

Connection Management

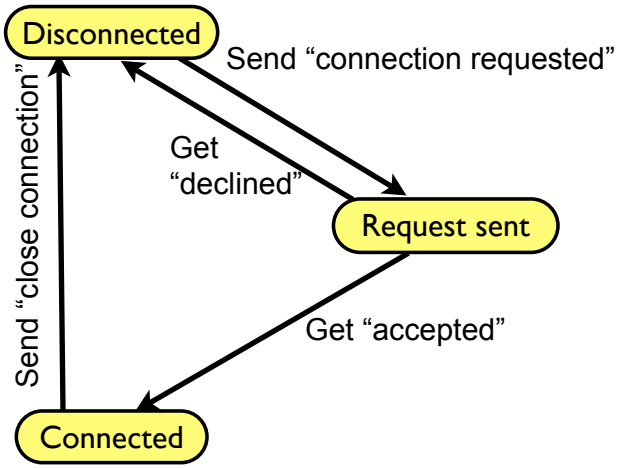
Networked Systems 3
Lecture 13

Lecture Outline

- Modelling protocol behaviour
- Managing transport connections
  - Connection establishment
  - Reliable data transfer
  - Connection tear down

Modelling Protocol Behaviour

- Can model protocols using a *finite state machine*
  - A set of *states* with *transitions* between them
  - The current state indicates what the system is doing at any time
  - Transitions show occurrence of events and the response of the system
- Can be used to define the behaviour of a protocol



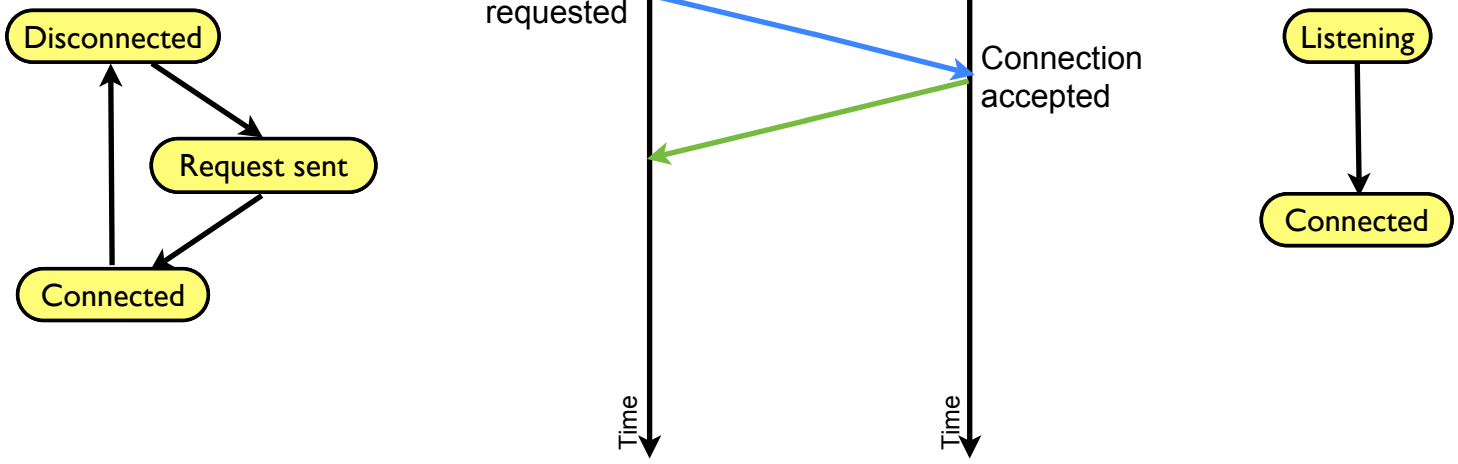
Example: partial state machine for opening and closing a connection

Managing Connections

- One role of transport layer: provide reliability
- How to reliably manage transport connections?
  - Setup a reliable connection over an unreliable connectionless network
  - Transport data without loss over an unreliable network
  - Agree to tear down a connection

