

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:

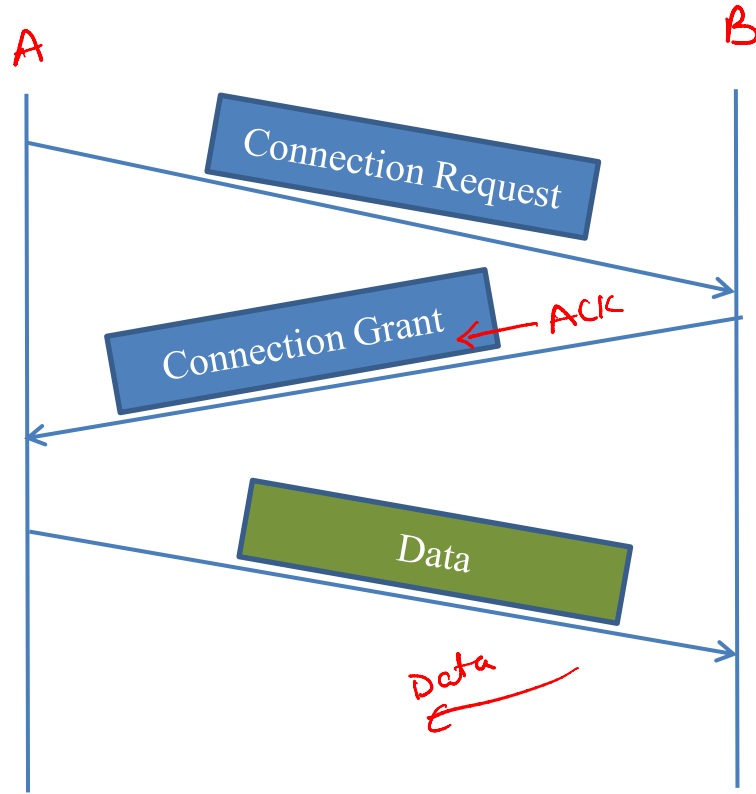
<http://commons.wikimedia.org> (Wikipedia, Wikimedia and workbooks); <http://www.sxc.hu> and <http://www.pixabay.com>

