TCP Connection Management

Created by

M Bateman, A Ruddle & C Allison

As part of the TCP View project

TCP

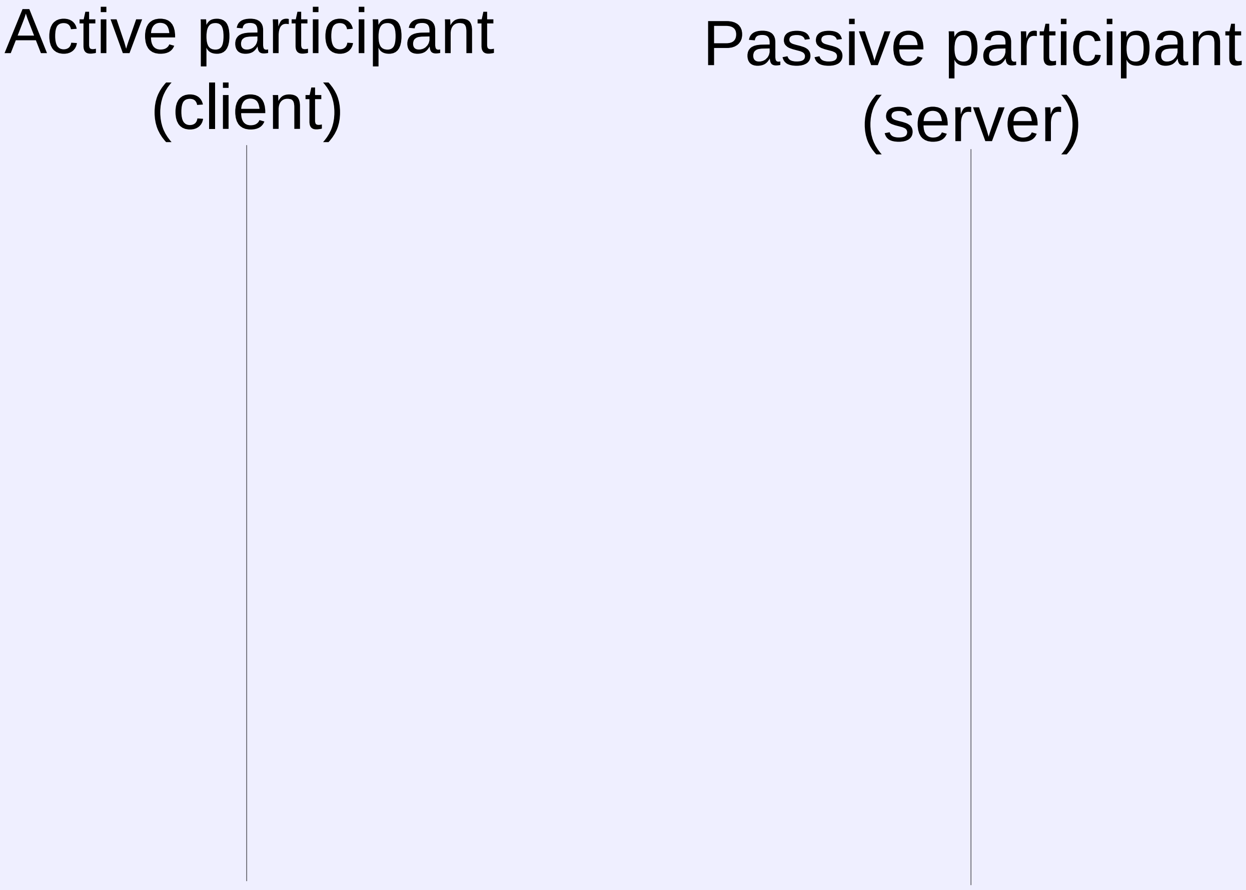
Connections

- Both sender an receiver must be ready before start of data transfer
  - Sender and receiver need to agree on parameters
  - Receive buffer size, initial sliding window variables
- Sender and receiver must agree when transfer is over
  - Both sides must discard state
- This is signalling
  - Setup/teardown state at the endpoints
  - Compare to ‘dialing’ in the telephone network

Connection Management

- Setup
  - 3-way handshake
- Transfer
  - Sliding window, data and ACKs in both directions
- Teardown
  - 4-way handshake
- Client-server model
  - Initiator (client) server
  - Listener (server) responds, provides service