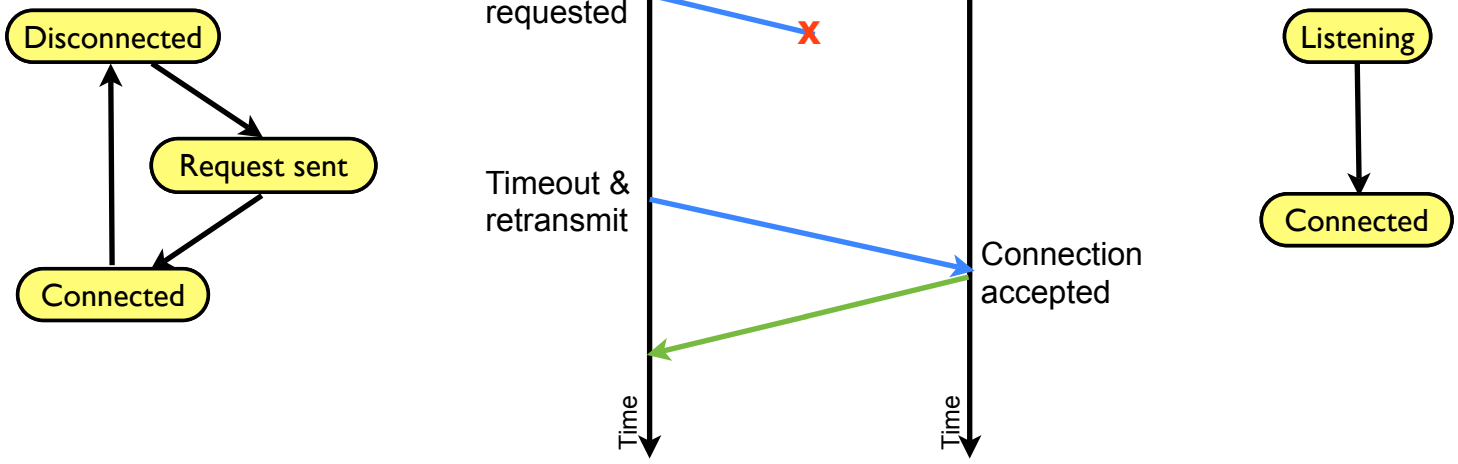
Connection Establishment

- How do two hosts agree to communicate?



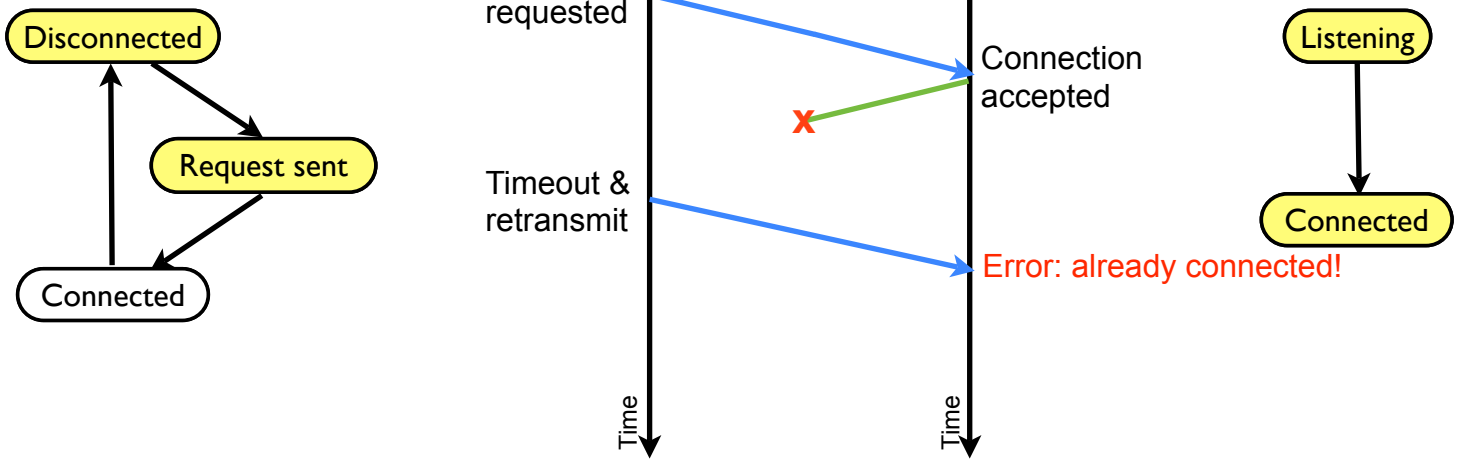
Connection Establishment

- What if the initial request is lost?



