TCP: Connection Management

Kameswari Chebrolu

All the figures used as part of the slides are either self created or from the public domain with either 'creative commons' or 'public domain dedication' licensing. The public sites from which some of the figures have been picked include: http://commons.wikimedia.org (Wikipedia, Wikimedia and workbooks); http://www.sxc.hu and http://www.pixabay.com

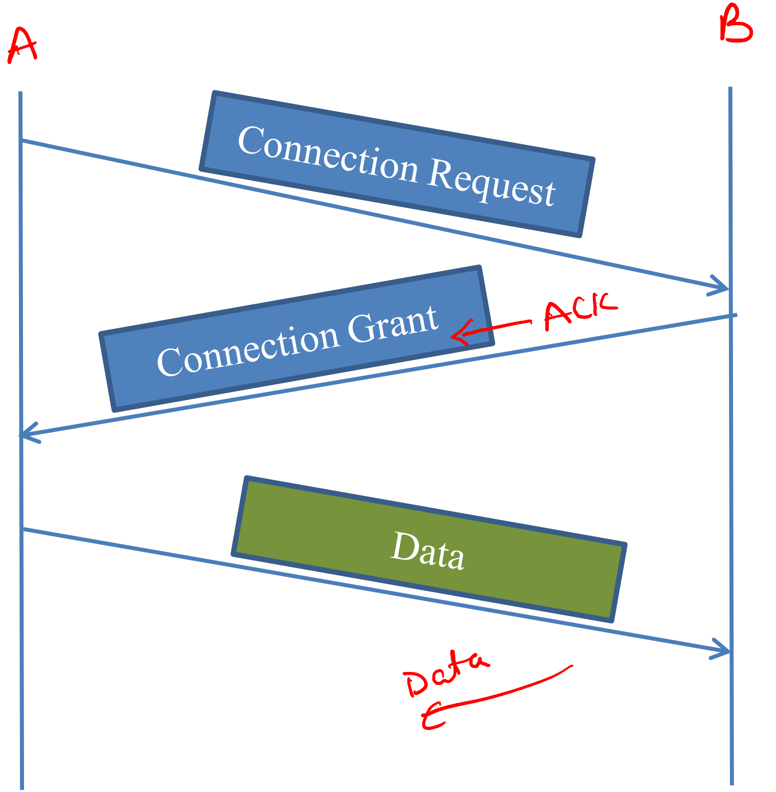
Background

- TCP is a connection oriented protocol
  - Processes can run on any type of machine in the Internet
- Connection establishment helps
  - Exchange and initiate state variables
    - MSS size, initial sequence number, ACK type
  - Allocate resources (buffer space)

