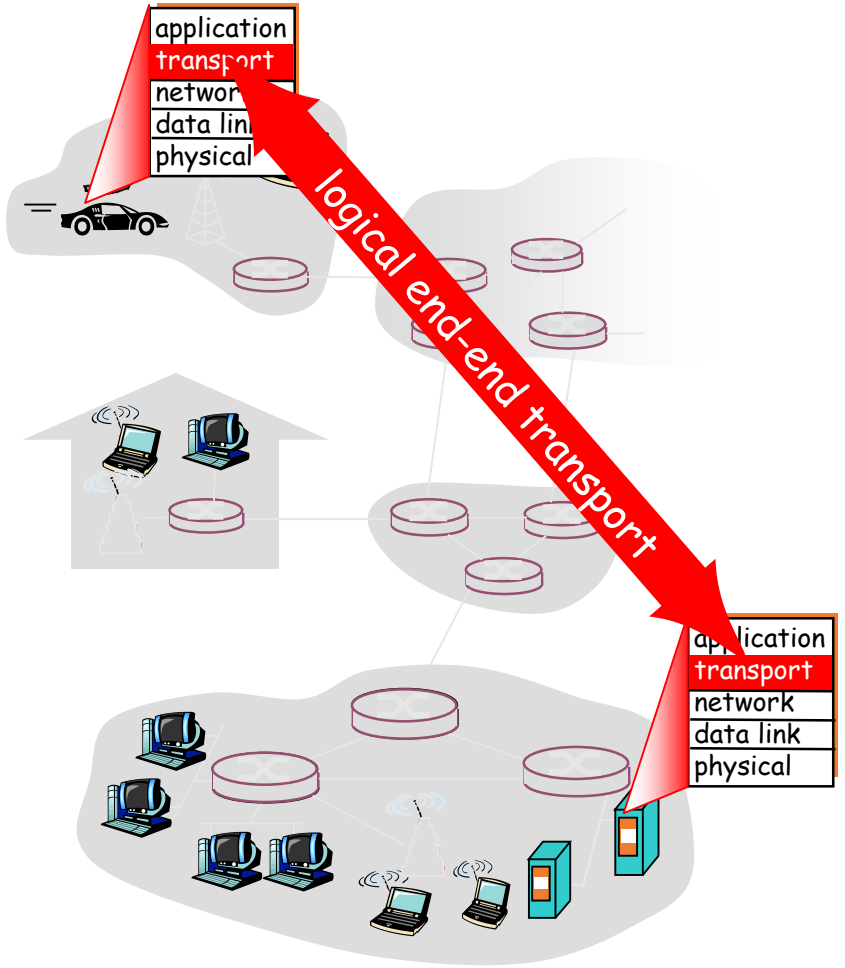
Transport Layer

Transport services and protocols

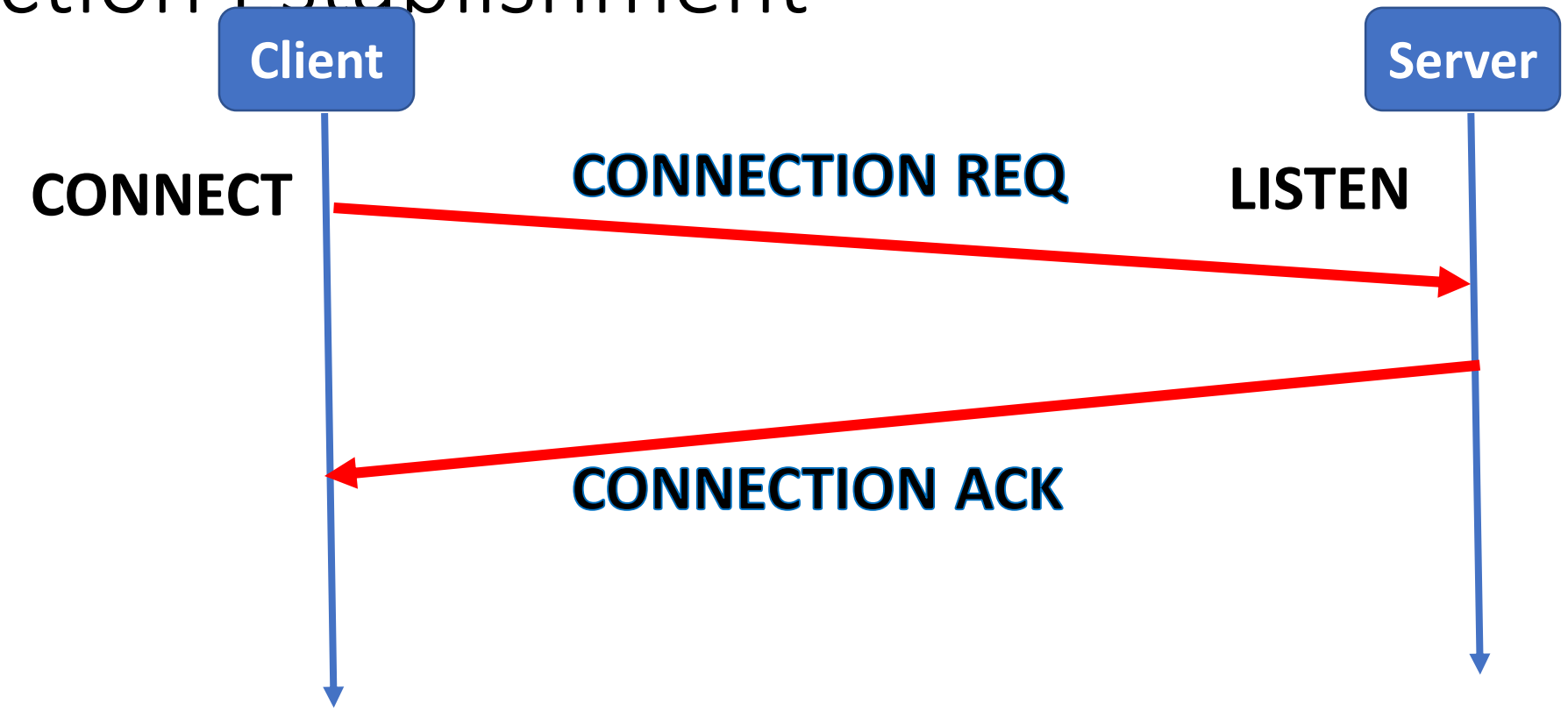
- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Transport Layer Services

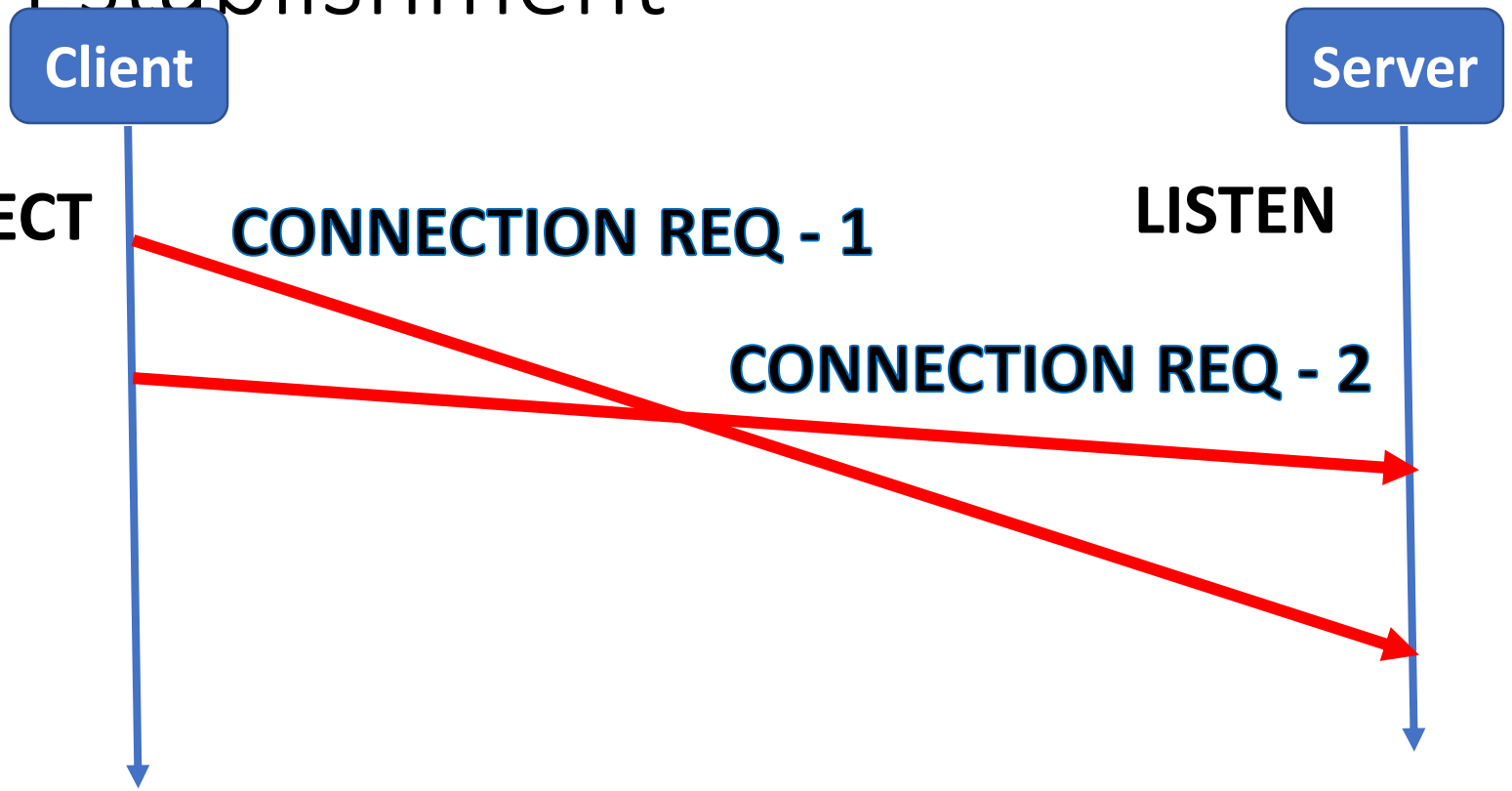
- Connection establishment
- Error and flow control
- Congestion control

Connection Establishment



- **This is a simple primitive for connection establishment – but does this work good?**

Connection Establishment



It may happen that the server has crashed and reinitiated the connection. So distinguishing between these two is essential

- **How will the server differentiate whether CONNECTION REQ-1 is a new connection request or a duplicate of the CONNECTION REQ-2?**

Connection Establishment

- Consider a scenario when the network can lose, delay, corrupt and duplicate packets (the underline network layer uses unreliable data delivery)

- Consider retransmission for ensuring reliability – every packet uses different paths to reach the destination

- Packets may be delayed and got struck in the network congestion, after the timeout, the sender assumes that the packets have been dropped, and retransmits the packets

Connection Establishment

- **Delayed duplicates** create a huge confusion in the packet switching network.

- A major challenge in packet switching network is to develop **correct** or **at least acceptable** protocols for handling delayed duplicates

How to handle delayed duplicates

- **Solution 1: Give each connection a unique identifier chosen by the initiating party and put in each segment**
  - Can you see any problem in this approach?

How to handle delayed duplicates

- **Solution 1: Give each connection a unique identifier chosen by the initiating party and put in each segment**
  - Can you see any problem in this approach?
  - Problem 1: Need to store the identifier (or sequence numbers) in a table at both sender and receiver side.

How to handle delayed duplicates

- **Solution 1: Give each connection a unique identifier chosen by the initiating party and put in each segment**
  - Can you see any problem in this approach?
  - Problem 1: Need to store the identifier (or sequence numbers) in a table at both sender and receiver side.

Connection Establishment – Handling Delayed Duplicates

- **Solution 2: Devise a mechanism to kill off aged packets that are still hobbling about (Restrict the packet lifetime)**

Connection Establishment – Handling Delayed Duplicates

- **Solution 2: Devise a mechanism to kill off aged packets that are still hobbling about (Restrict the packet lifetime)**

- Two ways to restrict packet lifetime
  - Timestamping each packet – define the lifetime of a packet in the network, need time synchronization across each router.

Connection Establishment – Handling Delayed Duplicates

- **Solution 2: Devise a mechanism to kill off aged packets that are still hobbling about (Restrict the packet lifetime)**

- Two ways to restrict packet lifetime
  - **Putting a hop count in each packet – initialize to a maximum value and decrement each time the packet traverses a single hop (most feasible implementation)**
  - Timestamping each packet – define the lifetime of a packet in the network, need time synchronization across each router.

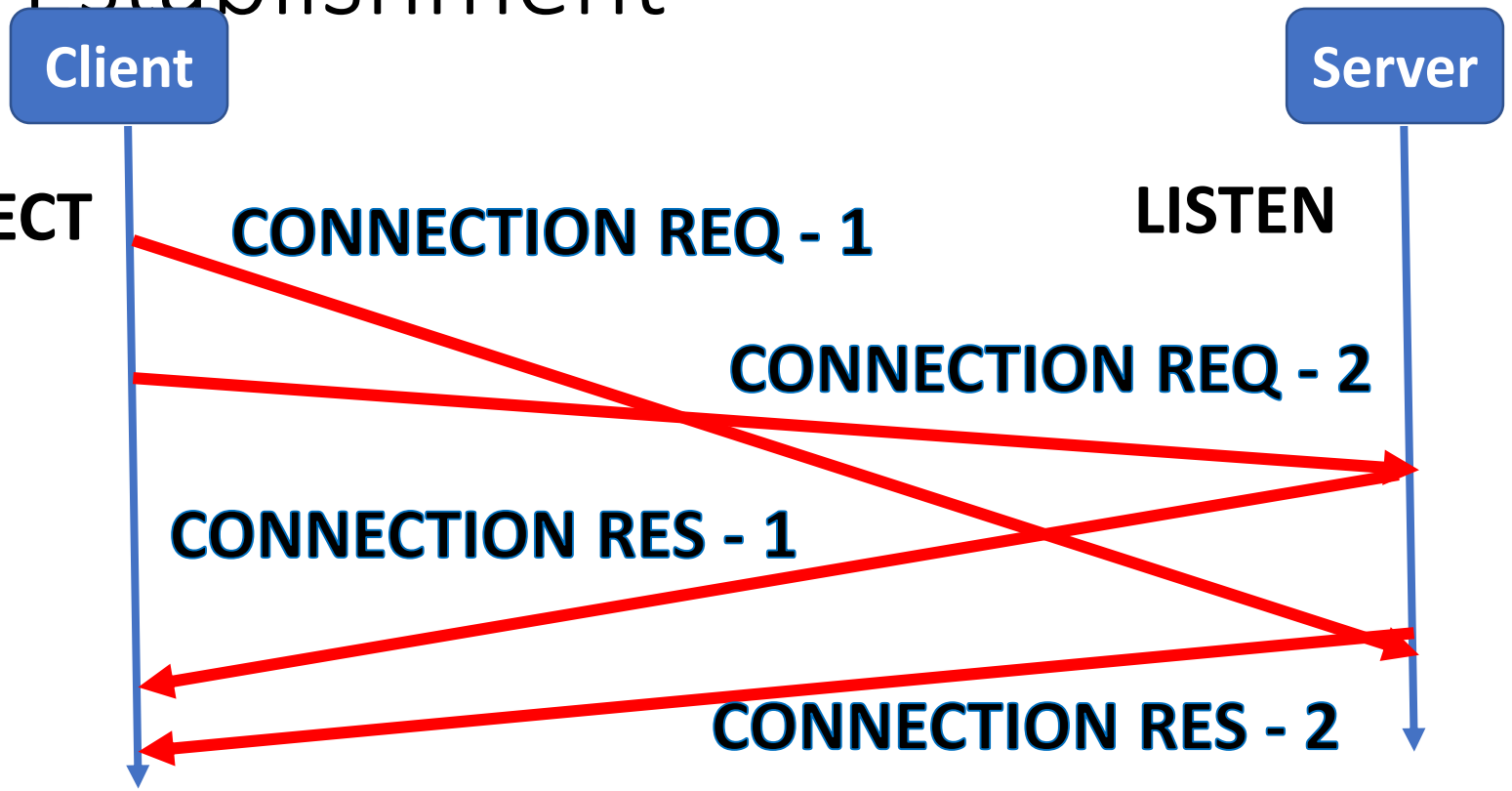
Connection Establishment – Handling Delayed Duplicates

- **Solution 2: Devise a mechanism to kill off aged packets that are still hobbling about (Restrict the packet lifetime)**

- Two ways to restrict packet lifetime
  - **Putting a hop count in each packet – initialize to a maximum value and decrement each time the packet traverses a single hop (most feasible implementation)**
  - Timestamping each packet – define the lifetime of a packet in the network, need time synchronization across each router.