Background

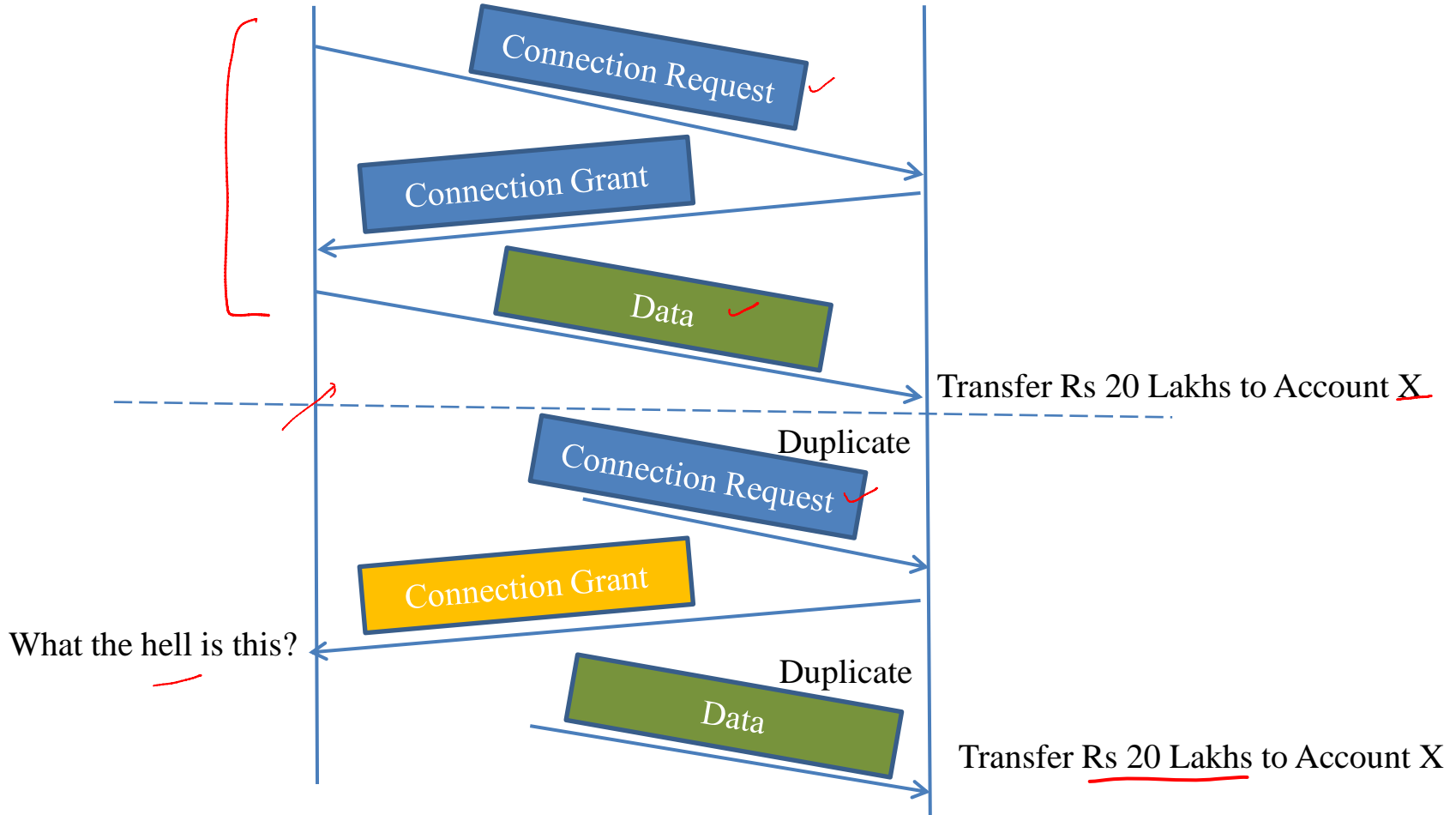


- TCP is a connection oriented protocol
 - Processes can run on any type of machine in the Internet
- Connection establishment helps
 - Exchange and initiate state variables
 - MSS size, initial sequence number, ACK type
 - Allocate resources (buffer space)
 - ↓ send Buffer
4KB - 1MB
 - ↓ receive Buffer
8KB