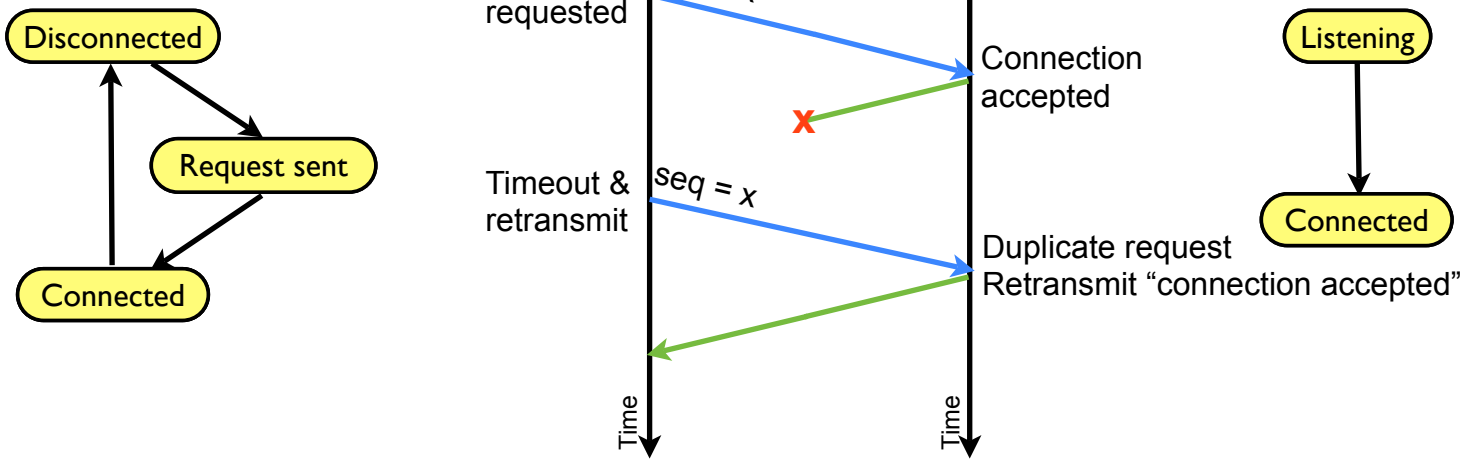
Connection Establishment

- What if the “connection accepted” reply is lost?



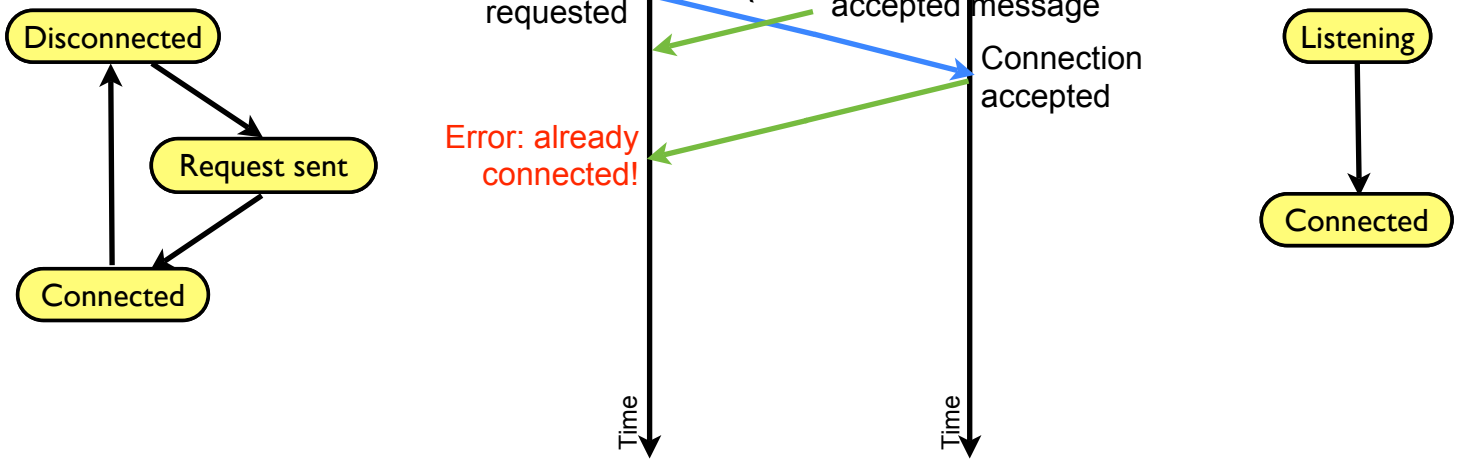
Connection Establishment

- What if the “connection accepted” reply is lost?
  - Sequence number in messages; random initial value



