there is no buffered data, reading from a TCP socket returns 0 to indicate EOF

Client Code

- TCP clients can connect to a server, which:
- takes care of establishing an endpoint address for the client socket.
- don't need to call bind first, the O.S. will take care of assigning the local endpoint address (TCP port number, IP address).
- Attempts to establish a connection to the specified server.
    - 3-way handshake

Reading from a TCP socket

- By default read() will block until data is available.
- Reading from a TCP socket may return less than max bytes (whatever is available).
- You must be prepared to read data 1 byte at a time!

Source : https://tutor4cs.wordpress.com/2013/03/01/mobile-computing-lecture-notes-for-unit-5/