

Transport Layer - I (Services)

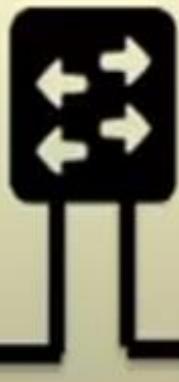
Application
Transport
Network
Data Link
Physical



Data Link
Physical



Network
Data Link
Physical



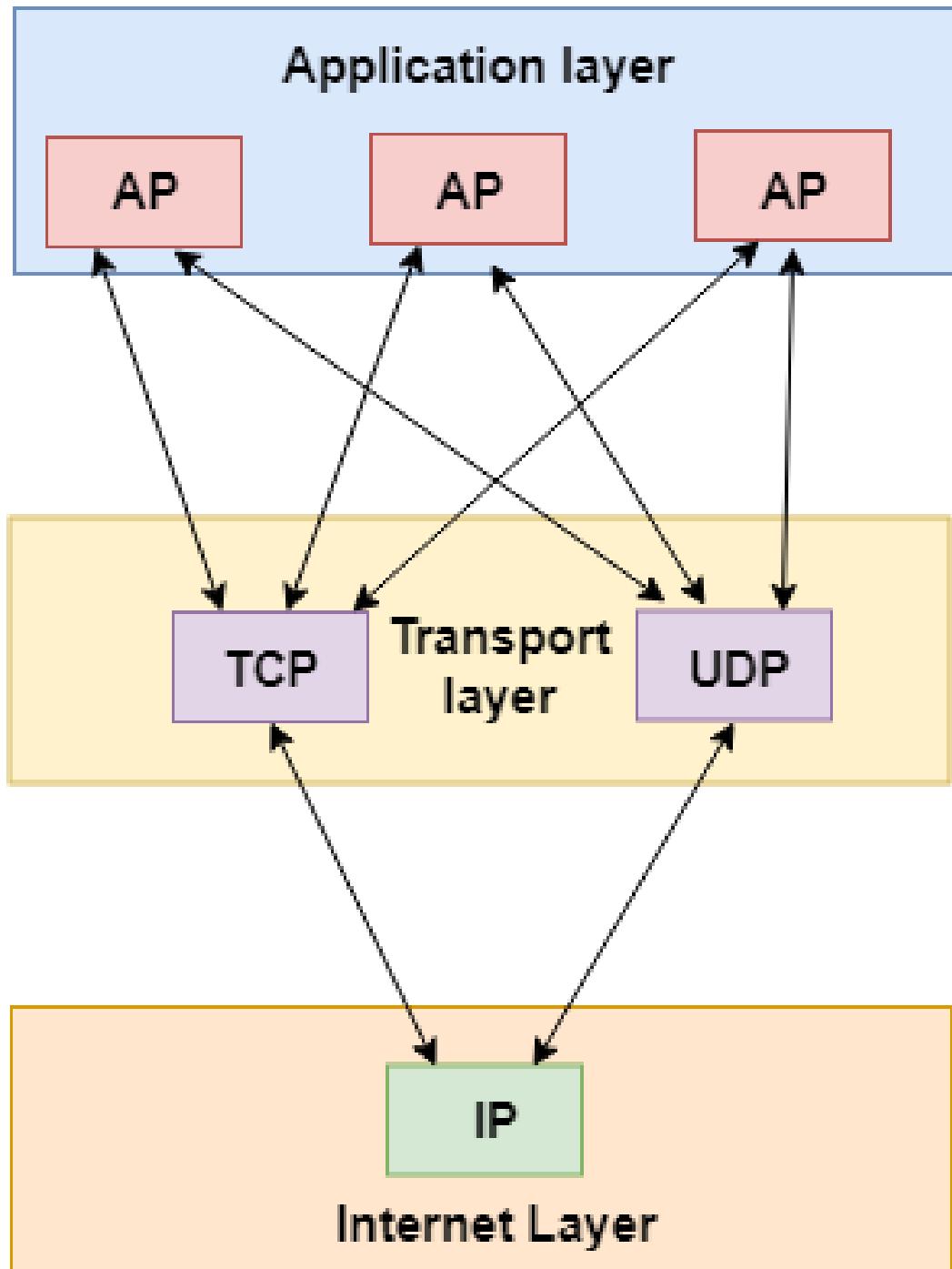
Data Link
Physical



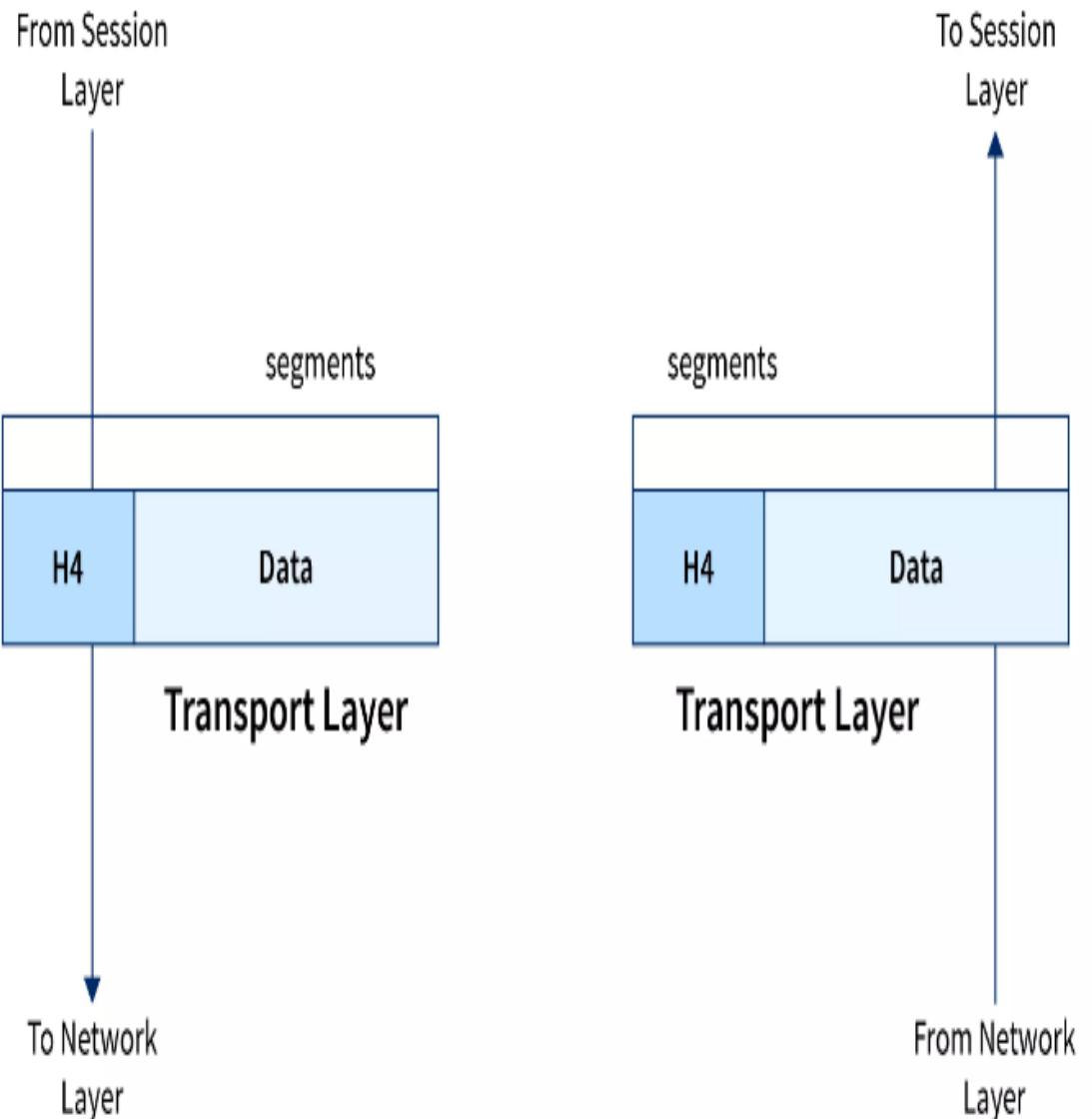
Application
Transport
Network
Data Link
Physical



The transport layer provides a logical communication between application processes running on different hosts. Although the application processes on different hosts are not physically connected, application processes use the logical communication provided by the transport layer to send the messages to each other.



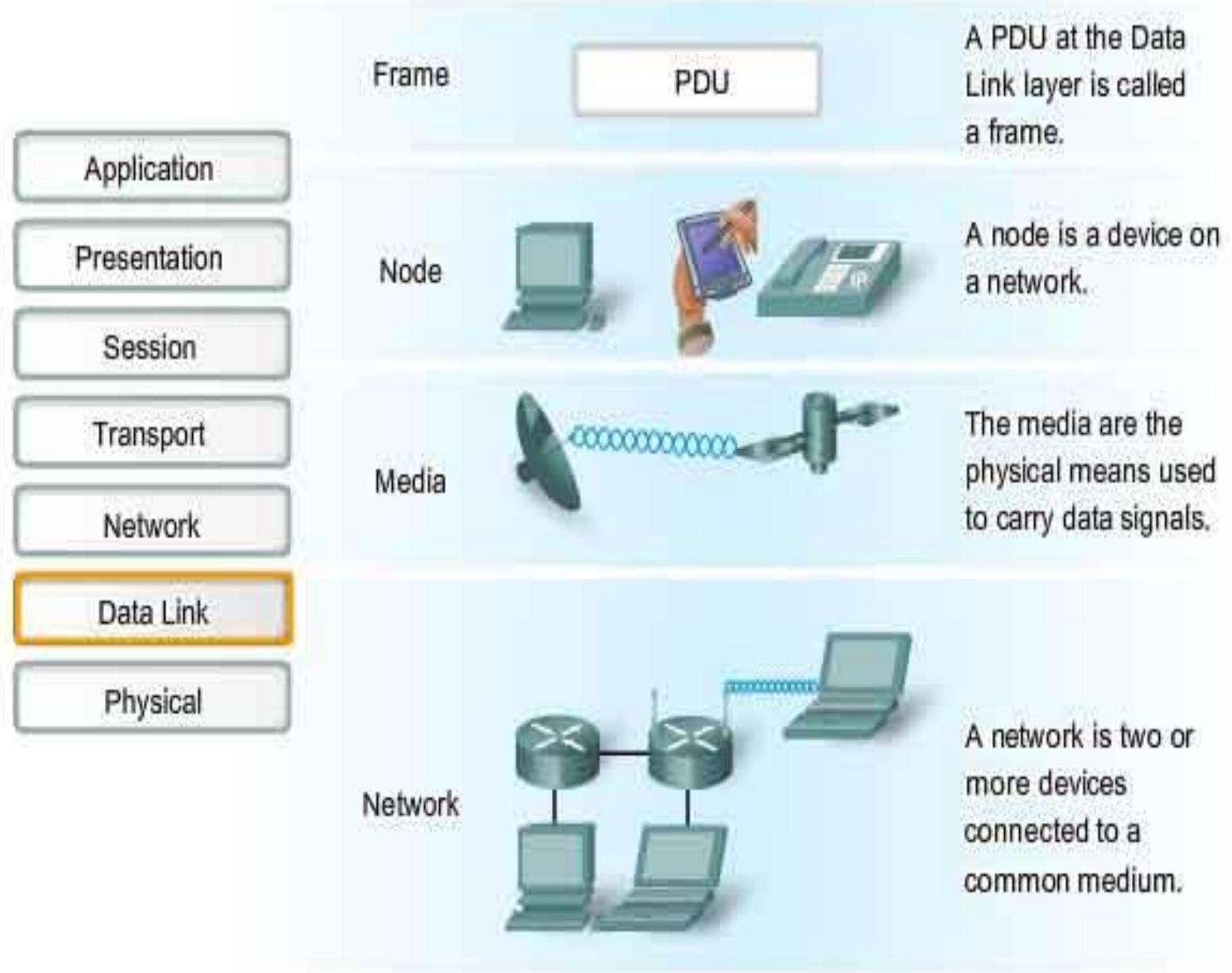
The transport layer is the fourth layer of the OSI model which is responsible for the process to process delivery of data. The network layer divides the data received from the transport layer in the form of packets. The network layer provides two ways of communication namely - connection-oriented and connection-less.



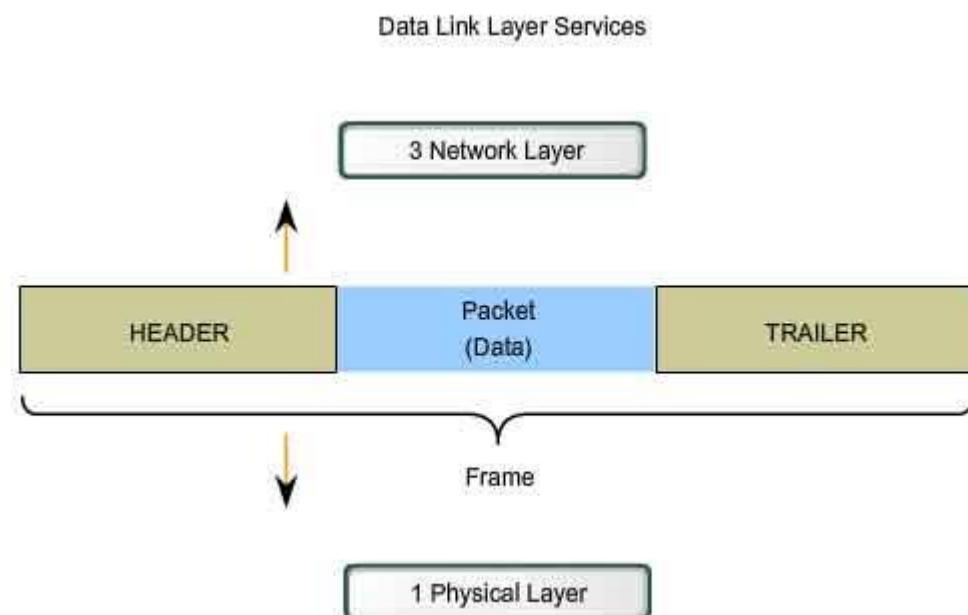
Data Link Layer Terms

Network layer: Handles the routing and sending of data between different networks. The most important protocols at this layer are IP and ICMP.

Data link layer: Handles communications between devices on the same network.

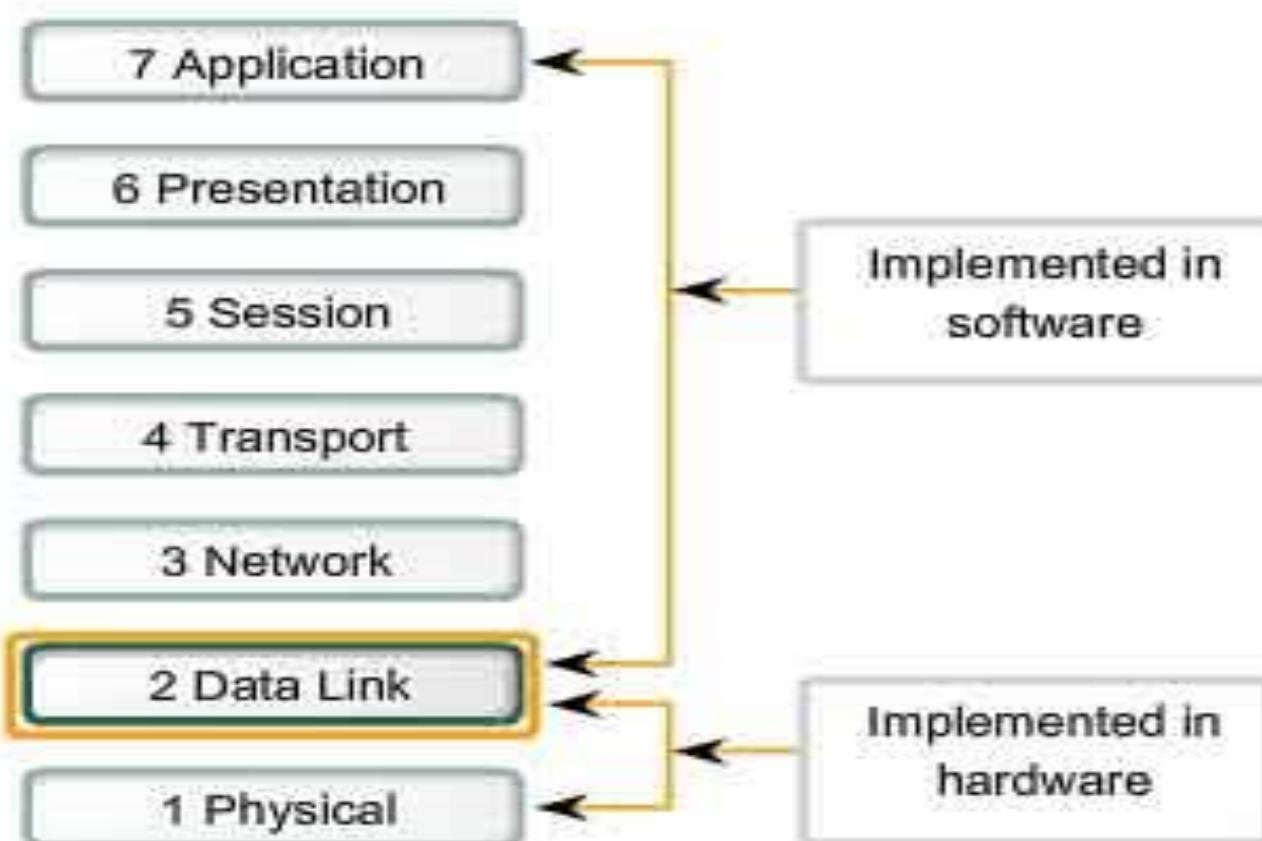


The Data Link layer is responsible for controlling the transfer of frames across the media.