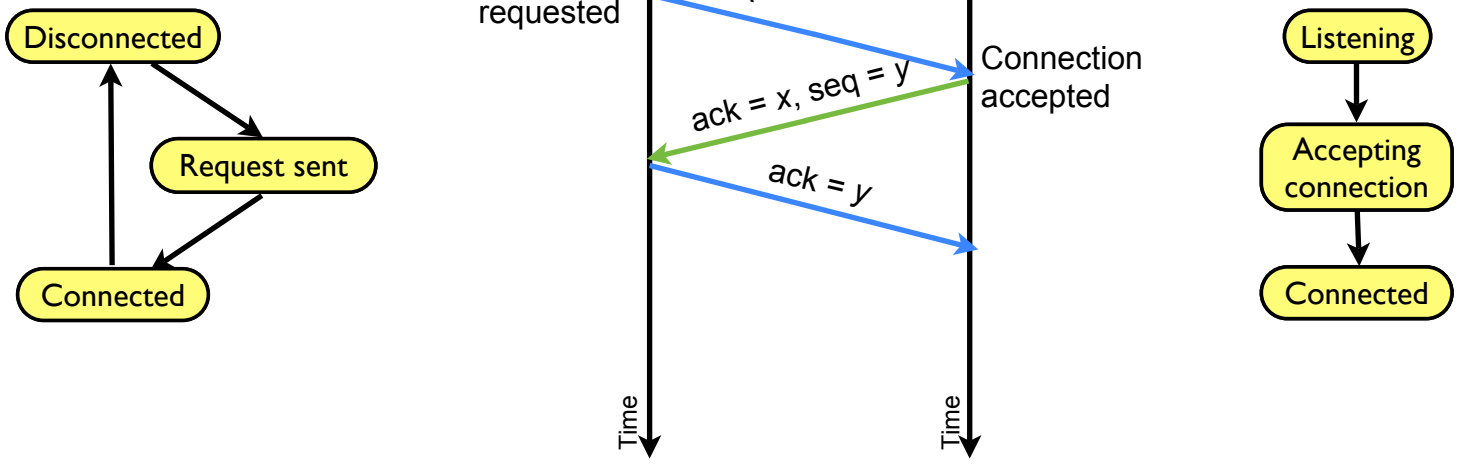
Connection Establishment

- What if data from an old connection is still in the network?



Connection Establishment

- Solution for robust connection establishment: use a three-way handshake

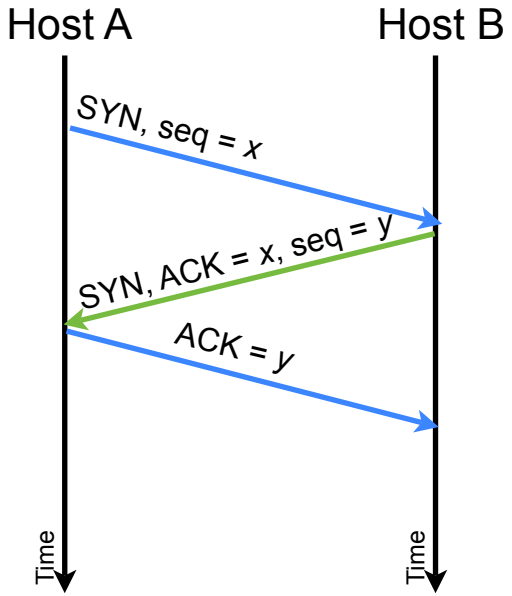


Three Way Handshake

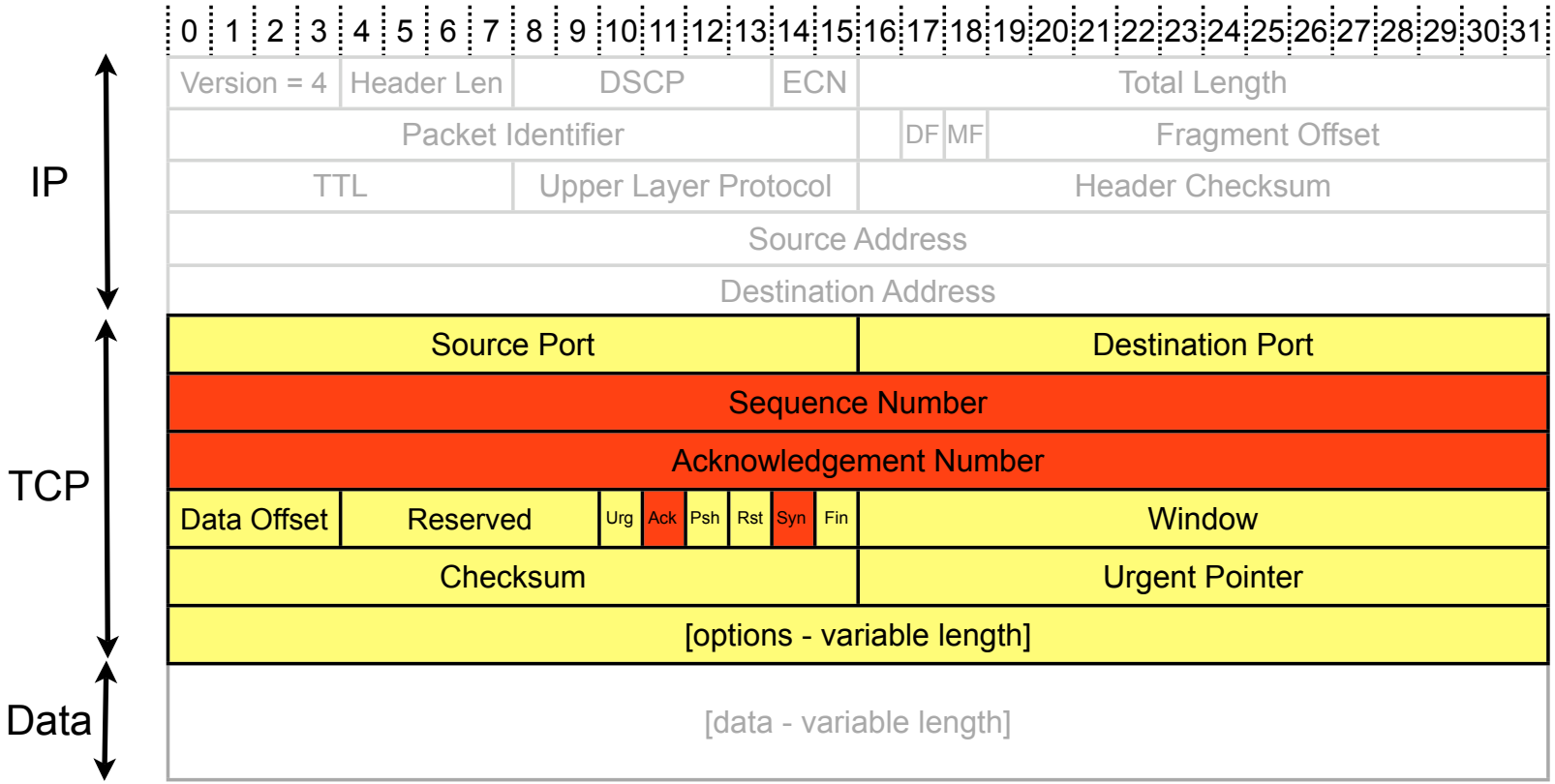
- Three way handshake ensures robustness
  - Delayed control messages generate an acknowledgement with incorrect sequence number
    - This is detected, and stops the connection establishment
  - Hosts cannot reuse initial sequence number until the *maximum packet lifetime* passed
    - Requires hosts to keep state regarding previous connections, to avoid reuse
    - Randomly chosen initial sequence number makes collisions unlikely if a host crash causes state to be lost