- **Design Challenge: We need to guarantee not only that a packet is dead, but also that all acknowledgements to it are also dead**

Connection Establishment



It may happen that the server has crashed and reinitiated the connection. So distinguishing between these two is essential

- **How will the server differentiate whether CONNECTION RES-1 is a new connection response or a response of the CONNECTION RES-2?**

Connection Establishment – Handling Delayed Duplicates

- **Design Challenge: We need to guarantee not only that a packet is dead, but also that all acknowledgements to it are also dead**

- **May be, we can use Solution 1 and 2 together for meeting the challenge**

Connection Establishment – Handling Delayed Duplicates

- Let us define a maximum packet lifetime T – If we wait a time T secs after a packet has been sent, we can be sure that all traces of it (packet and its acknowledgement) are now gone

Connection Establishment – Handling Delayed Duplicates

- Let us define a maximum packet lifetime T

- Rather than a physical clock (clock synchronization in the Internet is difficult to achieve), let us use a virtual clock – **sequence number will represent a virtual clock**

Connection Establishment – Handling Delayed Duplicates

- Let us define a maximum packet lifetime T
- **sequence number generated based on the clock ticks**
- **Label segments with sequence numbers that will not be reused within T secs.**
- **At most one packet with a given sequence number may be outstanding at any given time**

Connection Establishment – Handling Delayed Duplicates

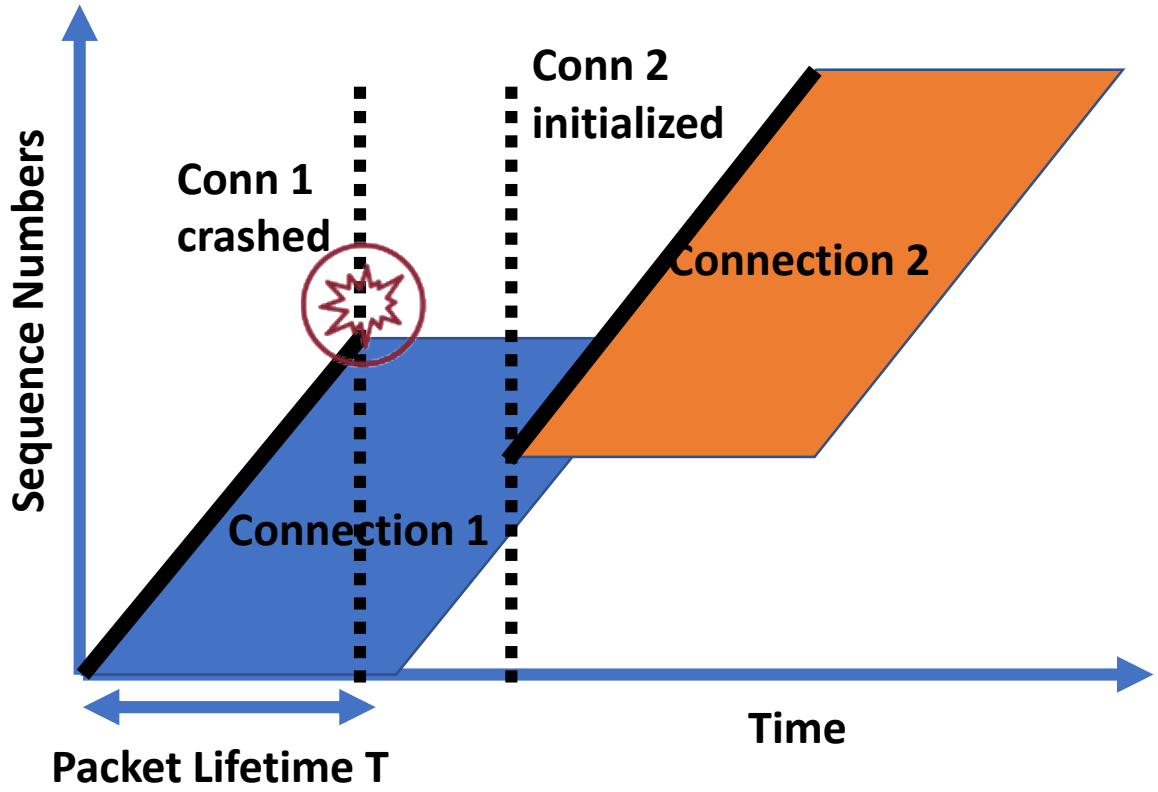
- **At most one packet with a given sequence number may be outstanding at any given time**

- If a receiver receives two segments having the same sequence number within a duration T, then one packet must be the duplicate. The receiver then discards the duplicate packets.

Connection Establishment – Handling Delayed Duplicates

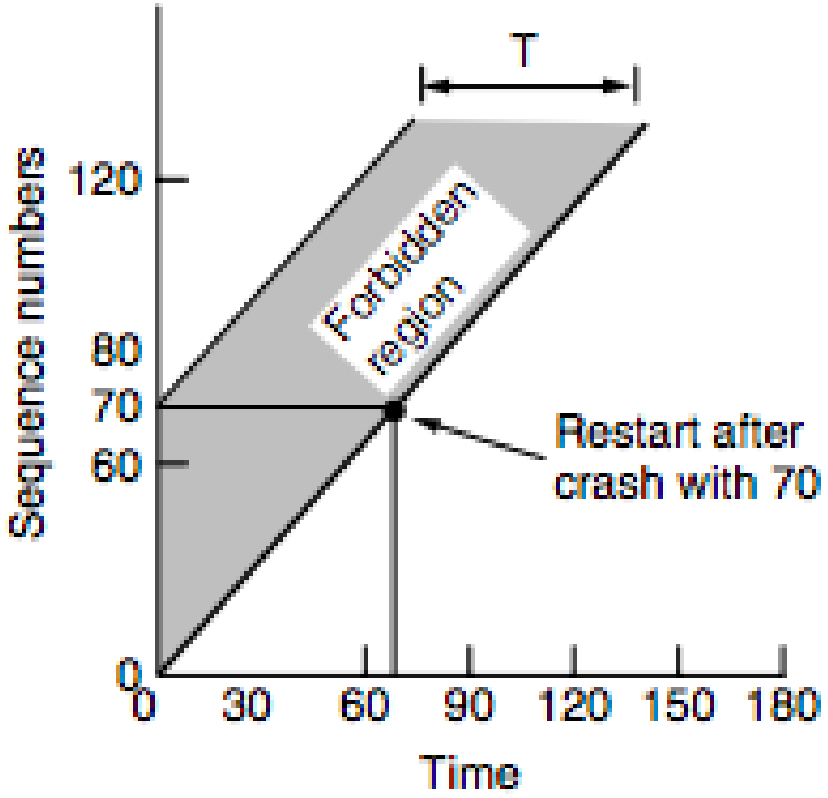
- **What happens in case of crash of sender/receiver??**
- **Where to start from, what sequence number to use??**

Why Initial Sequence Number is Important



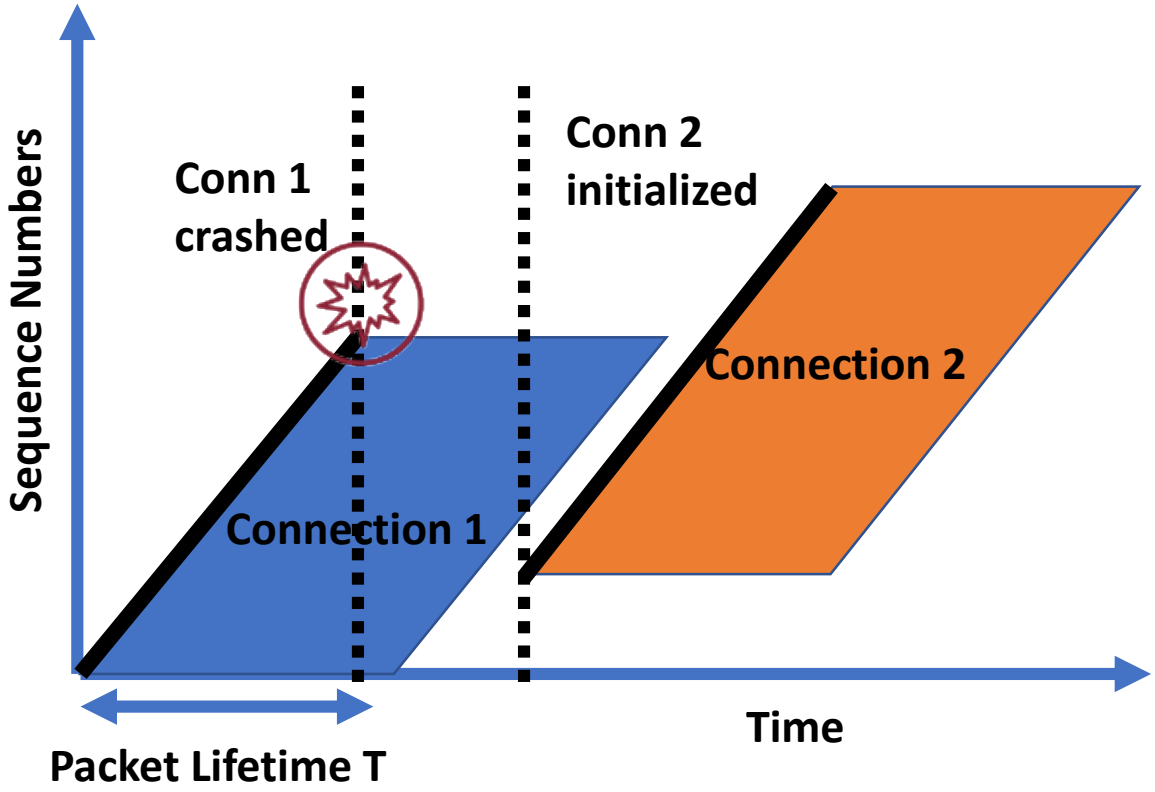
- A Delayed duplicate packet of connection 1 can create a confusion for connection 2

Connection Establishment – Handling Delayed Duplicates

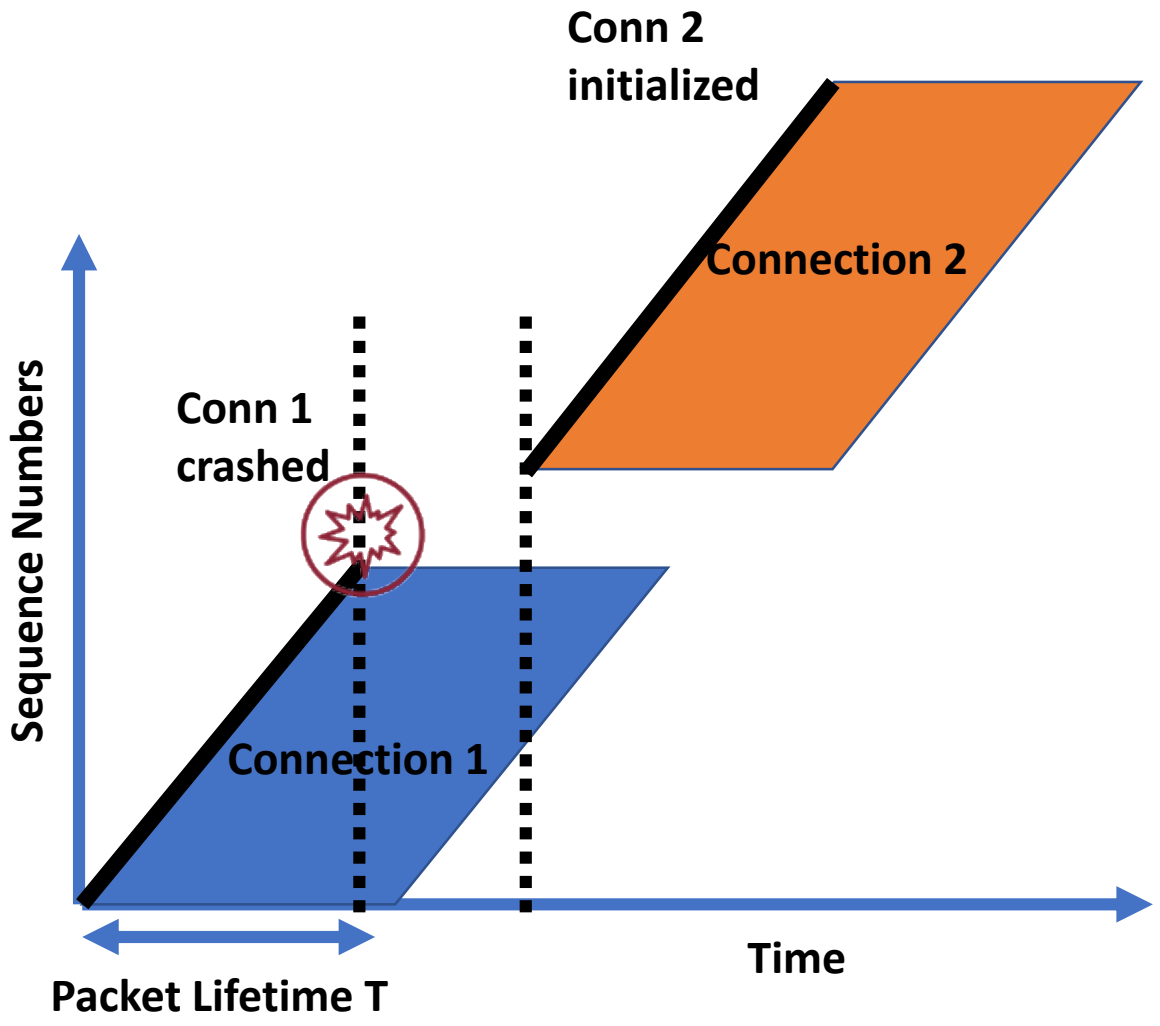


- **Adjust the initial sequence numbers properly - A host does not restart with a sequence number in the forbidden region, based on the sequence number it used before crash and the time duration T.**

What We Ideally Want? Or …



What We Ideally Want? Either …

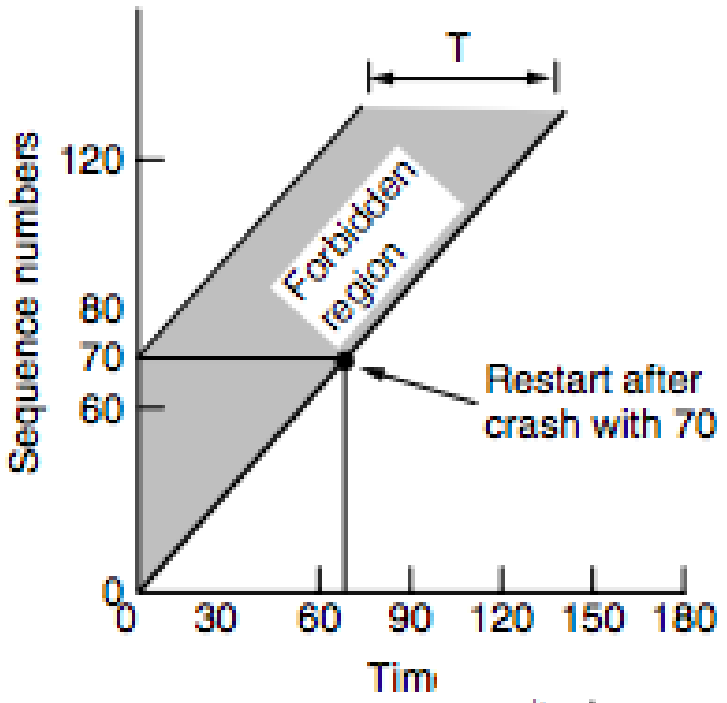


One problem while choosing a sequence number

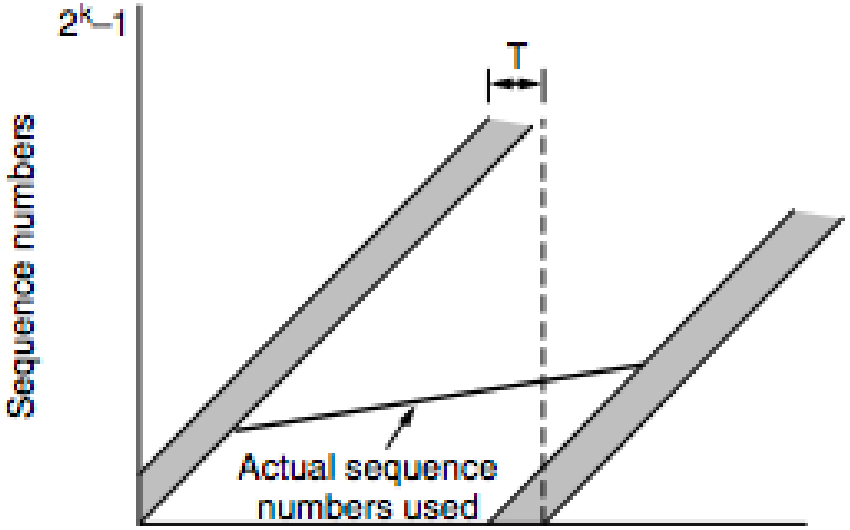
- The period T and the rate of packets per second determine the size of the sequence number