Connecting Upper Layer Services to the Media

The Data Link layer links the software and hardware layers.



Physical devices devoted to the Data Link layer have both hardware and software components.



PC NIC

Protocol Stack Implementation in a Host

Application

Transport

Network

Data Link

Physical

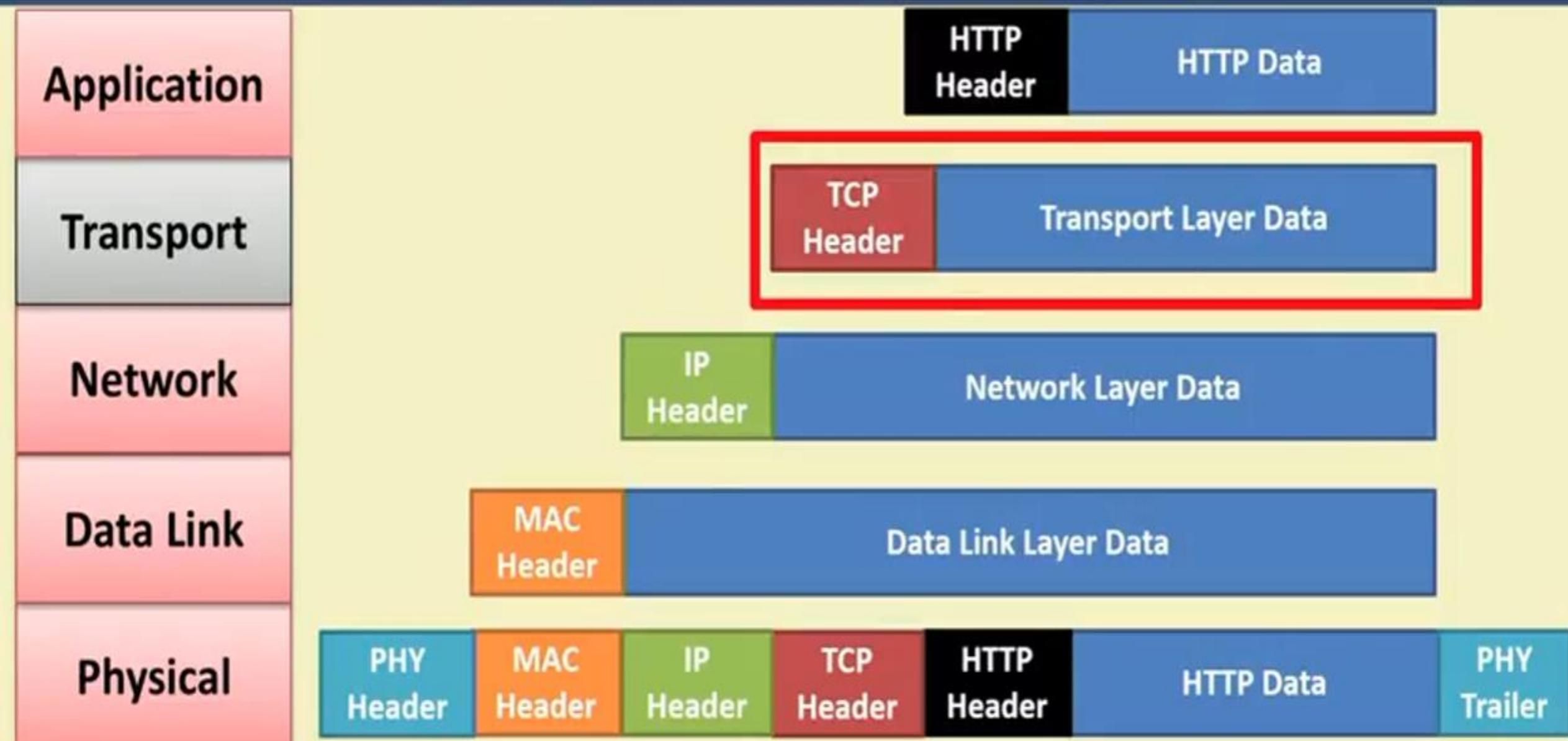
Software, Kernel

Firmware, Device Driver

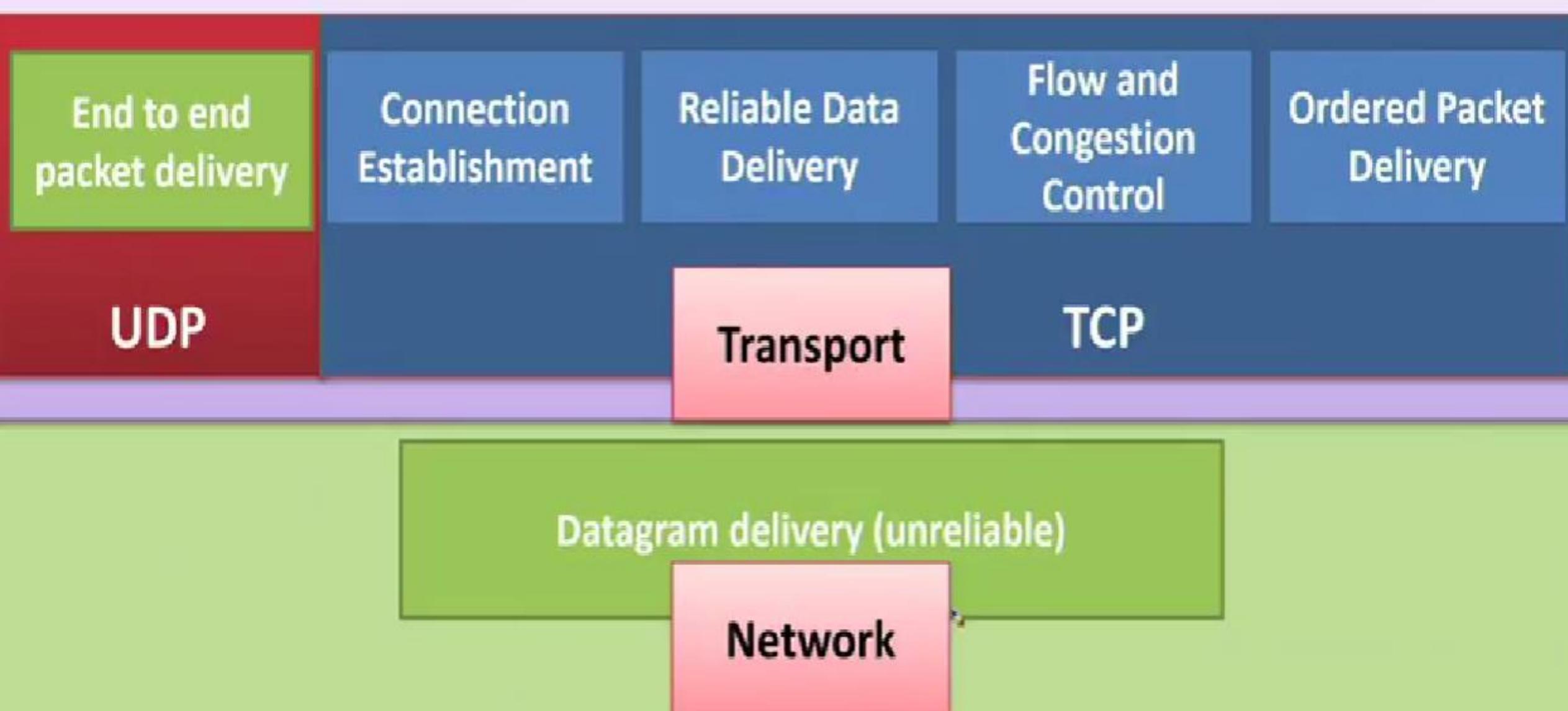
Hardware

NIC
Network Interface
Card

How Application Data Passes Through Different Layers



Transport Layer Services



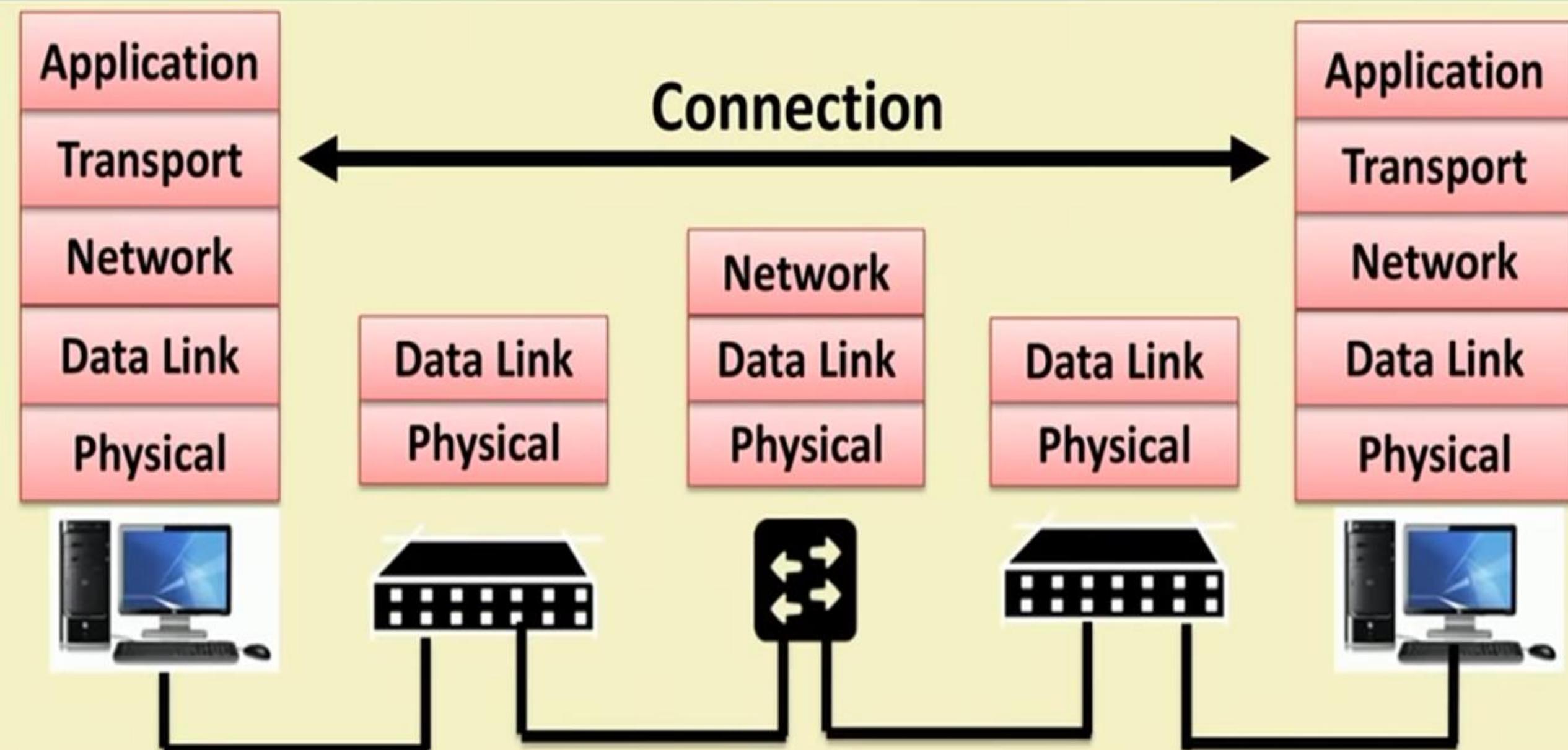
TCP uses Sliding Window mechanism at octet level. The window size can be variable over time. This is achieved by utilizing the concept of "Window Advertisement" based on :

1. **Buffer availability at the receiver**
2. **Network conditions (traffic load etc.)**

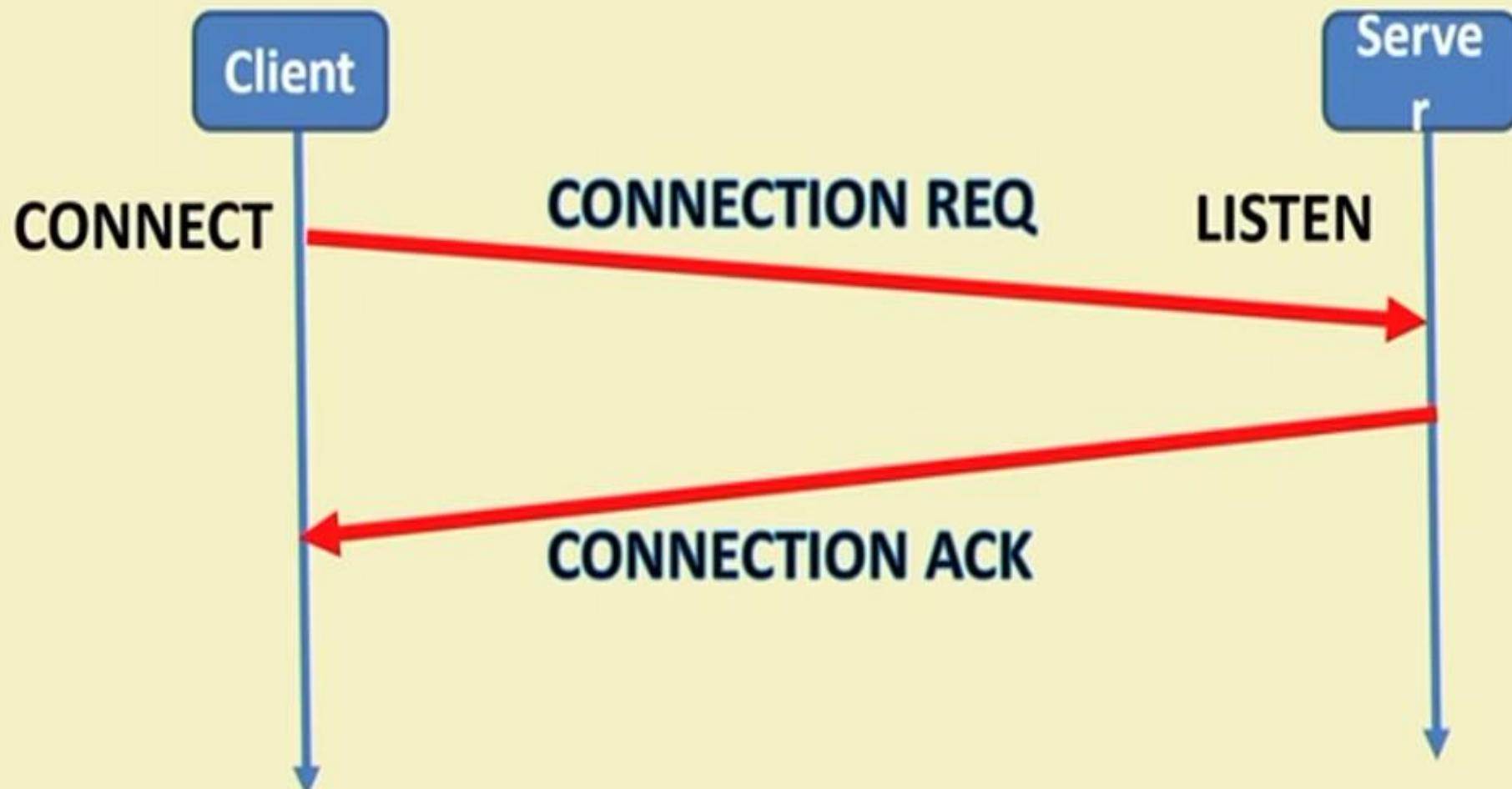
Flow Control

In the former case receiver varies its window size depending upon the space available in its buffers. The window is referred as RECEIVE WINDOW (Recv_Win). When receiver buffer begin to fill it advertises a small Recv_Win so that the sender does'nt send more data than it can accept. If all buffers are full receiver sends a "Zero" size advertisement. It stops all transmission. When buffers become available receiver advertises a Non Zero widow to resume retransmission. The sender also periodically probes the "Zero" window to avoid any deadlock if the Non Zero Window advertisement from receiver is lost. The Variable size Recv_Win provides efficient end to end flow control. The second case arises when some intermediate node (e.g. a router) controls the source to reduce transmission rate. Here another window referred as CONGESTION WINDOW (C_Win) is utilized. Advertisement of C_Win helps to check and avoid congestion.