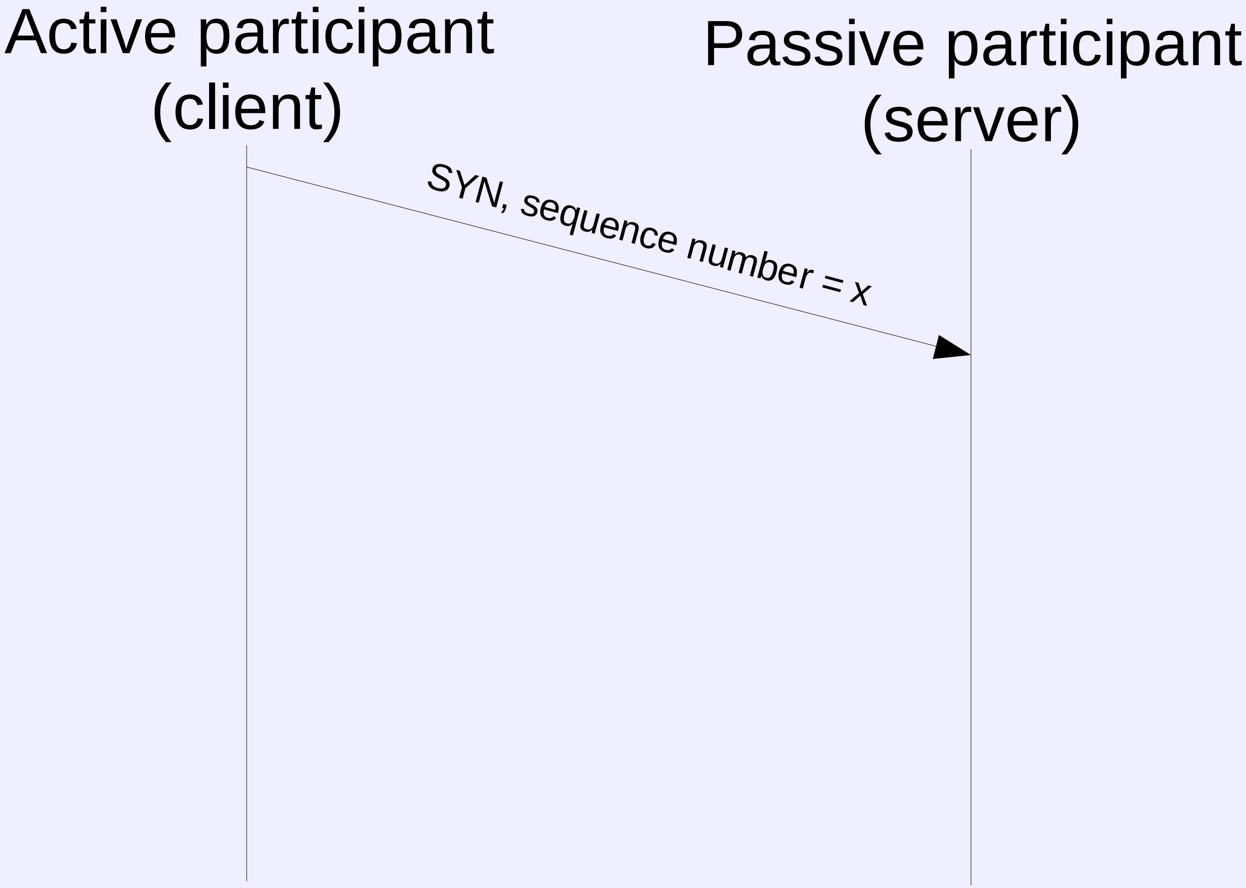
Connection Creation



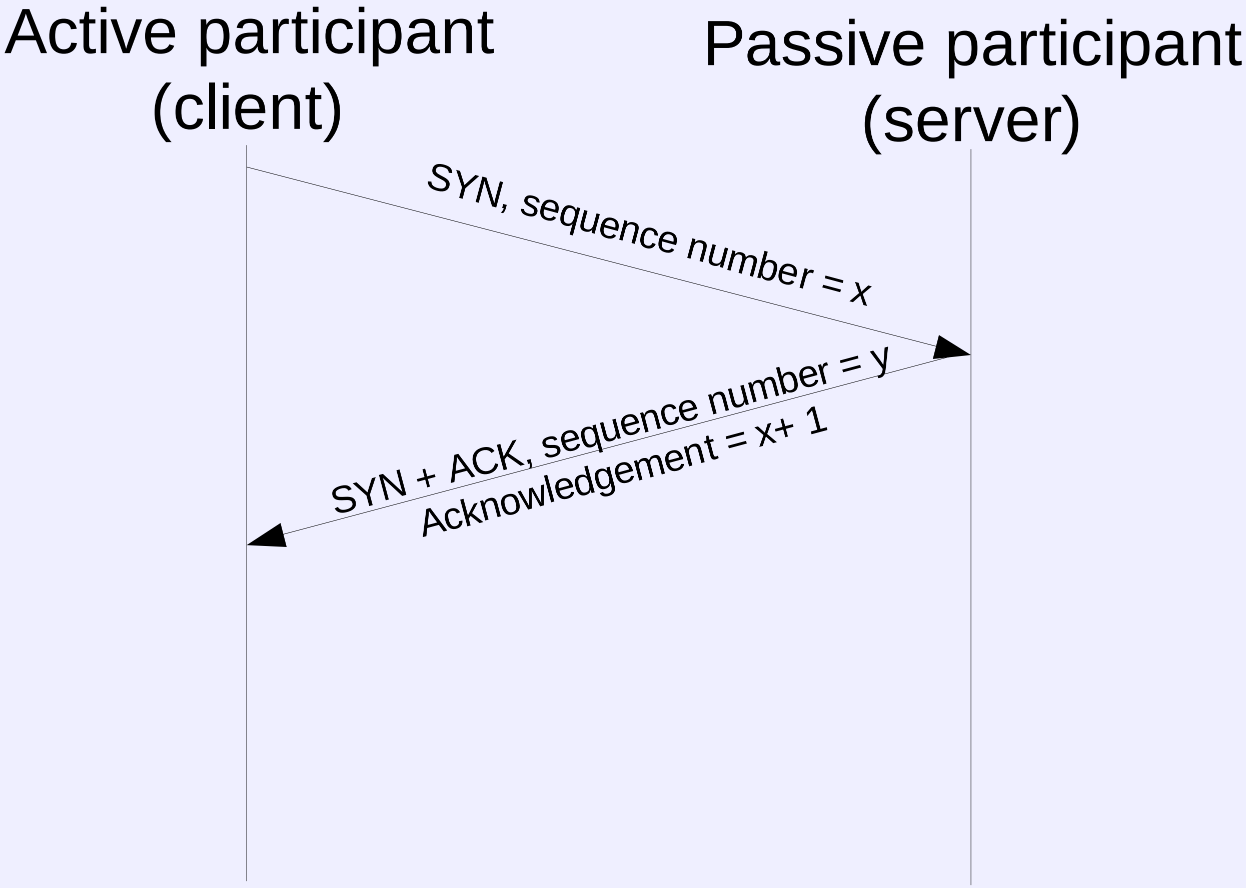
Active participant
(client)

Passive participant
(server)