How do We Ensure that Packet Sequence Numbers are Out of the Forbidden Region

- Two possible source of problems
  - A host sends too much data too fast on a newly opened connection



  - The data rate is too slow that the sequence number for a previous connection enters the forbidden region for the next connection



Adjusting the Sending Rate based on Sequence Numbers

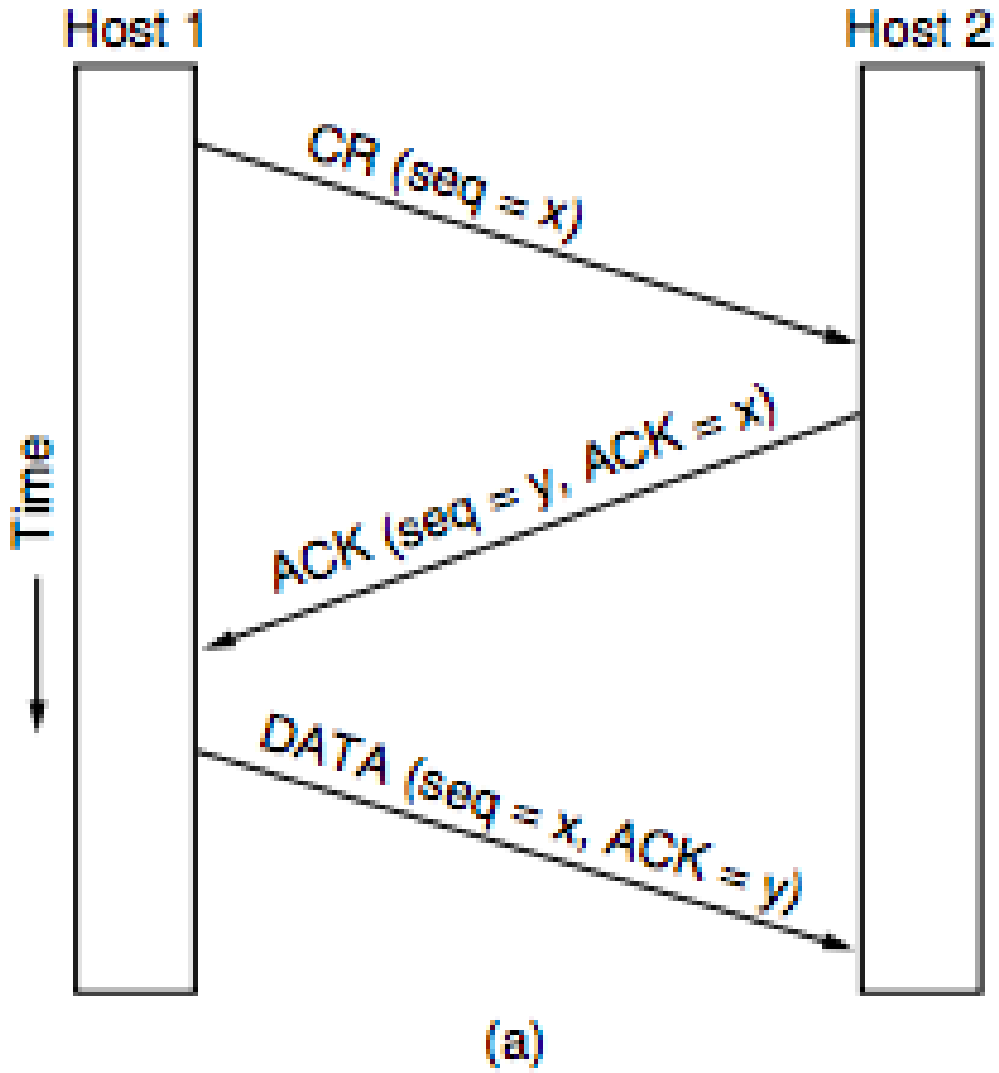
- The maximum data rate on any connection is one segment per clock tick
  - Clock ticks (inter-packet transmission duration) is adjusted based on the sequences acknowledged – **ensure that no two packets are there in the network with same sequence number**
  - **We call this mechanism as self-clocking (used in TCP)**
  - Ensures that the sequence numbers do not warp around too quickly (RFC 1323)

Adjusting the Sending Rate based on Sequence Numbers

- The maximum data rate on any connection is one segment per clock tick
  - Clock ticks (inter-packet transmission duration) is adjusted based on the sequences acknowledged – **ensure that no two packets are there in the network with same sequence number**
  - **We call this mechanism as self-clocking (used in TCP)**
  - Ensures that the sequence numbers do not warp around too quickly (RFC 1323)

- **We do not remember sequence number at the receiver:** Use a **three way handshake** to ensure that the connection request is not a repetition of an old connection request
  - The individual peers validate their own sequence number by looking at the acknowledgement (ACK)
  - **Positive synchronization among the sender and the receiver**

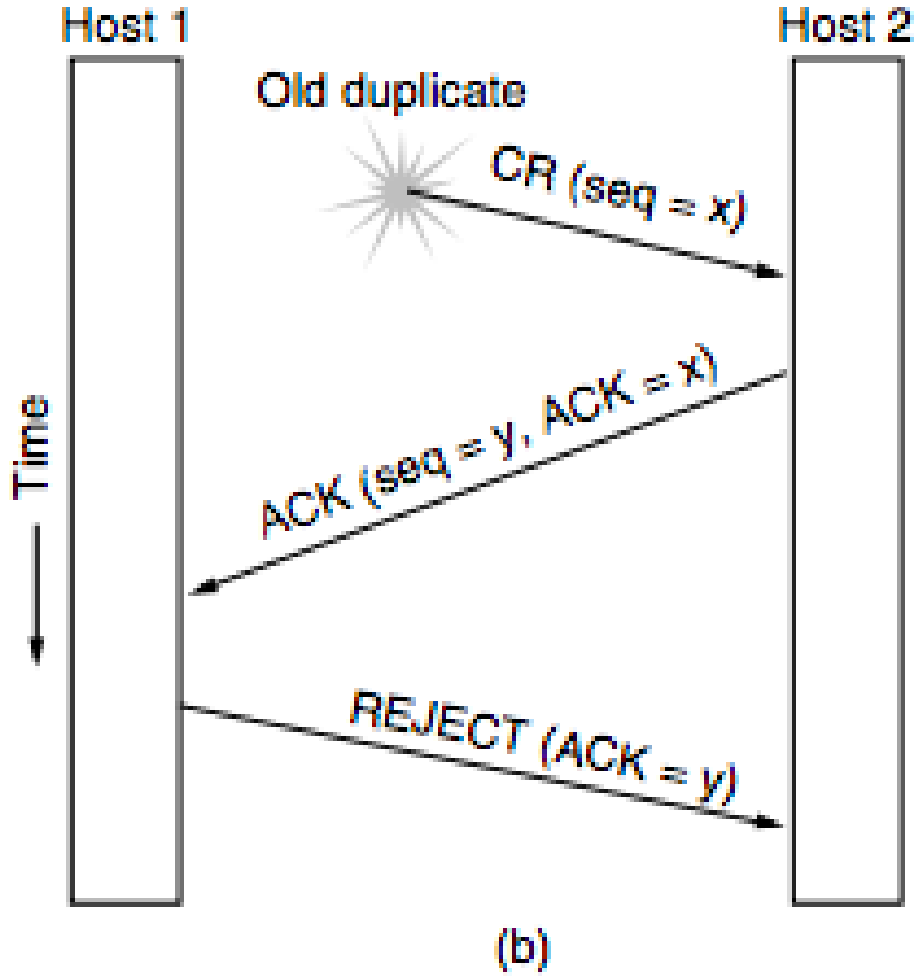
Three Way Handshake



(a)

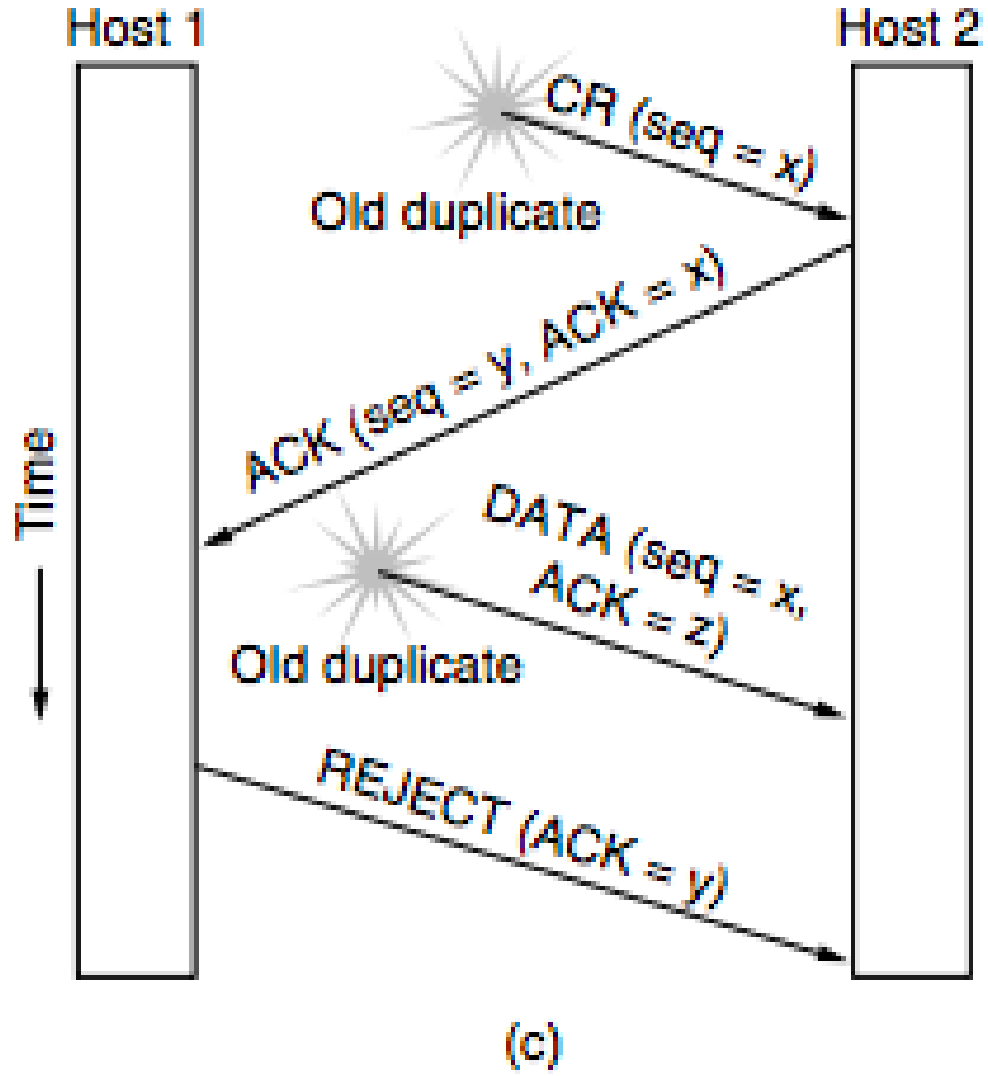
- By looking at the ACK, Host 1 ensures that Sequence number x does not belong to the forbidden region of any previously established connection
- By looking at the ACK in DATA, Host 2 ensures that sequence number y does not belong to the forbidden region of any previously established connection

Three Way Handshake – CONNECTION REQUEST is a Delayed Duplicate



Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Three Way Handshake – CONNECTION REQUEST and ACKNOWLEDGEMENT both are Delayed Duplicates



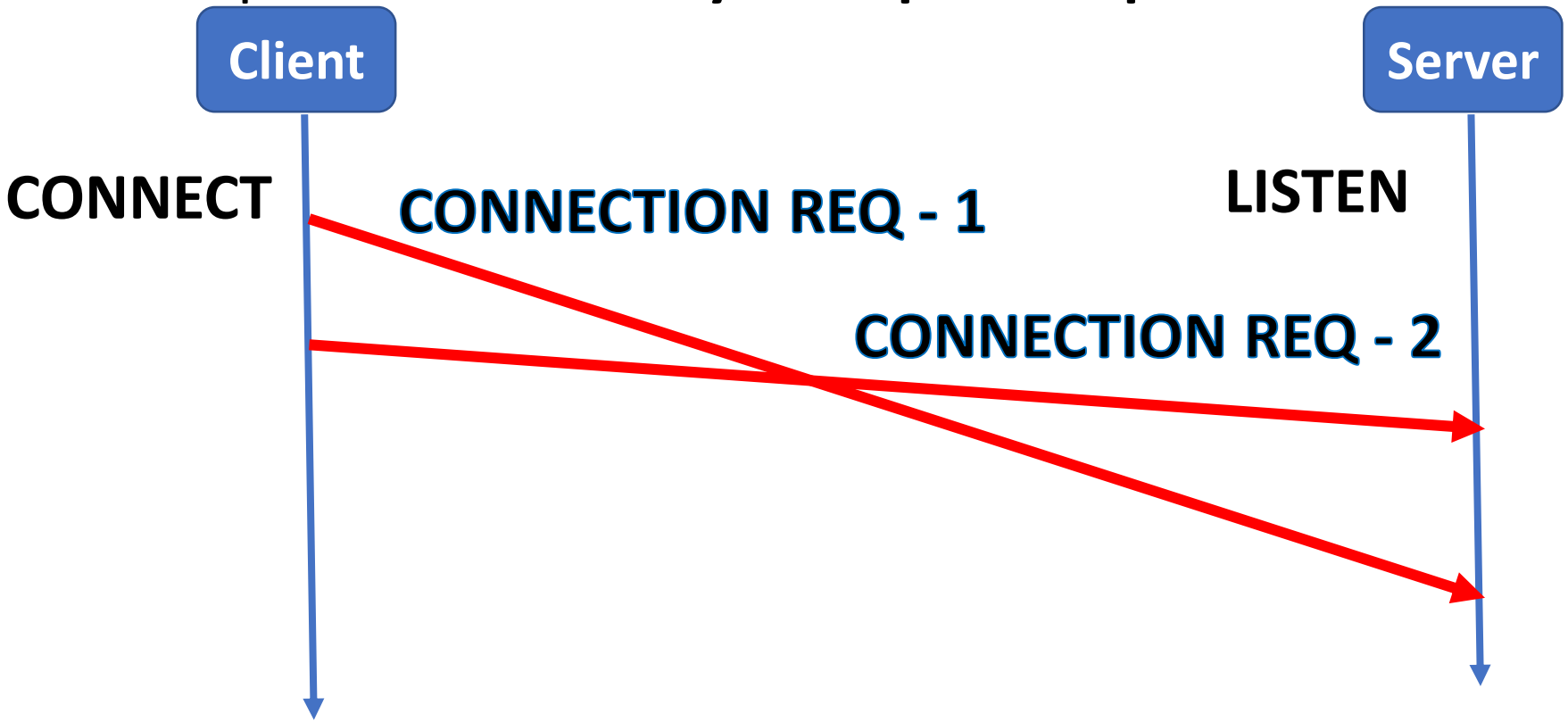
Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Sequence Number Adjustment