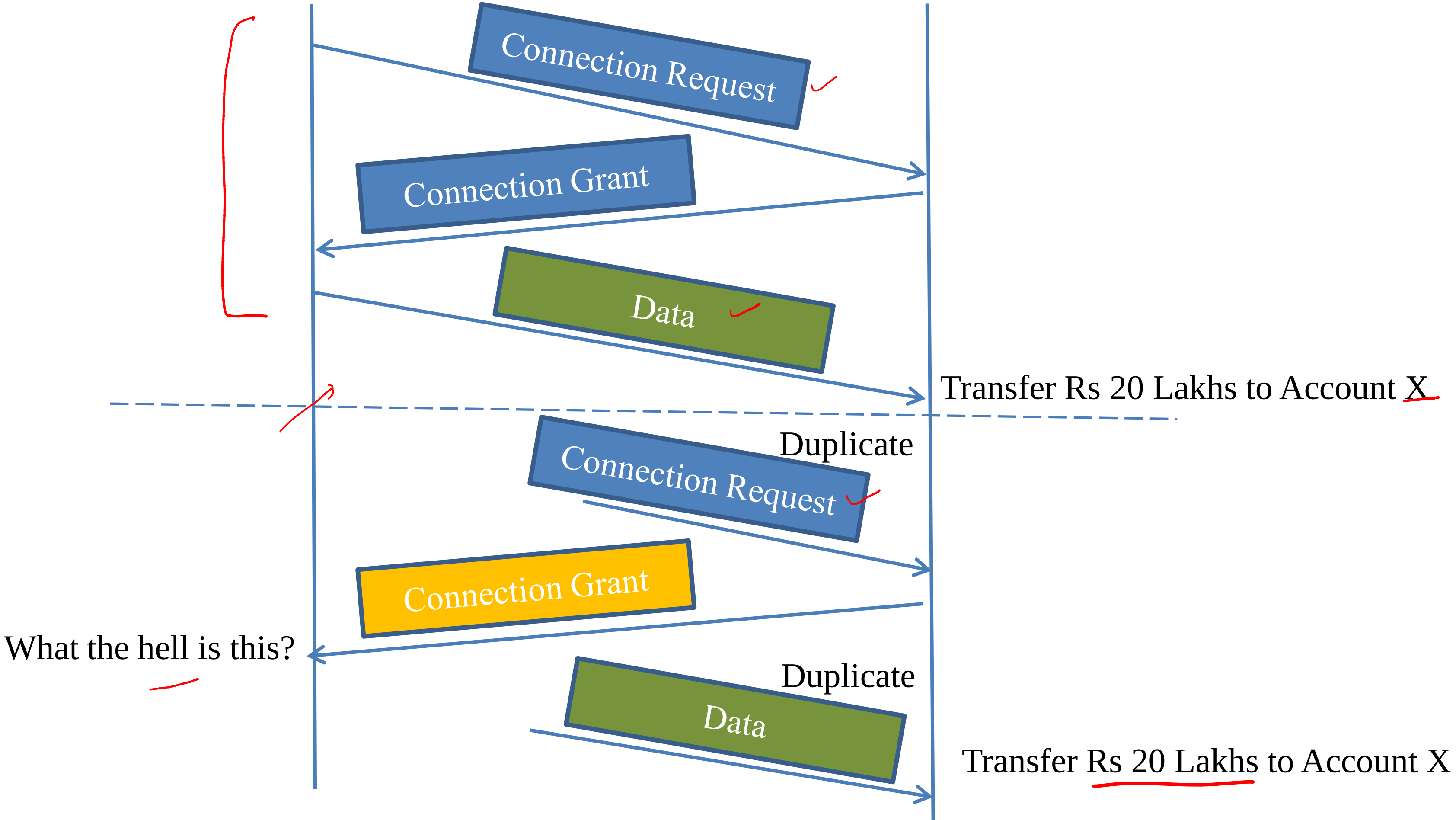
Send Buffer
4KB – MB

receive Buffer
8KB

Connection Setup



Problem



Connection Request
Connection Grant
Data
Transfer Rs 20 Lakhs to Account X
Duplicate
Connection Request
Connection Grant
What the hell is this?
Duplicate
Data
Transfer Rs 20 Lakhs to Account X

Solution

- TCP's famous three-way handshake (idea from Tomlinson)