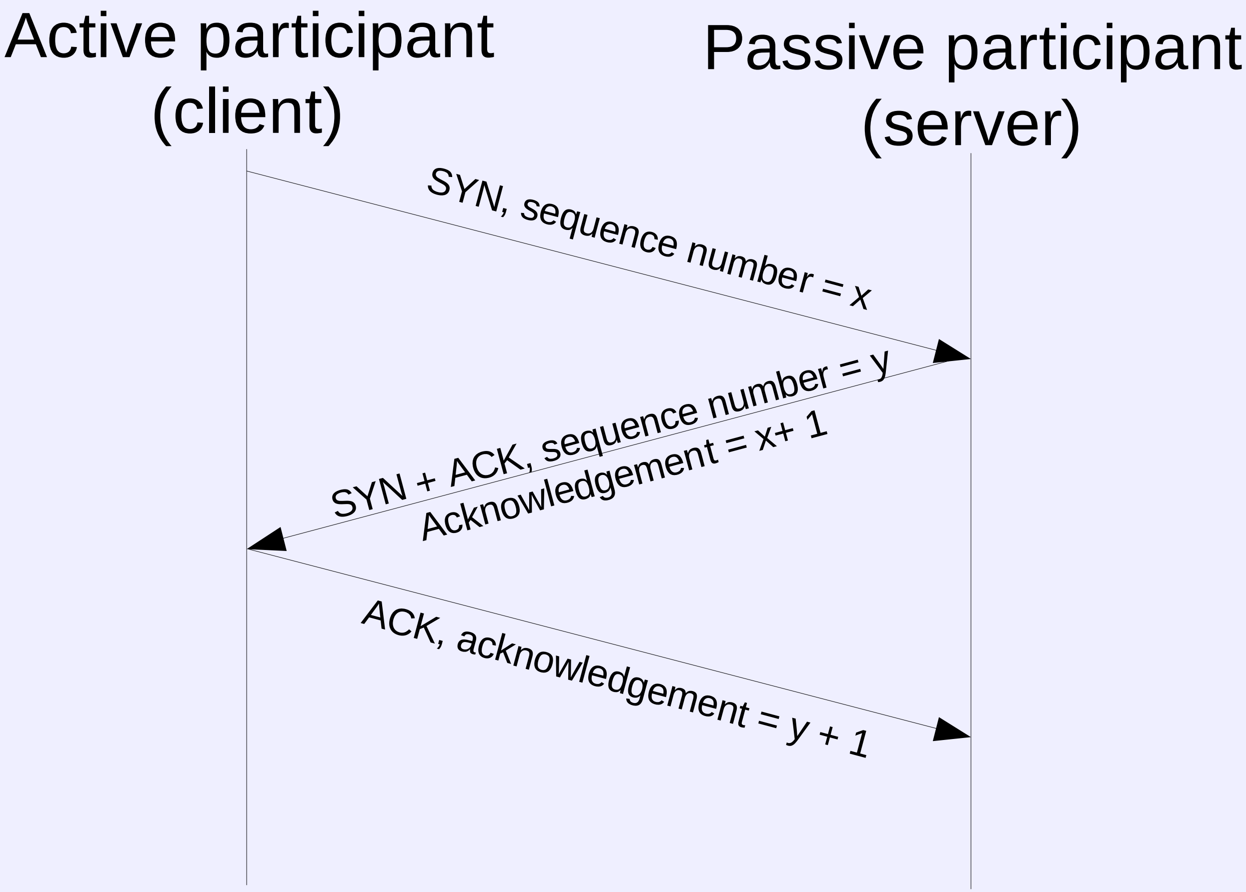
Connection Creation



Connection Creation



Connection Creation



Active participant
(client)

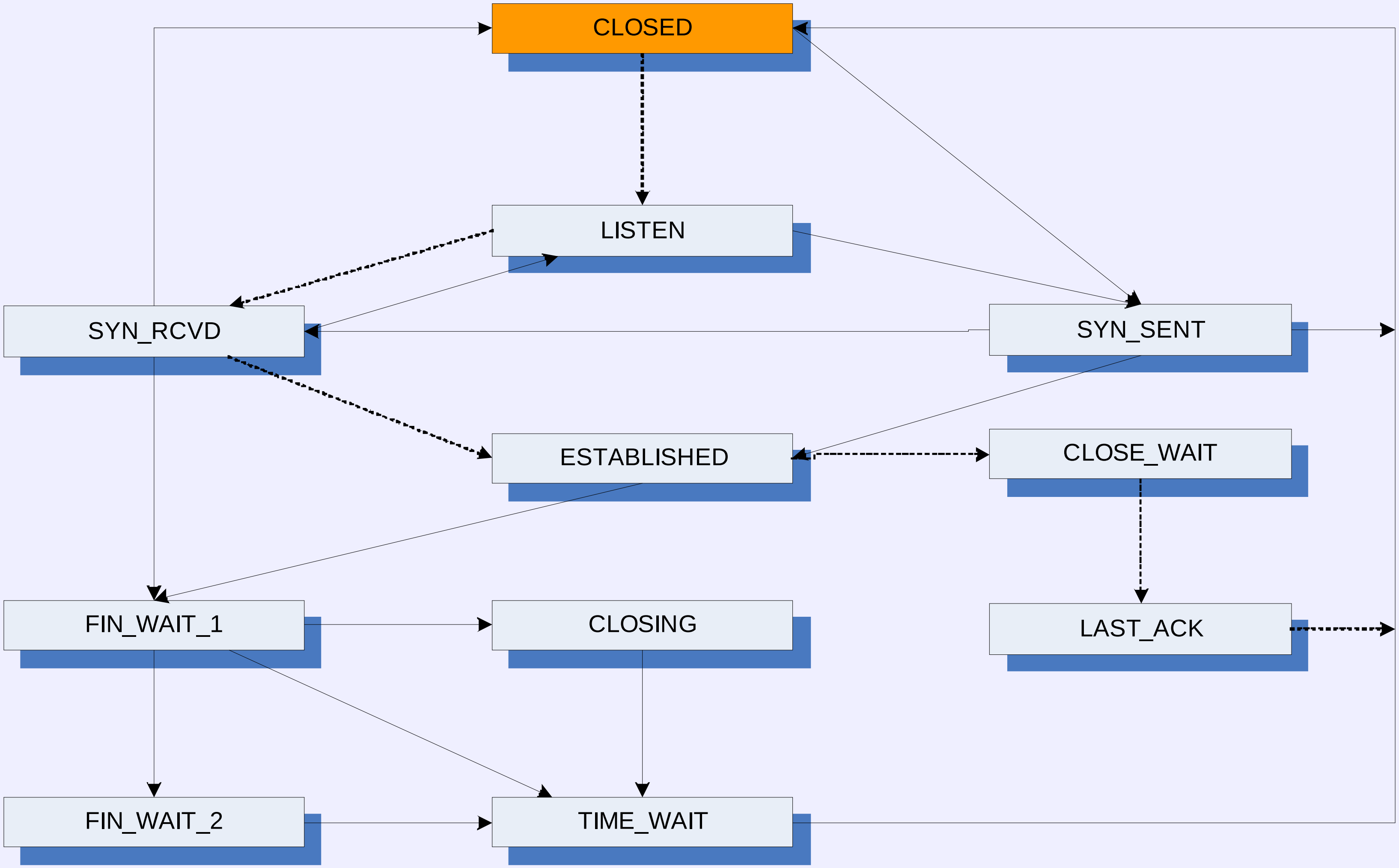
Passive participant
(server)