Transport Layer - II (Connection)



Connection Establishment

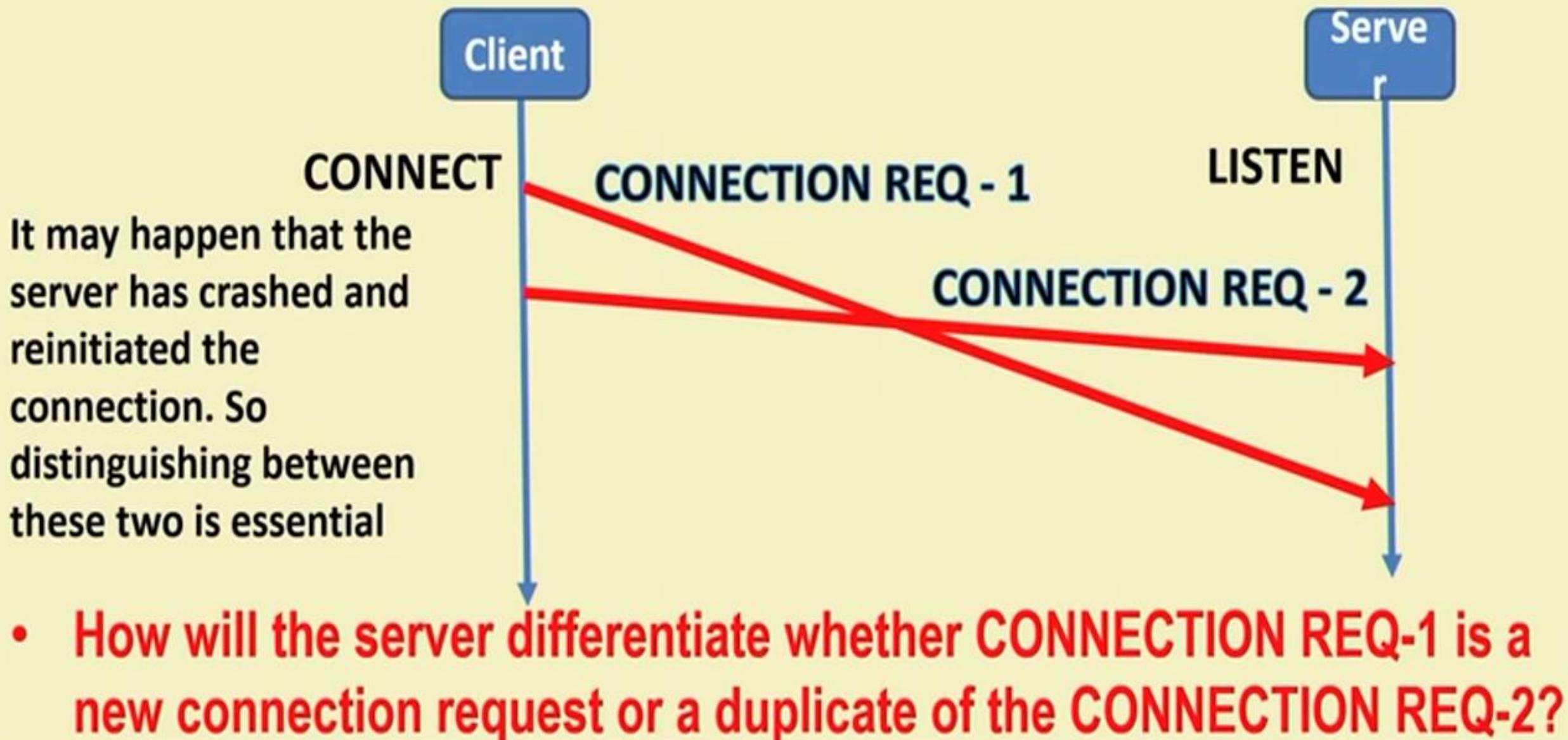


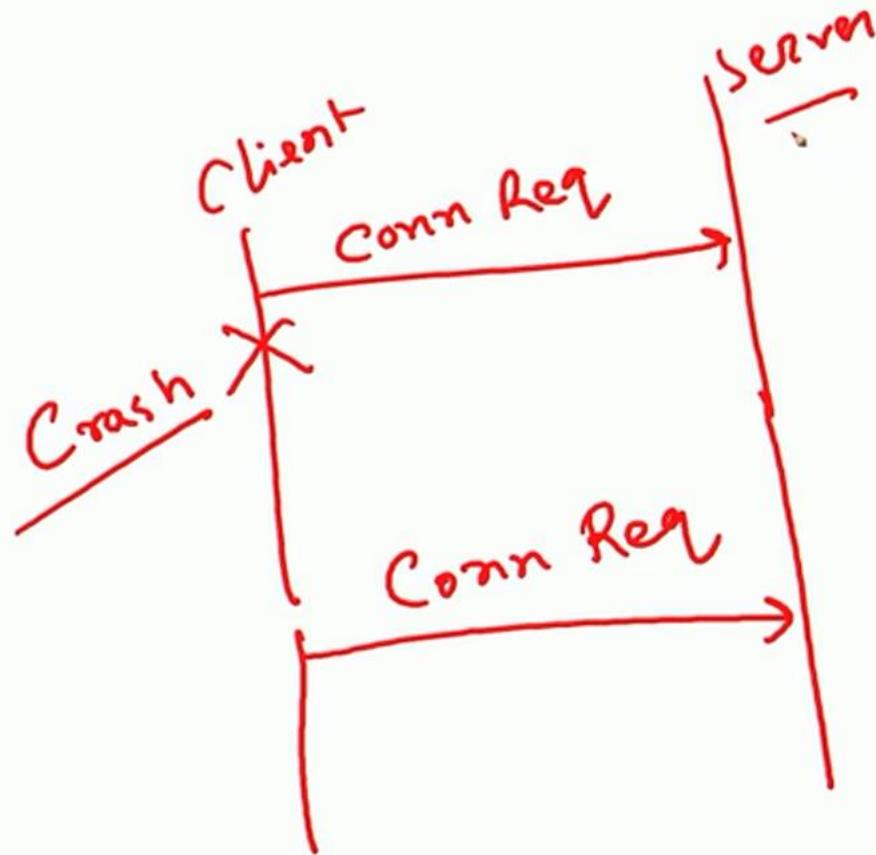
- This is a simple primitive for connection establishment – but does this work good?

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 stuck in the network congestion, after the timeout, the sender assumes that the packets have been dropped, and retransmits the packets

Connection Establishment



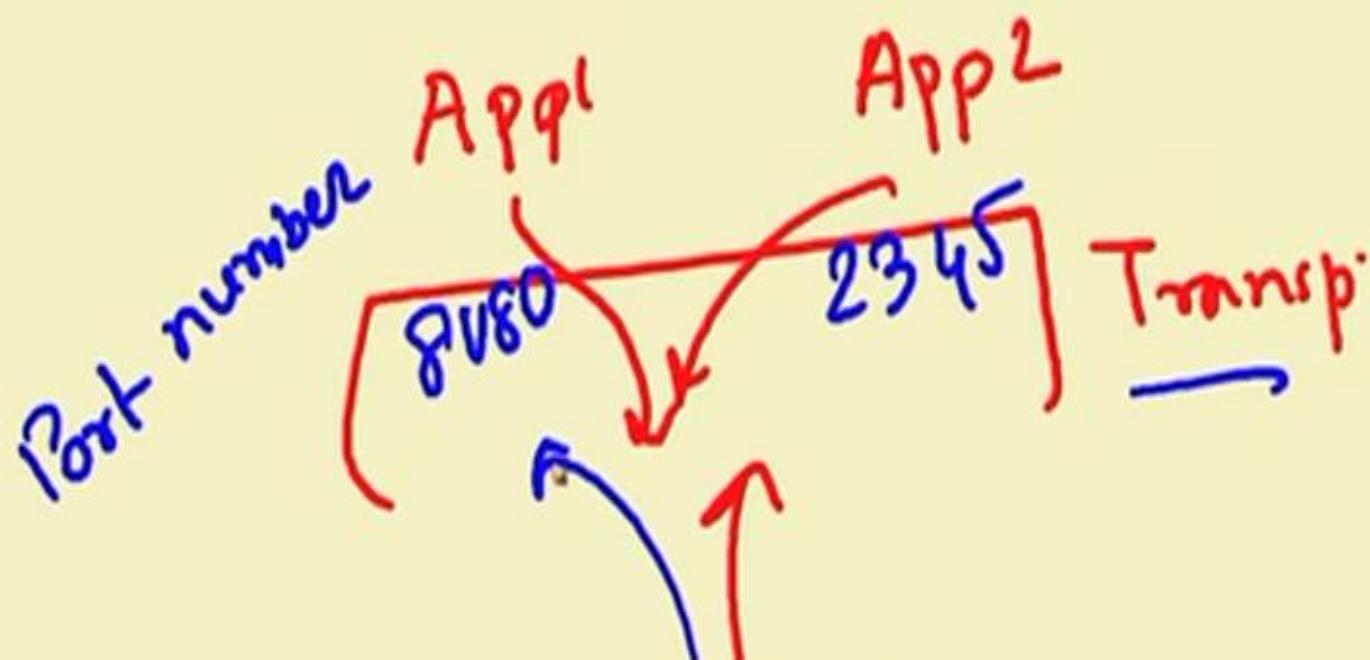


Connection Establishment

- Protocol correctness versus Protocol performance – an eternal debate in computer networks ...
- 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

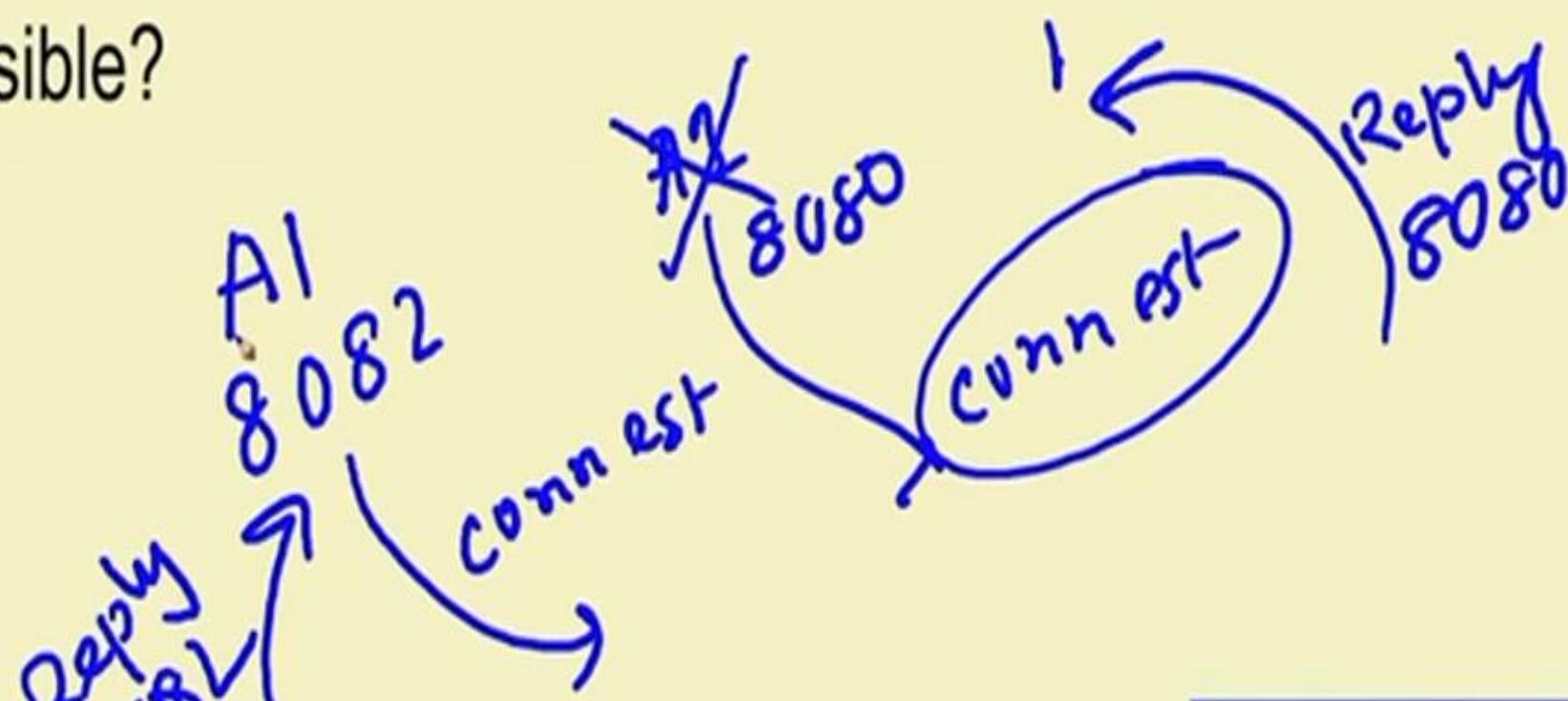
Connection Establishment – Handling Delayed Duplicates

- **Solution 1: Use Throwaway Transport Address (Port Numbers)**
 - Do not use a port number if it has been used once already – Delayed duplicate packets will never find their way to a transport process
 - Is this solution feasible?



Connection Establishment – Handling Delayed Duplicates

- Solution 1: Use Throwaway Transport Address (Port Numbers)
 - Do not use a port number if it has been used once already – Delayed duplicate packets will never find their way to a transport process
 - Is this solution feasible?

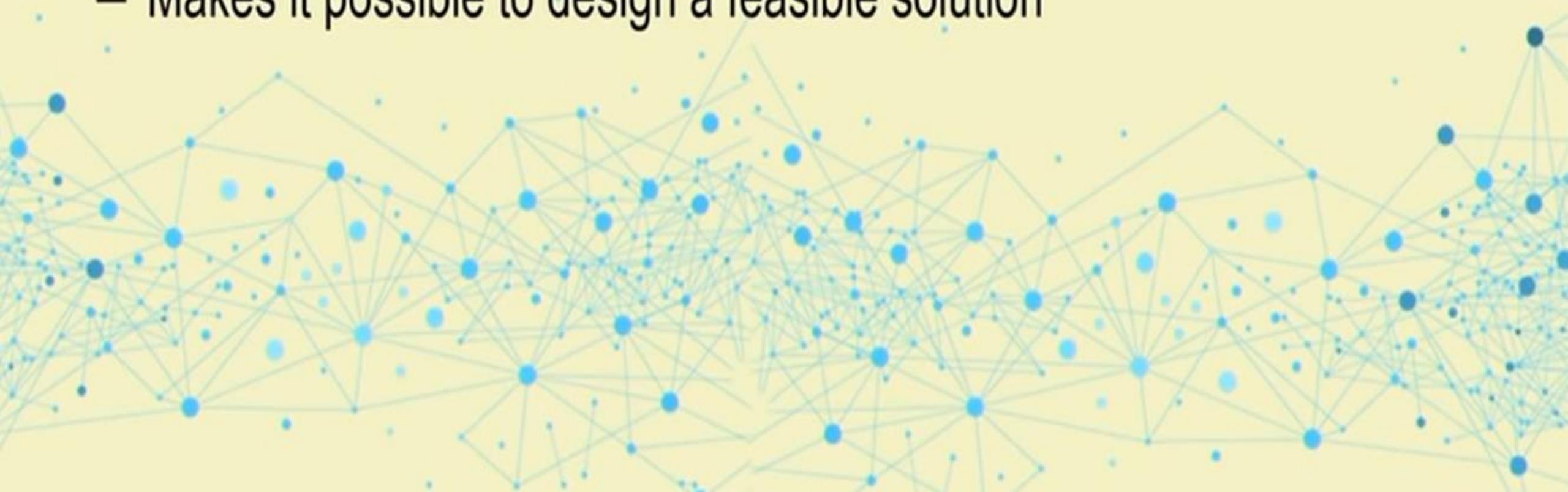


Connection Establishment – Handling Delayed Duplicates

- **Solution 1: Use Throwaway Transport Address (Port Numbers)**
 - Do not use a port number if it has been used once already – Delayed duplicate packets will never find their way to a transport process
 - Is this solution feasible?
- **Solution 2: Give each connection a unique identifier chosen by the initiating party and put in each segment**
 - Can you see any problem in this approach?