Example: TCP Connections

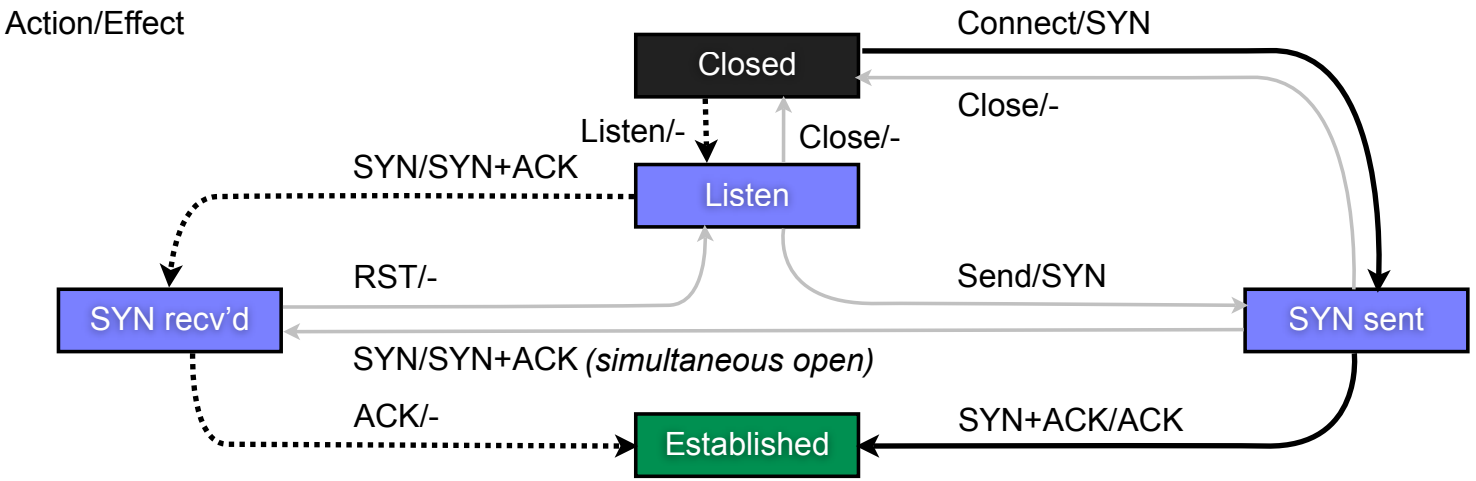
- TCP connections use a three way handshake
  - Use the SYN and ACK flags in the TCP header to signal connection progress
  - Packets contain sequence number and acknowledgement number