- **Two important requirements (*Tomlinson 1975, Selecting Sequence Numbers*)**
  - **Sequence numbers must be chosen such that a particular sequence number never refers to more than one byte (for byte sequence numbers) at any one time (how to choose the initial sequence number)**
  - **The valid range of sequence numbers must be positively synchronized between the sender and the receiver, whenever a connection is used (three way handshaking followed by the flow control mechanism – once connection is established, only send the data with expected sequence numbers)**

Recap

- Connection Establishment procedure – **Delayed Duplicates problem**



Recap

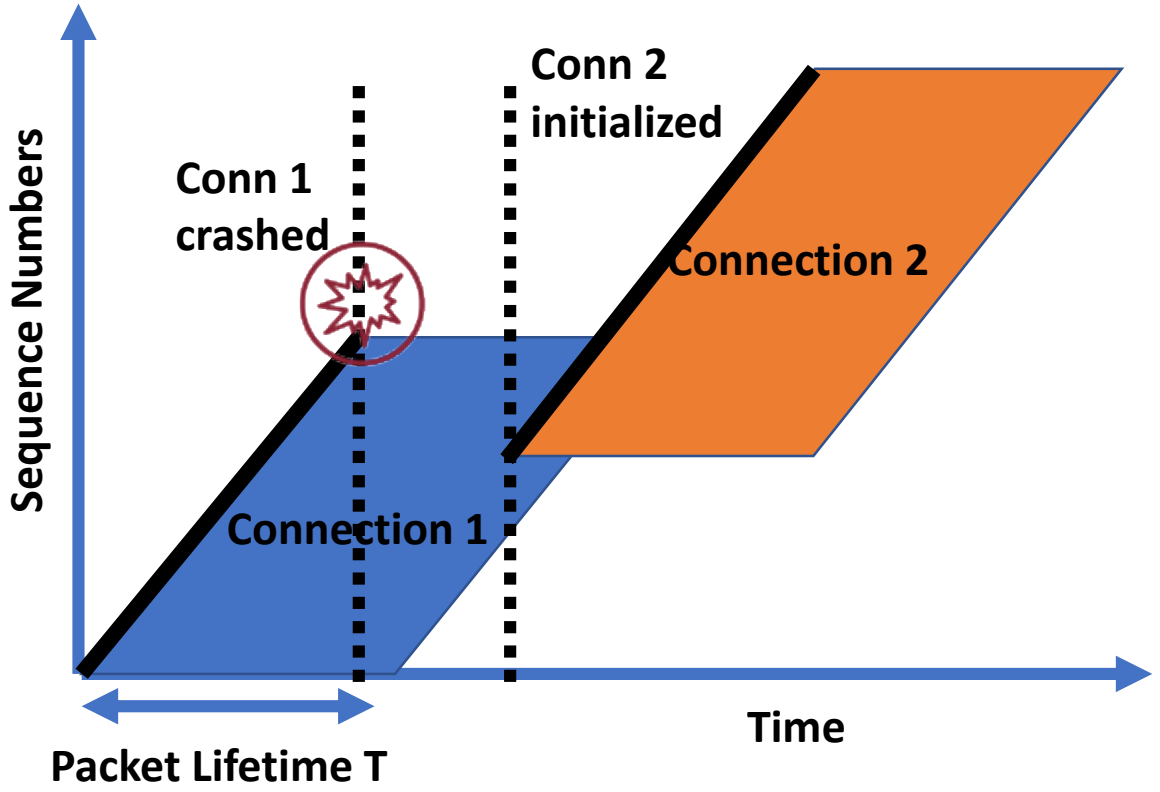
- **Connection Establishment – Handling Delayed Duplicates**

- **Solution:**
  - Let us define a maximum packet lifetime T
  - Label segments with sequence numbers that will not be reused within T secs.

- **At most one packet with a given sequence number may be outstanding at any given time**

Recap

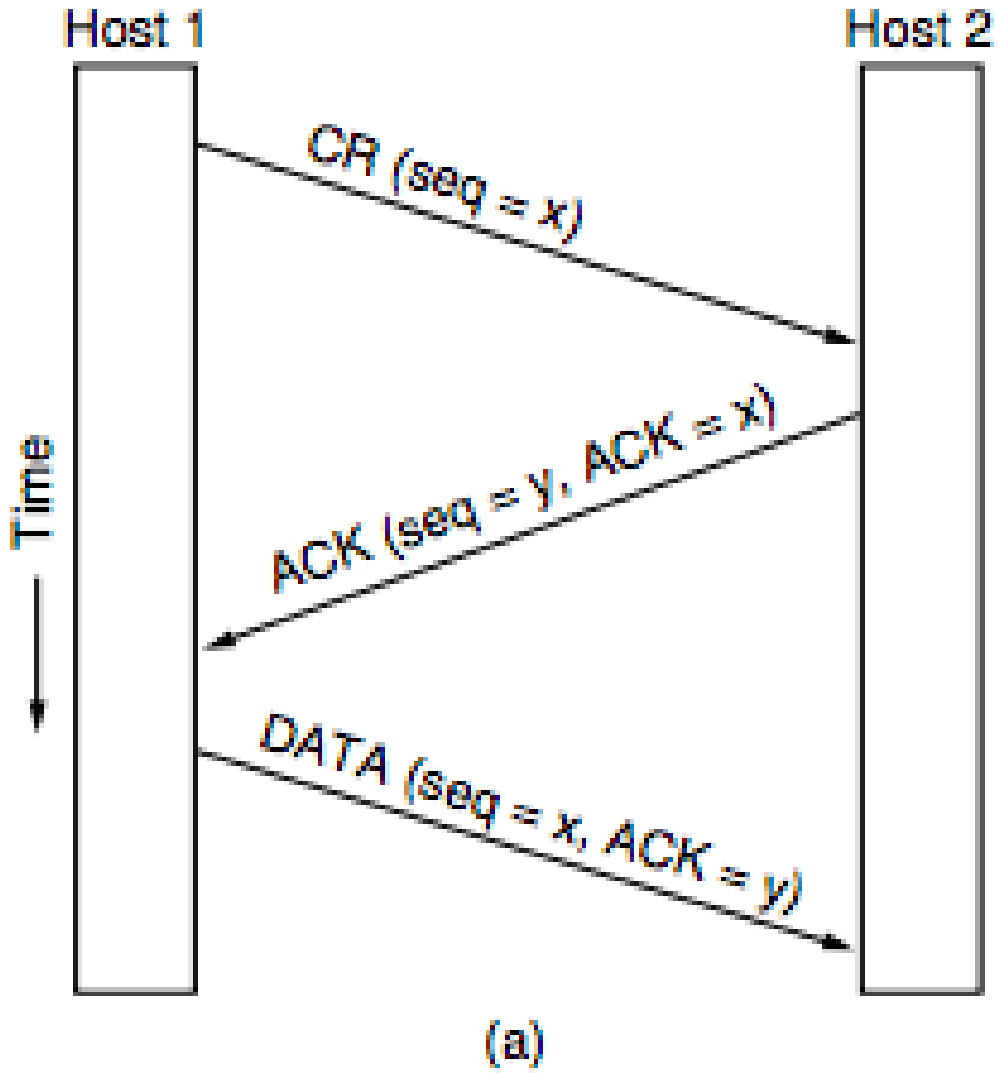
- Sequence Number Adjustment
  - **Choosing an Initial sequence number is important – as it does not clash with the sequence numbers currently used by another connection**



Recap

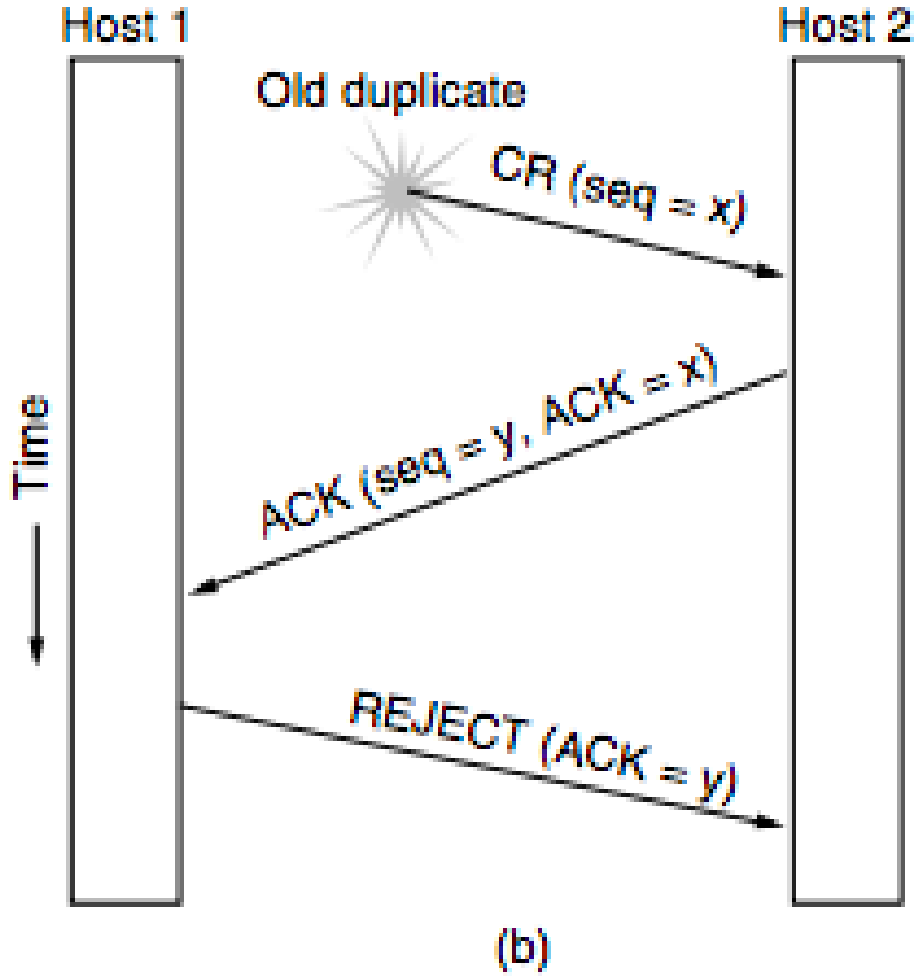
- Sequence Number Adjustment
  - **3-way handshake:** The valid range of sequence numbers must be positively synchronized between the sender and the receiver, whenever a connection is used.
  - Receiver need to inform sender correctly which connection it is acknowledging, so that the sender can accordingly distinguish an old connection request from the current/new one

Recap - Three Way Handshake



- By looking at the ACK, Host 1 ensures that Sequence number x does not belong to the forbidden region of any previously established connection
- By looking at the ACK in DATA, Host 2 ensures that sequence number y does not belong to the forbidden region of any previously established connection

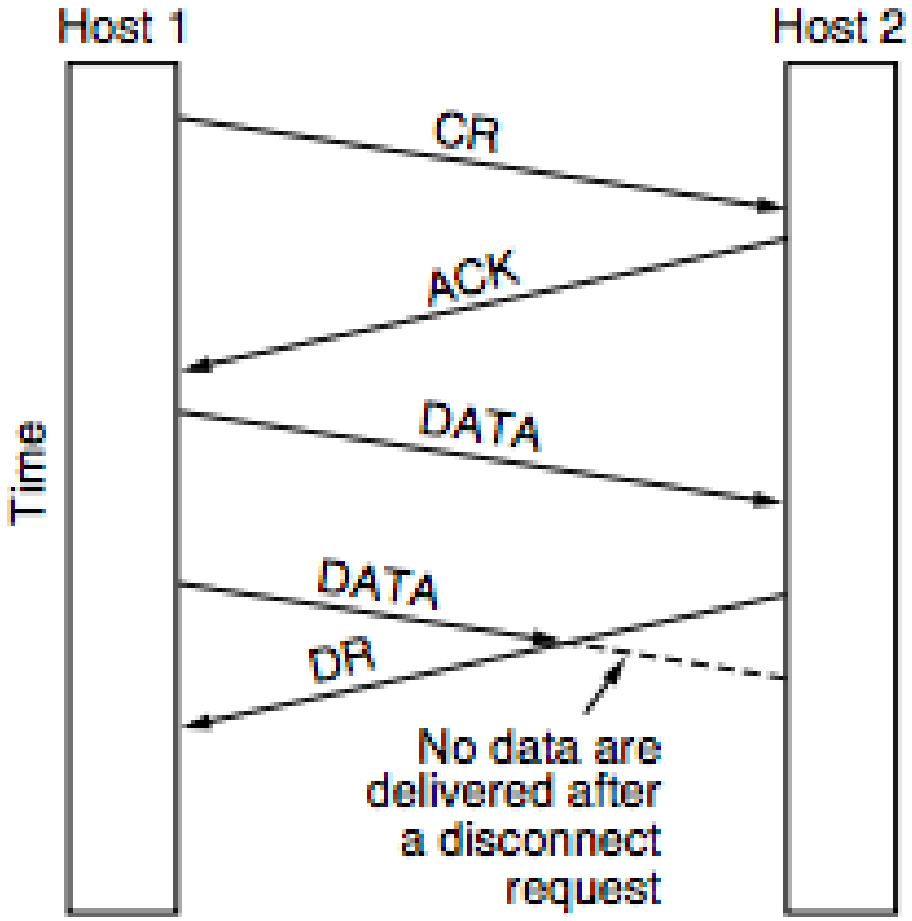
Three Way Handshake – CONNECTION REQUEST is a Delayed Duplicate



Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Connection Release – Asymmetric Release

- When one party hangs up, the connection is broken
- This may results in data loss

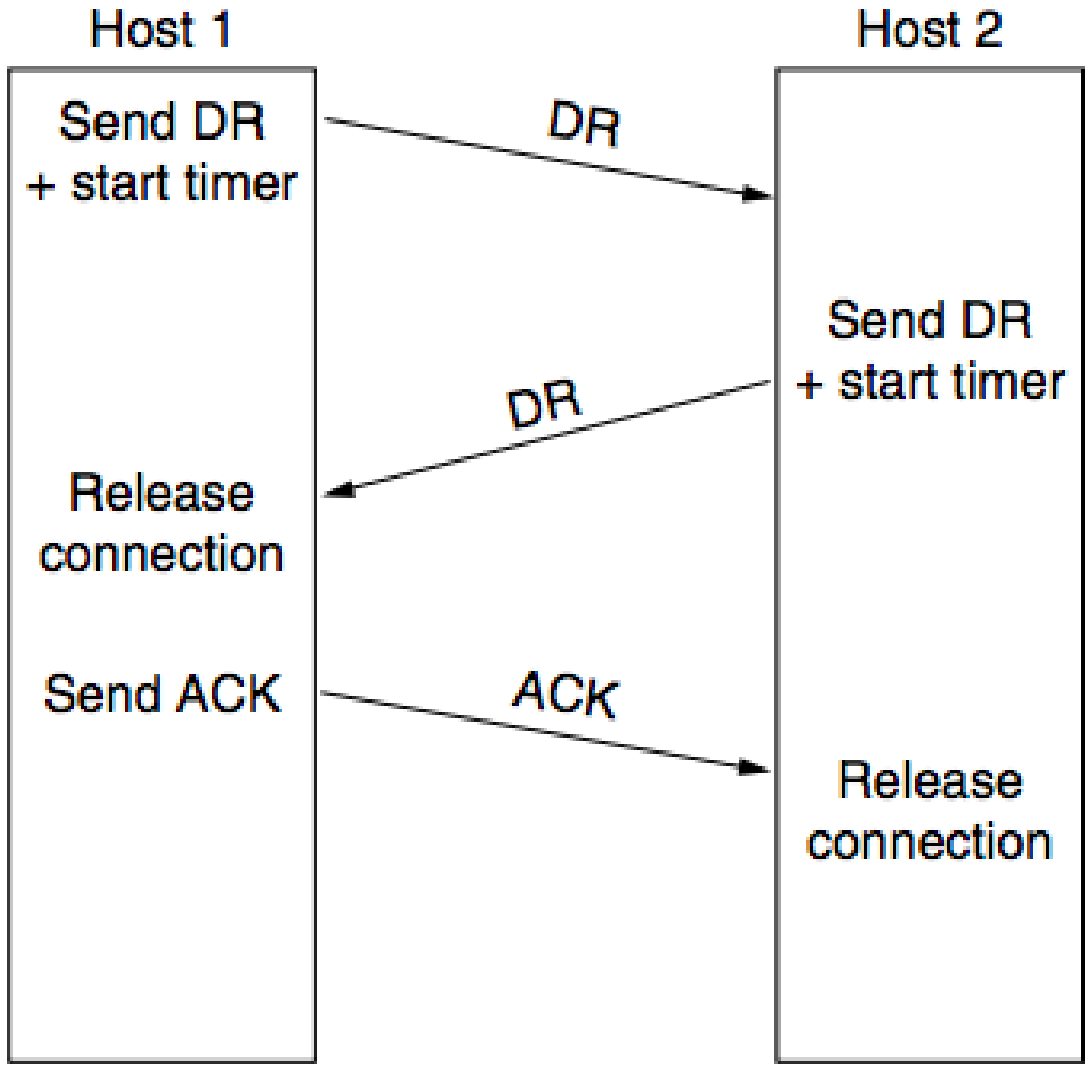


Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Connection Release – Symmetric Release