Connection Setup

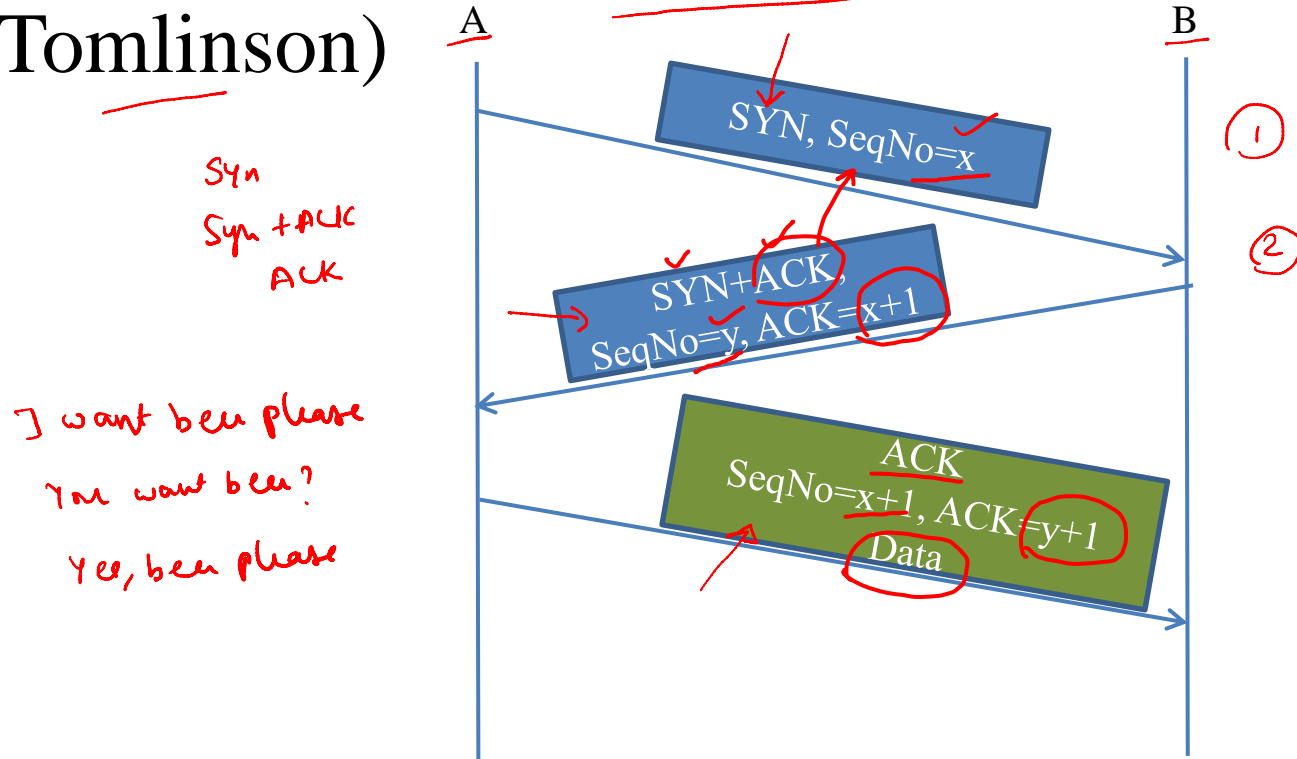


Problem

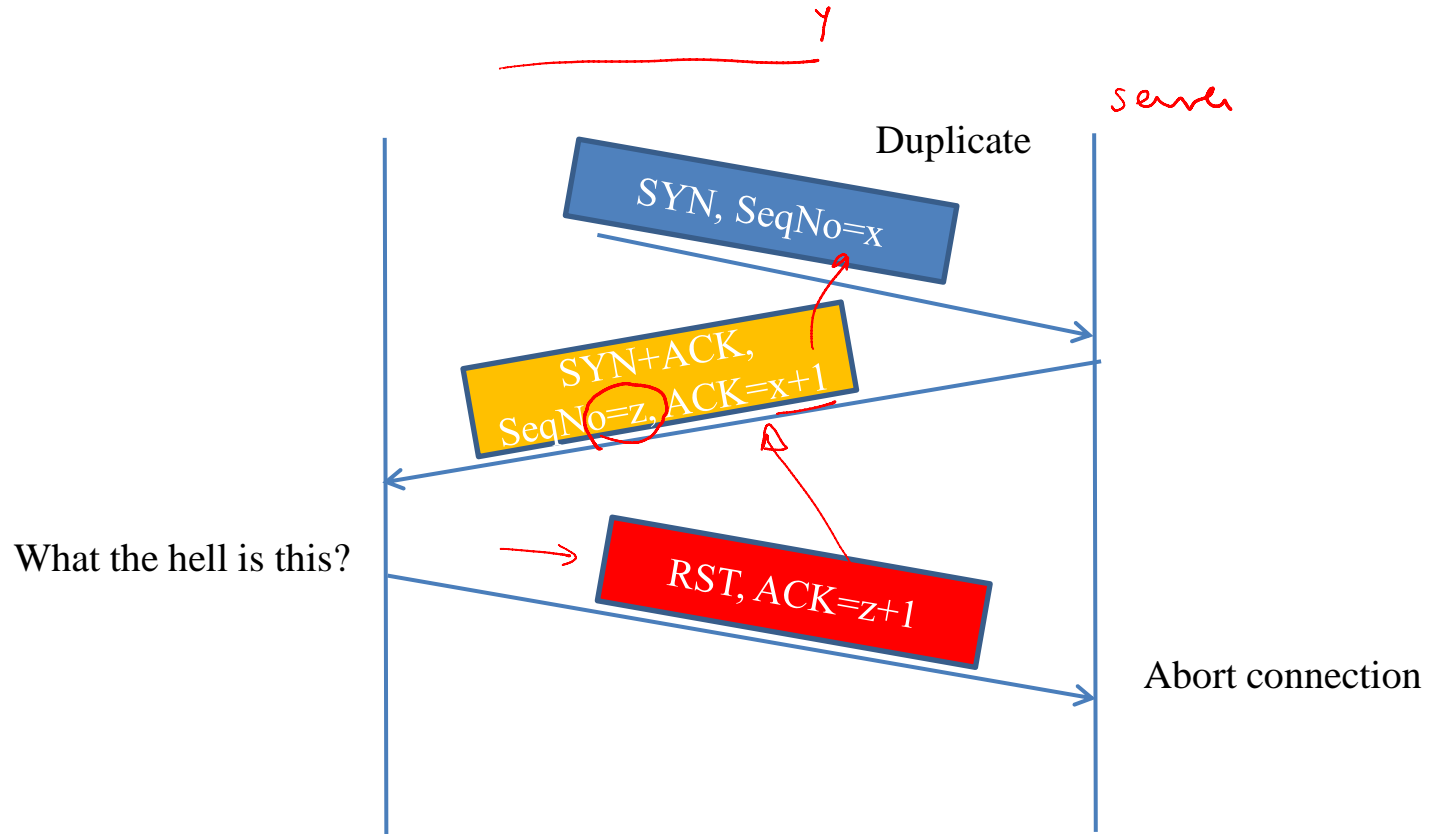