Connection Establishment – Handling Delayed Duplicates

- **Solution 3:** Devise a mechanism to kill off aged packets that are still hobbling about (Restrict the packet lifetime)
 - Makes it possible to design a feasible solution

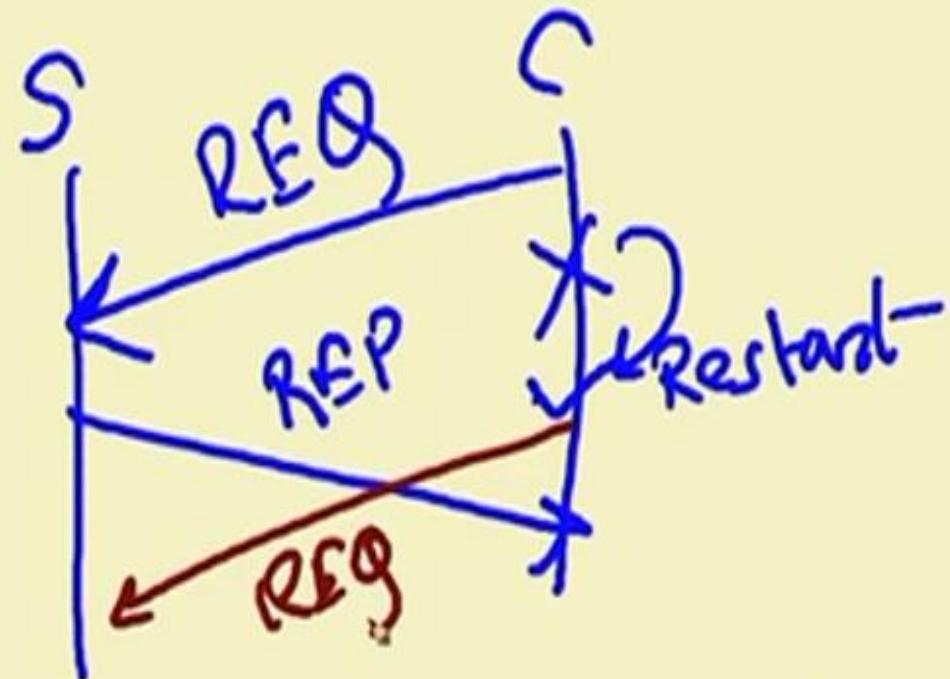


Connection Establishment – Handling Delayed Duplicates

- Three ways to restrict packet lifetime
 - **Restricted Network Design** – Prevents packets from looping (bound the maximum delay including congestion)
 - **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

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

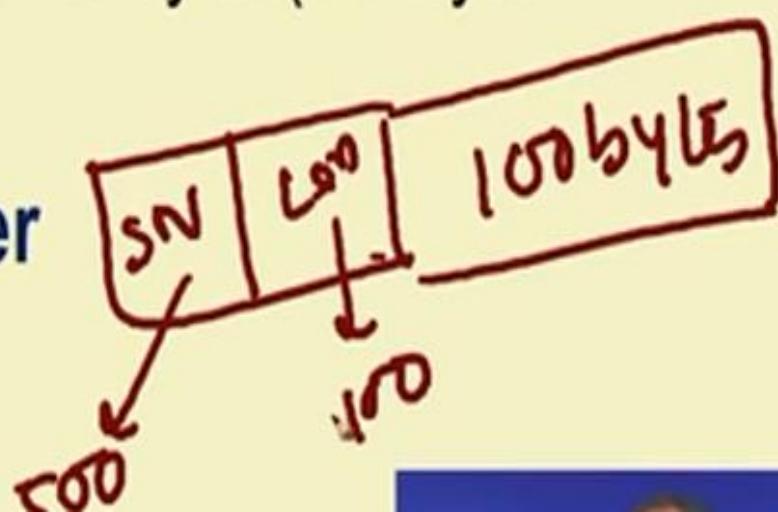
Connection Establishment – Handling Delayed Duplicates

- Label segments with sequence numbers that will not be reused within T secs.
- The period T and the rate of packets per second determine the size of the sequence number – at most one packet with a given sequence number may be outstanding at any given time



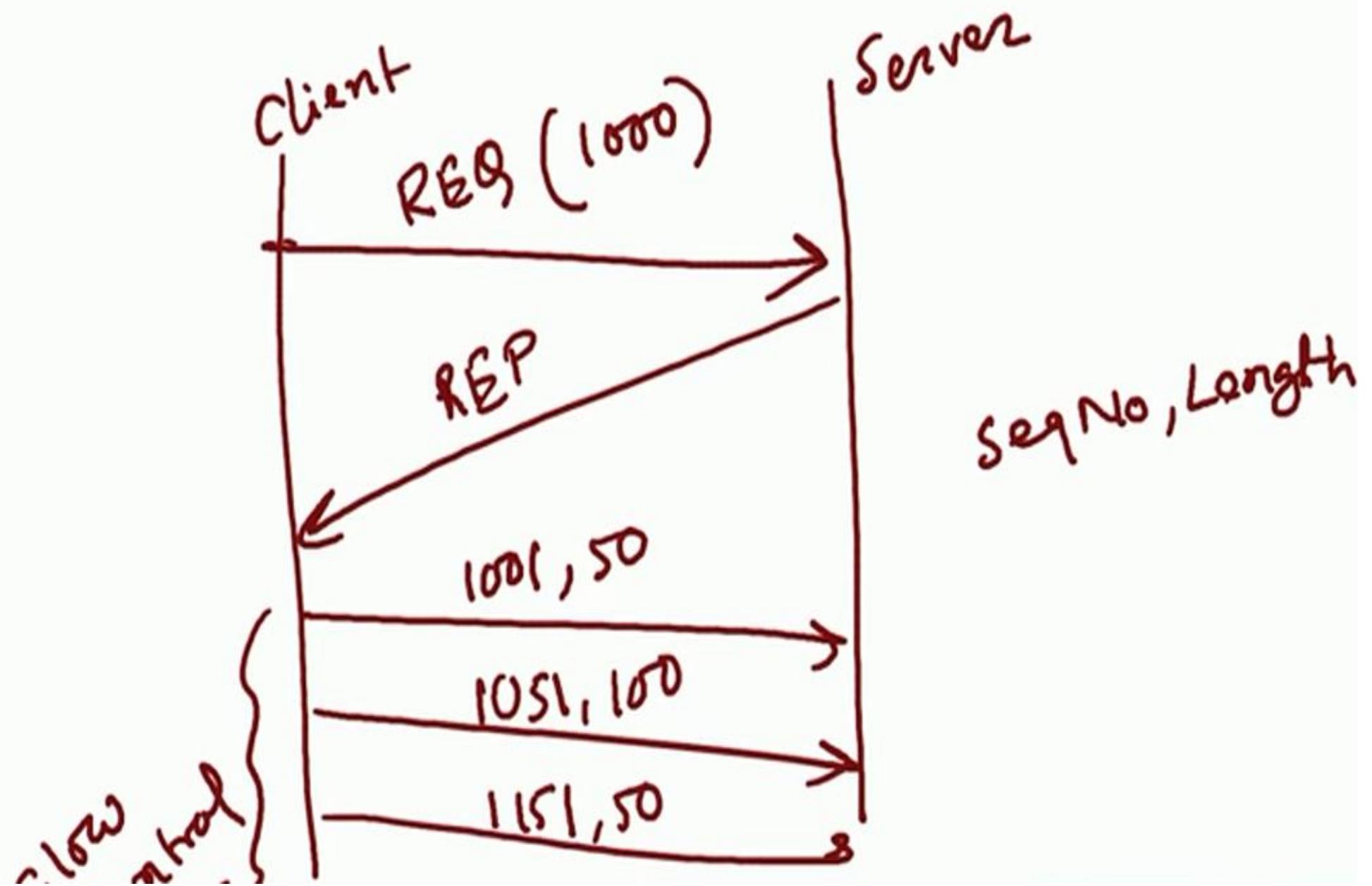
Sequence Number Adjustment

- Two important requirements (*Tomlinson 1975, Selecting Sequence Numbers*)
 - R1. 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

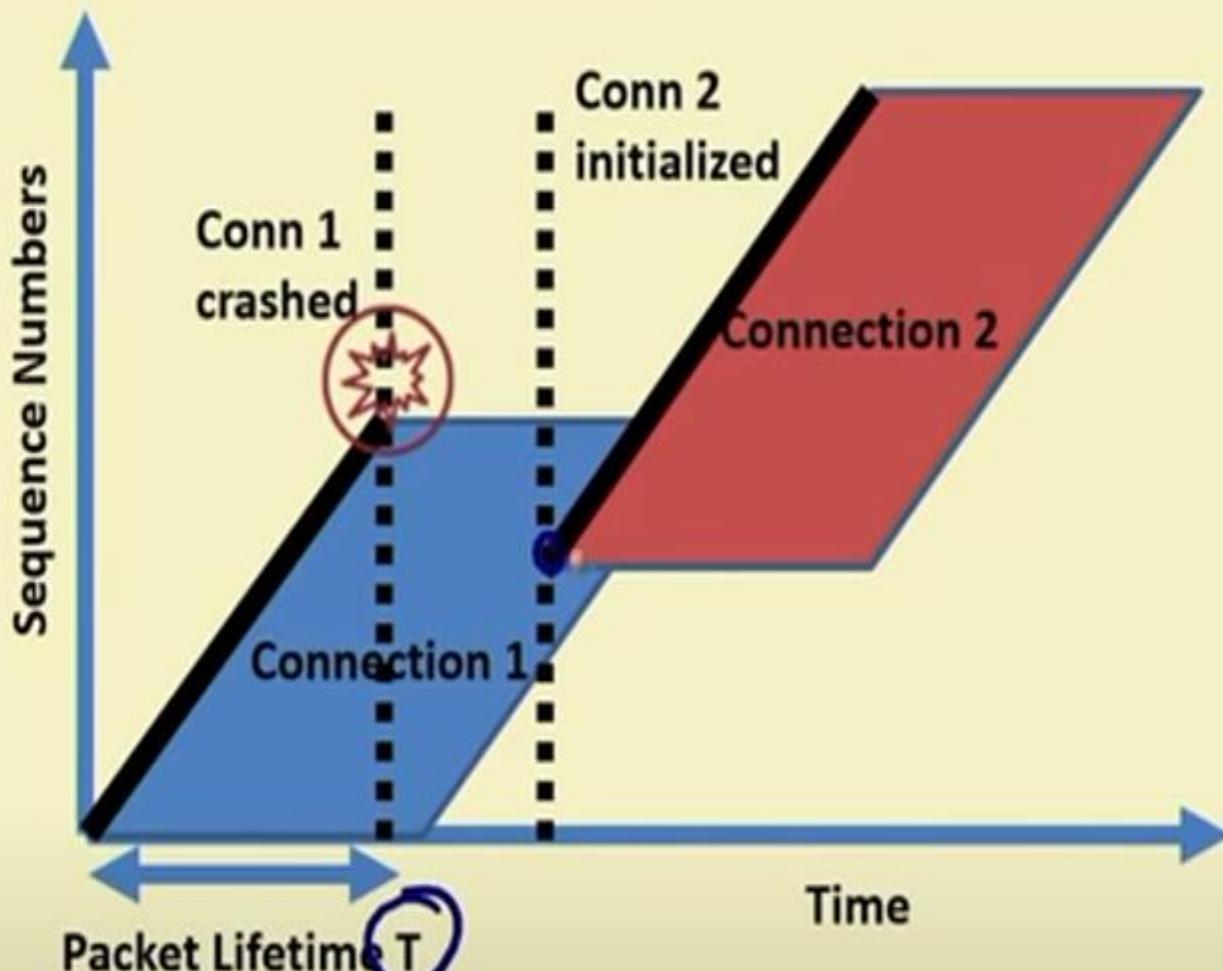


Sequence Number Adjustment

- Two important requirements (*Tomlinson 1975, Selecting Sequence Numbers*)
 - R2: 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

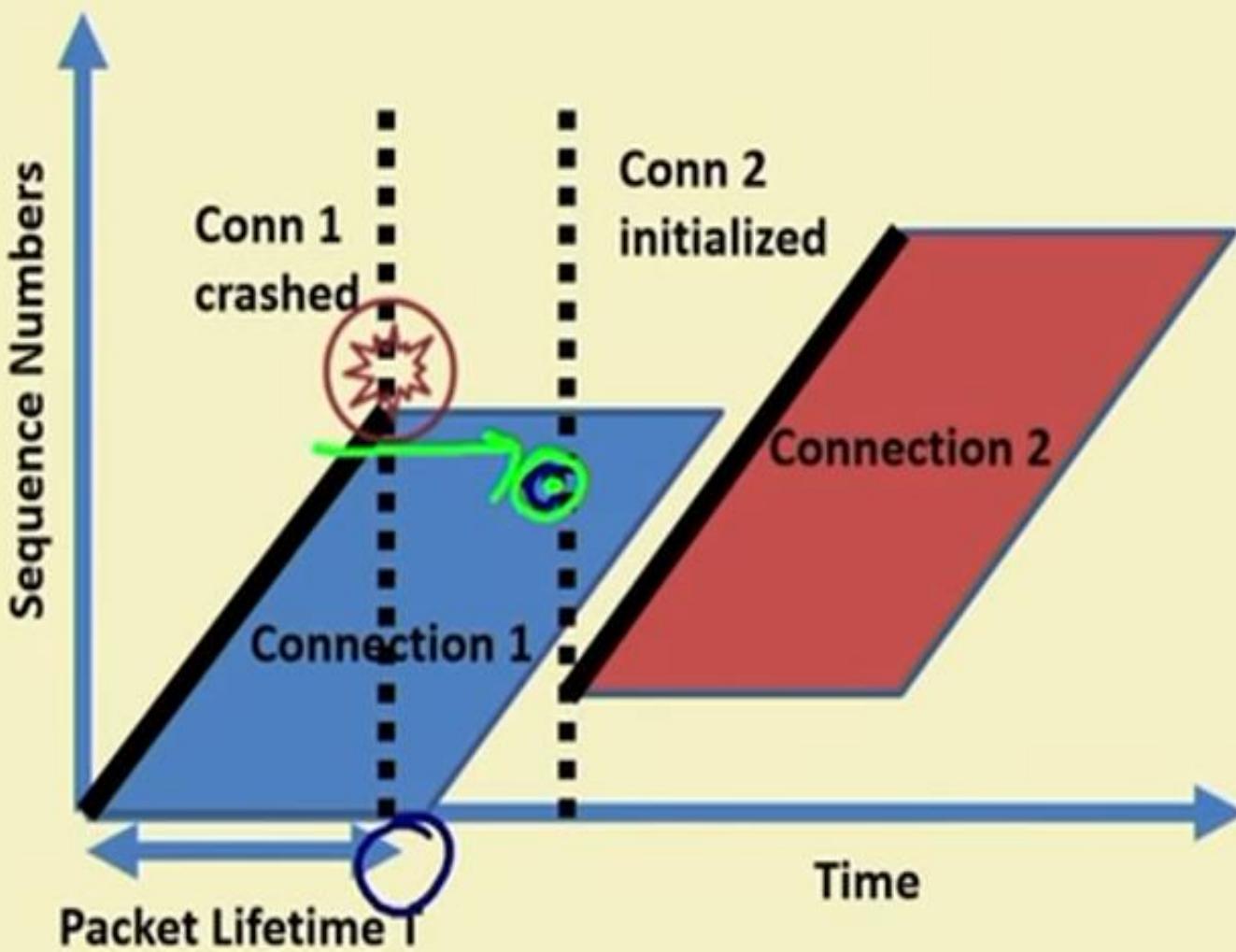


Initial Sequence Number during Connection Establishment

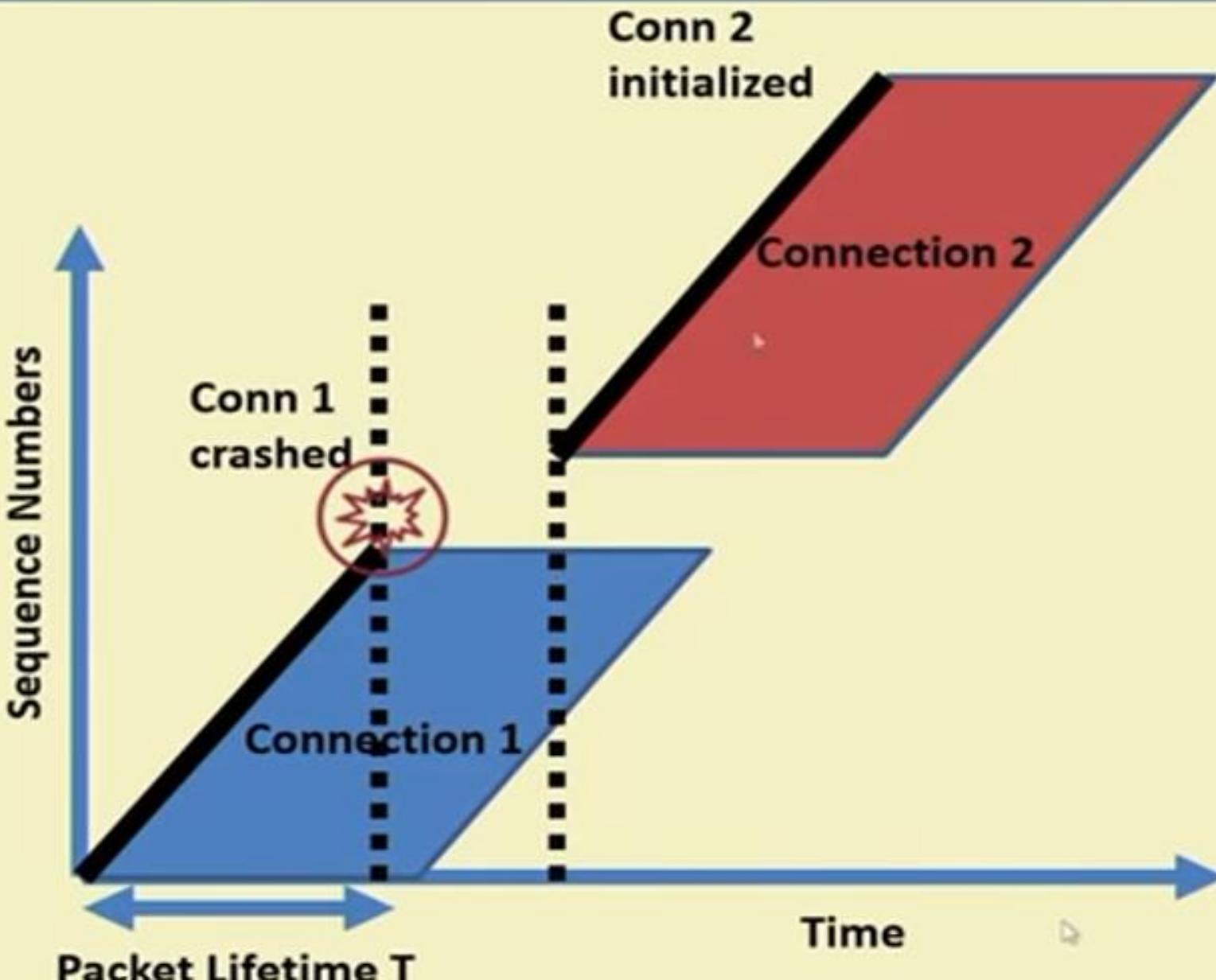


- A Delayed duplicate packet of connection 1 can create a confusion for connection 2