- Treats the connection as two separate unidirectional connections and requires each one to be released separately
- Does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.
- What can be a protocol for this?
  - **Host 1: “I am done”**
  - **Host 2: ”I am done too”**
- **Does this protocol work good always?**

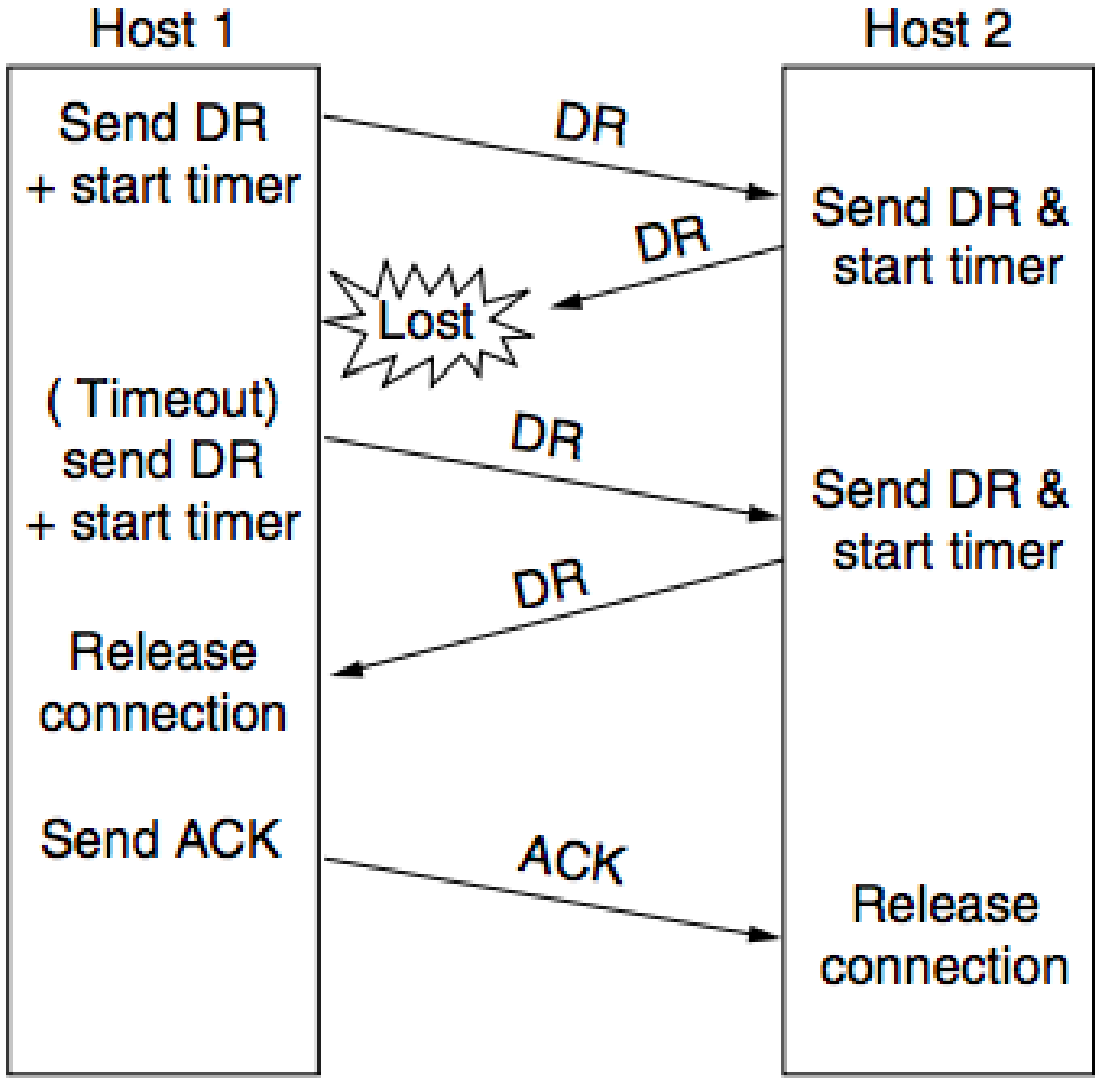
Connection Release



(a)

Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

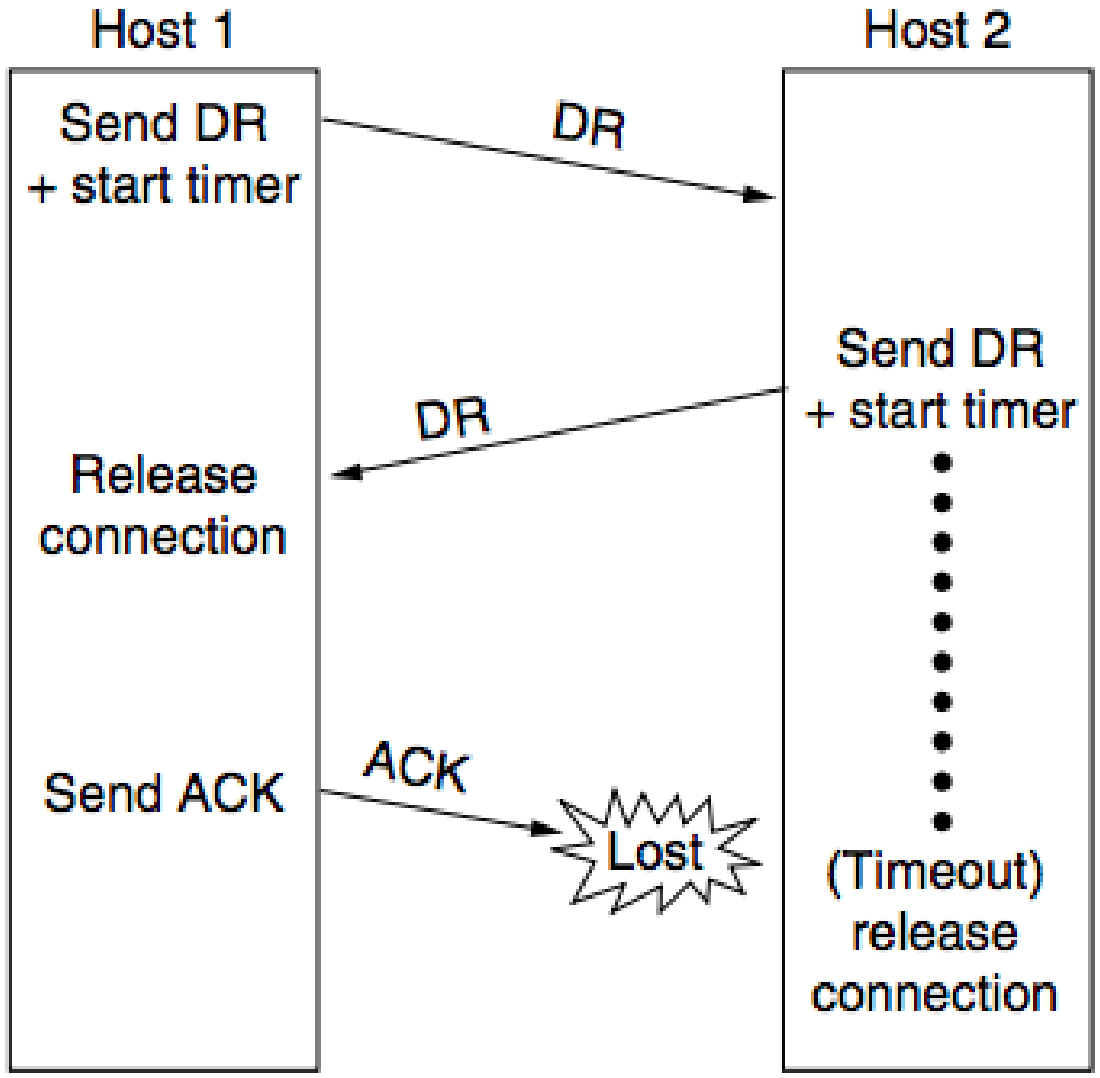
Connection Release – Response Lost



(c)

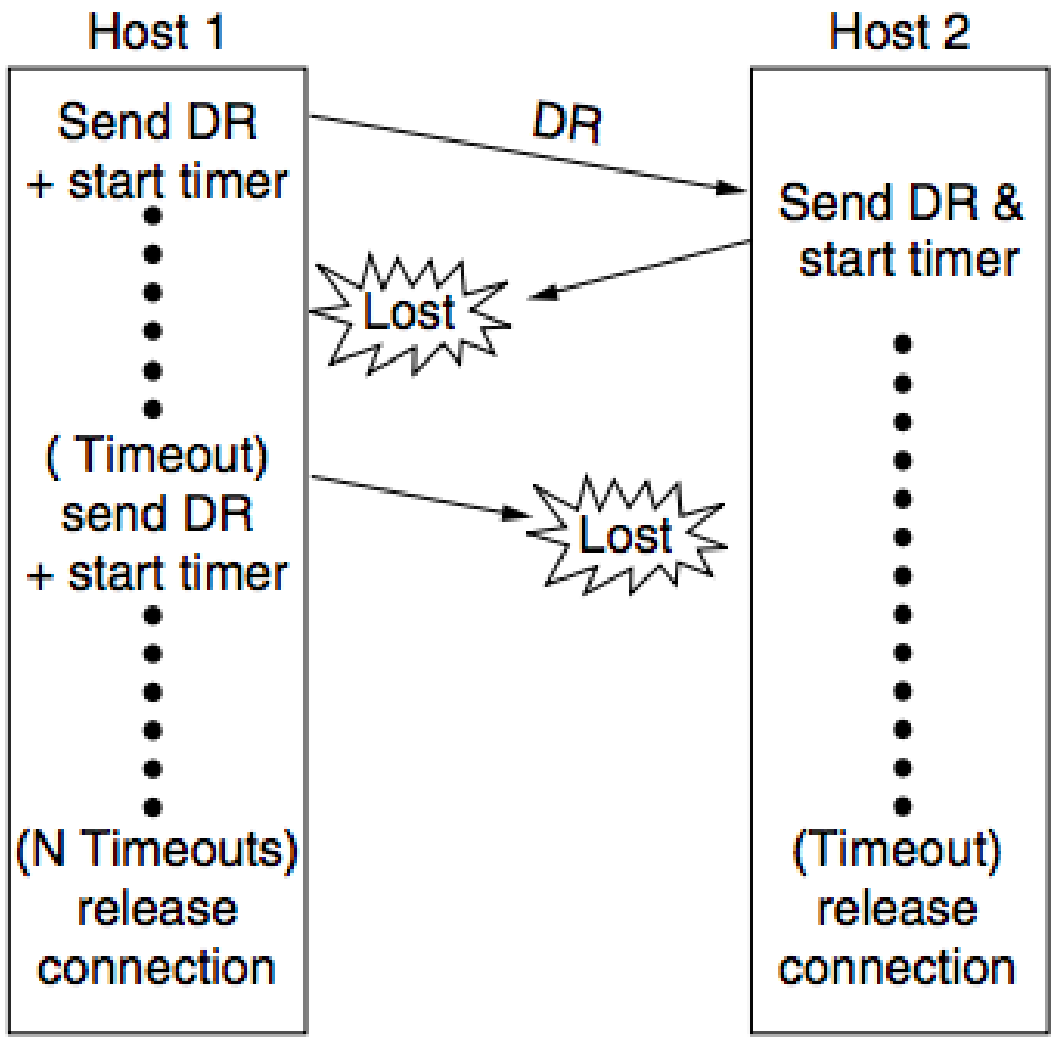
Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Connection Release – Final ACK Lost



Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Connection Release – Response Lost and Subsequent DRs Lost