Solution

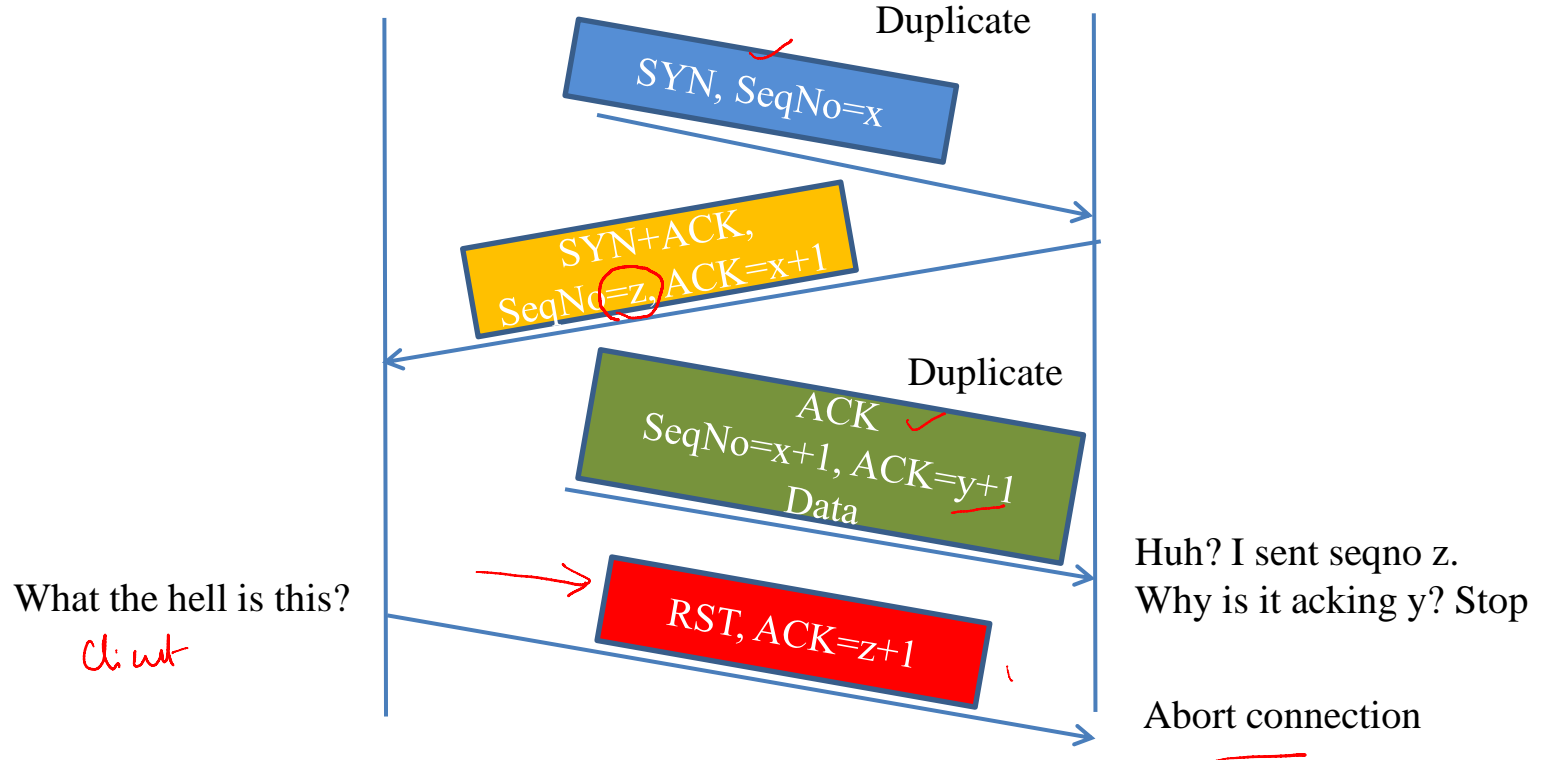
- TCP's famous three-way handshake (idea from Tomlinson)



