

TCP Details

Netprog: TCP Details

1

TCP Lingo

- When a client requests a connection, it sends a “SYN” segment (a special TCP segment) to the server port.
- SYN stands for *synchronize*. The SYN message includes the client’s ISN.
- ISN is Initial Sequence Number.

Netprog: TCP Details

2

More...

- Every TCP segment includes a *Sequence Number* that refers to the first byte of *data* included in the segment.
- Every TCP segment includes a *Request Number (Acknowledgement Number)* that indicates the byte number of the next data that is expected to be received.
 - All bytes up through this number have already been received.

Netprog: TCP Details

3

And more...

- There are a bunch of control flags:
 - URG: urgent data included.
 - ACK: this segment is (among other things) an acknowledgement.
 - RST: error - abort the session.
 - SYN: synchronize Sequence Numbers (setup)
 - FIN: polite connection termination.

Netprog: TCP Details

4

And more...

- MSS: Maximum segment size (A TCP option)
- Window: Every ACK includes a Window field that tells the sender how many bytes it can send before the receiver will have to toss it away (due to fixed buffer size).

Netprog: TCP Details

5

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!

Netprog: TCP Details

6

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)

Netprog: TCP Details

7

Sever 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)

Netprog: TCP Details

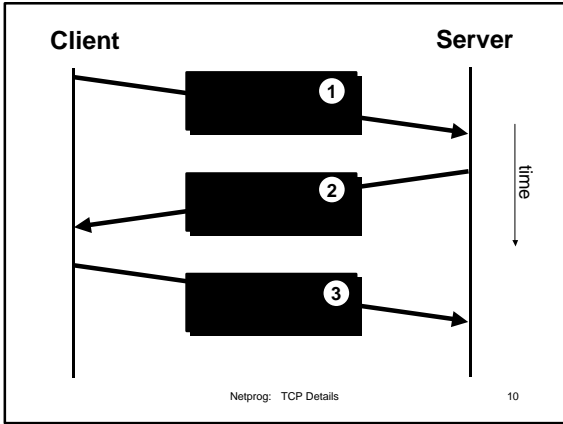
8

Finally

- When the Server's SYN is received, the client sends back an ACK with:
 - Request Number is Server's ISN+1

Netprog: TCP Details

9



TCP 3-way handshake

- ① Client: "I want to talk, and I'm starting with byte number $X+1$ ".
- ② Server: "OK, I'm here and I'll talk. My first byte will be called number $Y+1$, and I know your first byte will be number $X+1$ ".
- ③ Client: "Got it - you start at byte number $Y+1$ ".
- ④ Bill: "Monica, I'm afraid I'll syn and byte your ack"

Netprog: TCP Details 11

Why 3-Way?

- Why is the third message necessary?
- HINTS:
 - TCP is a reliable service.
 - IP delivers each TCP segment.
 - IP is not reliable.

Netprog: TCP Details 12

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.

Netprog: TCP Details

13

Buffering

- Keep in mind that TCP is (usually) part of the Operating System. It 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.

Netprog: TCP Details

14

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).

Netprog: TCP Details

15

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.
 - it has to stay, as it might need to be sent again!
- The TCP layer won't accept data from the application unless (or until) there is buffer space.

Netprog: TCP Details

16

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 (MSL) it resends the data.

Netprog: TCP Details

17

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.

Netprog: TCP Details

18

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.

Netprog: TCP Details

19

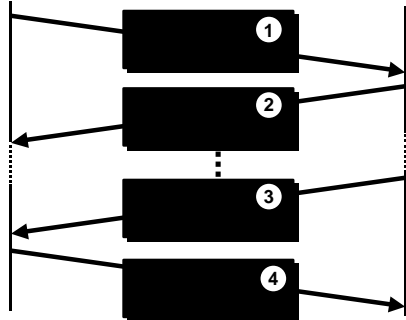
FIN

- Either end of the connection can initiate termination.
- A FIN is sent, which means the application is done sending data.
- The FIN is ACK'd.
- The other end must now send a FIN.
- That FIN must be ACK'd.

Netprog: TCP Details

20

App1 App2



Netprog: TCP Details

21

TCP Termination

- ① App1: "I have no more data for you".
- ② App2: "OK, I understand you are done sending."
dramatic pause...
- ③ App2: "OK - Now I'm also done sending data".
- ④ App1: "Roger, Over and Out, Goodbye,
Astalavista Baby, Adios, It's been real ..."
camera fades to black ...

Netprog: TCP Details

22

TCP TIME_WAIT

- Once a TCP connection has been terminated (the last ACK sent) there is some unfinished business:
 - What if the ACK is lost? The last FIN will be resent and it must be ACK'd.
 - What if there are lost or duplicated segments that finally reach the destination after a long delay?
- TCP hangs out for a while to handle these situations.

Netprog: TCP Details

23

Test Questions

- Why is a 3-way handshake necessary?
- Who sends the first FIN - the server or the client?
- Once the connection is established, what is the difference between the operation of the server's TCP layer and the client's TCP layer?
- What happens if a *bad guy* can guess ISNs?

Netprog: TCP Details

24