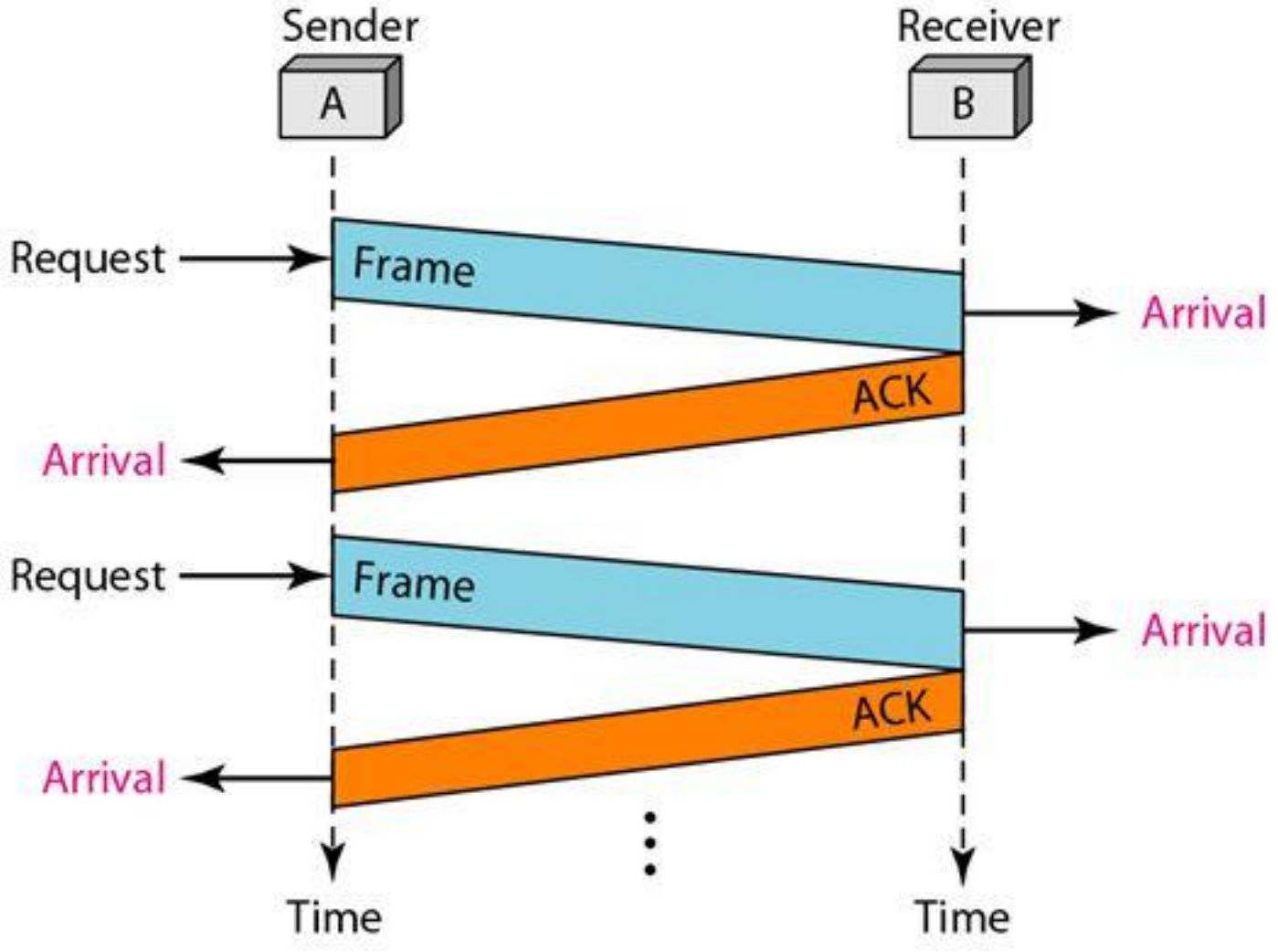


Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Flow Control Algorithms

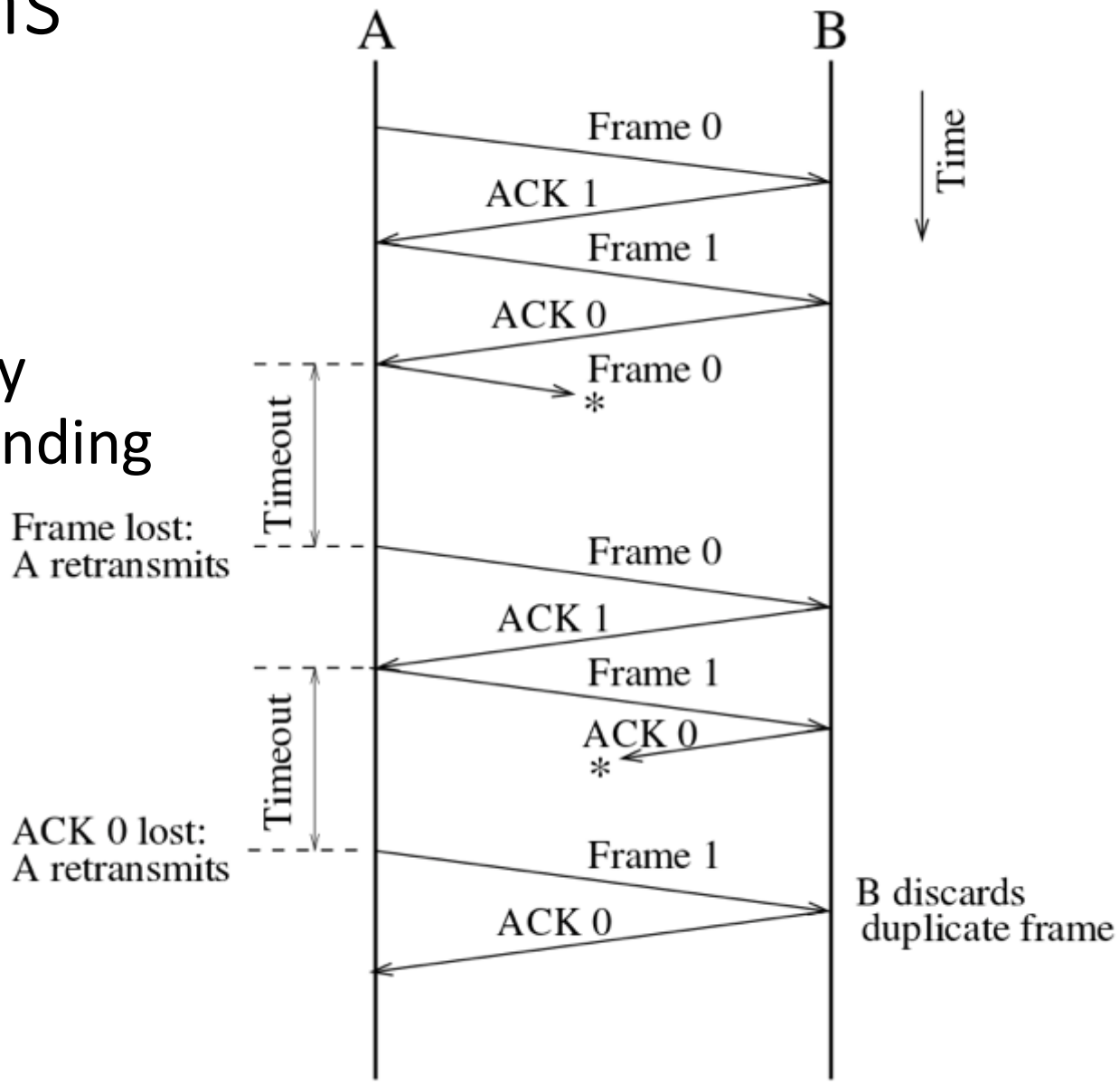
- **Stop and Wait Flow**

sender sends one frame and then waits for an acknowledgement before proceeding to send next frame.



Flow Control Algorithms

- **Stop and Wait (Noisy Channel):**
- Use sequence numbers to individually identify each frame and the corresponding acknowledgement
- **Automatic Repeat Request (ARQ)**



Flow Control Algorithms

- Three things to consider when designing a reliable flow control algorithm
  - Sequence Numbers – for distinguishing older packets and new ones (both data as well as ACKs).
  - Receiver Feedback – using ACKs and NAKs
  - Retransmission – after a fixed amount of Time, in case of lost data or ACKs

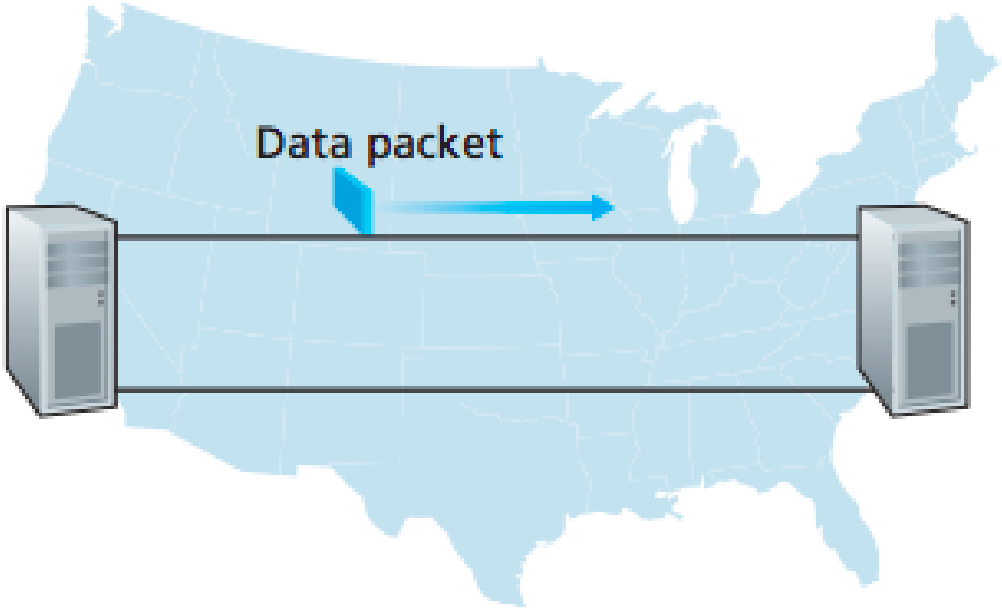
Stop-and-Wait ARQ:

- When a frame is sent, the sender starts the timeout counter.
- If acknowledgement of frame comes in time, the sender transmits the next frame in queue.
- If acknowledgement does not come in time, the sender assumes that either the frame or its acknowledgement is lost in transit. Sender retransmits the frame and starts the timeout counter.
- If a negative acknowledgement is received, the sender retransmits the frame.

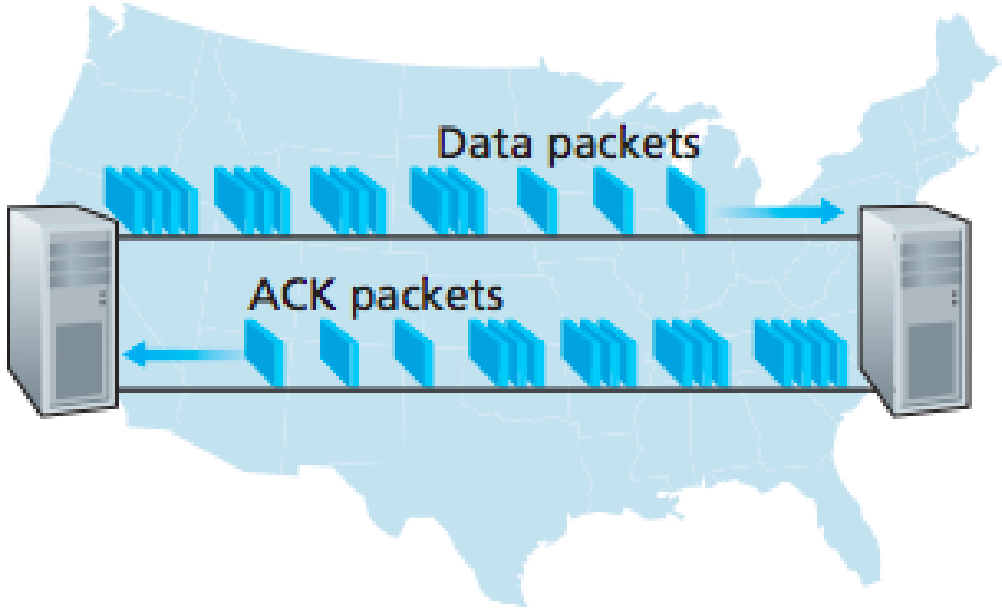
Problem with Stop and Wait

- Every packet needs to wait for the acknowledgement of the previous packet.
- For bidirectional connections – use two instances of the stop and wait protocol at both directions – further waste of resources
- A possible solution: Piggyback data and acknowledgement from both the directions
- Reduce resource waste based on **sliding window protocols (a pipelined protocol)**

Stop and Wait versus Sliding Window (Pipelined)



a. A stop-and-wait protocol in operation



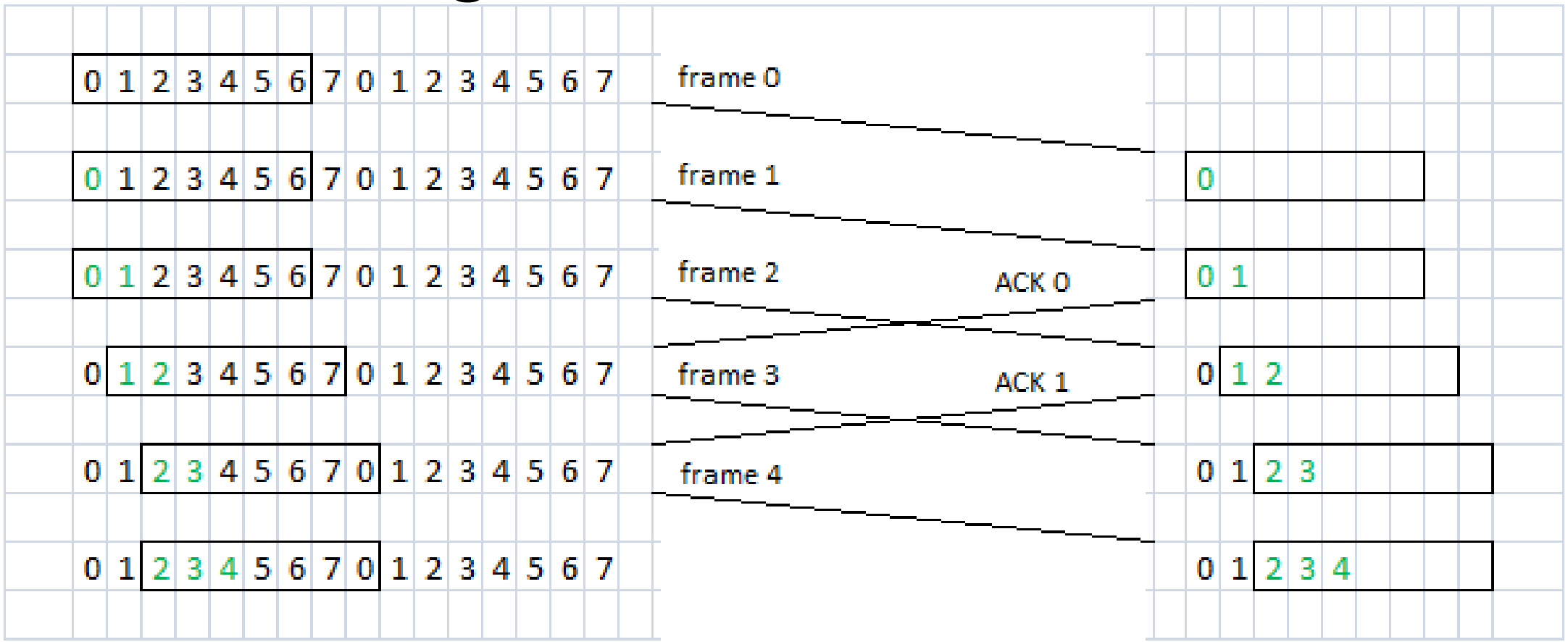
b. A pipelined protocol in operation

Source: Computer Networks, Kurose, Ross

Sliding Window Protocols

- Each outbound segment contains a sequence number – from 0 to some maximum (2^n-1 for a n bit sequence number)
- The sender maintains a set of sequence numbers corresponding to frames it is permitted to send (**sending window)**
- The receiver maintains a set of frames it is permitted to accept (**receiving window)**

