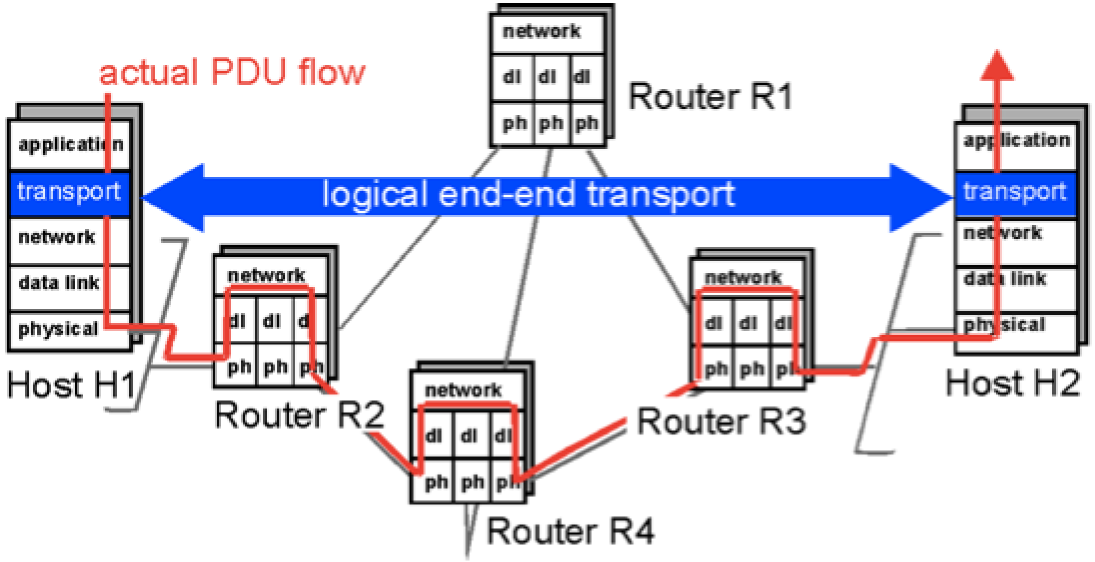
CS 31006: Computer Networks – Transport Layer Services



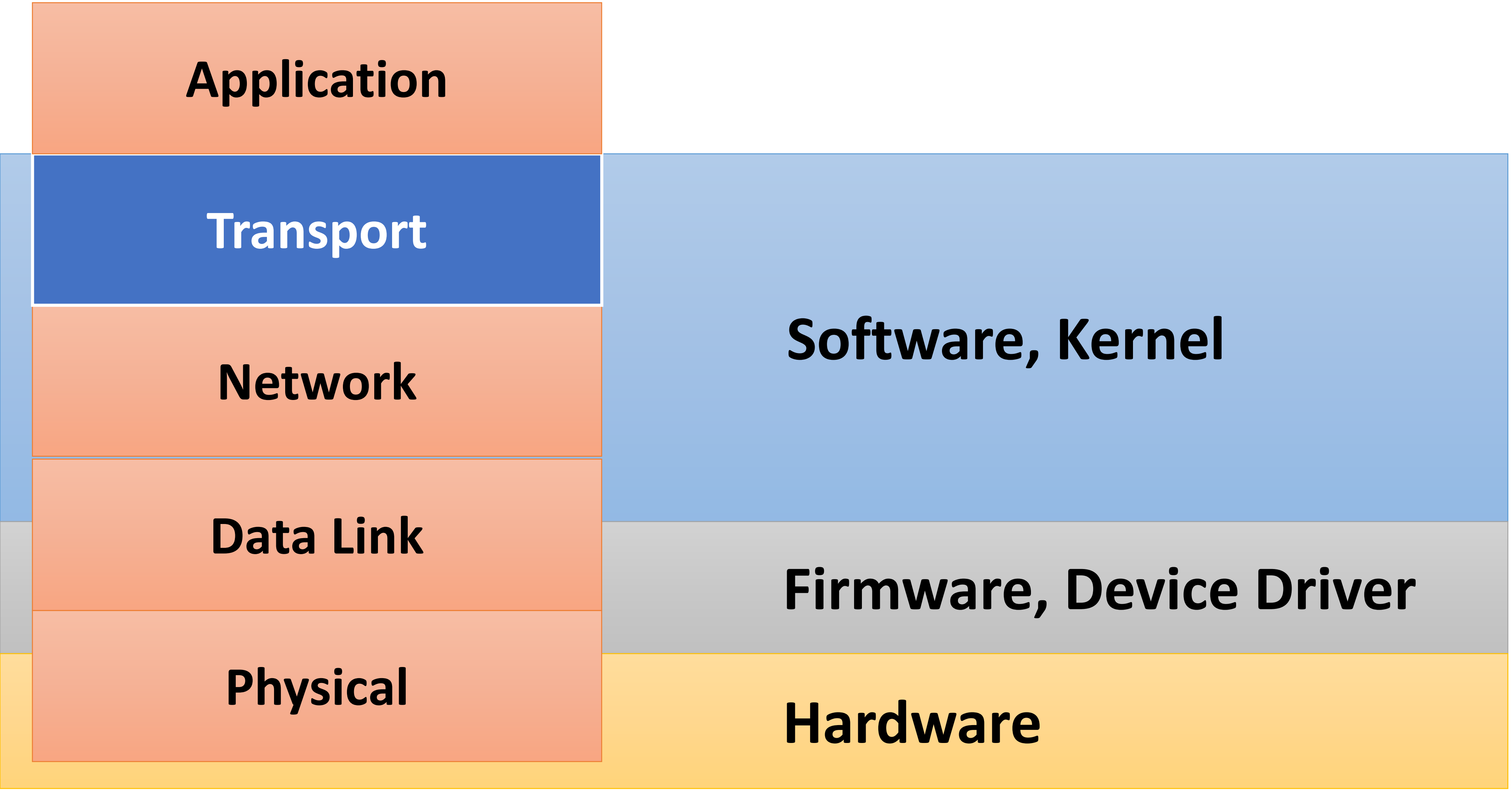
Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