Syn

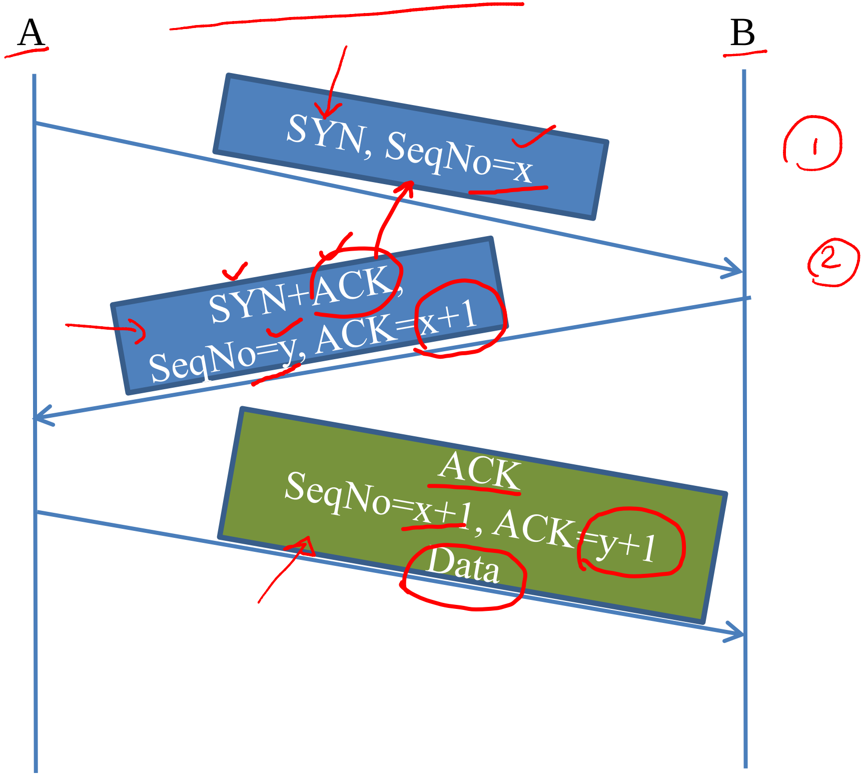
Syn + ACK

ACK

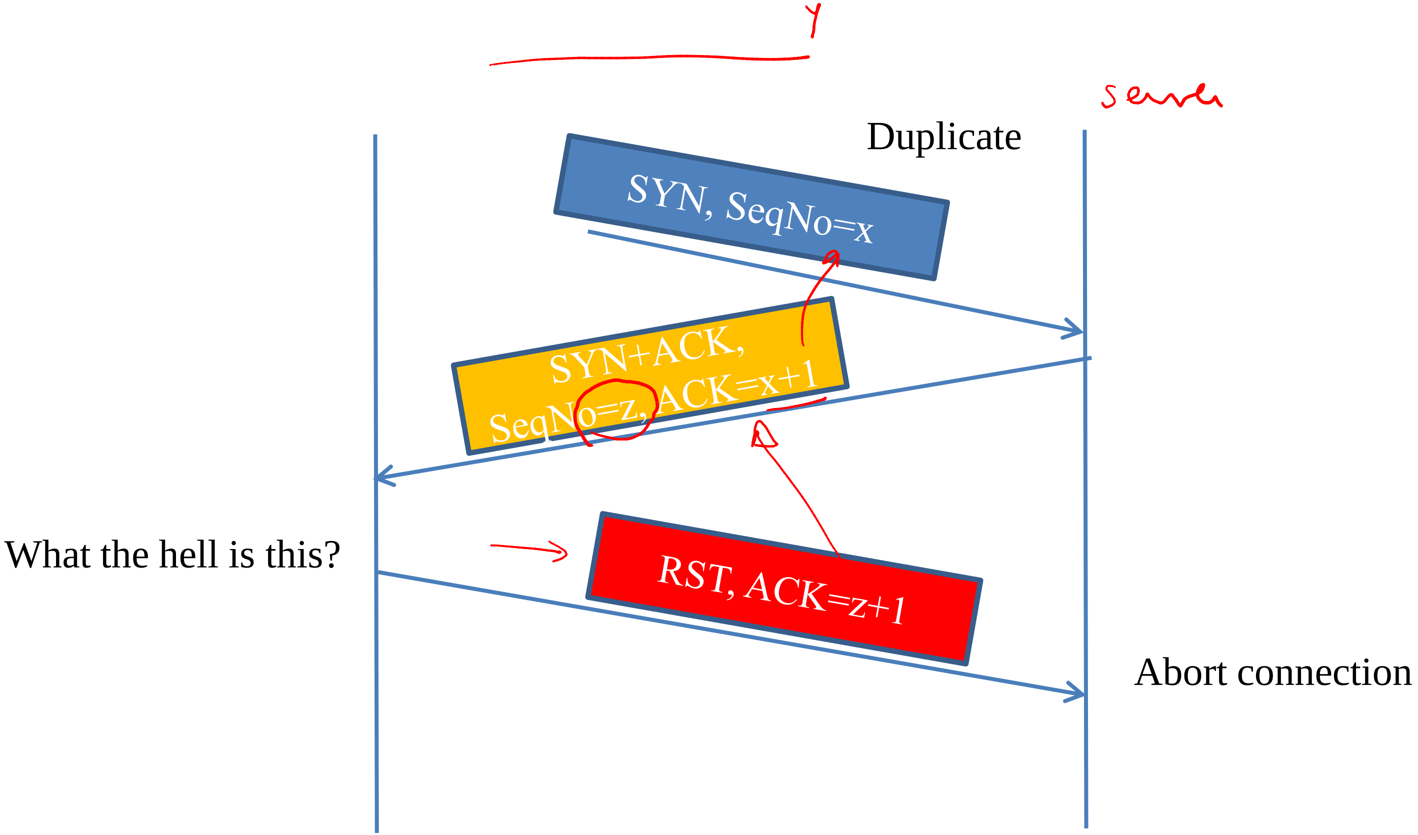
I want beer please

You want beer?

Yes, beer please

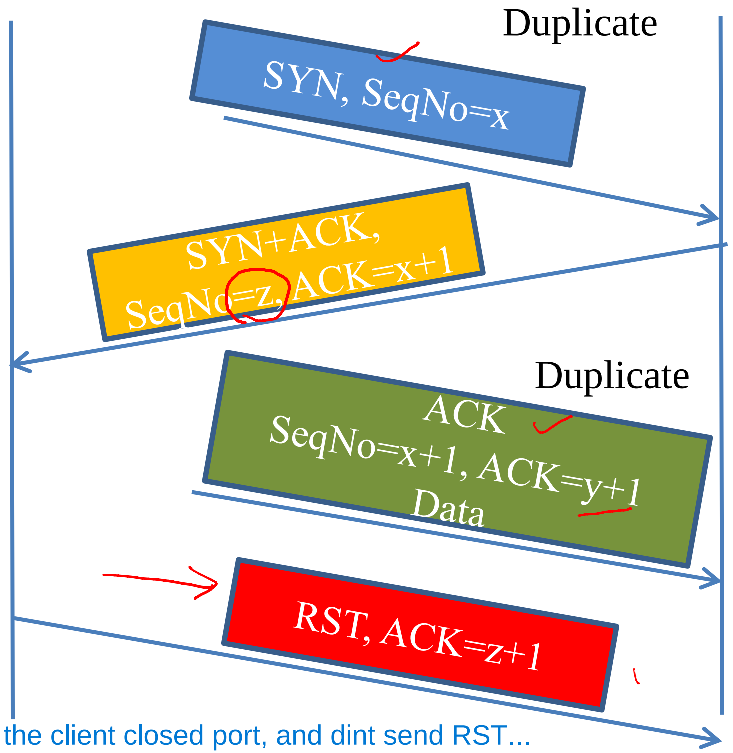


Case-1



Y
server
Duplicate
SYN, SeqNo=x
SYN+ACK, SeqNo=z, ACK=x+1
What the hell is this?
RST, ACK=z+1
Abort connection

Case-2



Duplicate

SYN, SeqNo=x

SYN+ACK,
SeqNo=z, ACK=x+1

Duplicate

ACK
SeqNo=x+1, ACK=y+1
Data

RST, ACK=z+1

What the hell is this?

Client

Huh? I sent seqno z.
Why is it acking y? Stop

Abort connection

even if the client closed port, and dint send RST...
so 3 way handshake not established .. so connection closed.

