CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

TCP Connections

TCP is connection-oriented protocol

This requires some overhead to set up and tear down connection

Understanding connection-establishment process important for understanding some efficiency and security issues

Connection establishment happens in three steps

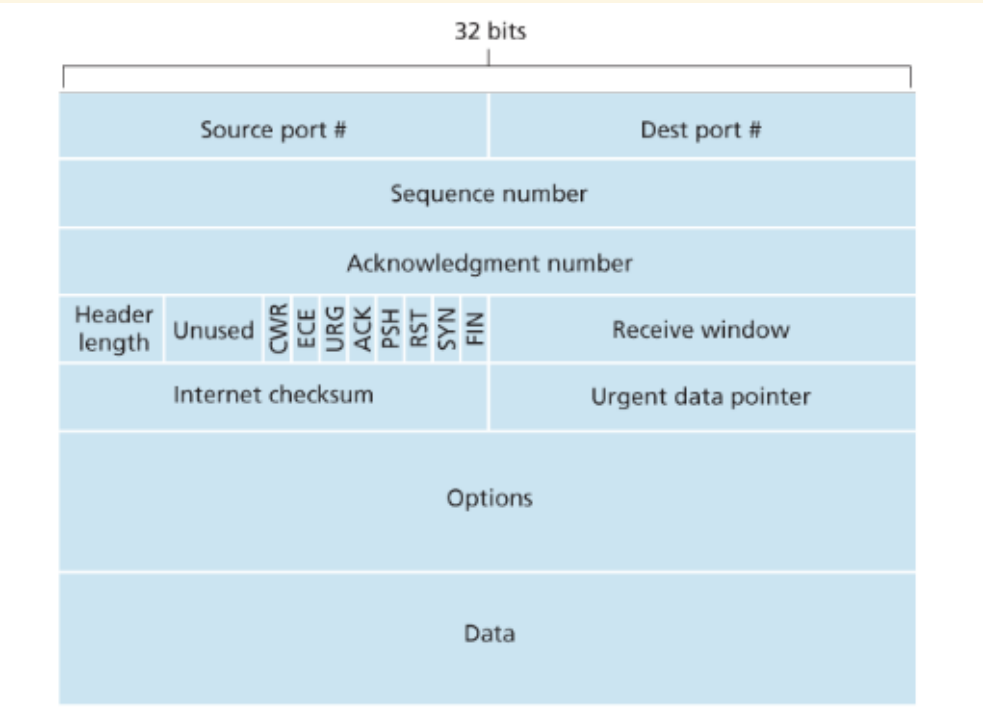
First step

Triggering event: Application decides it needs TCP connection

TCP client sends special segment to TCP server

Initial segment contains no application-layer data

Has special SYN flag set -- hence the name **SYN segment**


32 bits
Source port #
Dest port #
Sequence number
Acknowledgment number
Header length
Unused
CWR
ECE
URG
ACK
PSH
RST
SYN
FIN
Receive window
Internet checksum
Urgent data pointer
Options
Data

First step

Rather than use 0, client chooses random initial sequence number (which we'll denote `client_isn`)