SYN, sequence number = x

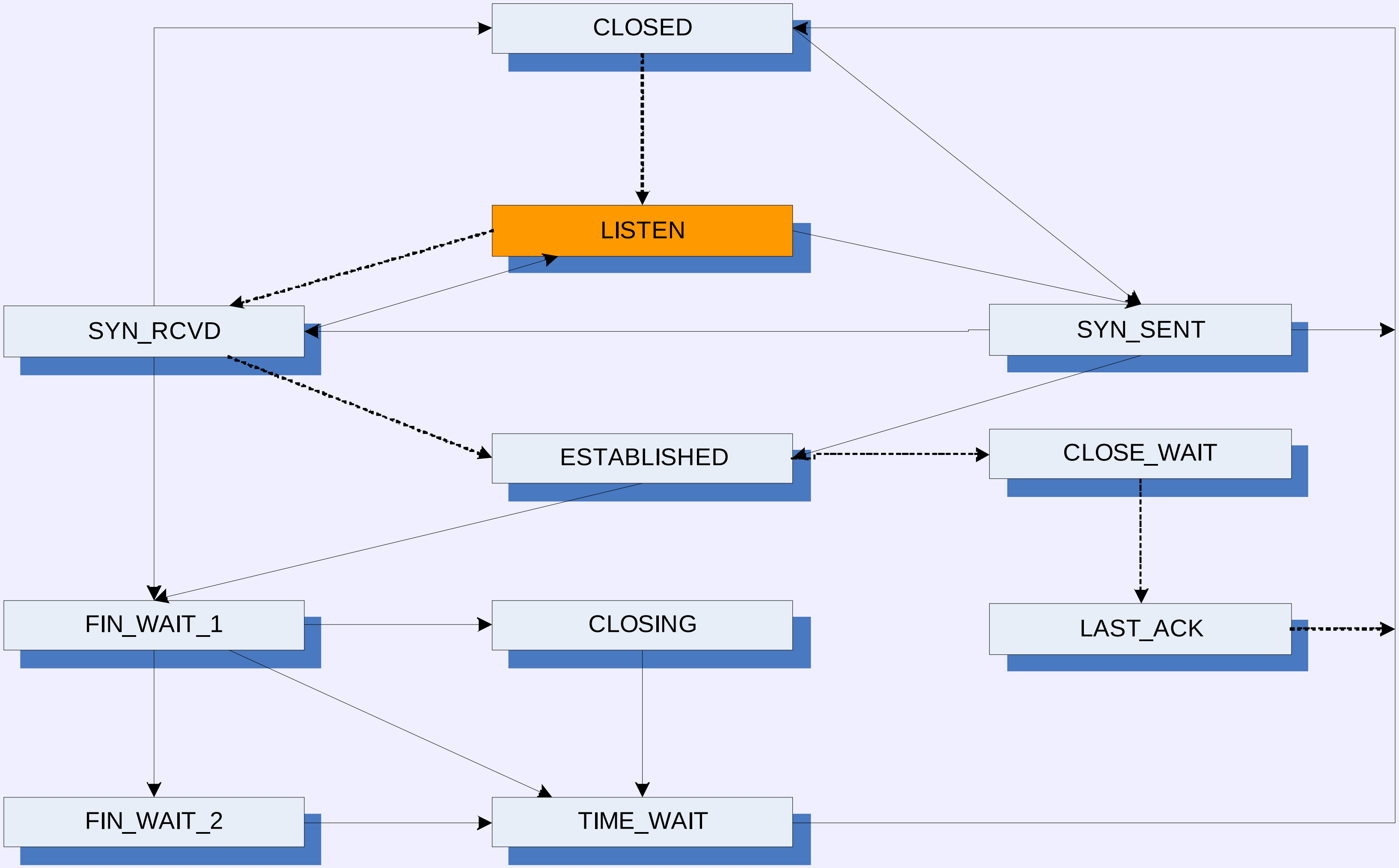
SYN + ACK, sequence number = y
Acknowledgement = x+ 1