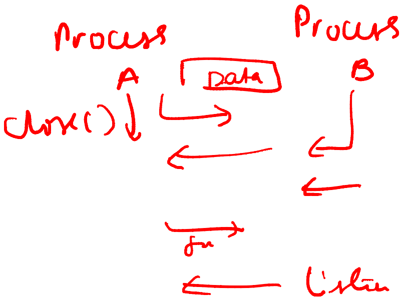
Initial Sequence Number (ISN)

- Why not start with Seqno zero?
- Segments from different connections can get mixed up
- Security risk when ISN's are predictable
- Original solution: Use a clock (e.g. increments every 4 microsec) to choose ISN
  - 32 bit sequence number wraps around in 4 hrs
- Current implementations use random ISN

IP → TTL

Connection Termination

- Asymmetric release (just hang-up) leads to loss of data



- Symmetric release
  - Treat connection as two separate unidirectional connections
  - Each side should be released separately

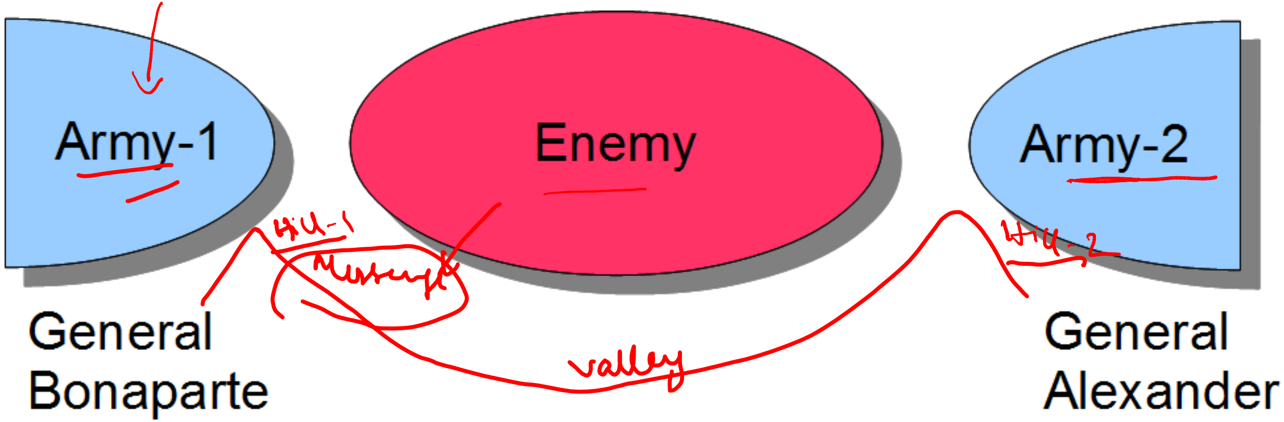
Two Army Problem

1. Lets attack on Sun @ 9am, please ACK

2. OK. fine. Lets attack, please ACK

3. OK. fine. ACK

CAN IT ATTACK NOW????

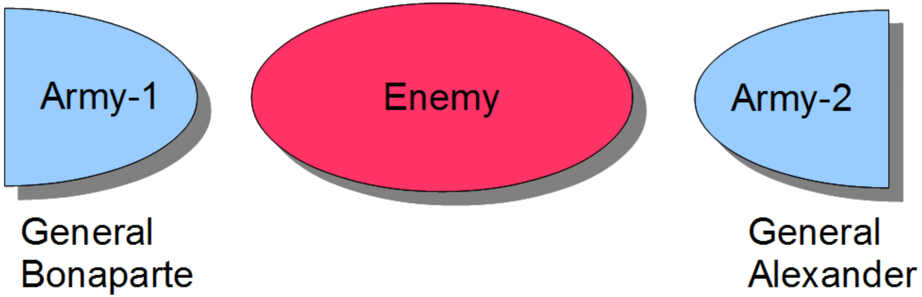


NO MATTER HOW MANY U USE, U CNT BE SURE

The attack will succeed *if and only if* both armies attack the enemy at the same time

What strategy to adopt?