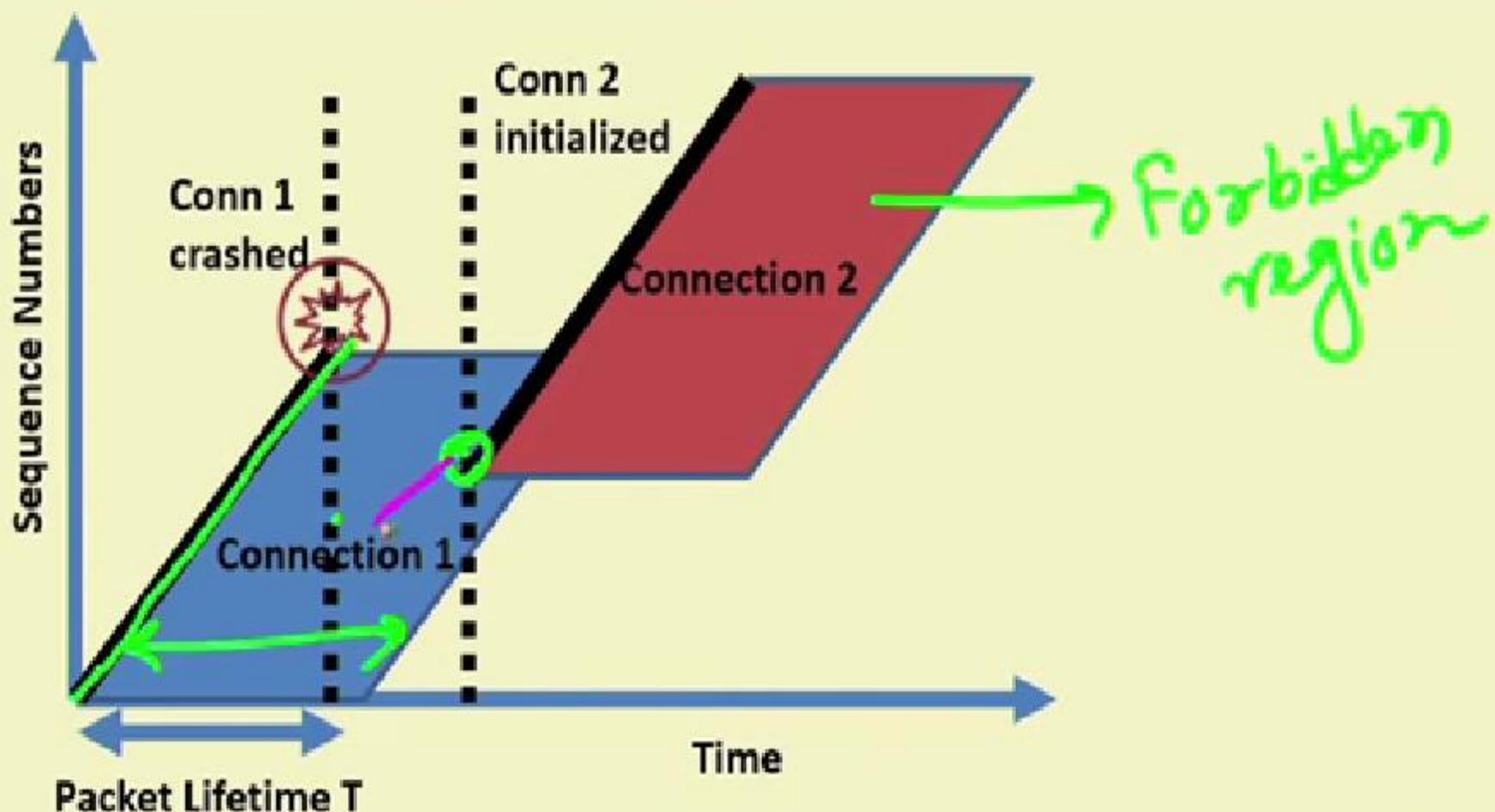
What We Ideally Want? Either ...



What We Ideally Want? Or ...

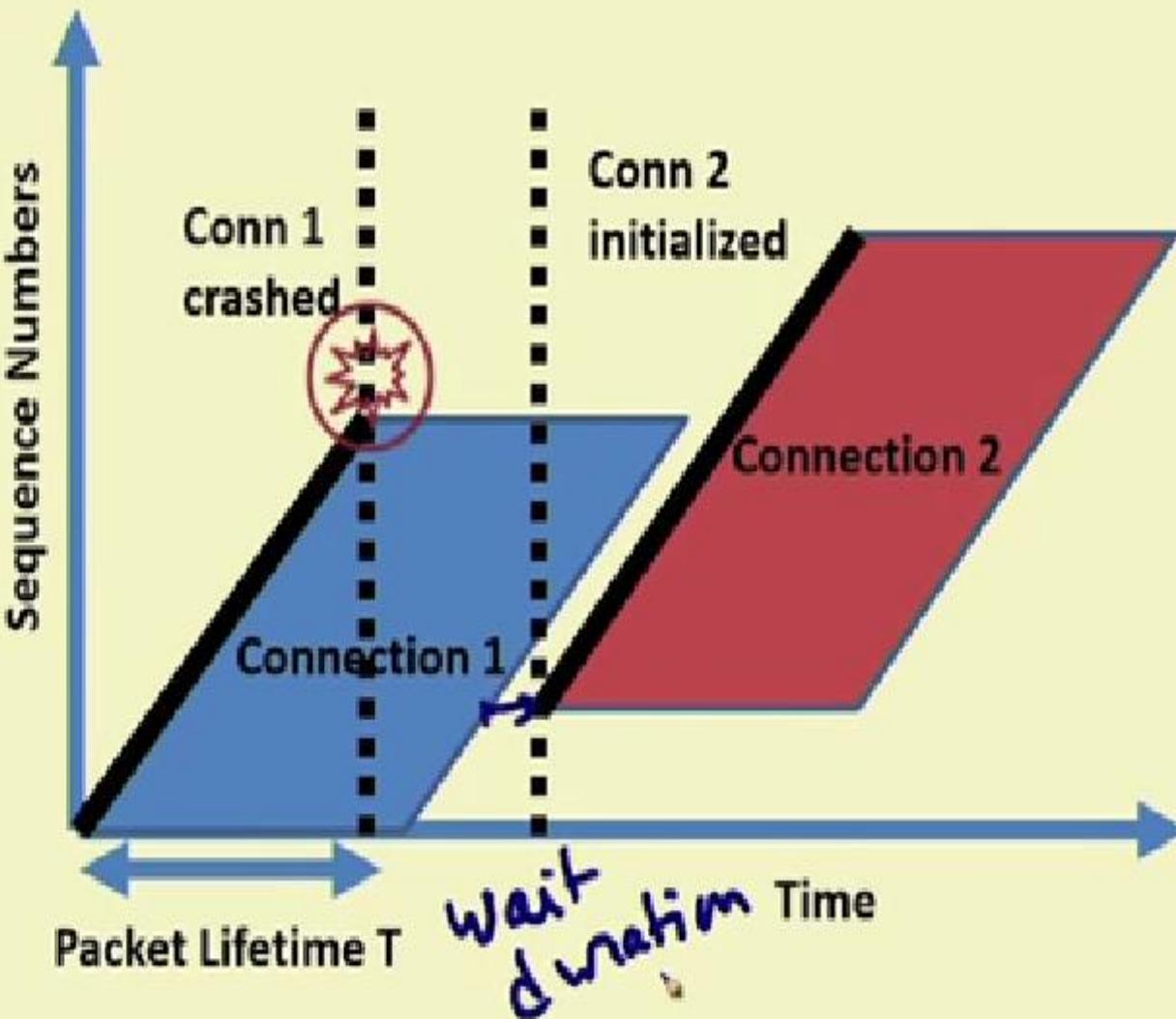


Initial Sequence Number during Connection Establishment

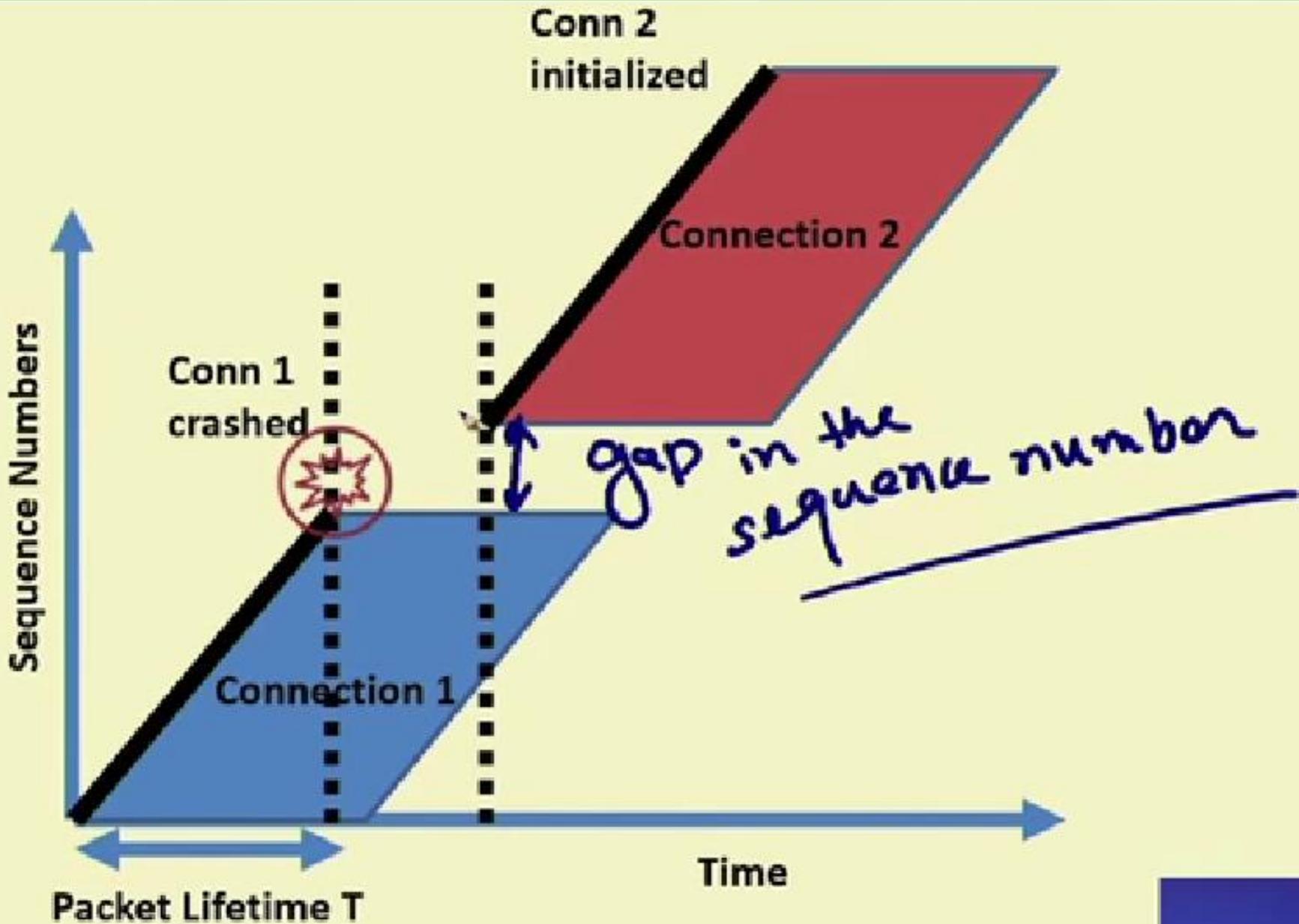


- A Delayed duplicate packet of connection 1 can create a confusion for connection 2

What We Ideally Want? Either ...

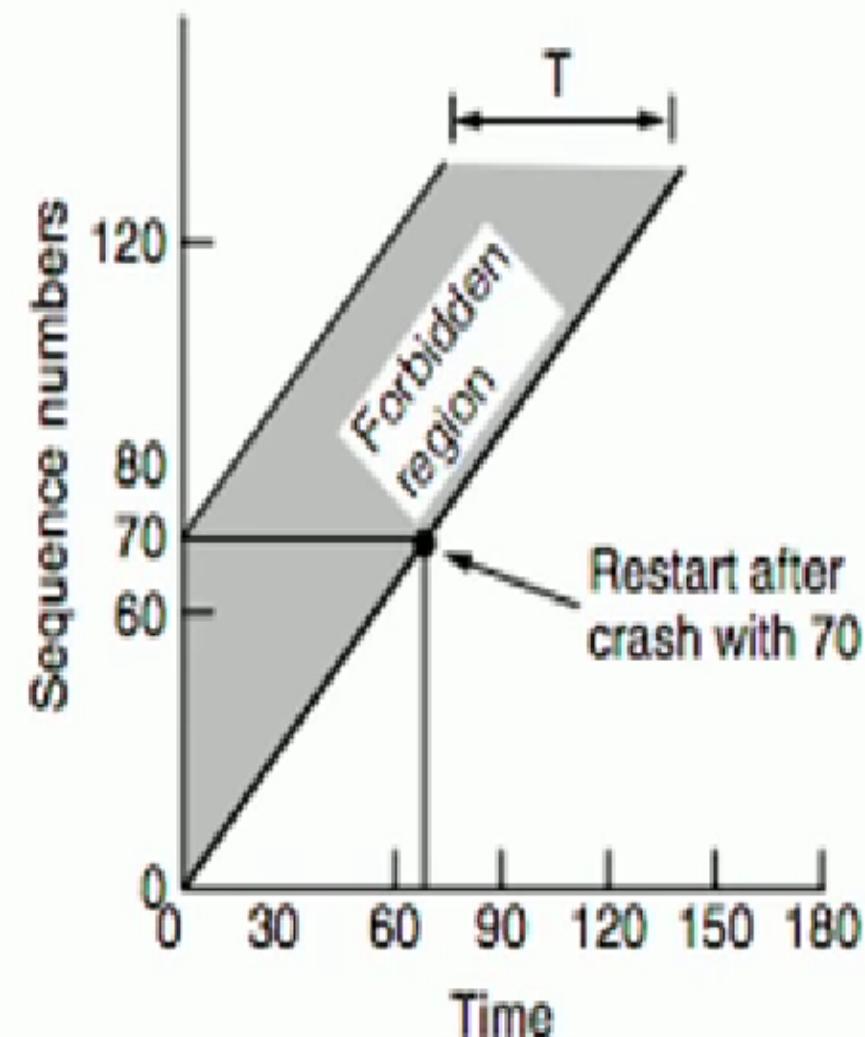


What We Ideally Want? Or ...