ACK, acknowledgement = y + 1

TCP


CLOSED
LISTEN
SYN_RCVD
SYN_SENT
ESTABLISHED
CLOSE_WAIT
FIN_WAIT_1
CLOSING
LAST_ACK
FIN_WAIT_2
TIME_WAIT

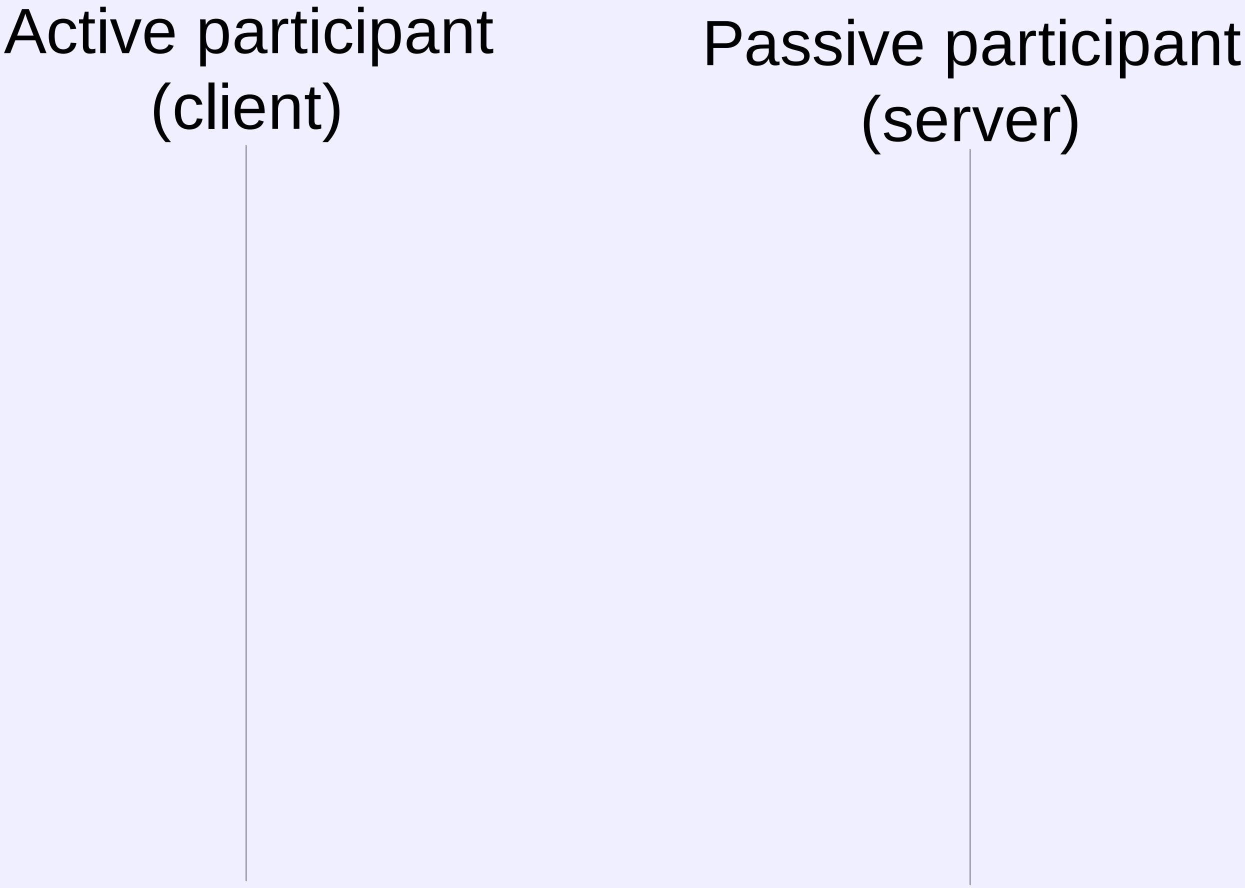
TCP


CLOSED
LISTEN
SYN_RCVD
SYN_SENT
ESTABLISHED
CLOSE_WAIT
FIN_WAIT_1
CLOSING
LAST_ACK
FIN_WAIT_2
TIME_WAIT

Data Transfer

- Connection is bi-directional
- ACKs can carry data

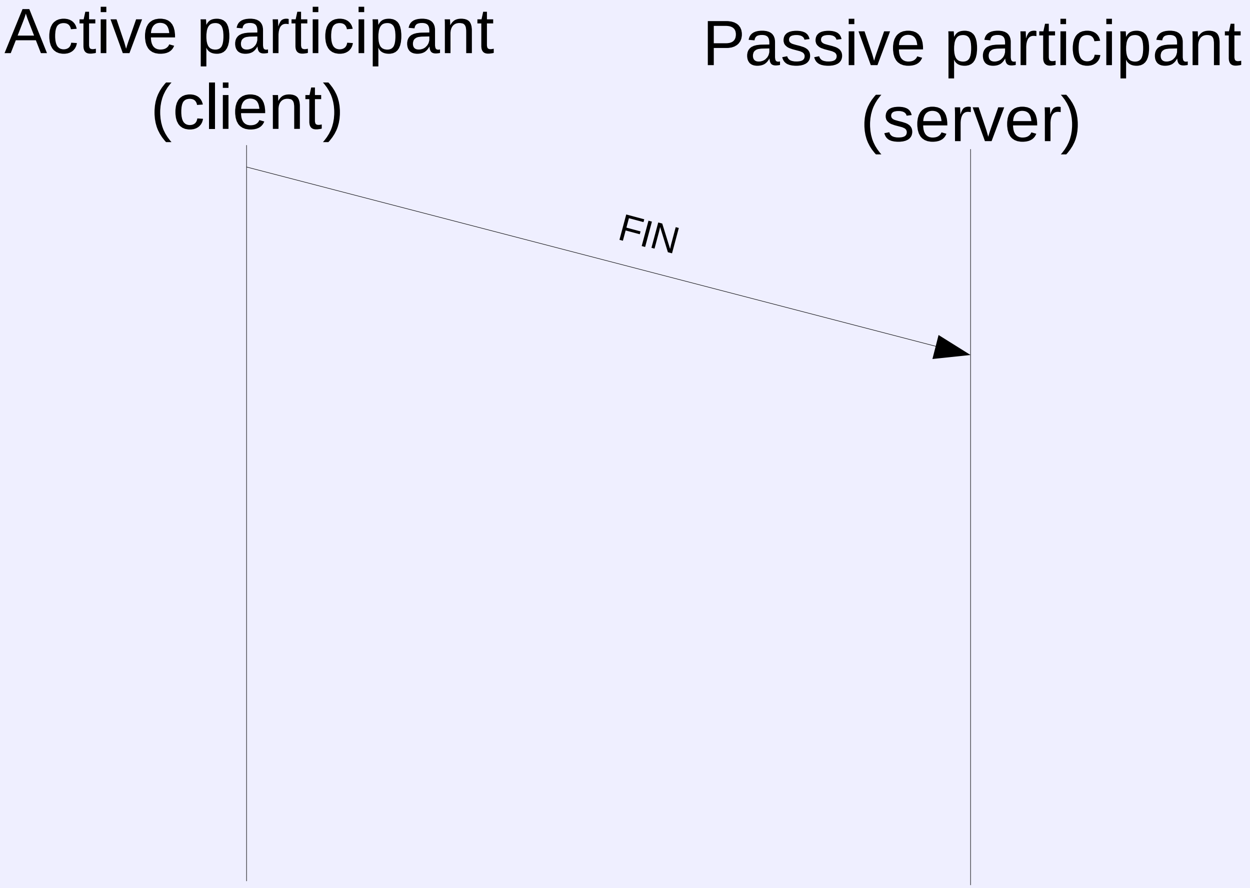
Connection Teardown



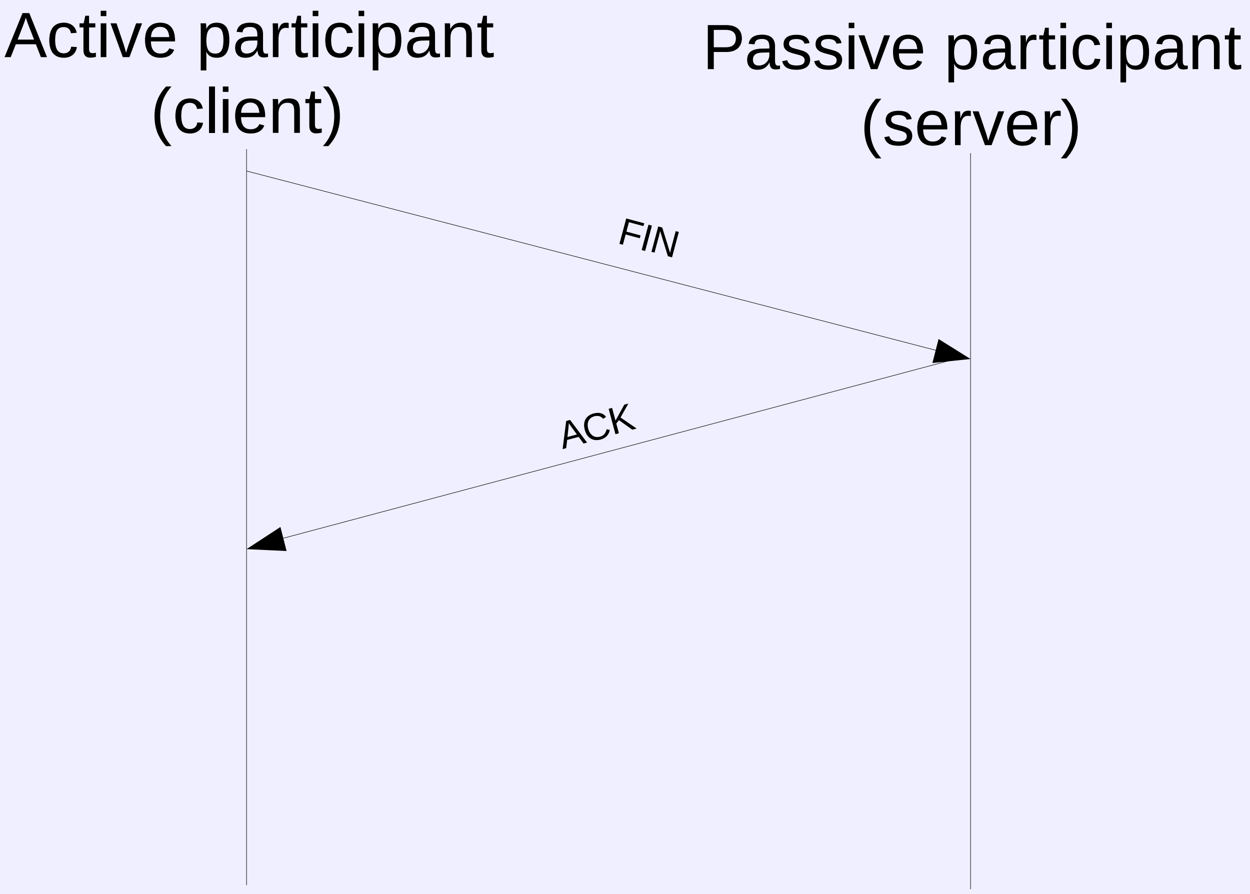
Active participant (client)