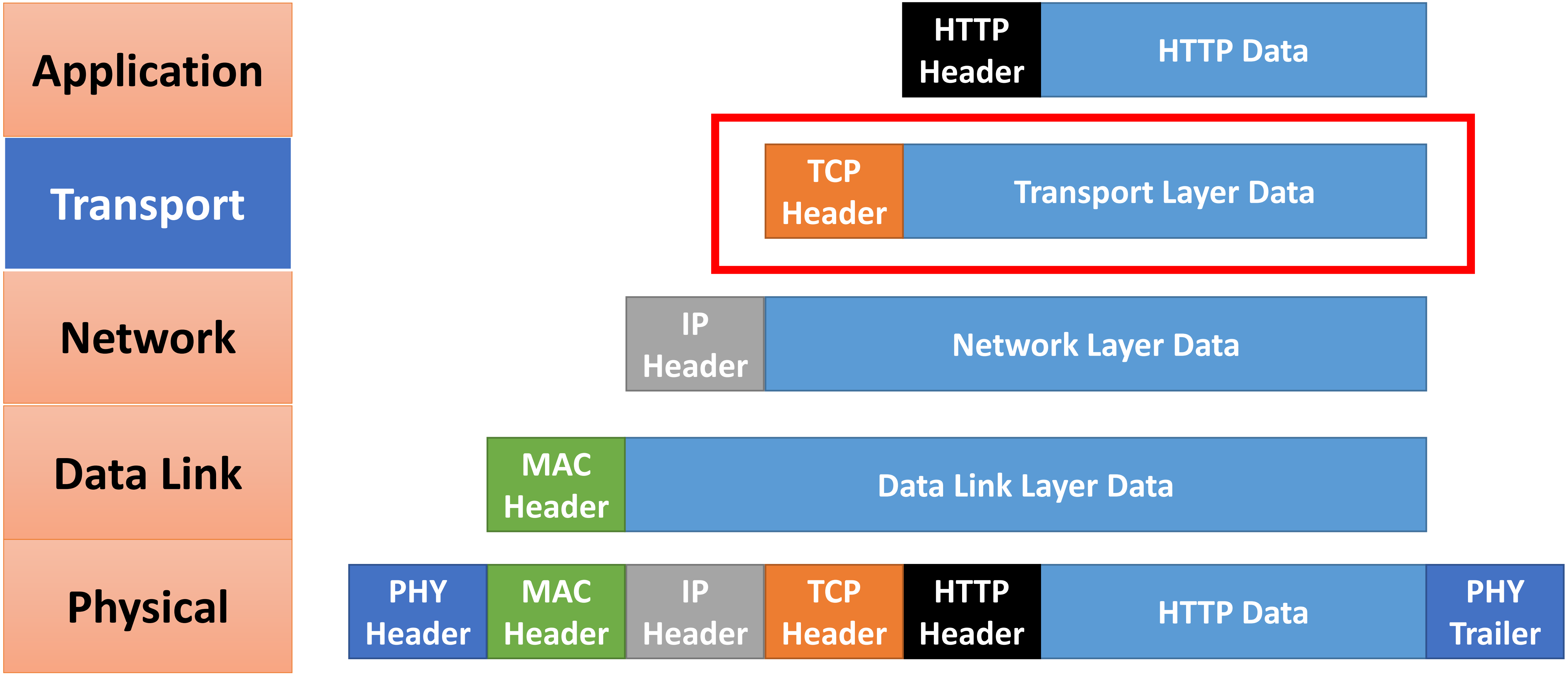
Rajat Subhra Chakraborty
rschakraborty@cse.iitkgp.ac.in