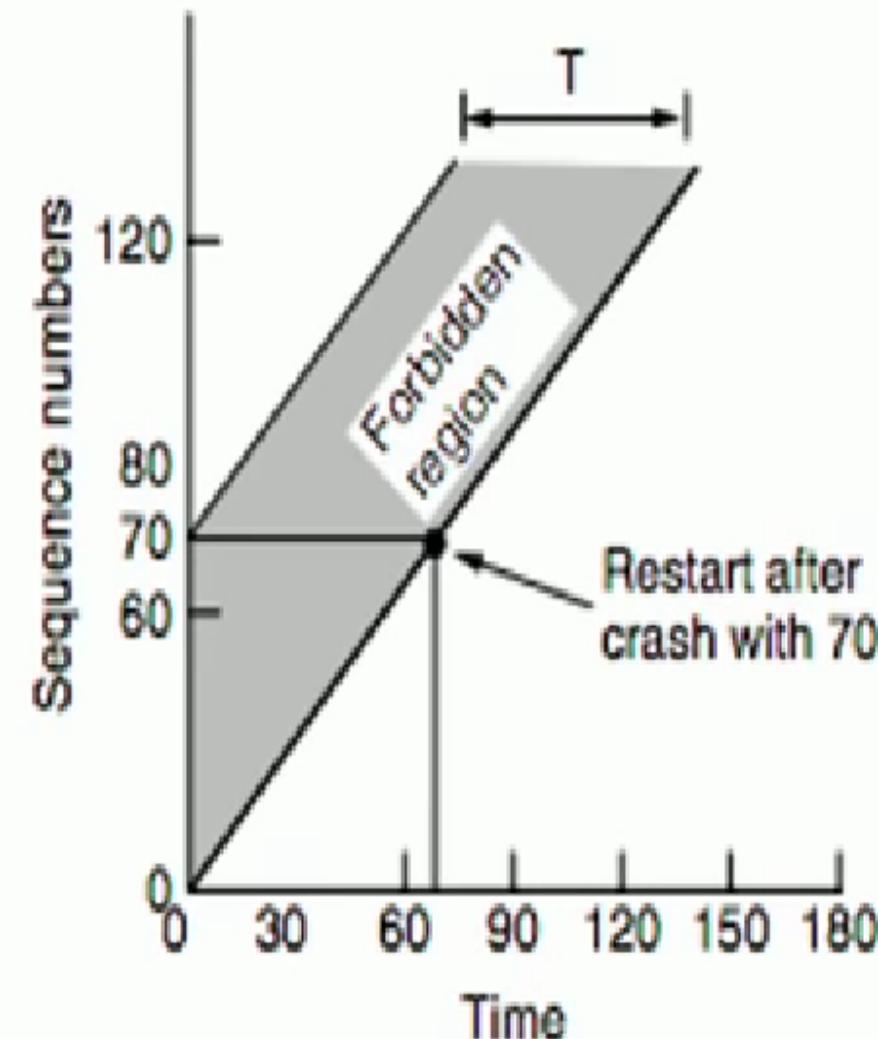
Connection Establishment – Handling Delayed Duplicates

- Receiver receives two segments having the same sequence number within a duration T
 - One packet must be the duplicate
 - The receiver discards the duplicate packets.



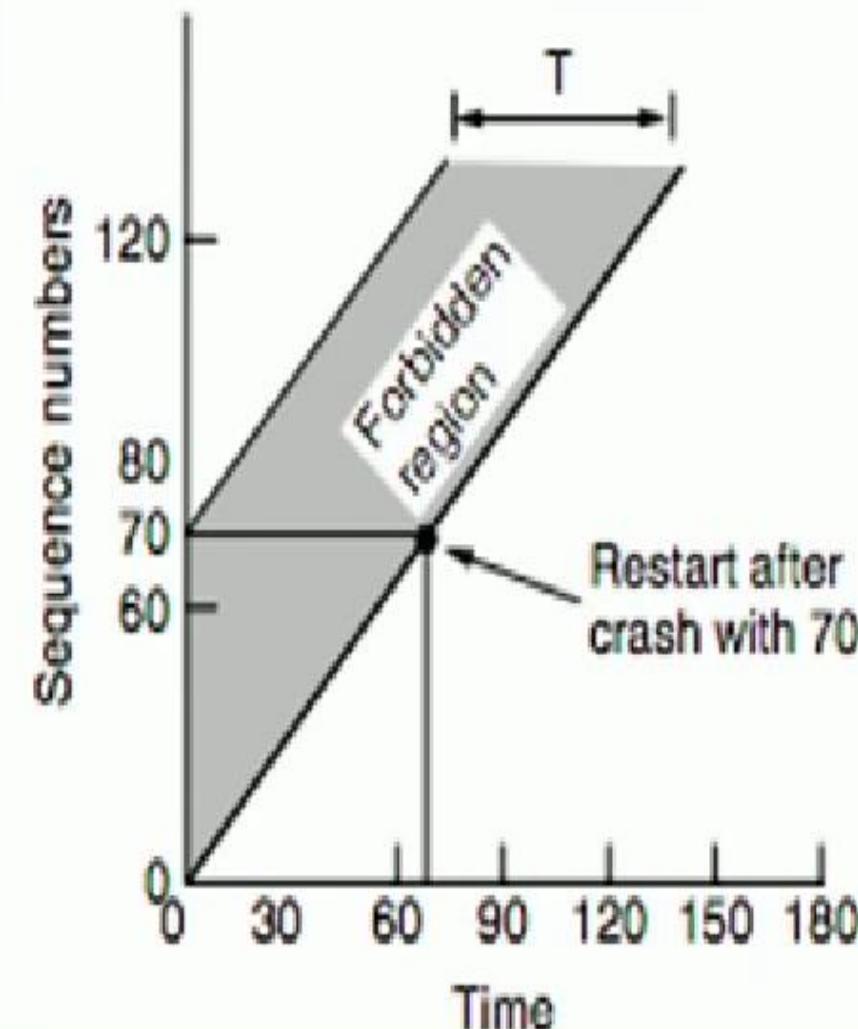
Connection Establishment – Handling Delayed Duplicates

- For a crashed device, the transport entity remains idle for a duration T after recovery, to ensure that all packets from the previous connection are dead – **not a good solution**



Connection Establishment – Handling Delayed Duplicates

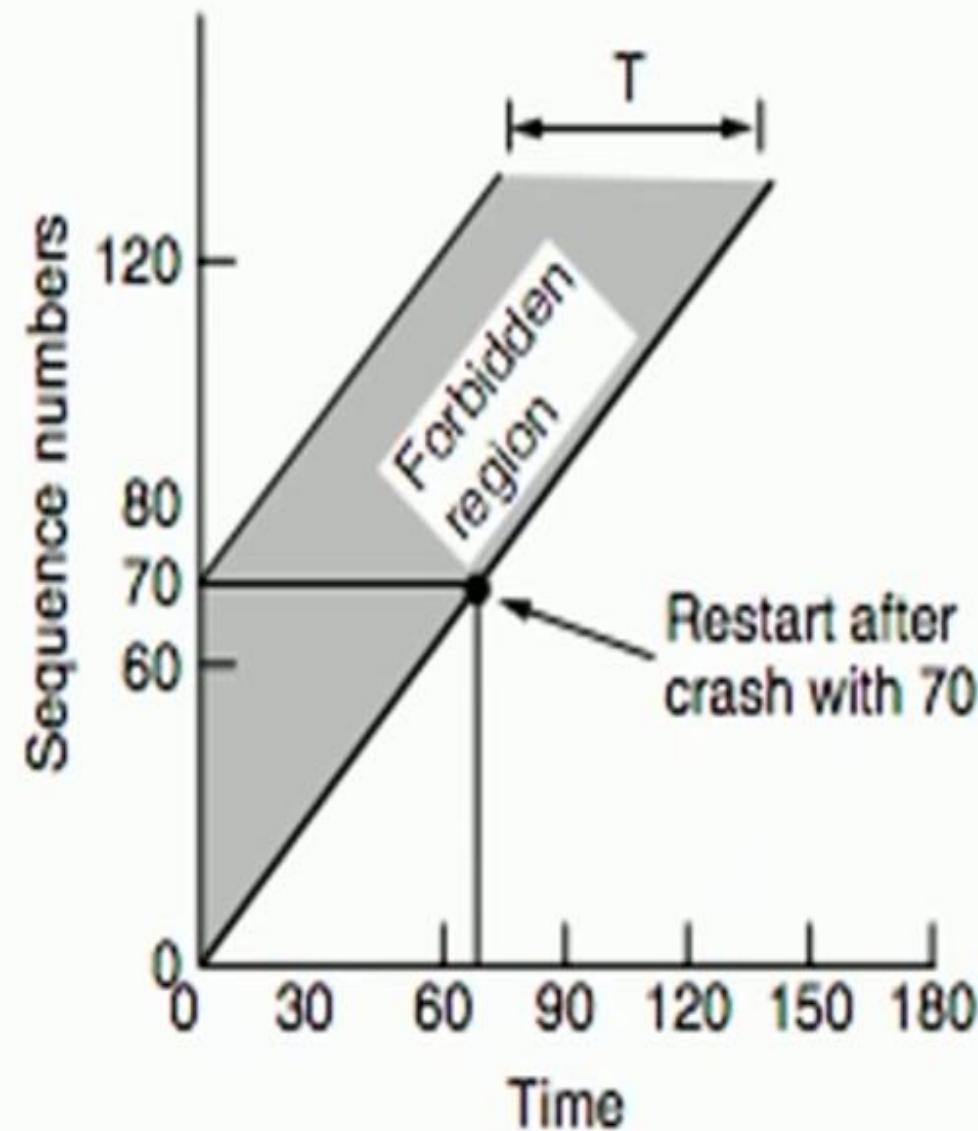
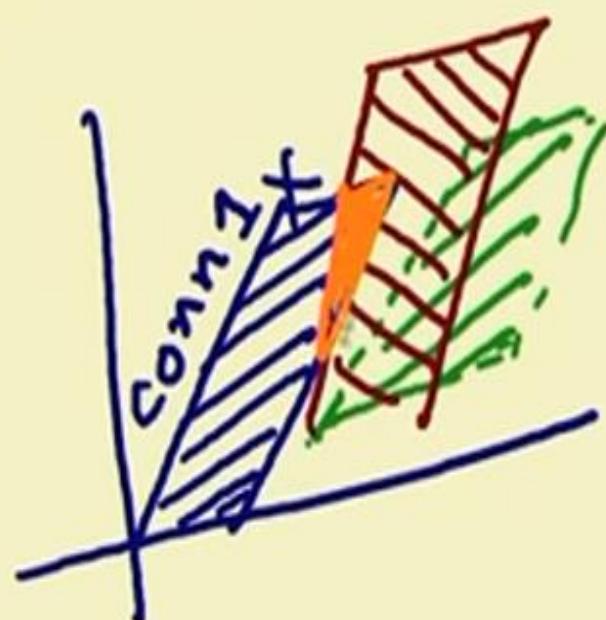
- **Adjust the initial sequence numbers properly** - A host does not restart with a sequence number in the **forbidden region**, based on the sequence number it used before crash and the time duration T.



Packet Sequence Numbers are Out of the Forbidden Region

Two possible source of problems

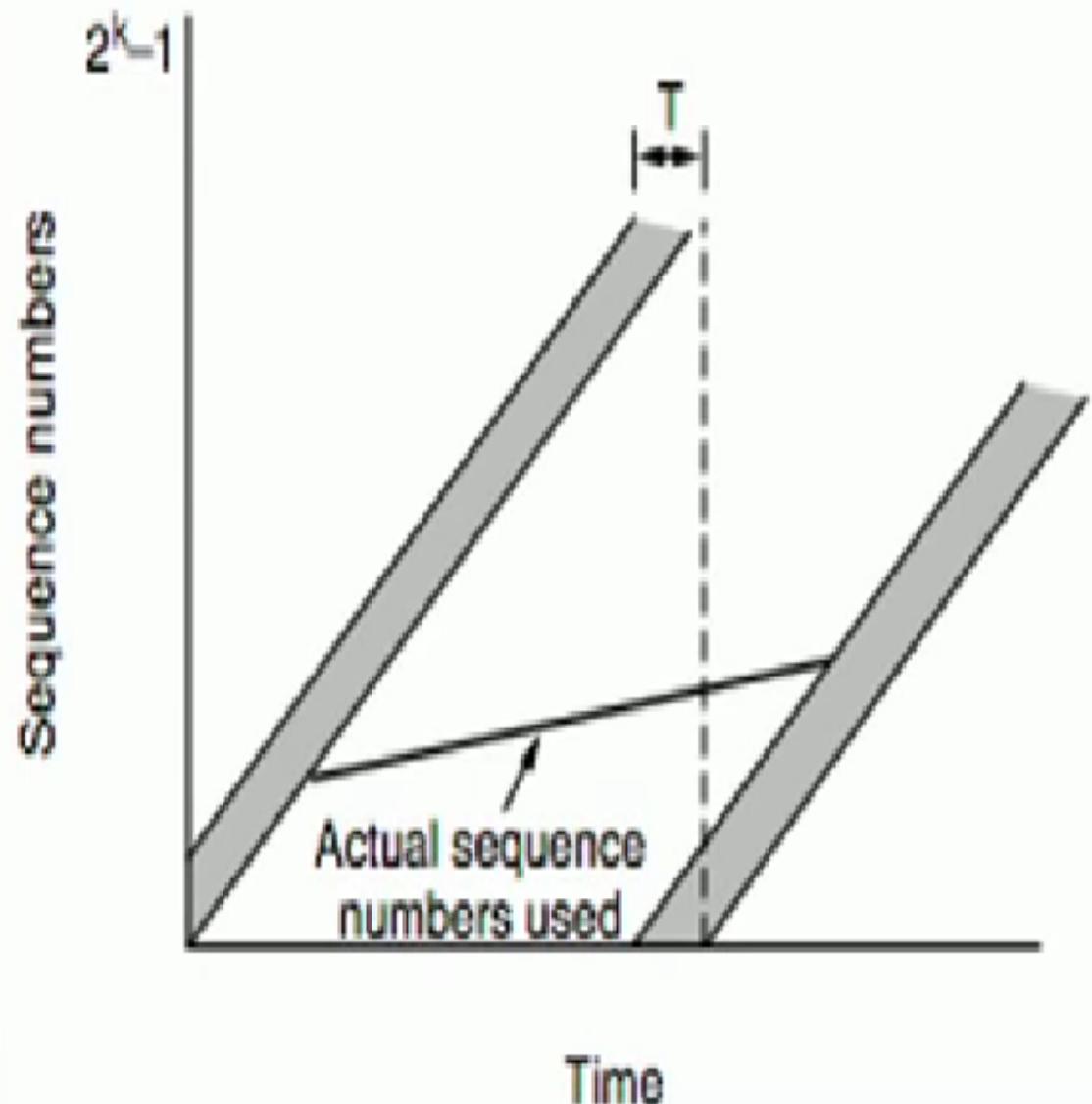
1. A host sends too much data too fast on a newly opened connection



Packet Sequence Numbers are Out of the Forbidden Region

Two possible source of problems

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



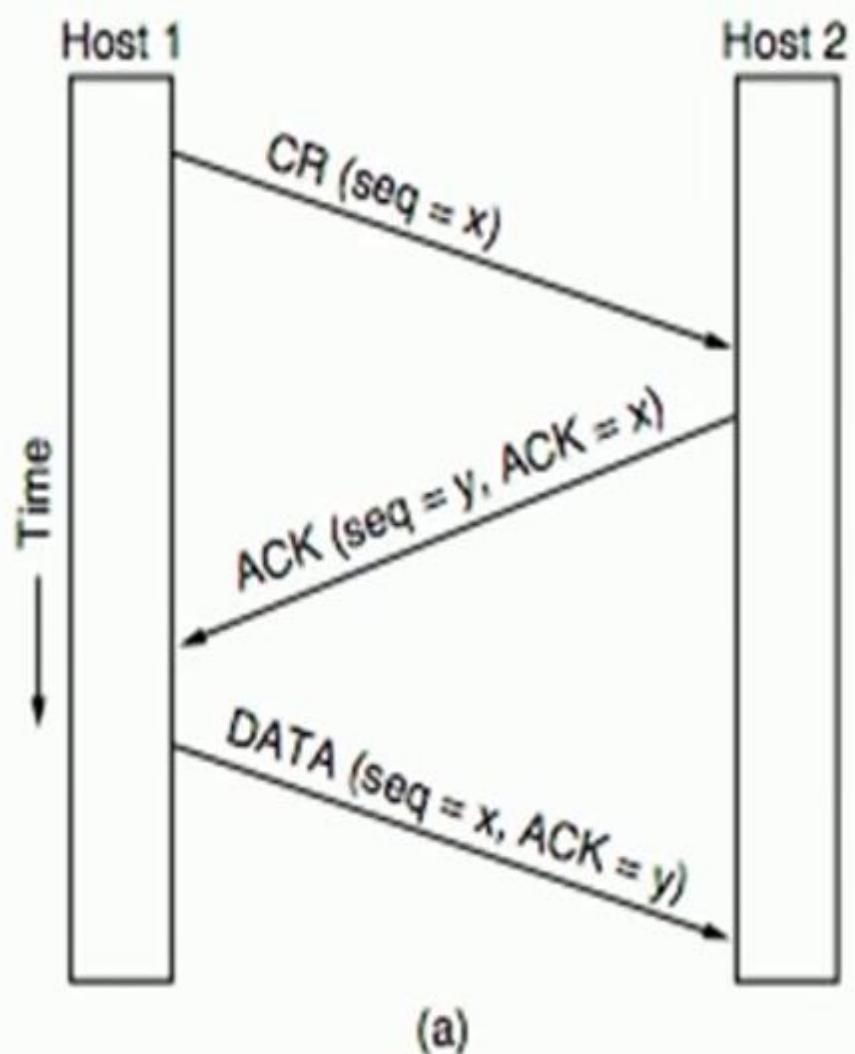
Adjusting the Sending Rate based on Sequence Numbers