Passive participant (server)

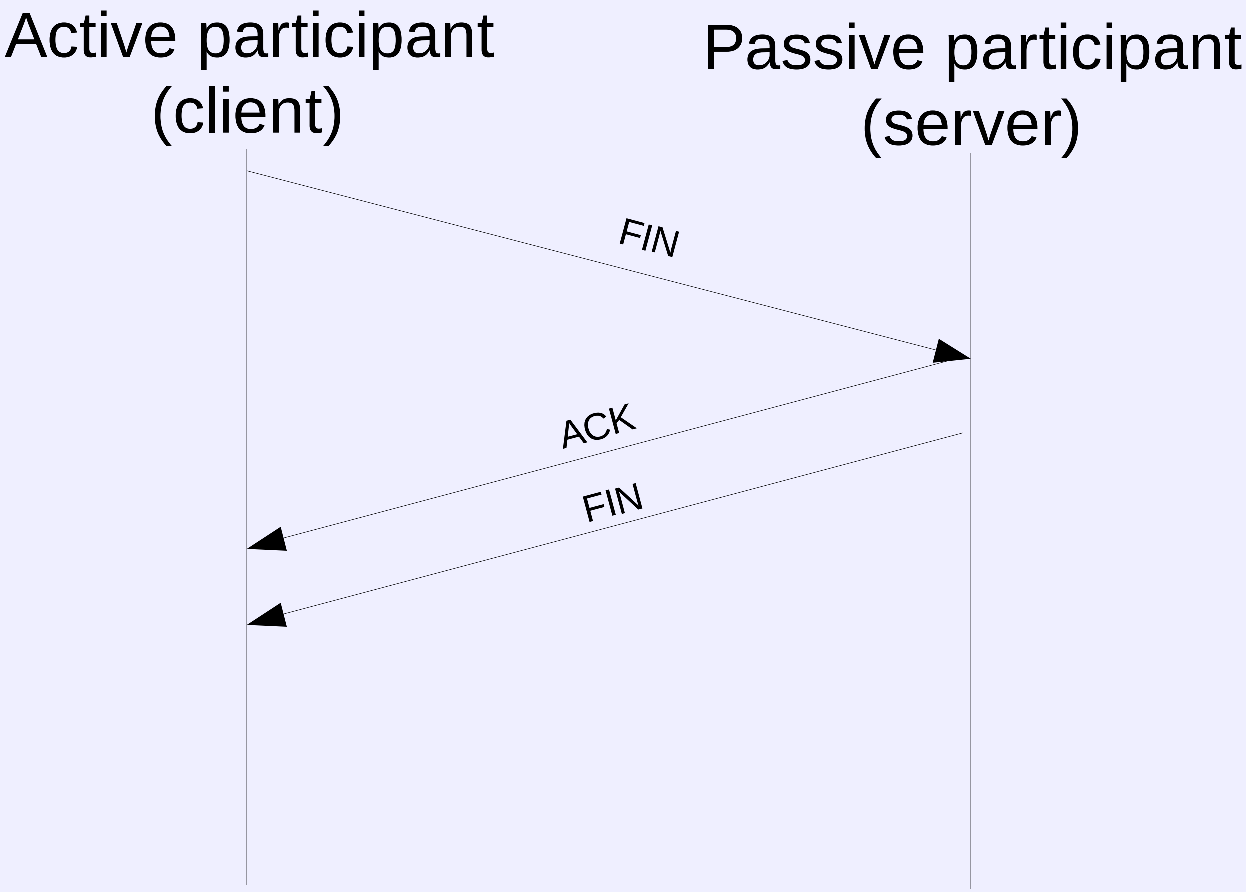
Connection Teardown



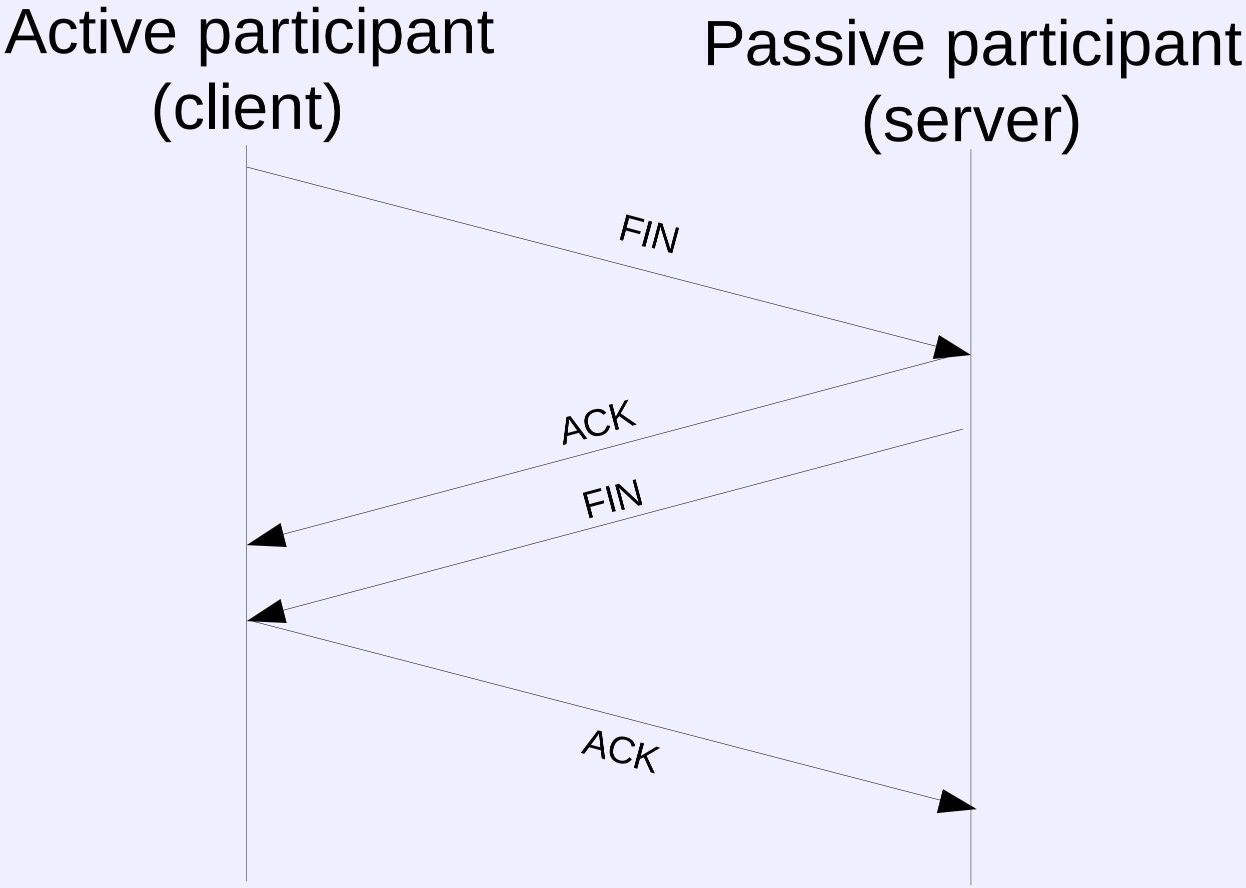
Connection Teardown



Connection Teardown



Connection Teardown



Active participant (client)