Sandip Chakraborty
sandipc@cse.iitkgp.ac.in

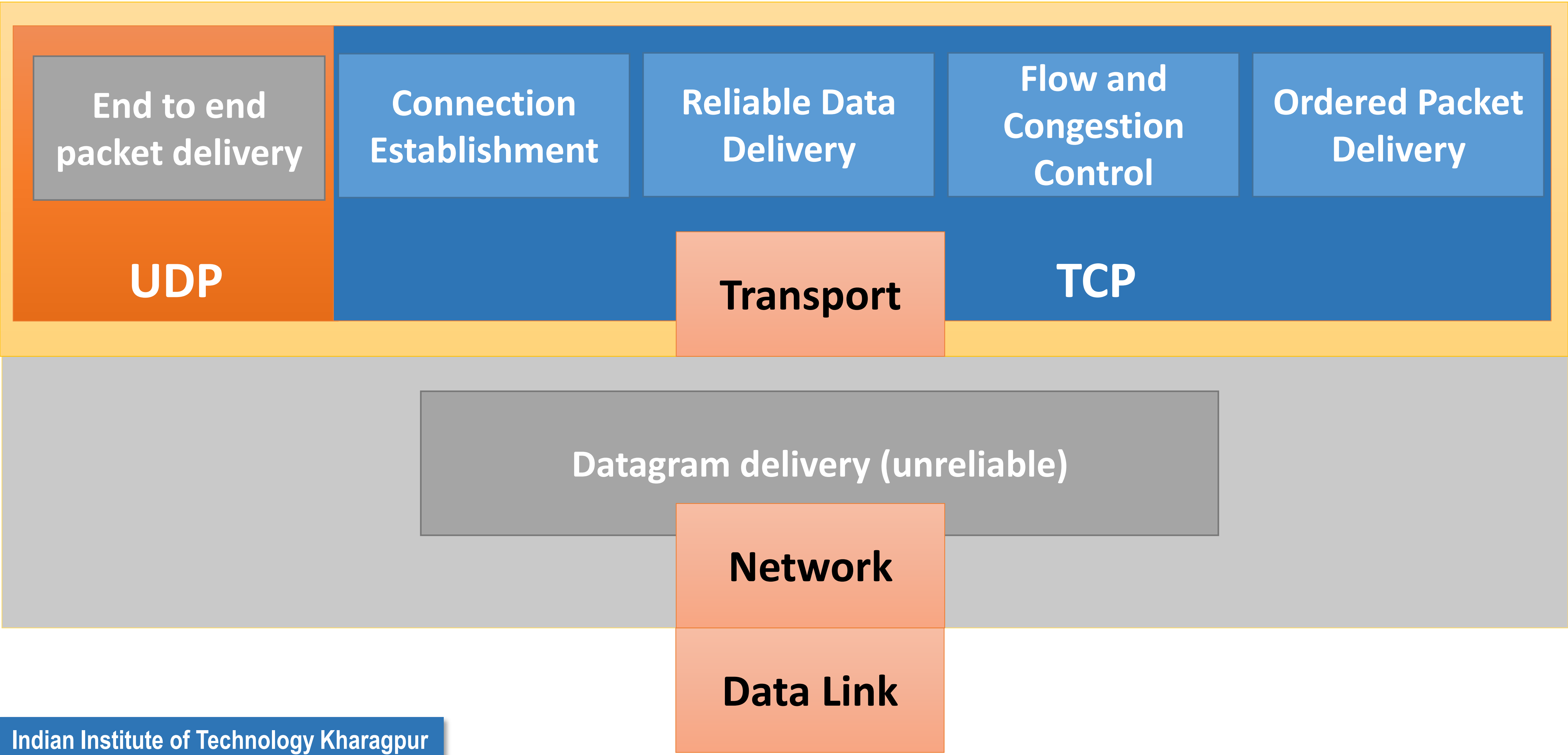
Protocol Stack Implementation in a Host



How Application Data Passes Through Different Layers



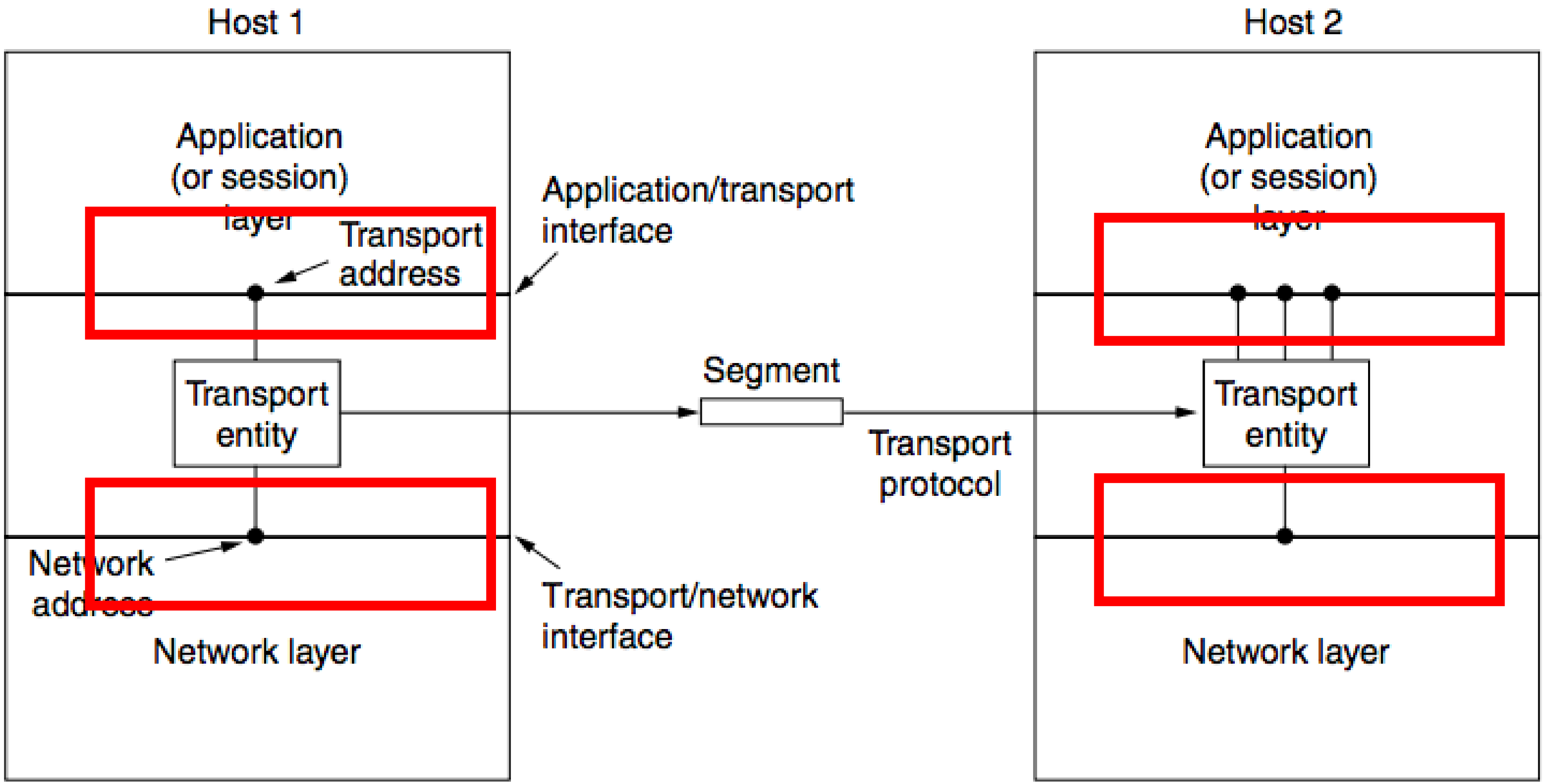