Example: TCP Connections



TCP State Machine



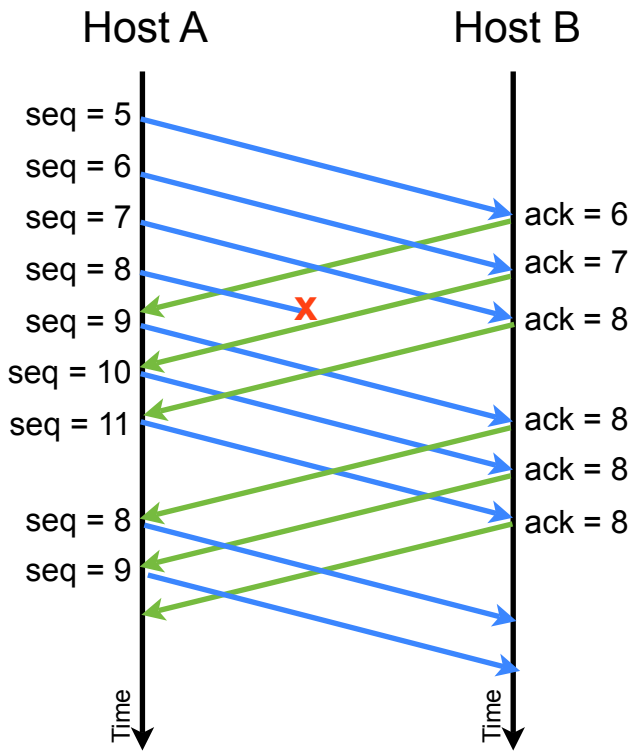
- Shows client and server on same state diagram
- Additional transitions allow simultaneous open

Reliable Data Transfer

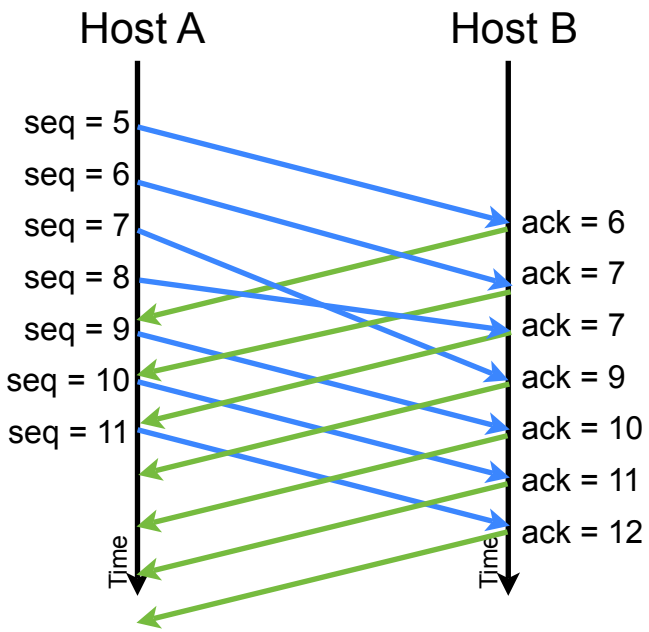
- Two approaches to reliable data transfer at the transport layer
  - End-to-end ARQ
    - Positive or negative acknowledgements
  - End-to-end FEC
    - Within each network layer packet
    - Across several network layer packets
  - Conceptually identical to operation at data link layer

Example: TCP

- TCP packets include sequence number and an (optional) acknowledgement number
  - Sequence number counts bytes transmitted
  - Send cumulative positive acknowledgements
    - Acknowledgement specifies the *next byte expected*
    - Only acknowledge contiguously data packets (sliding window protocol, so several data packets in flight)
  - Duplicate acknowledgement implies loss
  - Retransmit lost packets



Example: TCP



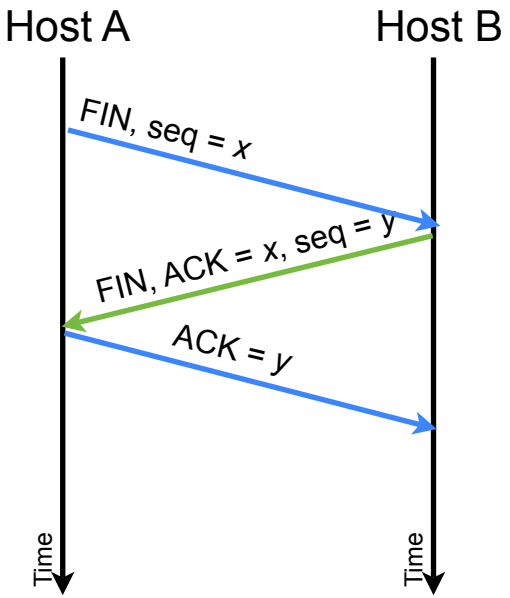
- Packet reordering also causes duplicate ACKs
  - Gives appearance of loss, when the data was merely delayed
- TCP uses *triple duplicate ACK* to indicate loss
  - Three identical ACKs in a row
  - Slightly delays response to loss, but makes TCP more robust to reordering

Example: TCP

- Problem with cumulative ACKs: don't signal any packets received after the highest contiguously received packet
  - E.g. if packets 1, 2, 3, 5, 6, and 7 are received, the ACK will show 4 as the next packet outstanding, but won't mention packets 5, 6, and 7
  - Leads to unnecessary retransmissions
  - Solution: Selective ACK ("SACK") option to TCP
    - But only supported by 68% of web servers, ten years after standardisation... [Medina, Allman, & Floyd, ACM CCR, Apr 2005]