Relevance



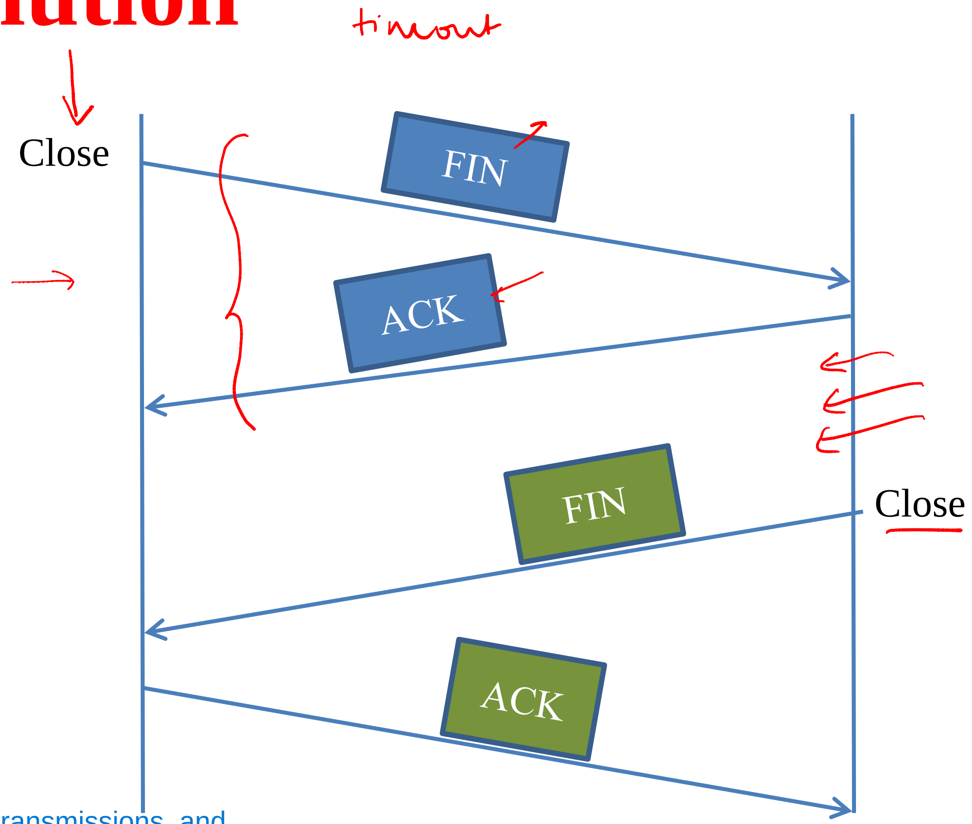
The attack will succeed *if and only if* both armies attack the enemy at the same time

What strategy to adopt?

If neither side is ready to disconnect unless it is sure the other side is ready to disconnect, disconnect will never happen