Transport Layer Services



Transport Layer – Interfacing with Application and Network

Port Number

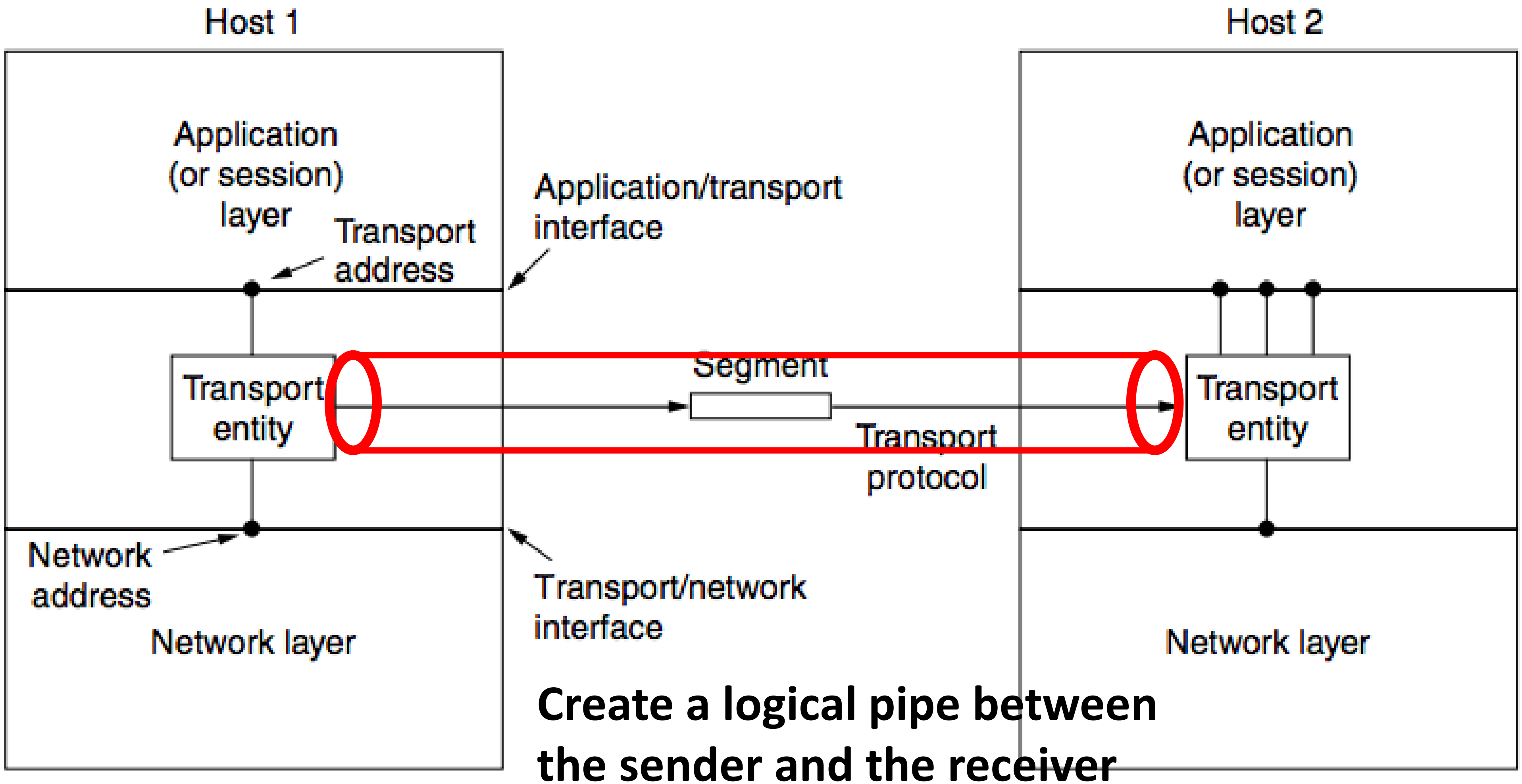
IP Address



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

Transport Layer – Interfacing with Application and Network

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



Create a logical pipe between the sender and the receiver and monitor the data transmission through this pipe