- 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

Adjusting the Sending Rate based on Sequence Numbers

- 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

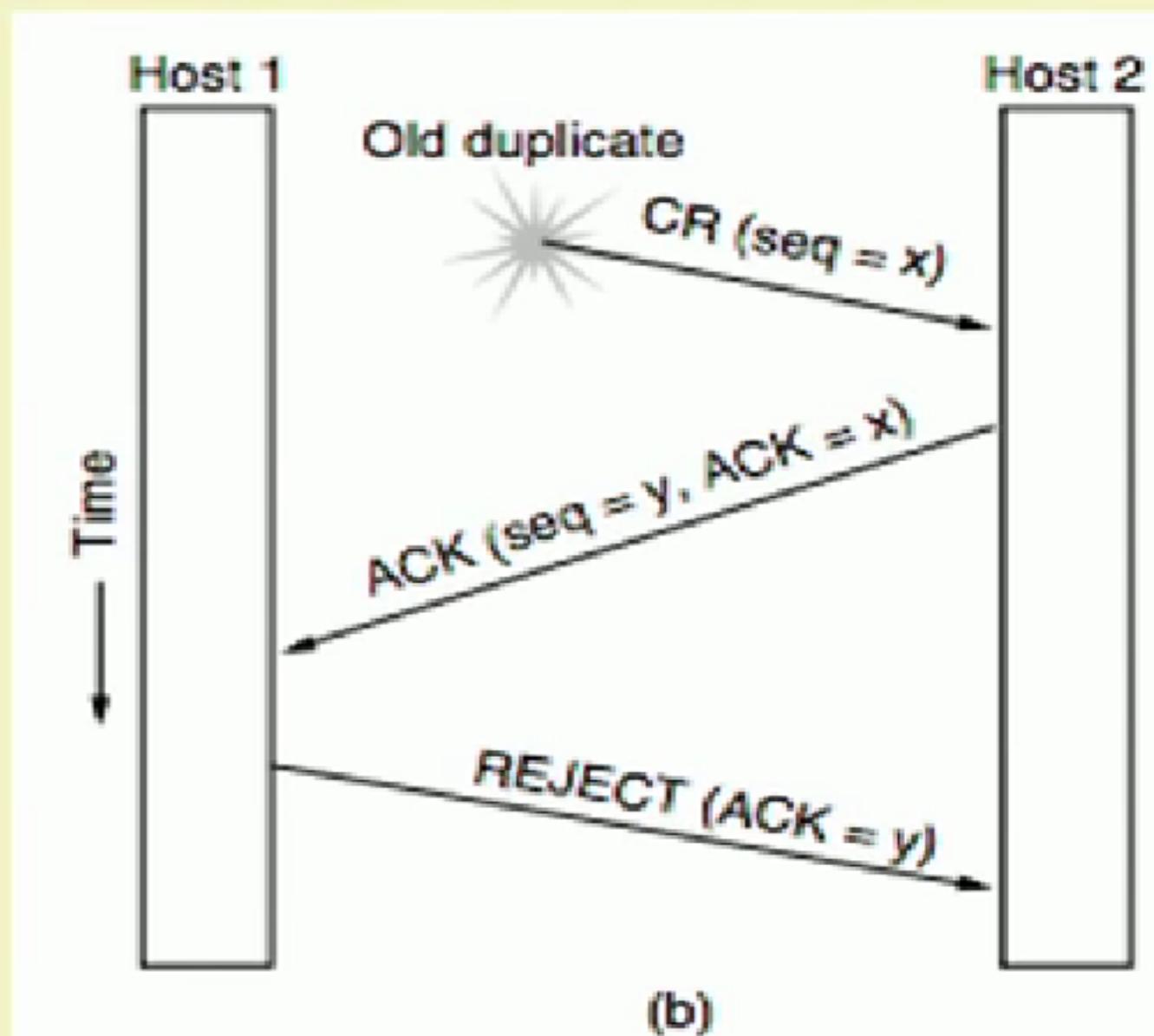
Three Way Handshake



- 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

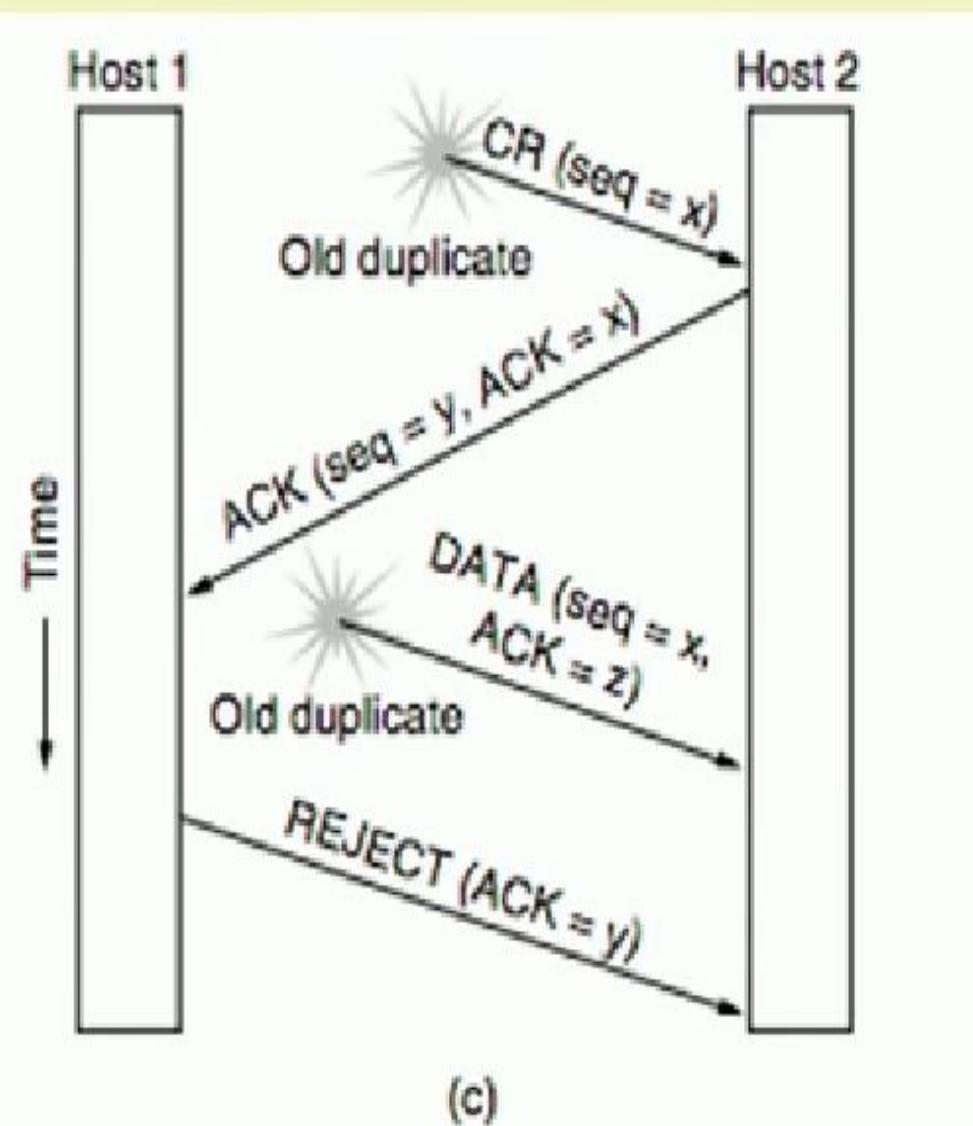
CONNECTION REQUEST is a Delayed Duplicate

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



CONNECTION REQUEST and ACK both are Delayed Duplicates

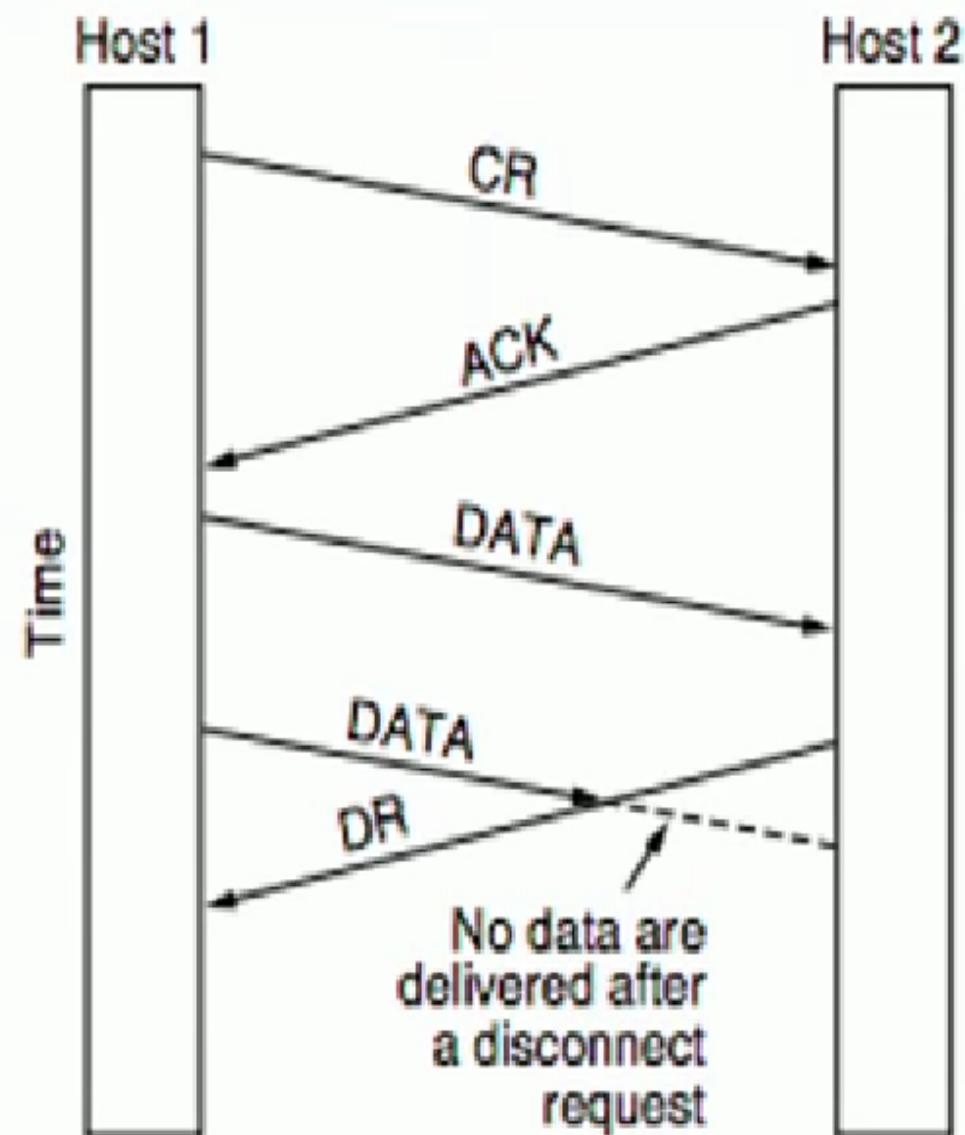
Source: Computer Networks
(5th Edition) by Tanenbaum,
Wetherell



Connection Release – Asymmetric Release

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

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



Connection Release – Symmetric Release

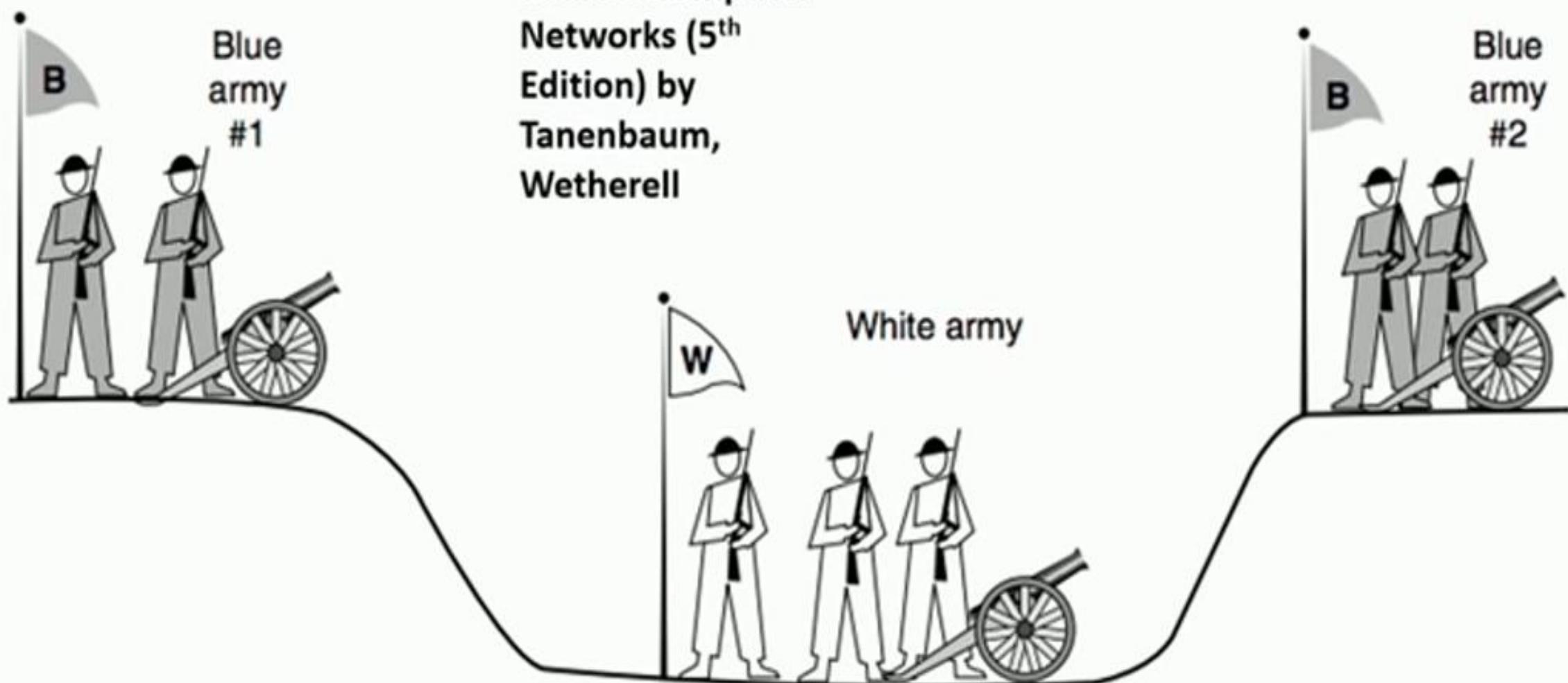
- Treats the connection as two separate unidirectional connections and requires each one to be released separately
- Does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.
- What can be a protocol for this?