Sliding Window Protocols – Sending Window and Receiving Window



Sliding window Protocol

Source: http://ironbark.xtelco.com.au/subjects/DC/lectures/13/

Sliding Window for a 3 bit Sequence Number

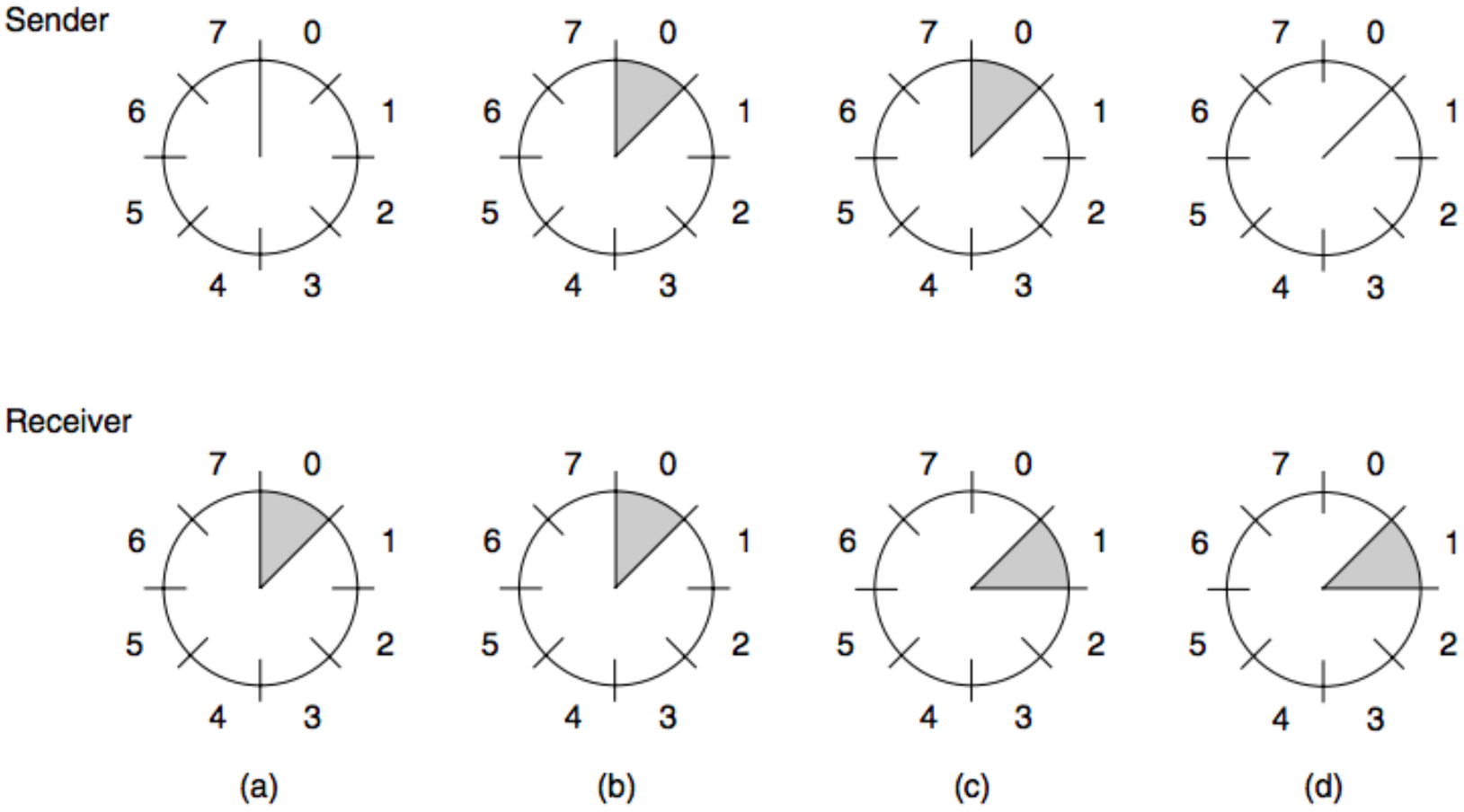


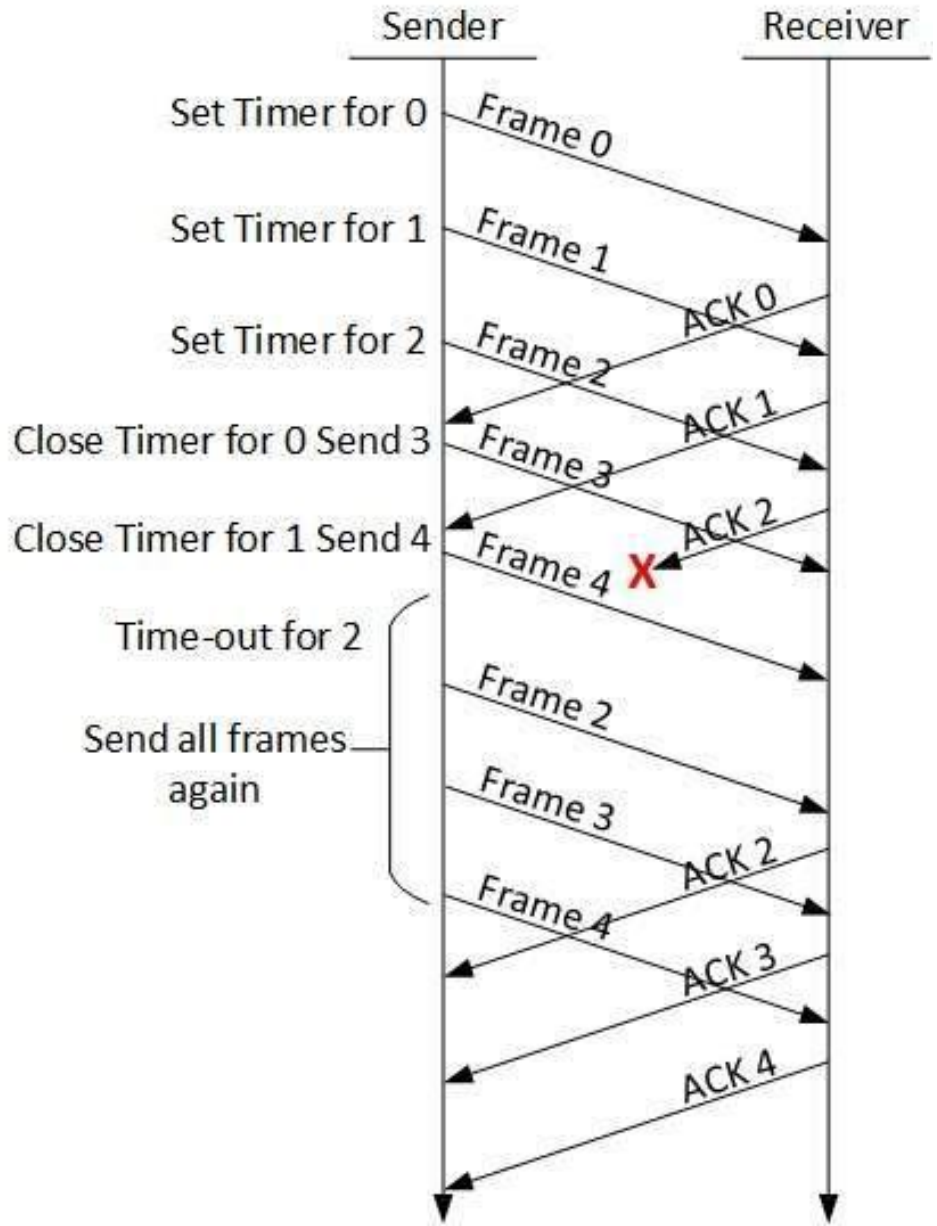
Figure 3-15. A sliding window of size 1, with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Sliding Window Protocols in Noisy Channels

- **A timeout occurs if a segment (or the acknowledgment) gets lost**
- **How does the flow and error control protocol handle a timeout?**
- **Go Back N ARQ:** If segment N is lost, all the segments from segment N are retransmitted
- **Selective Repeat (SR) ARQ:** Only the lost packets are selectively retransmitted
  - **Negative Acknowledgement (NAK) or Selective Acknowledgements (SACK):** Informs the sender about which packets need to be retransmitted (not received by the receiver)

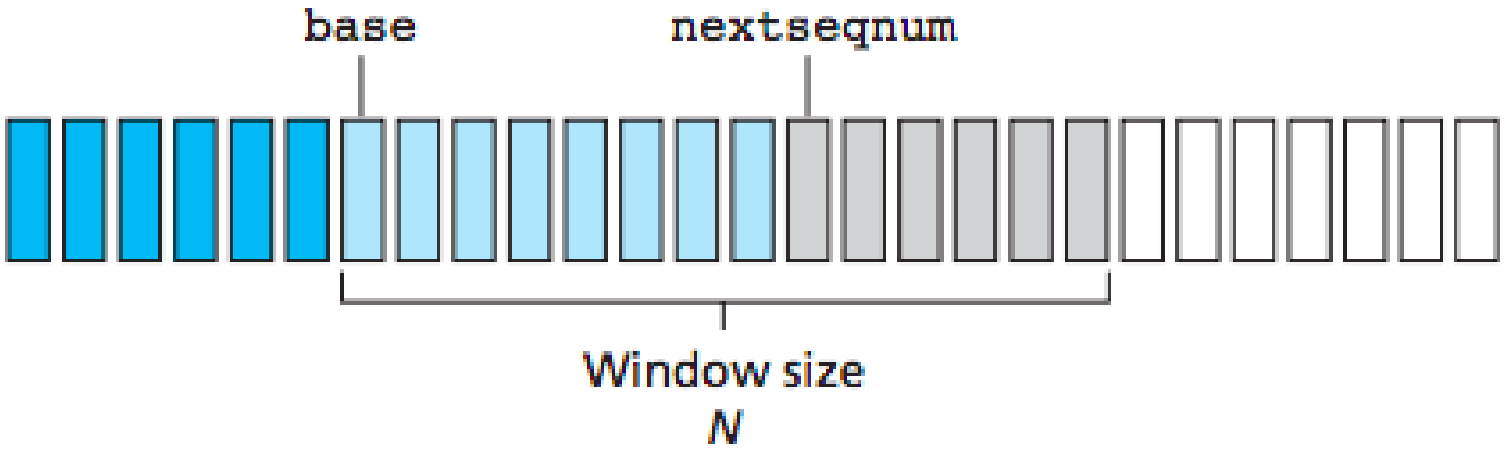
Go Back N ARQ

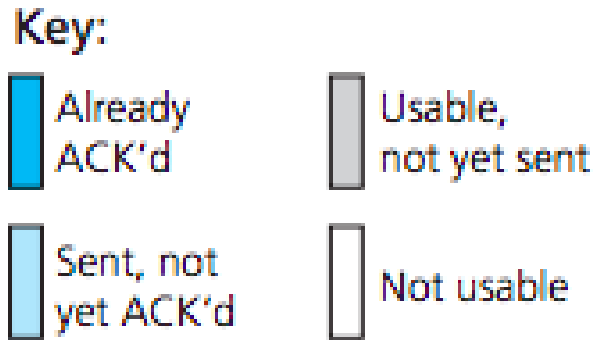


Source

https://www.tutorialspoint.com/data_communication_computer_network/data_link_control_and_protocols.htm

Go Back N ARQ – Sender Window Control





Source: Computer Networks, Kurose, Ross

Go Back N ARQ – A Bound on Window Size

- **Outstanding Frames –** Frames that have been transmitted, but not yet acknowledged

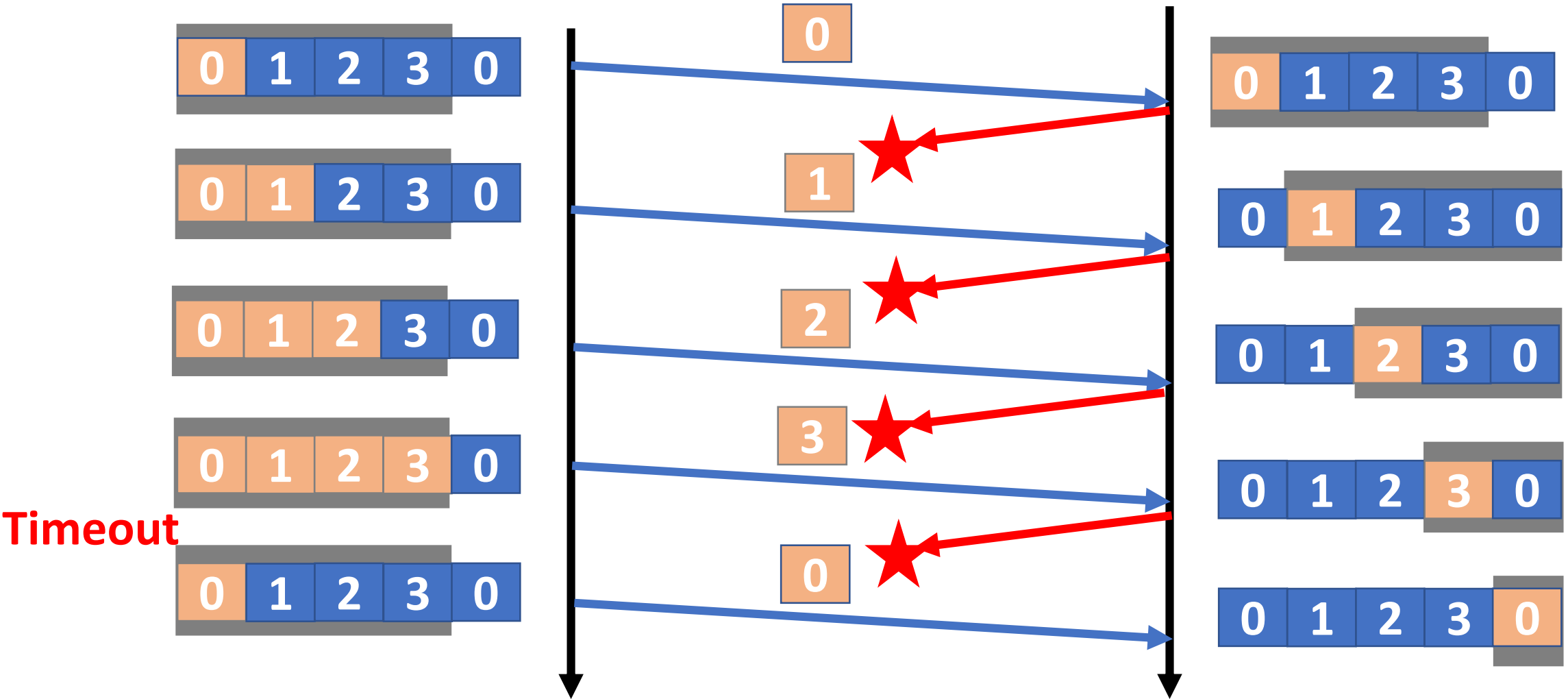
- **Maximum Sequence Number (MAX_SEQ): MAX_SEQ+1** distinct sequence numbers are there
  - **0,1,…,MAX_SEQ**

- **Maximum Number of Outstanding Frames (=Window Size): MAX_SEQ**

- **Example:** Sequence Numbers (0,1,2,…,7) – 3 bit sequence numbers, number of outstanding frames = 7 **(Not 8)**