Case-1



Case-2



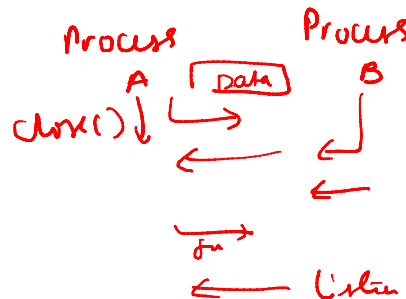
Initial Sequence Number (ISN)

- 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

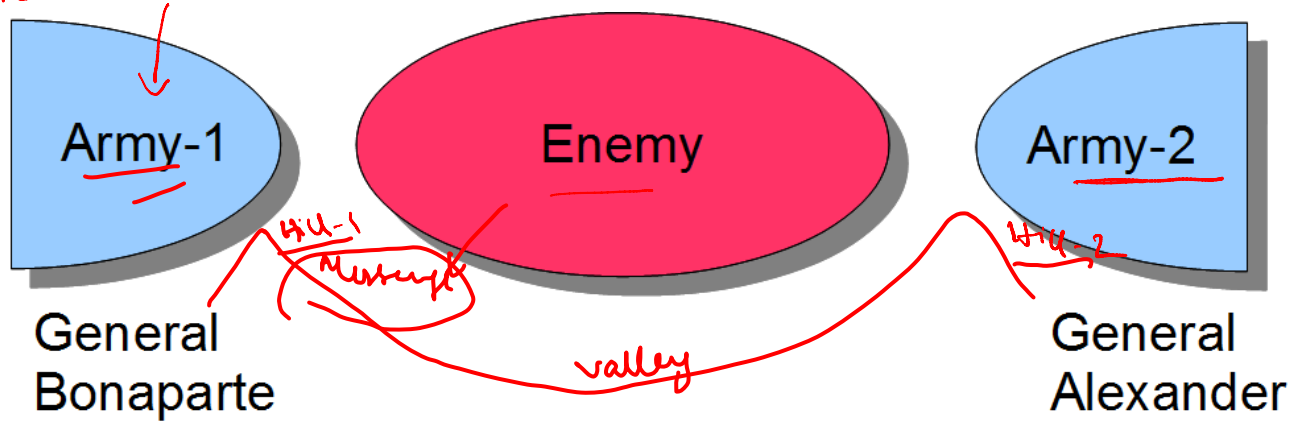
Connection Termination

- 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



1. Let's attack on Sun @ 9am, Please ACK
2. OK fine. Let's attack, Please ACK
3. OK fine. ACK

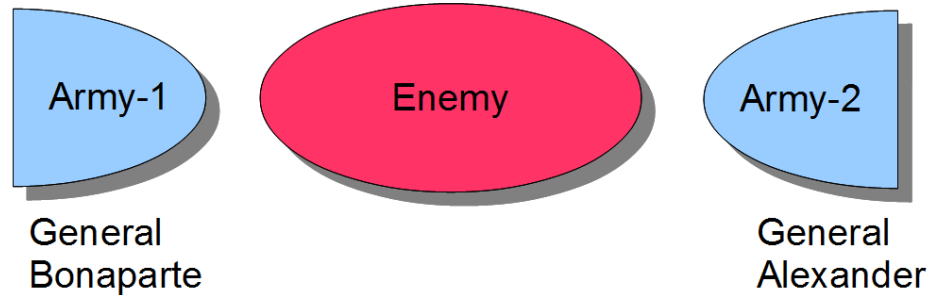
Two Army Problem



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

What strategy to adopt?