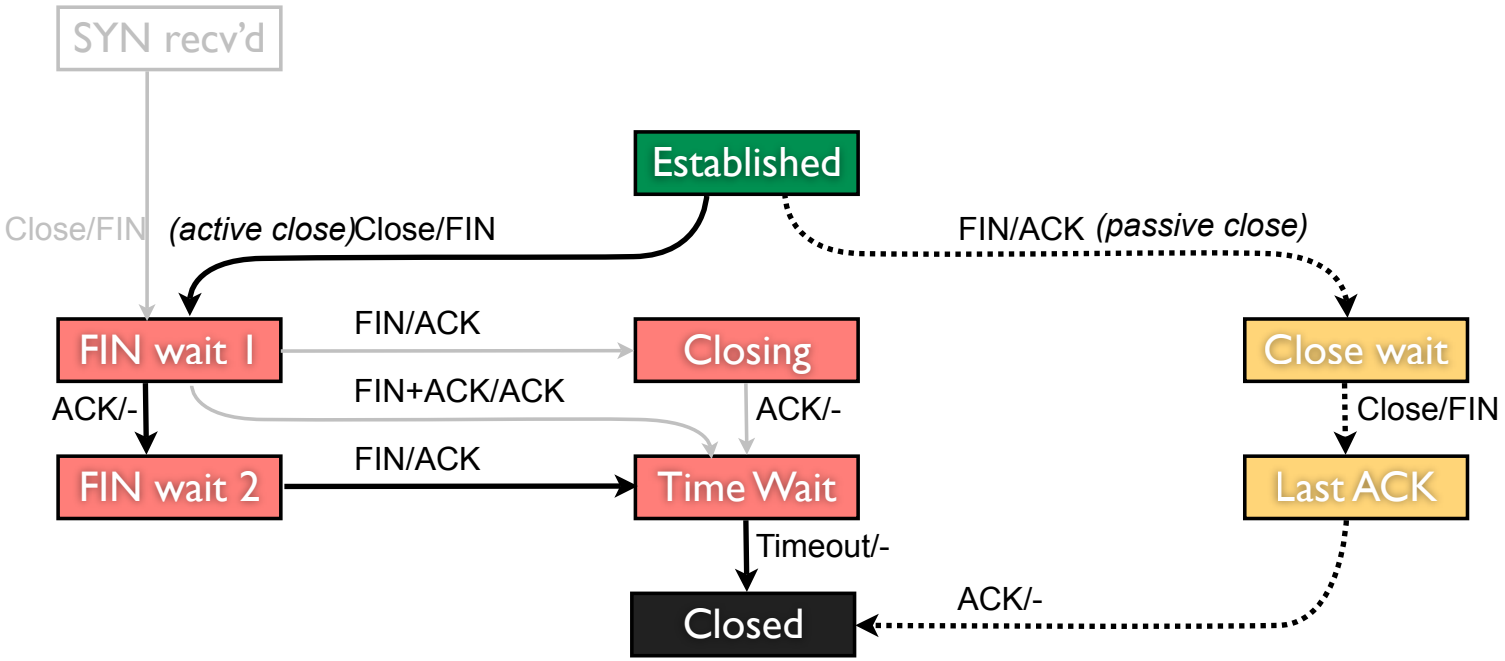
Connection Tear Down

- Three way handshake to tear down a connection
- What happens if the last ACK is lost?
  - A has closed the connection, so cannot resend the ACK; B is still waiting
  - Unavoidable problem → B must eventually give up, without knowing if the last packet arrived
  - Data sent on last packet is potentially lost