Passive participant (server)

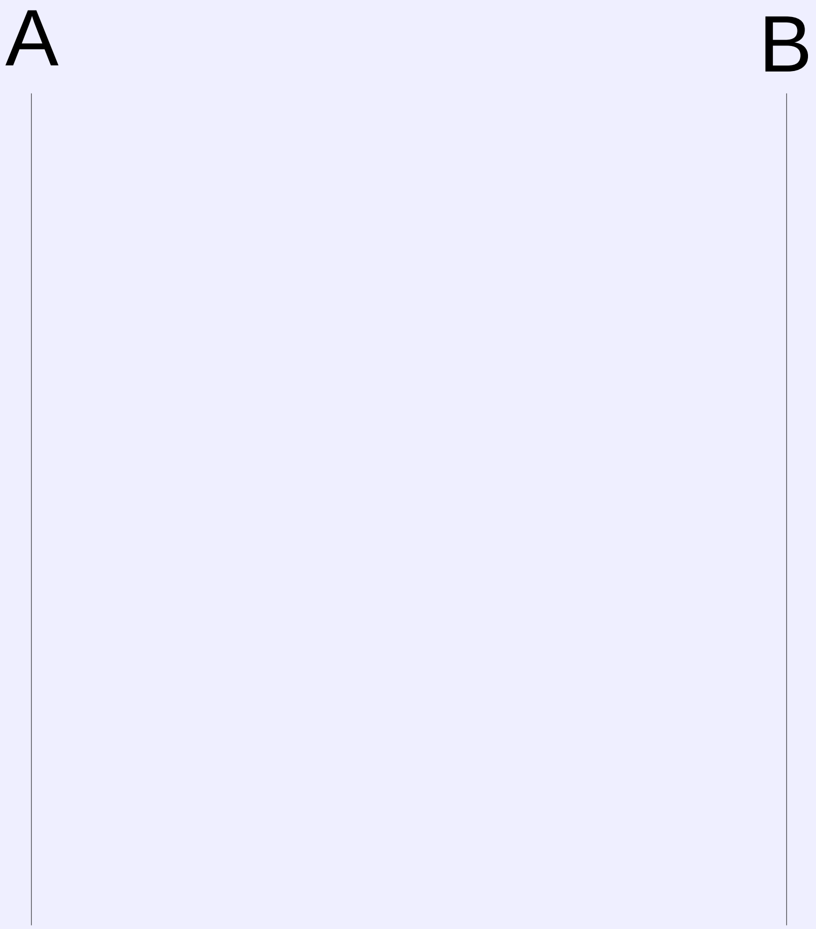
The TIME_WAIT State

- We wait 2 * maximum segment lifetime (60 seconds) before completing the close
- Why?
  - ACK might have been lost so FIN would be resent
  - Could interfere with subsequent connections

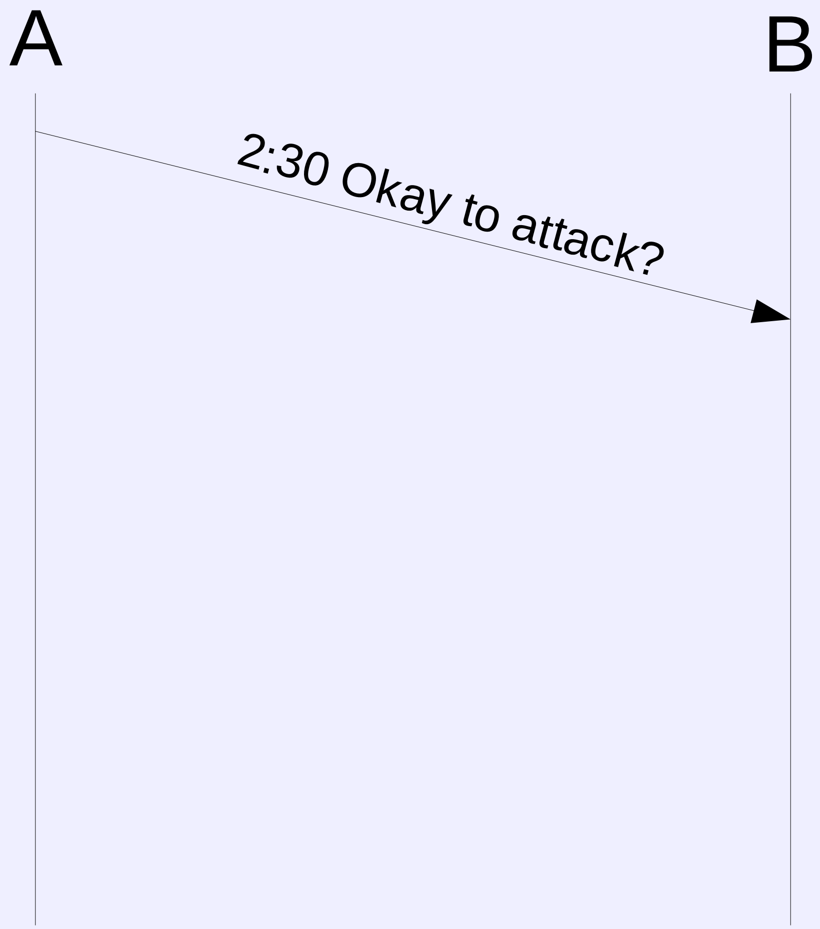
‘Two-Army’ Problem

- Can we use messages and retries to synchronise two machines so they are guaranteed to do some operation at the same time?