Transport Service Primitives

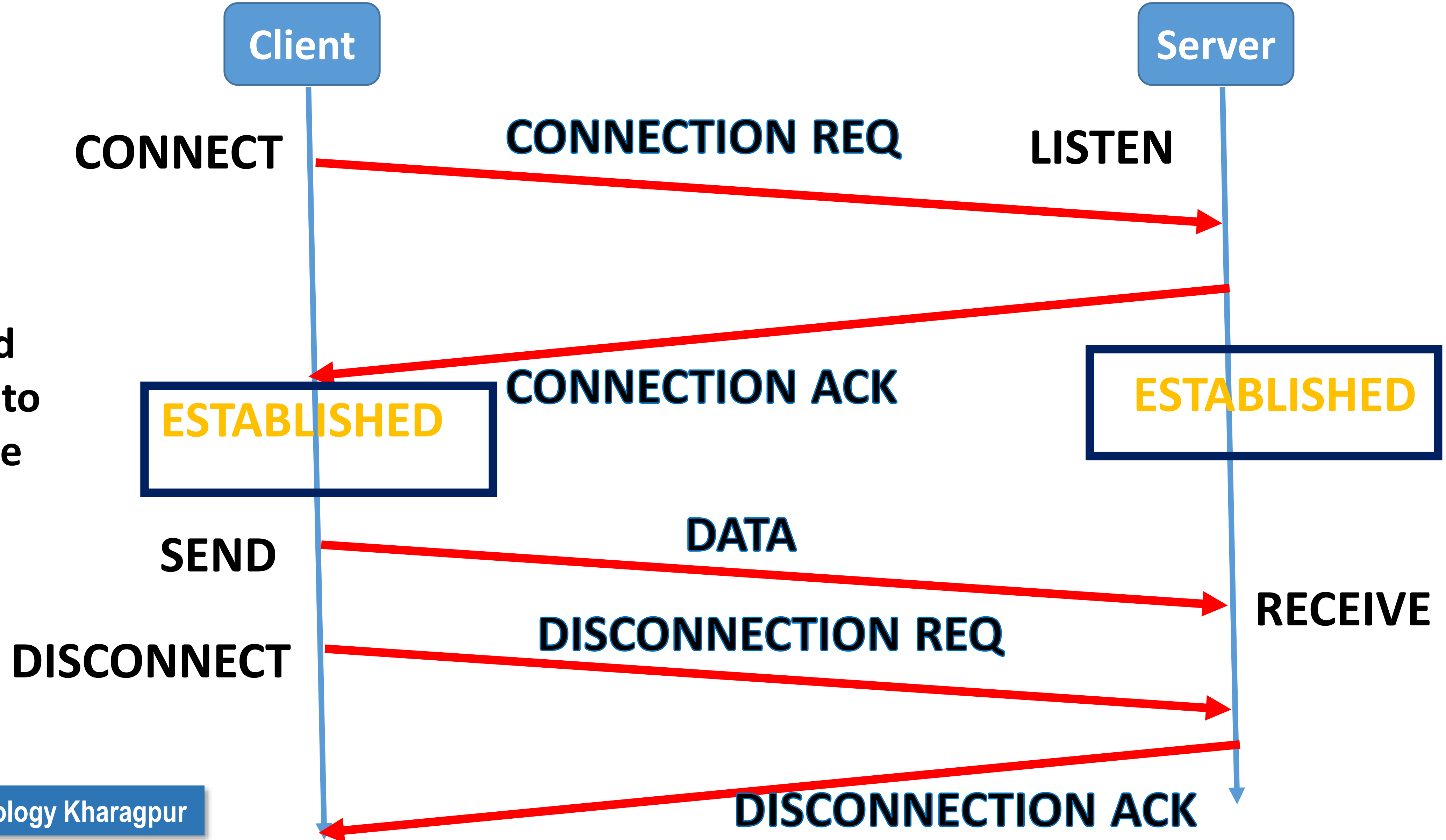
- To allow users to access transport service, the transport layer must provide some operations to the application programs.
- Let us look into a hypothetical transport service primitives, that are provided to the application layer

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	Request a release of the connection