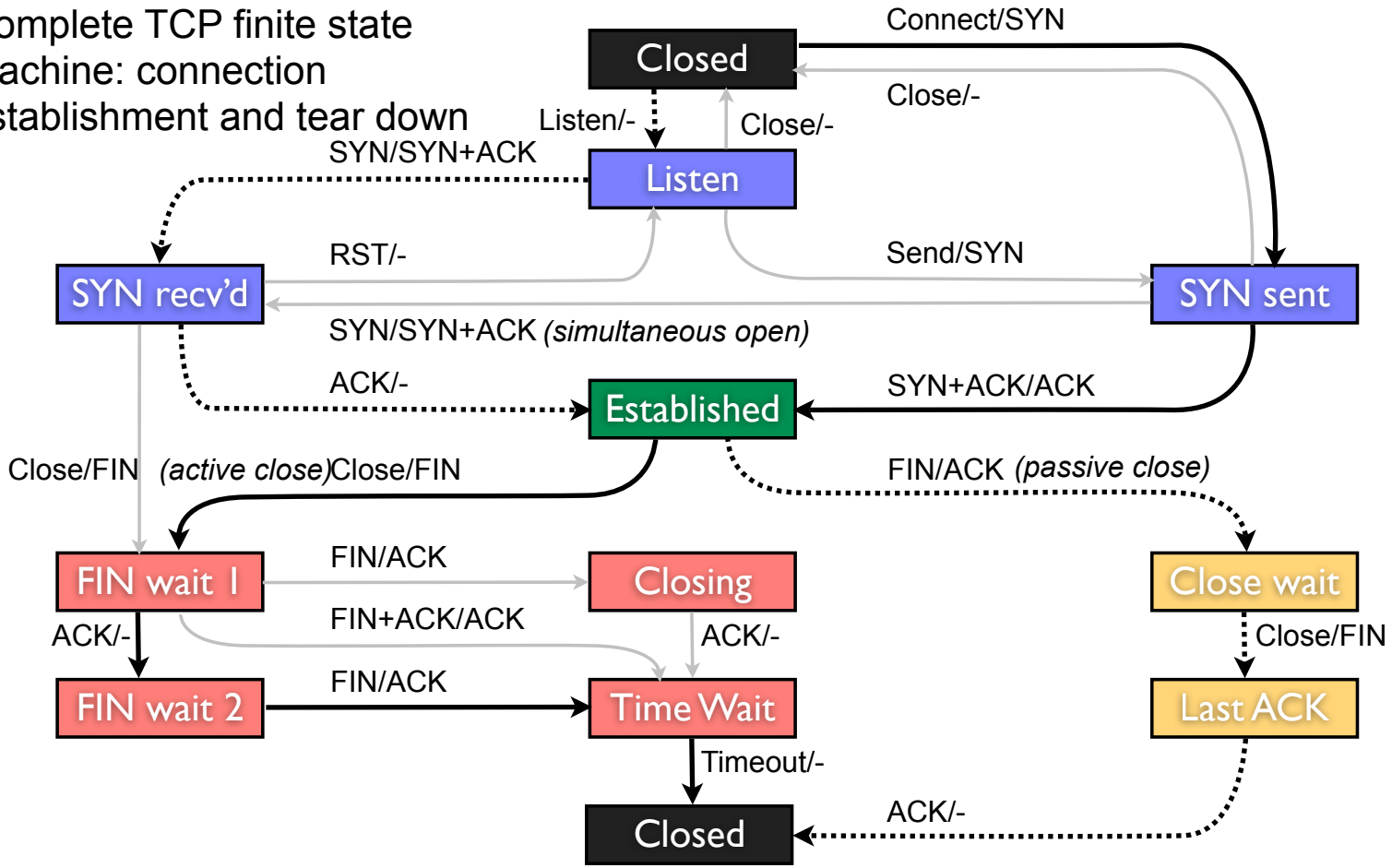
TCP State Machine

- TCP uses a three way handshake to close connection
- Signalled by the FIN bit in the packet header



TCP State Machine

Complete TCP finite state machine: connection establishment and tear down



Note: RFC 793 has a mistake in this diagram

TCP Connection Progress

```
Open connection:
192.168.0.4.49159 > 130.209.240.1.80: S 1033471698:1033471698(0) win 65535
130.209.240.1.80 > 192.168.0.4.49159: S 3518203430:3518203430(0) ack 1033471699 win 5792
192.168.0.4.49159 > 130.209.240.1.80: . ack 3518203431 win 65535
Send "GET /index.html HTTP/1.1":
192.168.0.4.49159 > 130.209.240.1.80: P 1033471699:1033471725(26) ack 3518203431 win 65535
130.209.240.1.80 > 192.168.0.4.49159: . ack 1033471725 win 5792
Send "Host: www.dcs.gla.ac.uk":
192.168.0.4.49159 > 130.209.240.1.80: P 1033471725:1033471746(21) ack 3518203431 win 65535
130.209.240.1.80 > 192.168.0.4.49159: . ack 1033471746 win 5792
Send blank line:
192.168.0.4.49159 > 130.209.240.1.80: P 1033471746:1033471748(2) ack 3518203431 win 65535
130.209.240.1.80 > 192.168.0.4.49159: . ack 1033471748 win 5792
Receive web page:
130.209.240.1.80 > 192.168.0.4.49159: . 3518203431:3518204879(1448) ack 1033471748 win 5792
130.209.240.1.80 > 192.168.0.4.49159: . 3518204879:3518206327(1448) ack 1033471748 win 5792
192.168.0.4.49159 > 130.209.240.1.80: . ack 3518206327 win 65160
Server closes connection:
130.209.240.1.80 > 192.168.0.4.49159: FP 3518206327:3518207344(1017) ack 1033471748 win 5792
192.168.0.4.49159 > 130.209.240.1.80: . ack 3518207345 win 64143
130.209.240.1.80 > 192.168.0.4.49159: . ack 1033471748 win 5792
192.168.0.4.49159 > 130.209.240.1.80: F 1033471748:1033471748(0) ack 3518207345 win 65535
130.209.240.1.80 > 192.168.0.4.49159: . ack 1033471749 win 5792
```

Questions?