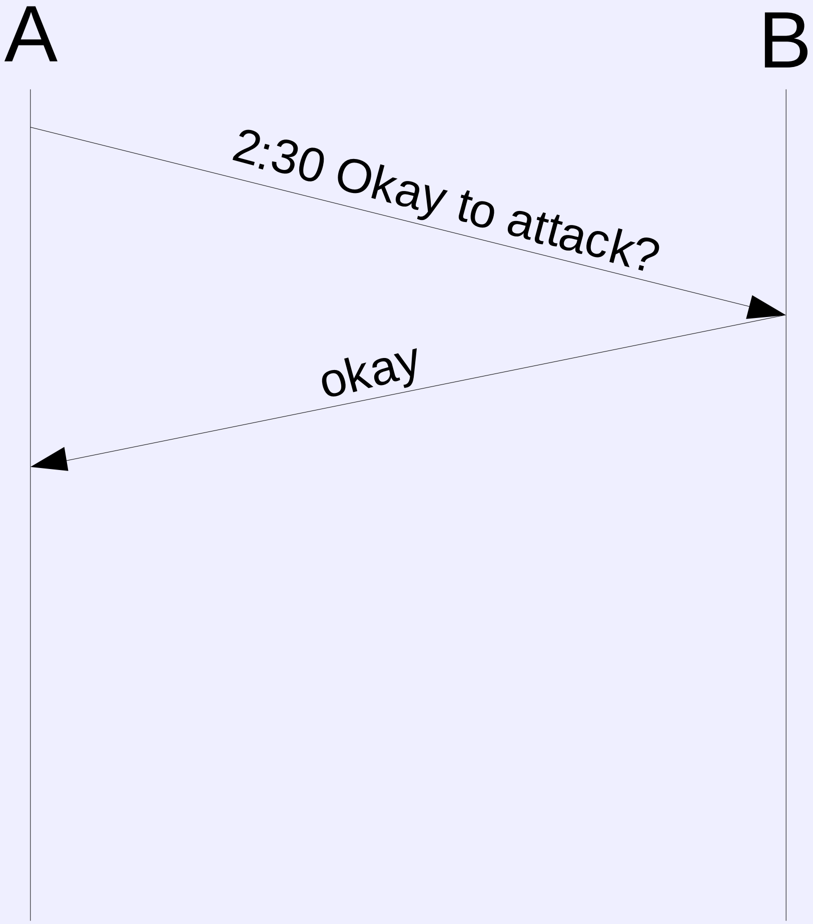
Two-Army Problem



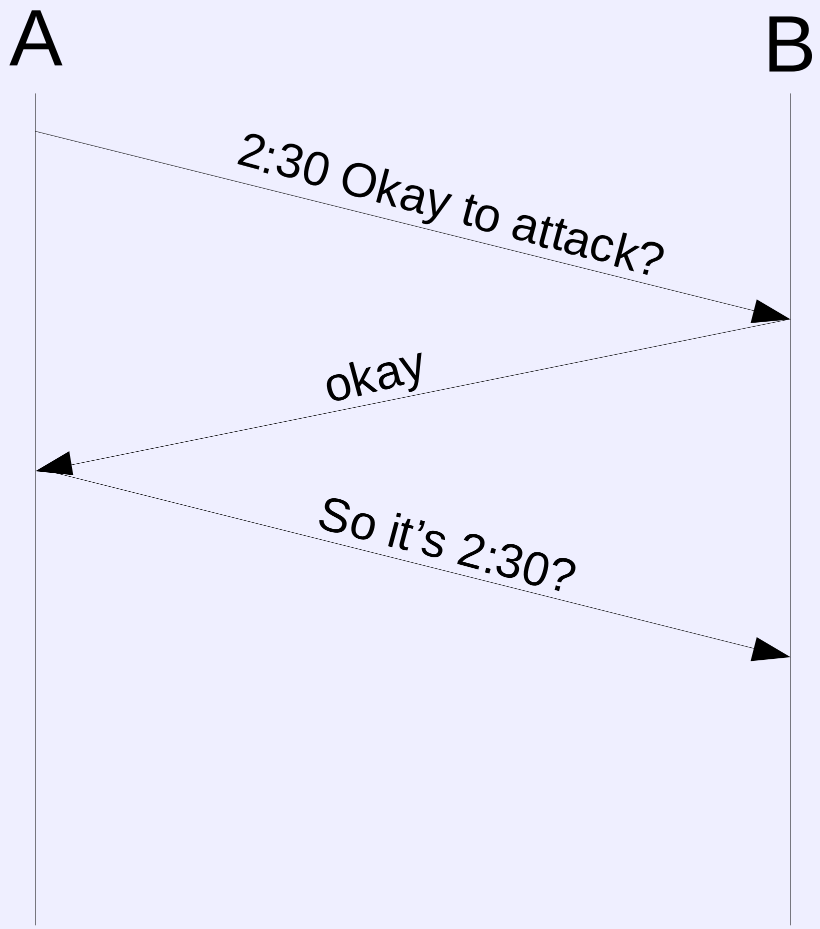
Two-Army Problem



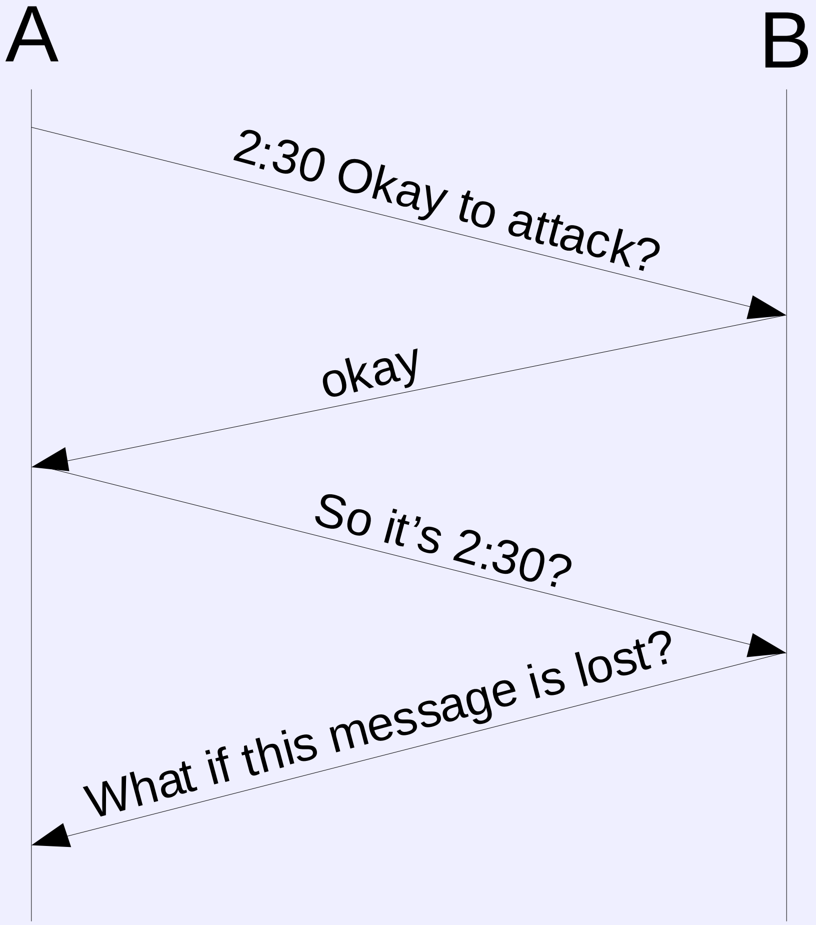
Two-Army Problem



Two-Army Problem



Two-Army Problem



A

B

2:30 Okay to attack?

okay

So it’s 2:30?

What if this message is lost?