Relevance



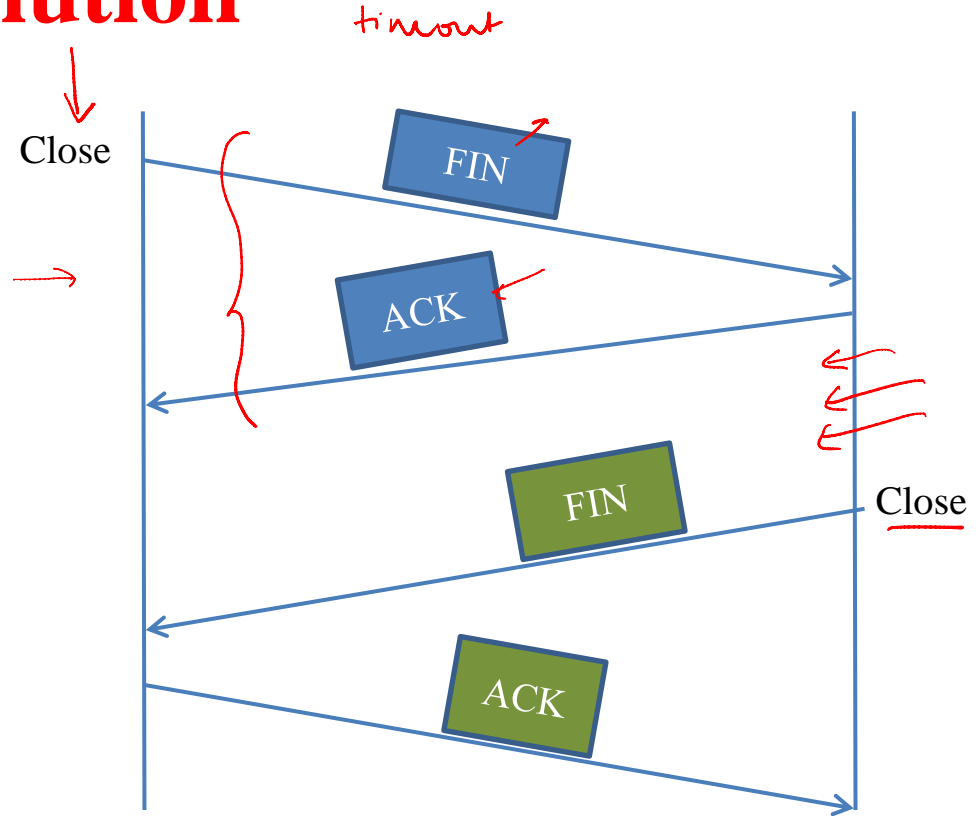
The attack will succeed *if and only if* both armies attack the enemy at the same time

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

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

Solution



The diagram illustrates the TCP state transitions and the flow of data and control packets. It is divided into three main sections: Connection Establishment, Data Exchange, and Connection Termination.

Connection Establishment (3-way Handshake)

- Start:** The connection begins in the **CLOSED** state.
- Step 1:** A **CONNECT SYN** packet is sent from the client to the server.
- Step 2:** The server responds with a **SEND/SYN** packet.
- Step 3:** The client responds with a **SYN+ACK/ACK** packet.
- States:** The client's state transitions from **CLOSED** to **SYN RECEIVED** and then to **ESTABLISHED**. The server's state transitions from **CLOSED** to **LISTEN** and then to **SYN SENT**.

Data Exchange

- Established:** Once the connection is established, data exchange occurs between the client and server.
- Transitions:** The client can transition from **ESTABLISHED** to **FIN WAIT 1** (if it wants to close) or **SYN RECEIVED** (if it receives a new SYN packet).

Connection Termination

Active Close

- FIN WAIT 1:** The client sends a **FIN/ACK** packet to the server.
- CLOSING:** The server receives the **FIN/ACK** packet and sends a **FIN+ACK/ACK** packet back to the client.
- TIME WAIT:** The client receives the **FIN+ACK/ACK** packet and enters the **TIME WAIT** state. A timeout period is shown.
- CLOSED:** After the timeout, the client transitions back to the **CLOSED** state.

Passive Close

- CLOSE WAIT:** The server receives the **FIN/ACK** packet and enters the **CLOSE WAIT** state.
- LAST ACK:** The server sends a **CLOSE/FIN** packet to the client.
- ACK/-:** The client receives the **CLOSE/FIN** packet and sends an **ACK/-** packet back to the server.
- CLOSED:** The server receives the **ACK/-** packet and transitions back to the **CLOSED** state.

Handwritten Annotations:

- Conn. Est.:** Points to the 3-way handshake process.
- Conn. Term:** Points to the connection termination process.
- Data exchange occurs:** Points to the **ESTABLISHED** state.
- ACK/-:** Points to the acknowledgment packet sent by the client in the passive close process.
- FIN/ACK:** Points to the **FIN/ACK** packet sent by the client in the active close process.
- FIN+ACK/ACK:** Points to the **FIN+ACK/ACK** packet sent by the server in the active close process.
- CLOSE/FIN:** Points to the **CLOSE/FIN** packet sent by the server in the passive close process.
- Timeout:** Points to the **TIME WAIT** state in the active close process.
- Go back to start:** Points to the **CLOSED** state at the end of the active close process.

Time-Wait State



- Wait in time-wait for $2 * \text{MSL}$ (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



"bind failed"

Conn 1:

remote Port #

src dest, src dest
ip ip port port

data

→ closed.

Conn 2:

Abort

RST

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