This randomness is done for security

As always, TCP segment encapsulated in IP datagram and sent

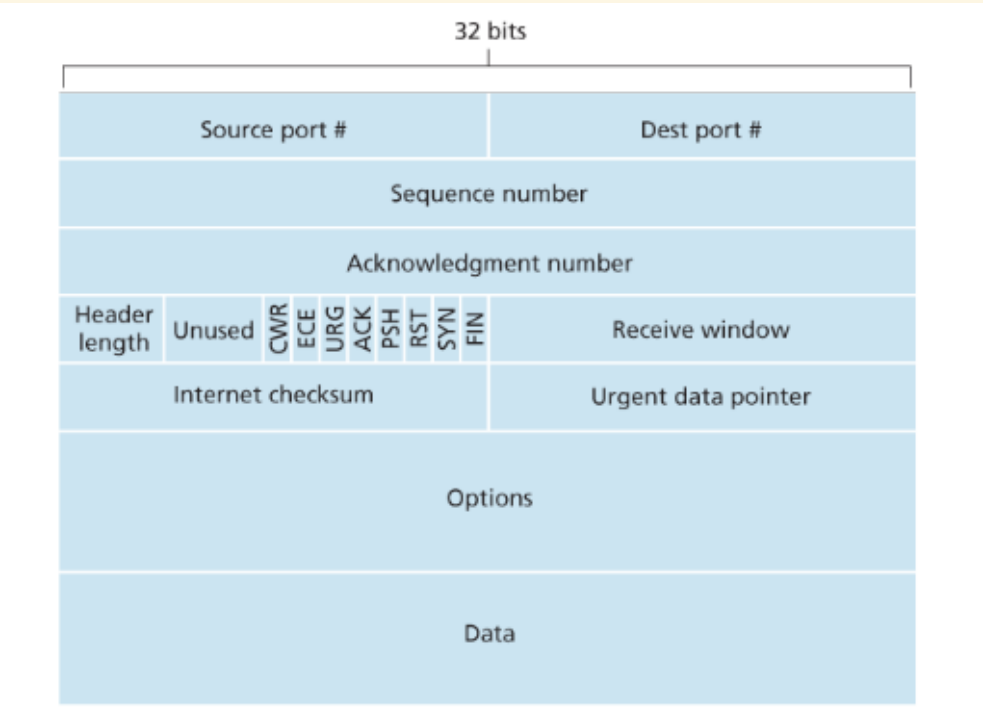
Second step

Triggering event: Server receives SYN segment from client

Server allocates buffers and variables necessary to maintain connection

Once buffers established, server replies with another special segment to let client know connection has been granted

Reply also has SYN set, and includes ACK number `client_isn + 1`


32 bits
Source port #
Dest port #
Sequence number
Acknowledgment number
Header length
Unused
CWR
ECE
URG
ACK
PSH
RST
SYN
FIN
Receive window
Internet checksum
Urgent data pointer
Options
Data

Second step

Unsurprisingly, this segment is called SYNACK segment

SYNACK segment, like SYN, cannot contain application-layer data

Server also chooses random initial sequence number, `server_icn`, for its segments

Third step

Triggering event: Client receives SYNACK segment

Client allocates buffers and variables for connection

Client sends ACK for SYNACK segment

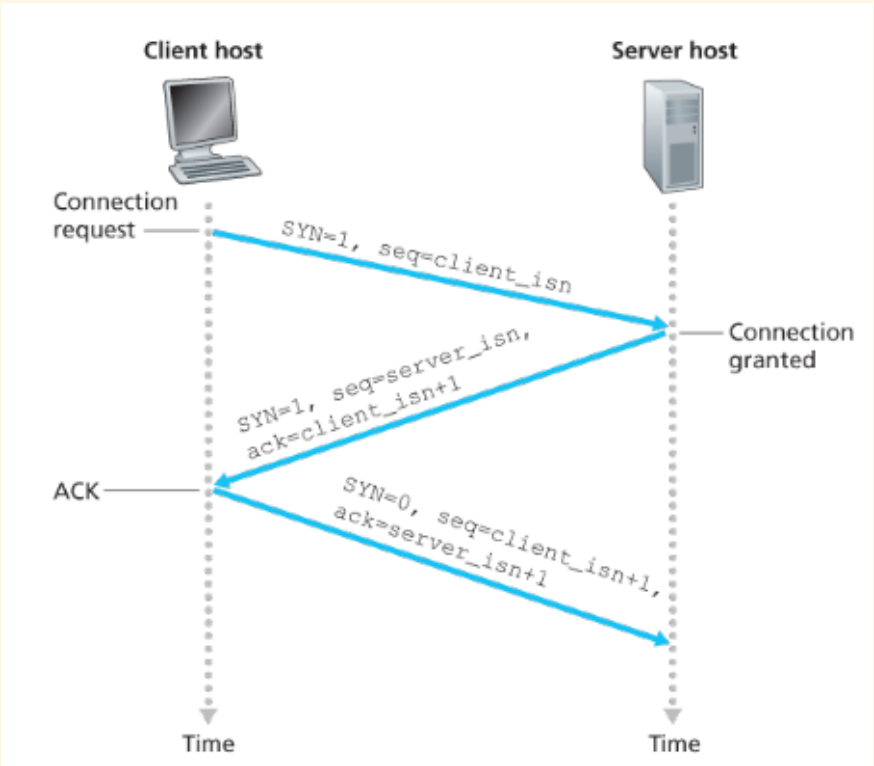
Third step

New ACK does *not* have SYN flag because connection already established

ACK contains appropriate ACK number:

```
server_isn + 1
```

Even though it is considered part of connection setup, this ACK segment is first one that can carry application data


Client host
Server host
Connection request
SYN=1, seq=client_isn
Connection granted
SYN=1, seq=server_isn, ack=client_isn+1
ACK
SYN=0, seq=client_isn+1, ack=server_isn+1
Time
Time