Connection Release – Symmetric Release

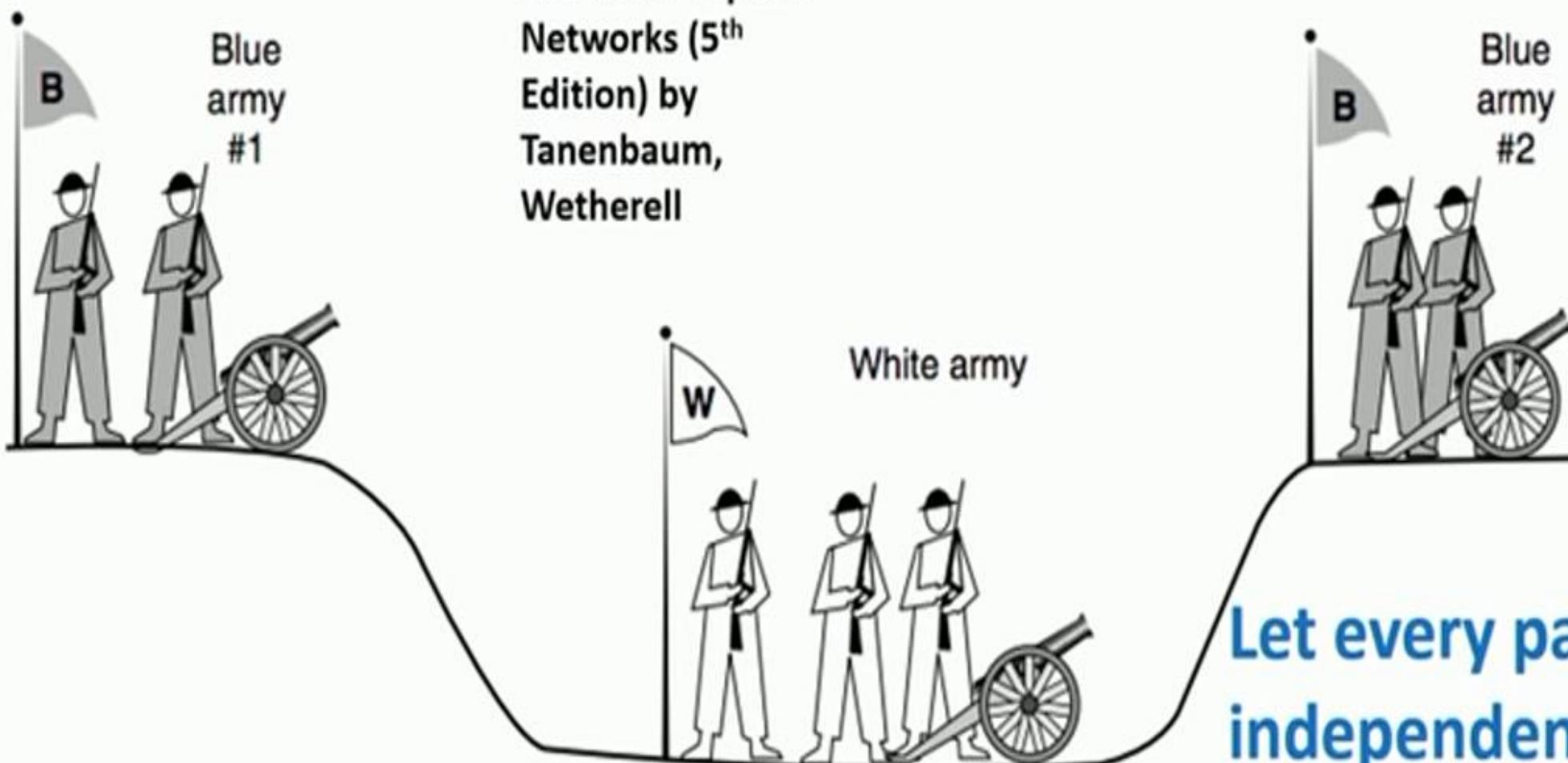
- Treats the connection as two separate unidirectional connections and requires each one to be released separately
- Does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.
- What can be a protocol for this?
 - Host 1: “I am done”
 - Host 2: “I am done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

The Two Army Problem

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



The Two Army Problem

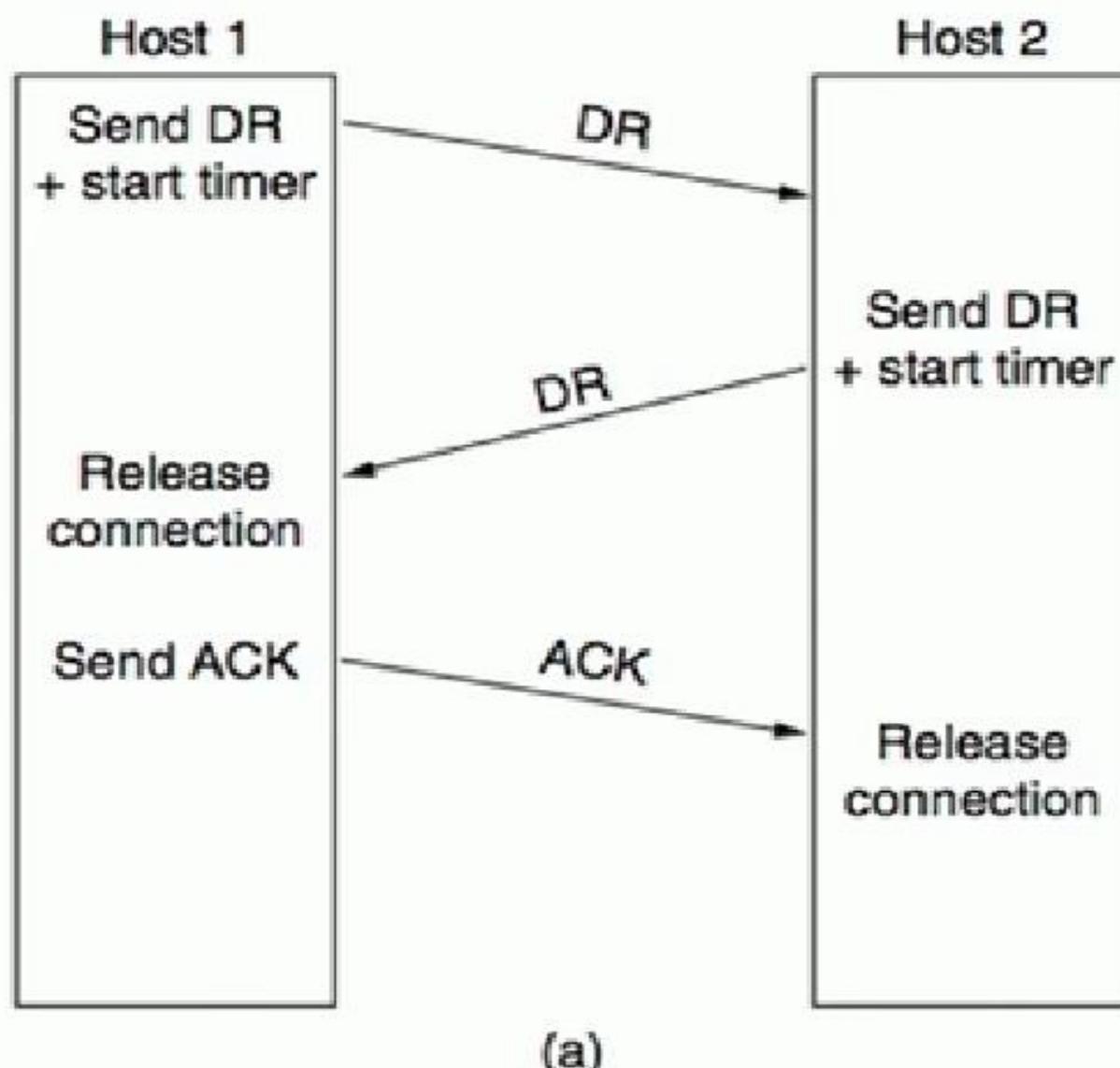


Source: Computer
Networks (5th
Edition) by
Tanenbaum,
Wetherell

Let every party take
independent
decisions

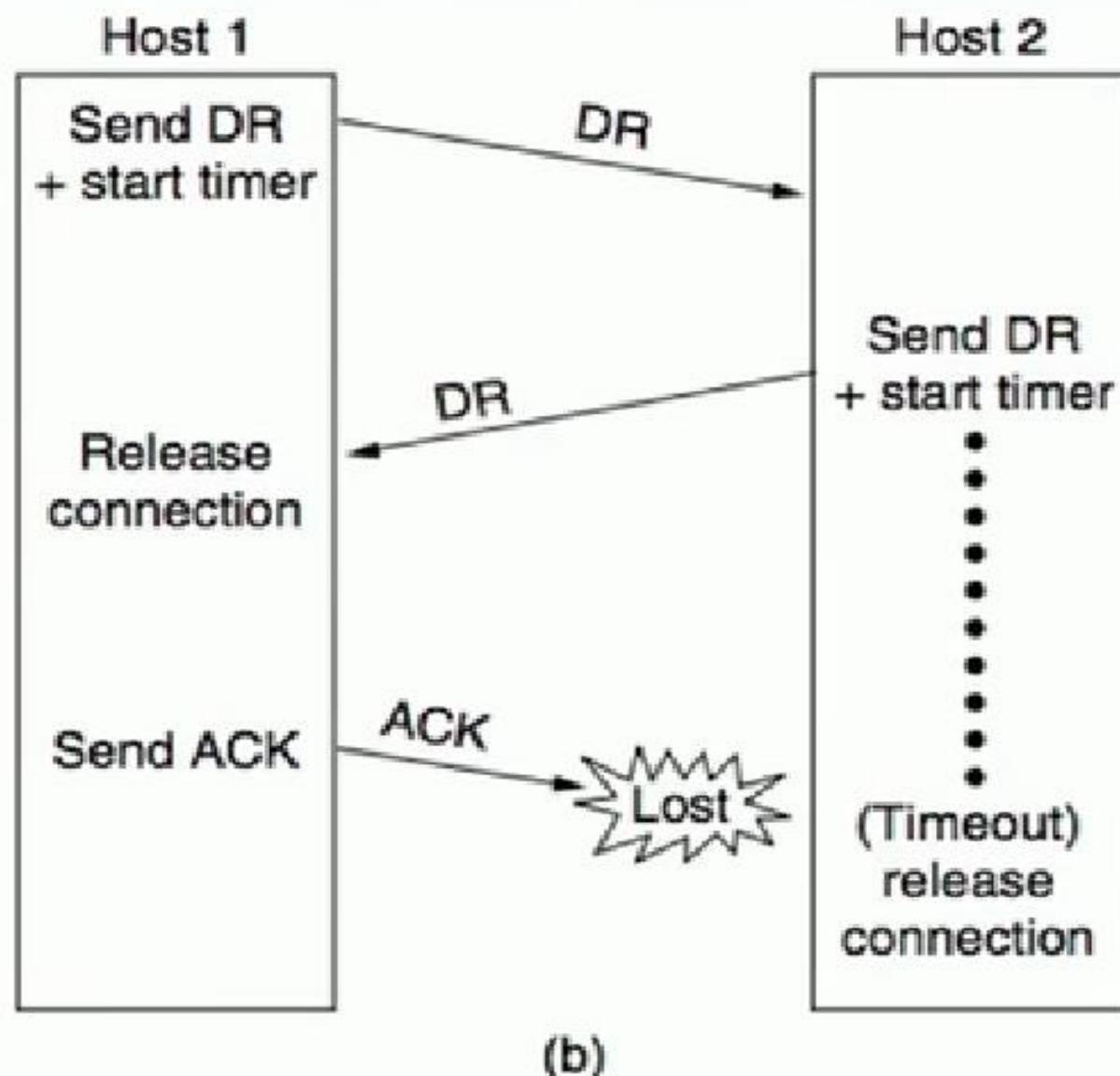
No protocol exists to solve this

Connection Release



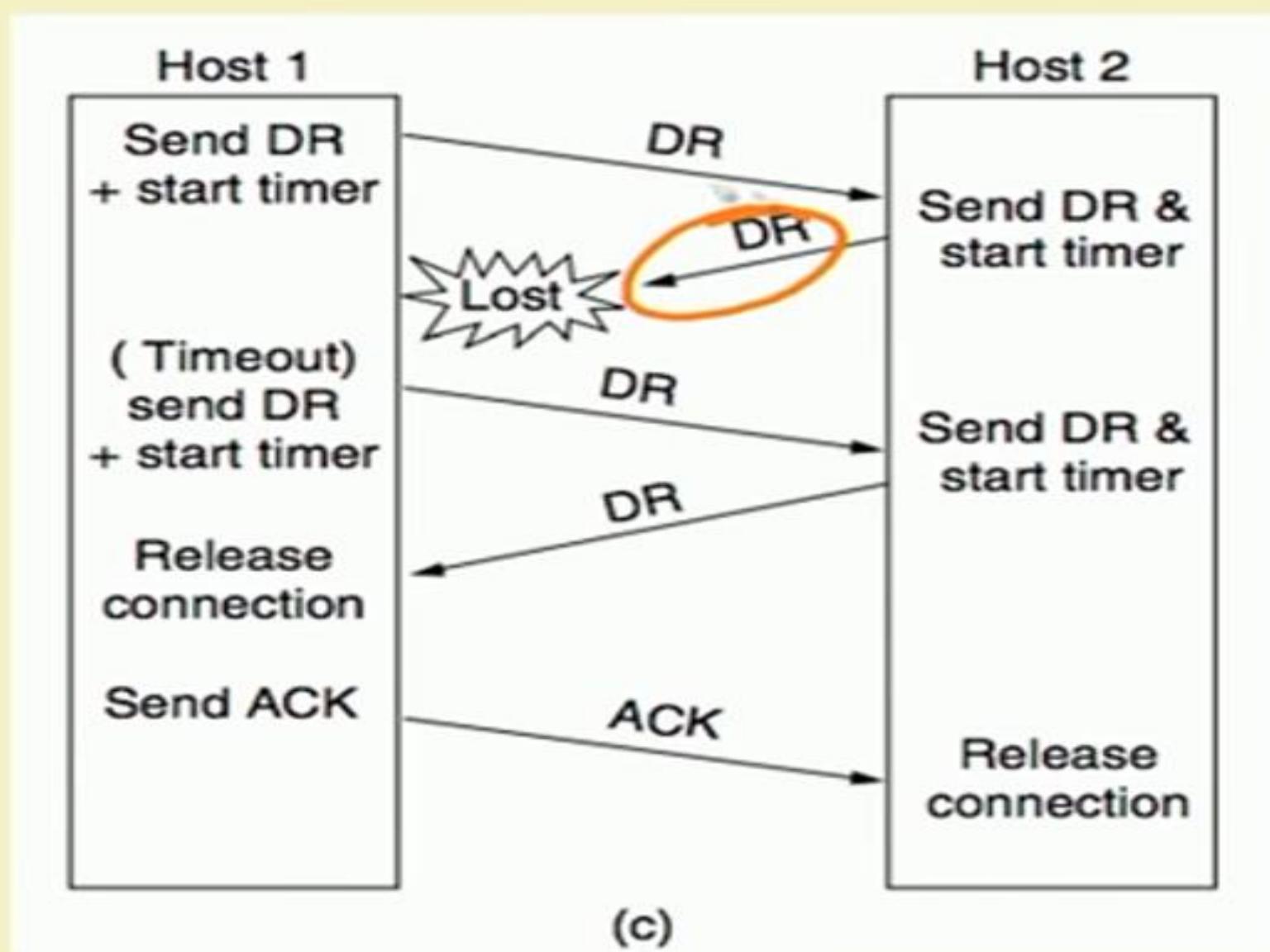
Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Connection Release – Final ACK Lost

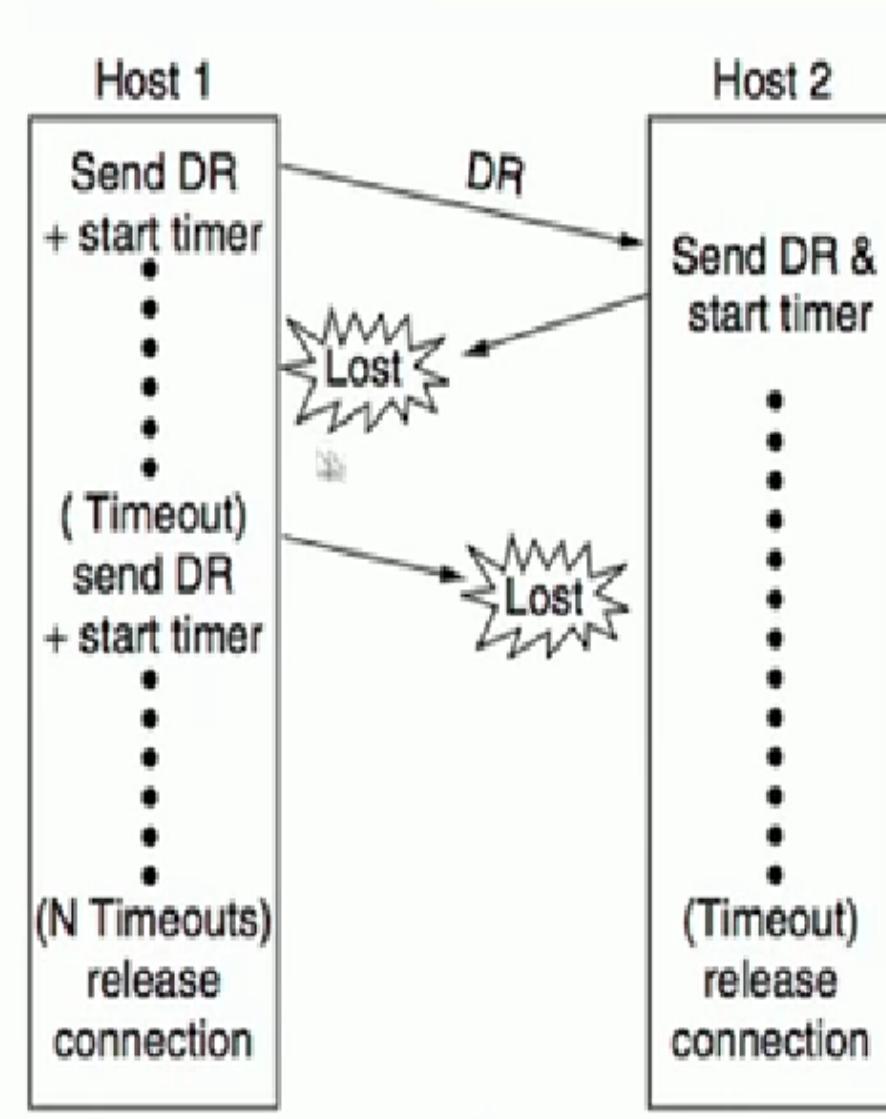


Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Connection Release – Response Lost



Connection Release – Response Lost and Subsequent DRs Lost



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

(d)

Flow Control and Reliability