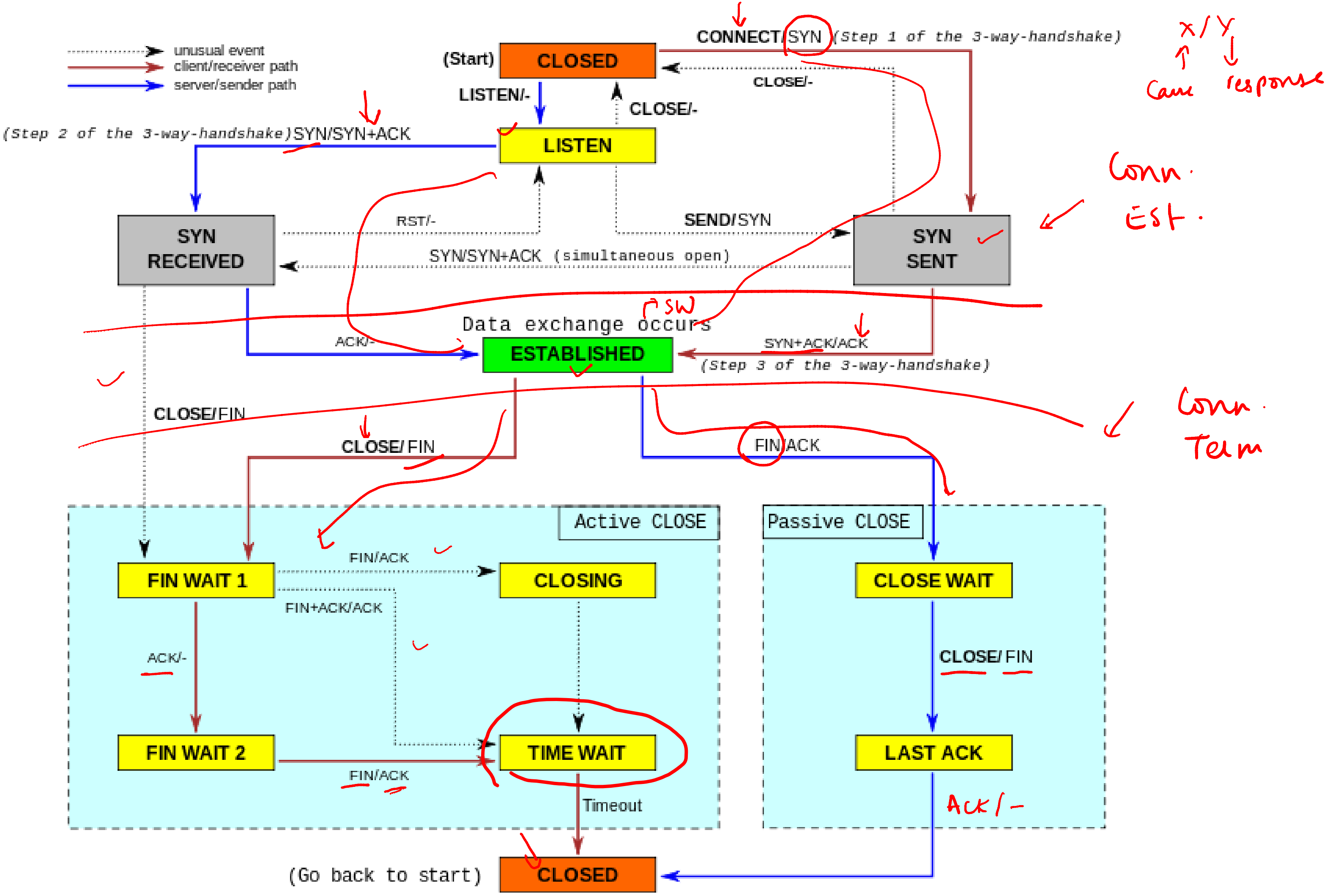
Solution

- Follows simple two-way handshake
- Each side independently closes connection

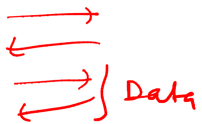


u will still use time-outs and Retransmissions, and if u do too many retrans then close()

TCP State Diagram



Time-Wait State



- Wait in time-wait for 2*MSL (maximum segment lifetime)
    - Helps clear out older packets in the network; prevents them from interfering with new connection
    - Time spent in time-wait range from 30sec to 2 min

→ reuse Port #

"bind failed"

Conn 1 : src ip, dst ip, src Port . dst Port

data

→ closed.

Conn 2 :

Abort

RST

if we establish a new connection with same port,dsp,src ip .. then if prev packet which got delayed reaches now , then seq num not mathc and it leads to ABORT RST.

so u wait for sometime to all those packets handling... TIMEOUT state.. then release those ports for new connection .... if u try to connect .. it gives BINDING ERROR.

Summary

- TCP is a connection oriented protocol
- Connection management complicated by the fact that packets can get retransmitted, delayed, delivered out of order etc
- Connection establishment governed by 3-way handshake
- Connection termination is based on symmetric release and managed by 2-way handshake
- Ahead: Sliding window action in the established state