Because three messages are sent, this is known as **three-way handshake**

TCP connections must also be ended when communication is done so resources can be recovered

Tearing down TCP connections is also multi-step process

Occurs when application calls `close`

Either client or server can initiate tear-down -- we will assume client initiated in our examples

Closing also initiated by sending special segment

In this case, FIN bit set in header

Server must ACK segment, but does not need to set FIN bit, to close communication in client -> server direction

Connection must also be closed in server -> client direction, and is done in same way

Server sends FIN packet, which client then ACKs

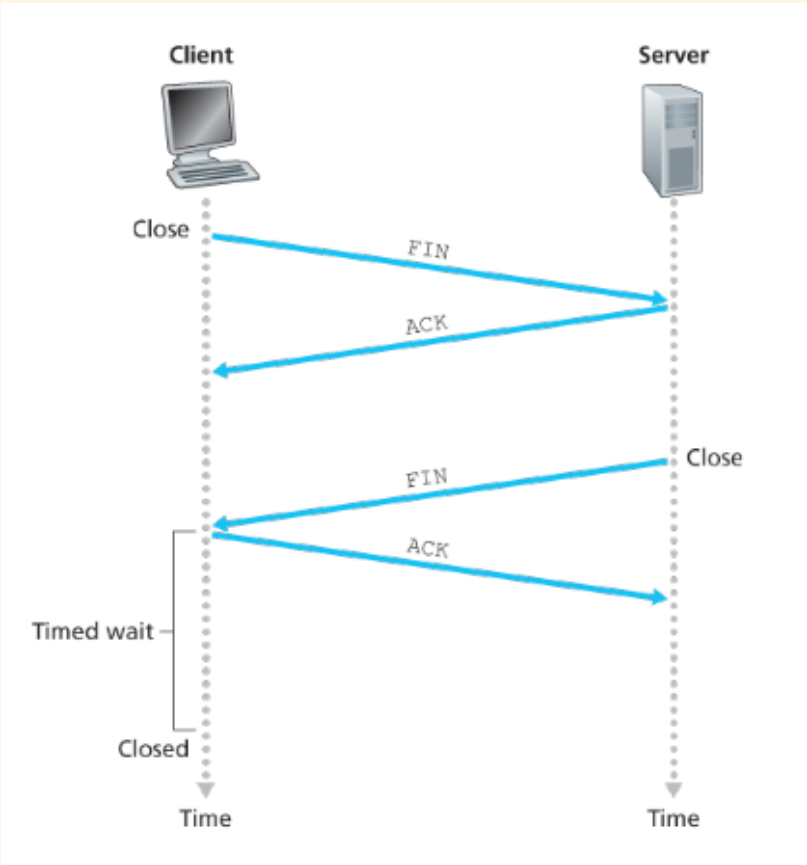
Remember that client and server could be interchanged in our examples

Once both FINs have been ACK'd, brief timeout ensues at last ACK sender (client, in our case) to ensure ACK was not lost

Once client confident that ACK was not lost, which would have resulted in server resending FIN, connection finally closed