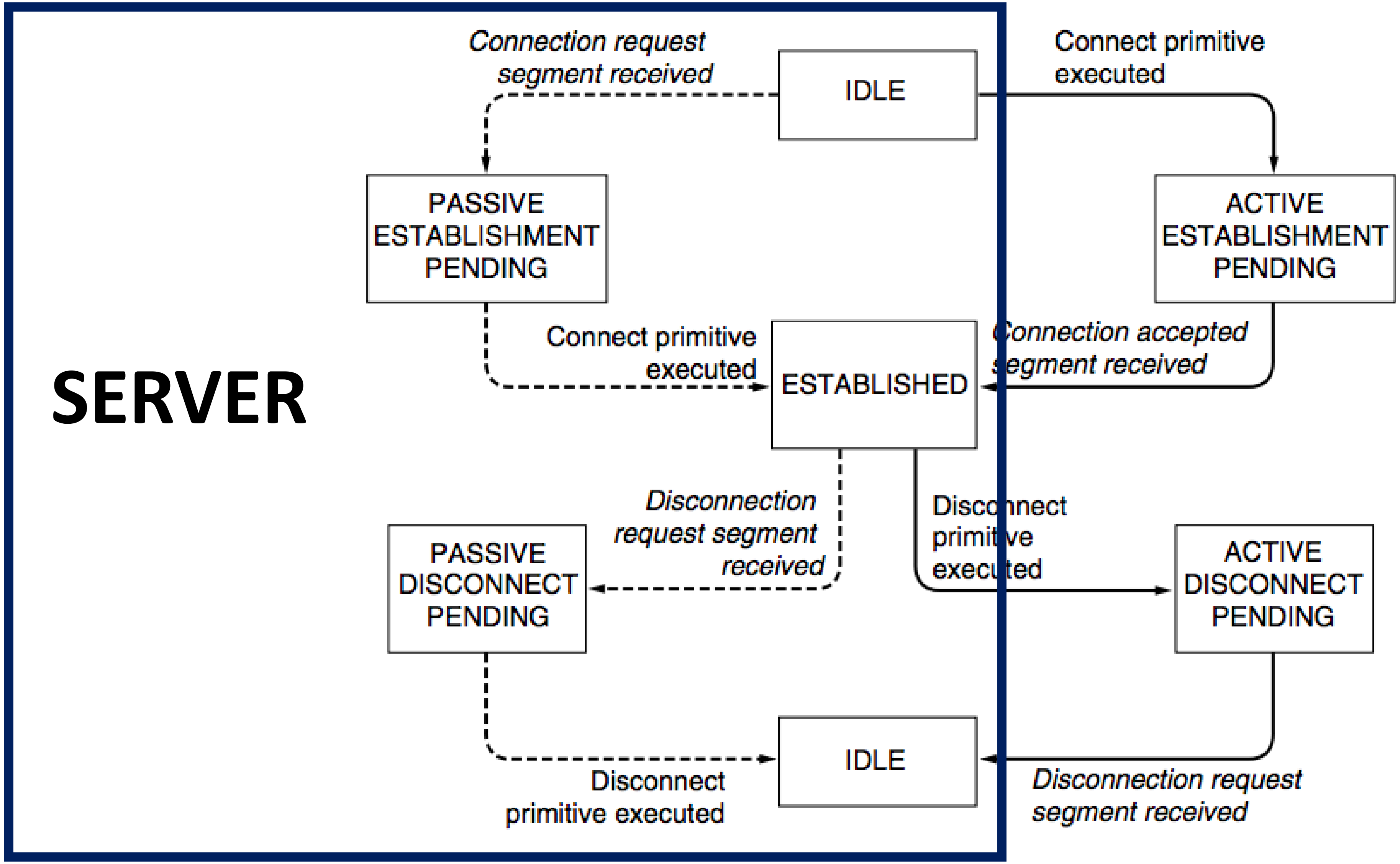
The transport layer needs to remember the state of the pipe, so that appropriate actions can be taken. We need a stateful protocol for transport layer.