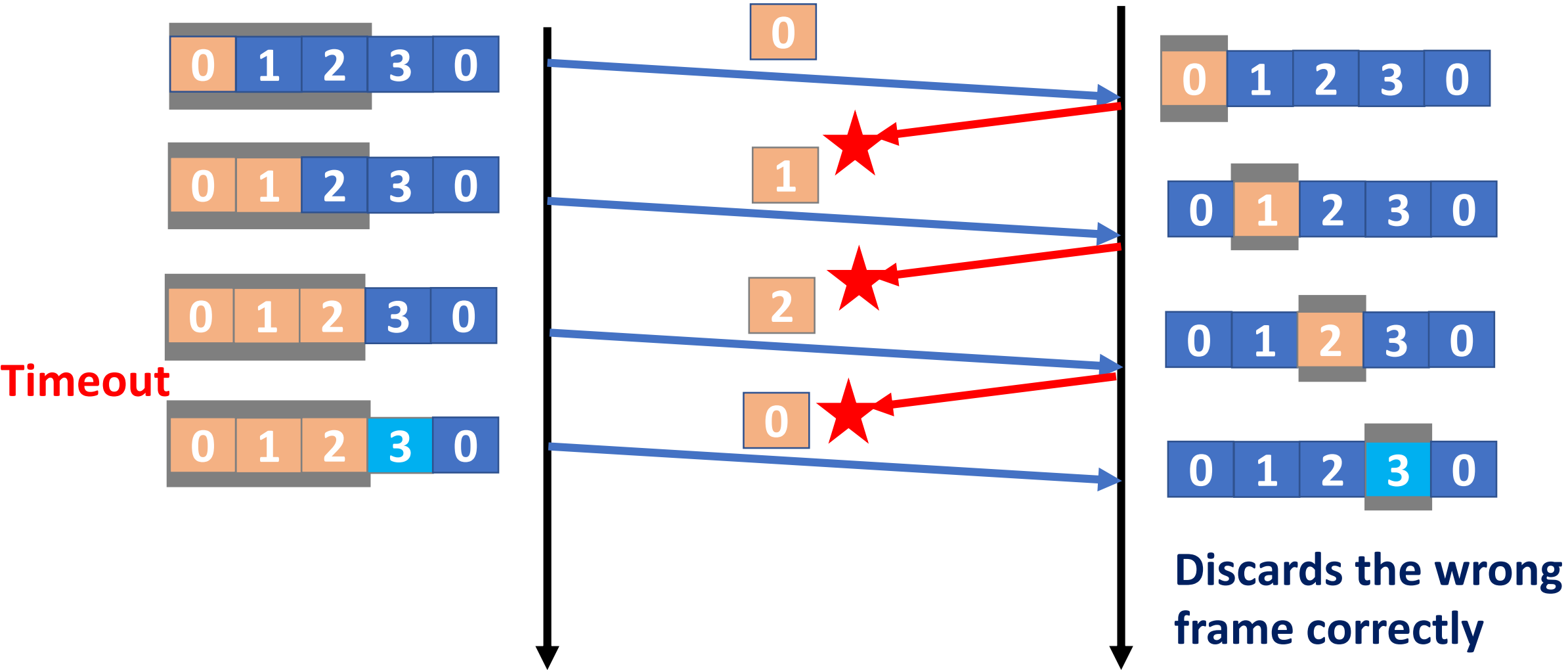
Go Back N ARQ – A Bound on Window Size

- **Let MAX_SEQ = 3, Window Size = 4**

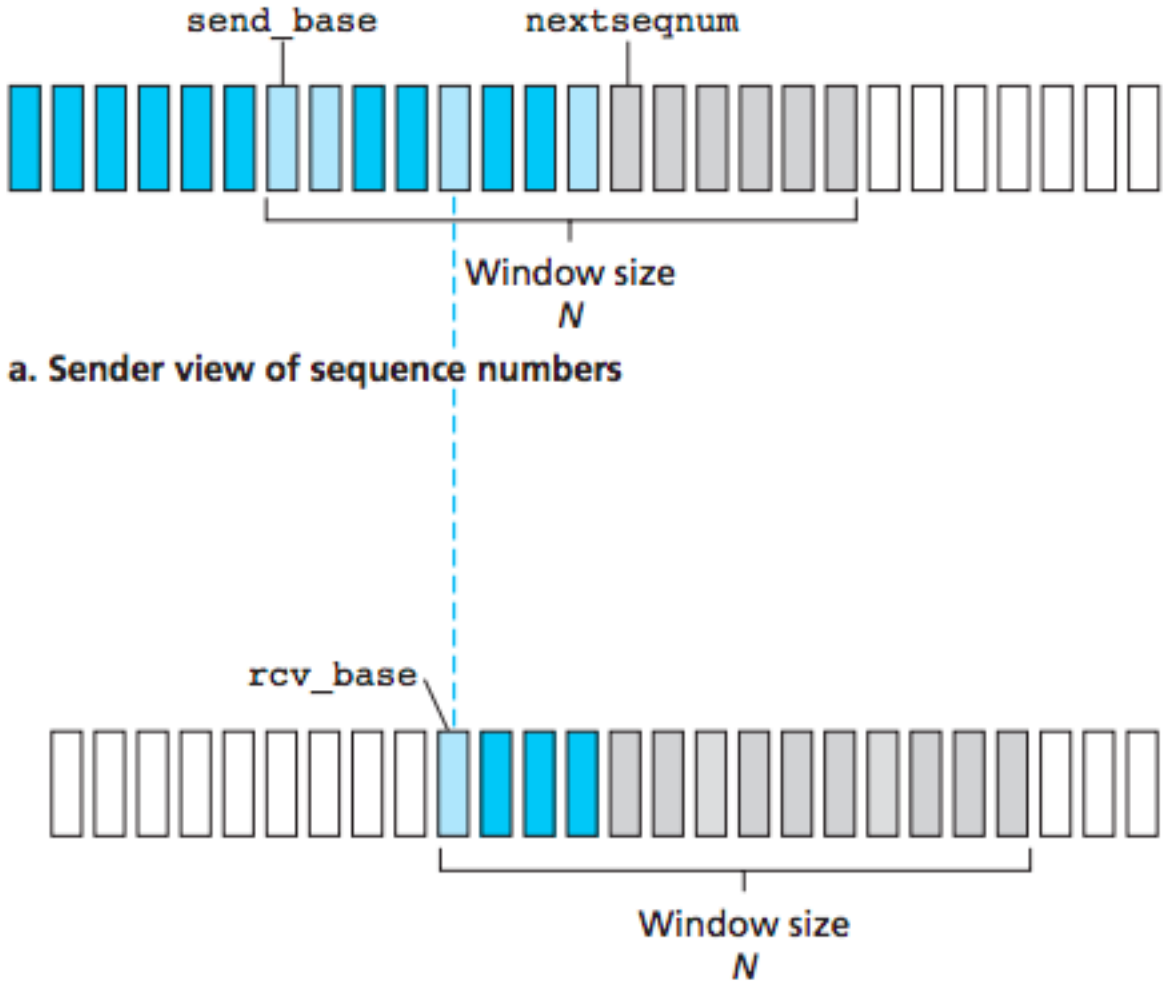


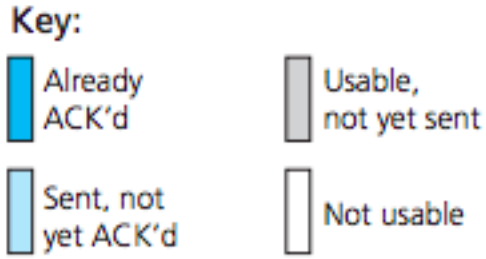
Go Back N ARQ – A Bound on Window Size

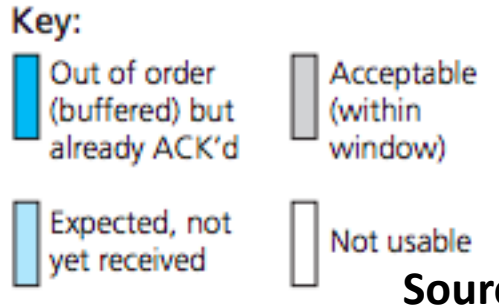
- **Let MAX_SEQ = 3, Window Size = 3**



Selective Repeat (SR) – Window Control

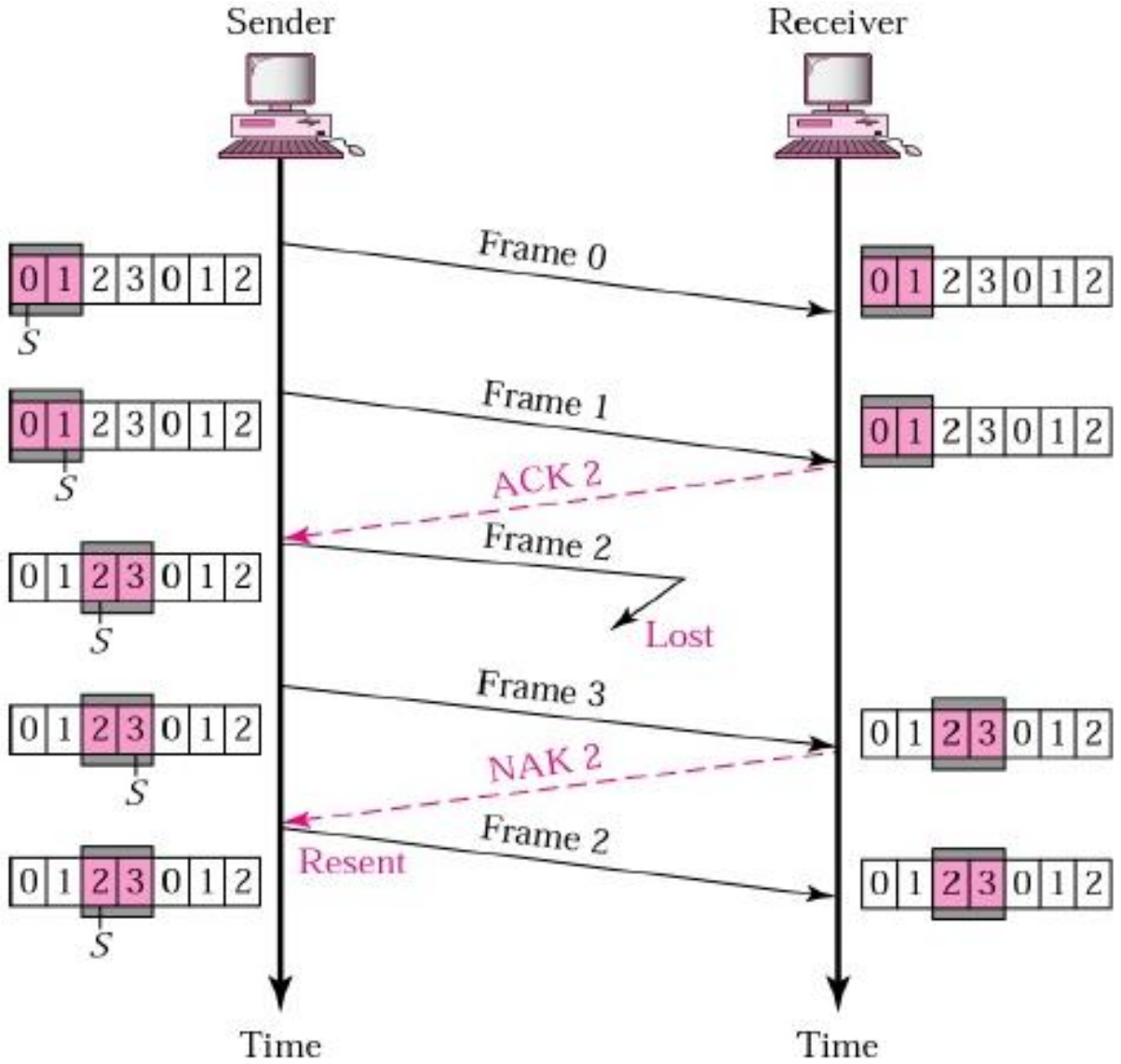






Source: Computer Networks, Kurose, Ross

Selective Repeat ARQ

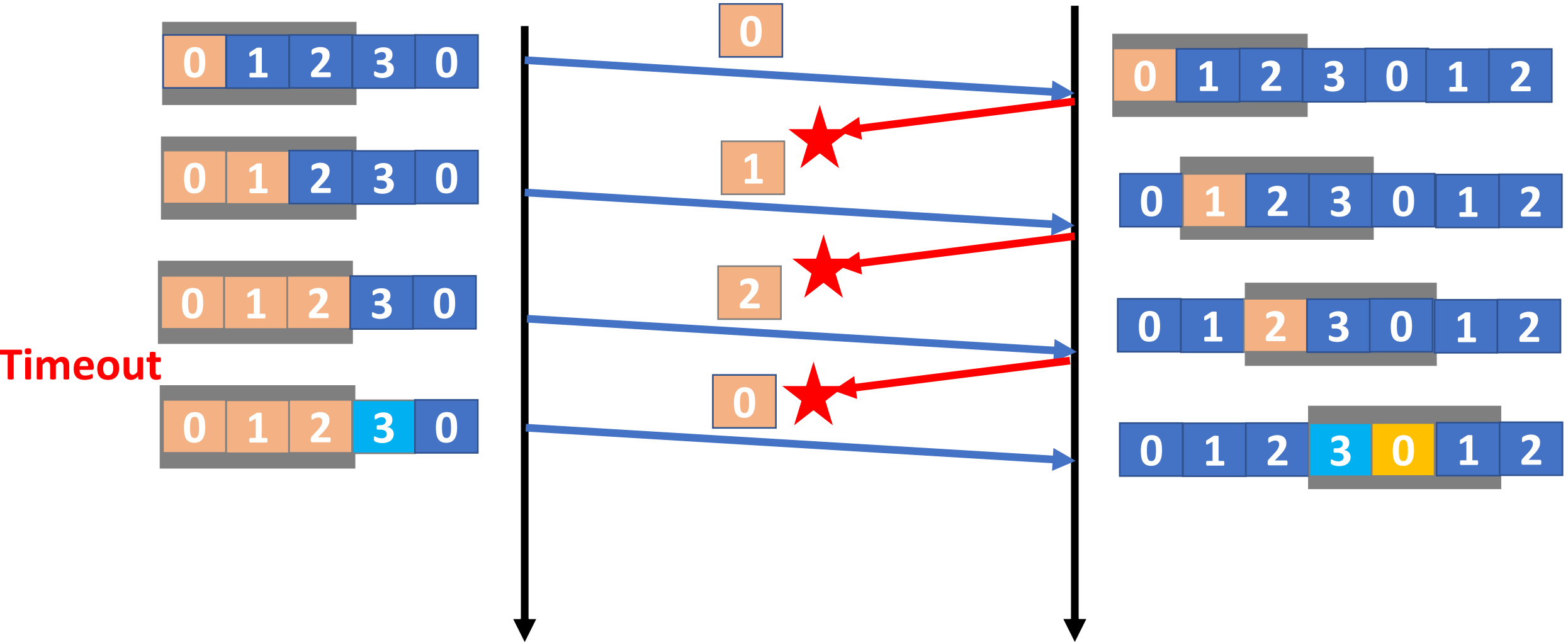


Selective Repeat – A Bound on Window Size

- **Maximum Sequence Number (MAX_SEQ): MAX_SEQ+1** distinct sequence numbers are there
  - **0,1,…,MAX_SEQ**
- **Maximum Number of Outstanding Frames (=Window Size): (MAX_SEQ+1)/2**
- **Example:** Sequence Numbers (0,1,2,…,7) – 3 bit sequence numbers, number of outstanding frames (window size) = 4

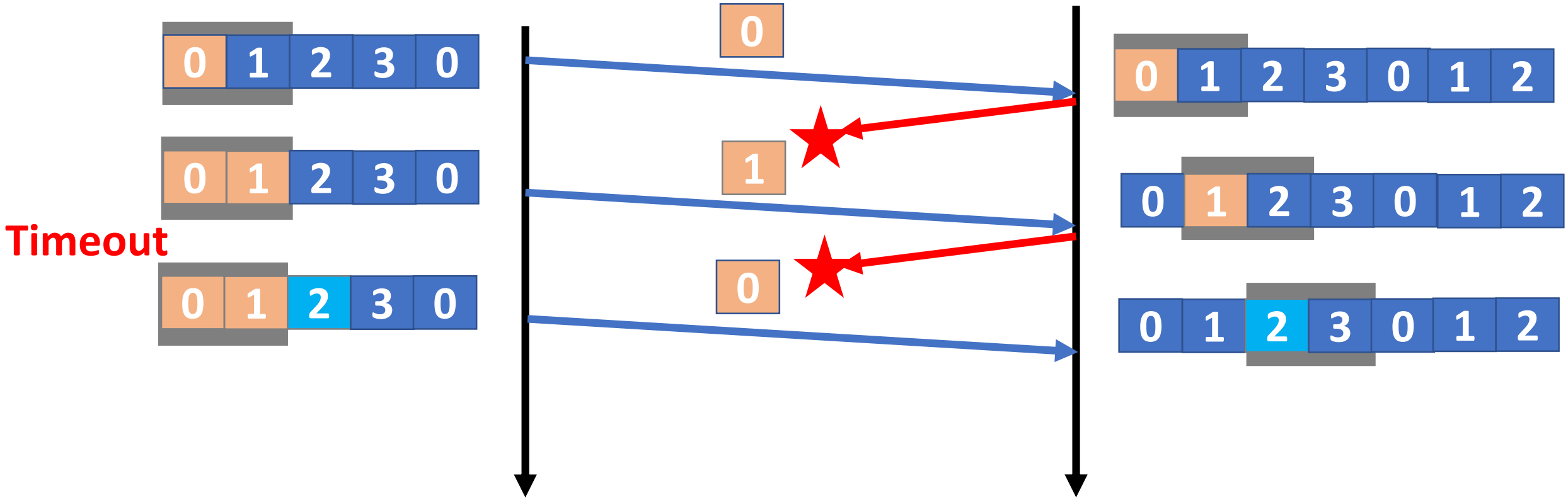
Selective Repeat – A Bound on Window Size

- **Let MAX_SEQ = 3, Window Size = 3 [(MAX_SEQ+1)/2+1]**



Selective Repeat – A Bound on Window Size

- **Let MAX_SEQ = 3, Window Size = 2**



Discards the wrong frame correctly

Bandwidth Delay Product

- **Bandwidth Delay Product (BDP) = Link Bandwidth x Link Delay** – an important metric for flow control

- Consider Bandwidth = 50 Kbps, one way transit time (delay) = 250 msec
  - BDP 12.5 Kbit
  - Assume 1000 bit segment size; BDP = 12.5 segments

- Consider the event of a segment transmission and the corresponding ACK reception – this takes a round trip time (RTT) – twice the one way latency.

- Maximum number of segments that can be outstanding during this duration = 12.5 x 2 = 25 segments

Bandwidth Delay Product – Implication on Window Size

- Maximum number of segments that can be outstanding within this duration = 25 + 1 (as the ACK is sent only when the first segment is received) = 26
  - This gives the maximum link utilization – **the link will always be busy in transmitting data segments**

- Let **BD** denotes the number of frames equivalent to the BDP, **w** is the maximum window size

- So, **w = 2BD + 1** gives the maximum link utilization – **this is an important concept to decide the window size for a window based flow control mechanism**

Implication of BDP on Protocol Design Choice

- Consider the link bandwidth = 1Mbps, Delay = 1ms
- Consider a network, where segment size is 1 KB (~1000 bytes)
- Which protocol is better for flow control?
  - (a) stop and wait,
  - (b) Go back N,
  - (c) Selective Repeat

- **BDP = 1 Mbps x 1ms = 1 Kb (~1000 bits)**
- **The segment size is eight times larger than the BDP -> the link can not hold an entire segment completely**
- **Sliding window protocols do not improve performance**
- **Stop and Wait is better – less complexity**