Typical wait time somewhere in 30 second to 2 minute range

Once connection closed, resources can be reclaimed

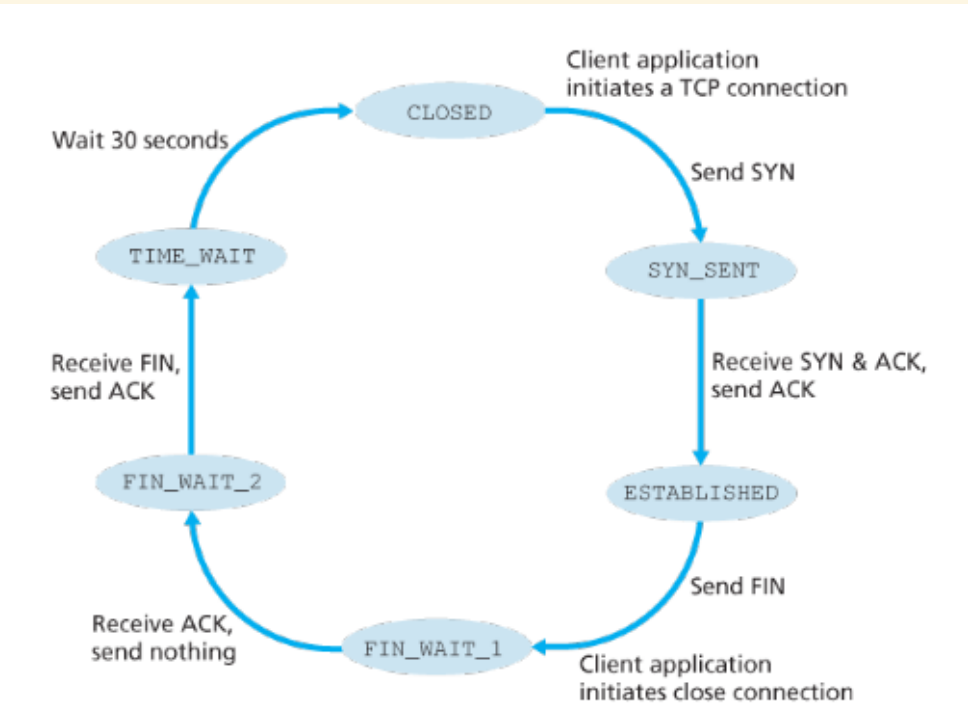

Client
Server
Close
FIN
ACK
Close
FIN
ACK
Timed wait
Closed
Time
Time

Like other protocols, often helpful to think of TCP in terms of *states*

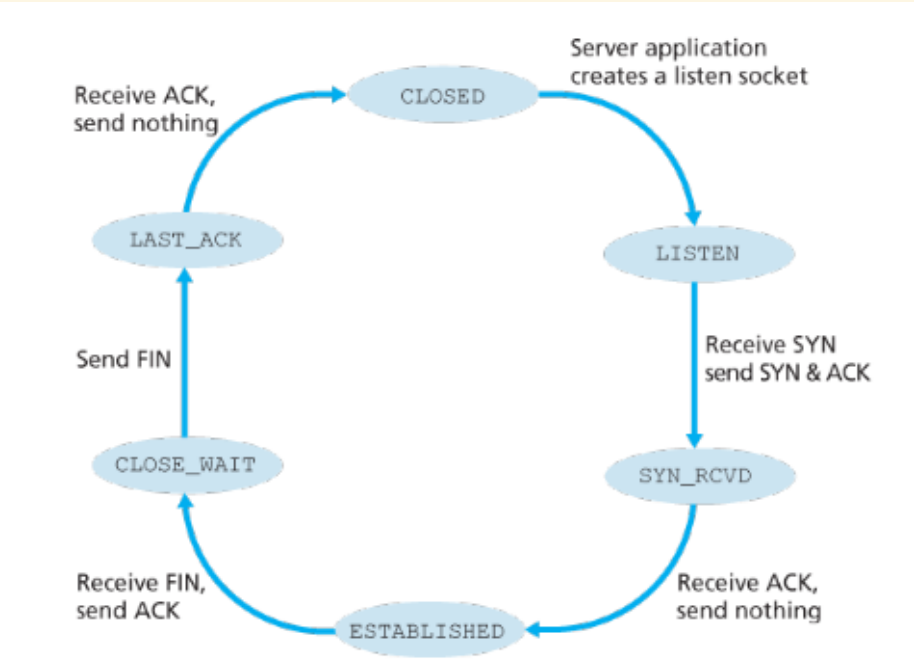
For event-driven programs, finite-state machines can be helpful to model (and code) behavior

Full FSMs for TCP not given here, but book has basic example of states TCP goes through under normal operation

Sender


Client application
initiates a TCP connection
CLOSED
Send SYN
SYN_SENT
Receive SYN & ACK,
send ACK
ESTABLISHED
Send FIN
Client application
initiates close connection
FIN_WAIT_1
Receive ACK,
send nothing
FIN_WAIT_2
Receive FIN,
send ACK
TIME_WAIT
Wait 30 seconds

Receiver


Server application
creates a listen socket
CLOSED
Receive ACK,
send nothing
LAST_ACK
LISTEN
Send FIN
Receive SYN
send SYN & ACK
CLOSE_WAIT
SYN_RCVD
Receive FIN,
send ACK
ESTABLISHED
Receive ACK,
send nothing

We do consider one "special case" where TCP connection fails:

What happens if client tries to connect to port where nothing is listening?

Another special TCP packet used to reply

TCP reset message has RST bit set in header

Essentially tells client nothing is listening on that port and to please not ask again

If reset not sent, client might assume SYN packet was lost

Reset may not be sent in some cases

Firewall between sender and receiver could be set up to drop any packets to certain ports, so packets never even reach receiver

This is helpful for security