Transport Service Primitive – Connection Establishment

The client and server needs to remember the state

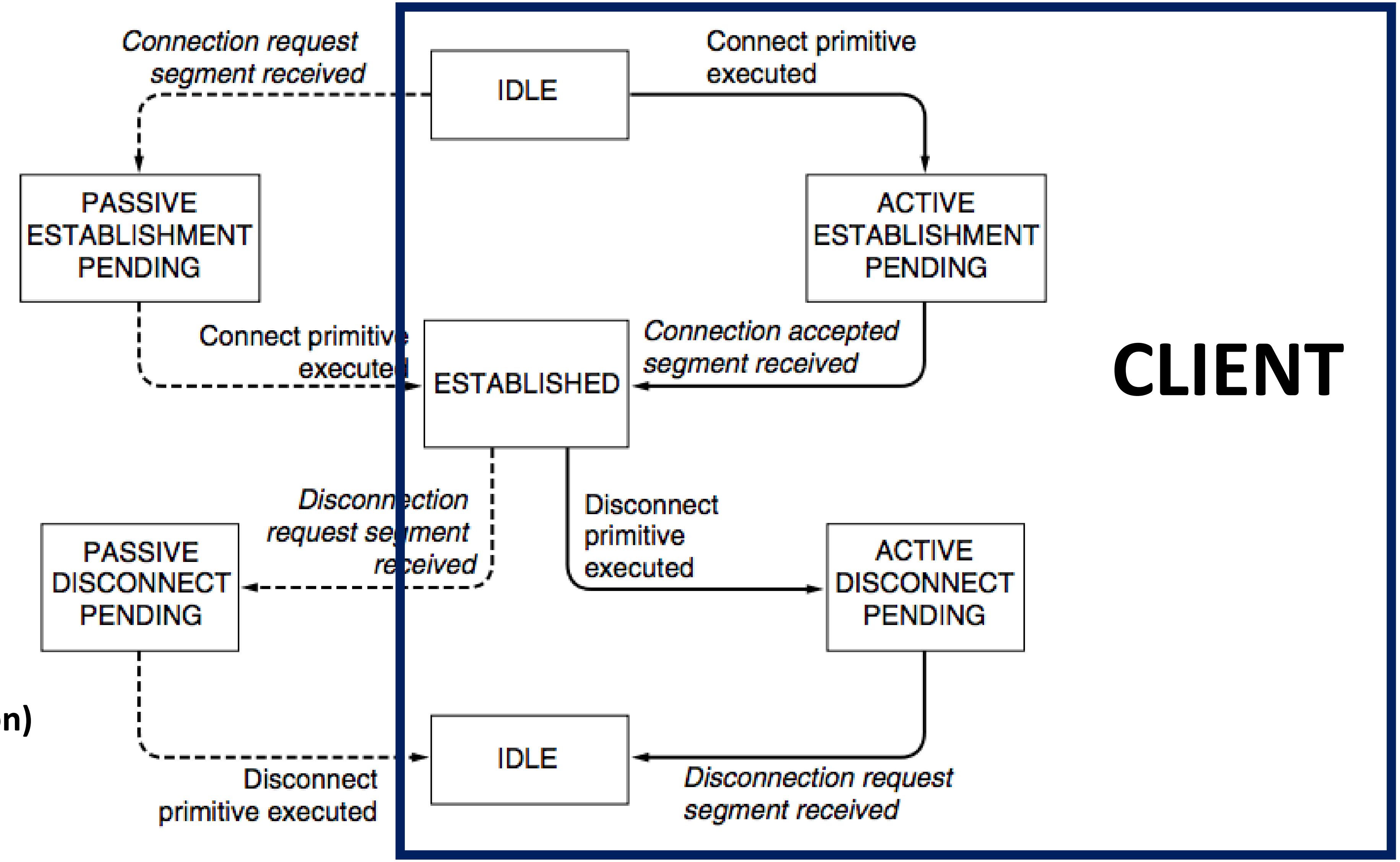


Transport Layer Protocol – State Diagram



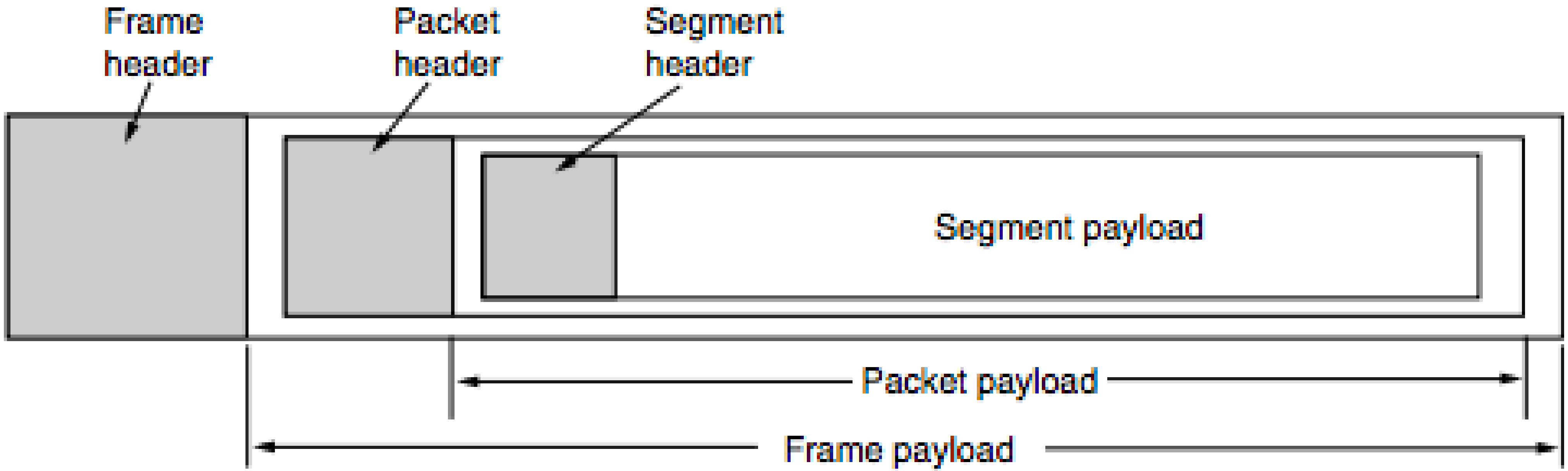
Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Transport Layer Protocol – State Diagram



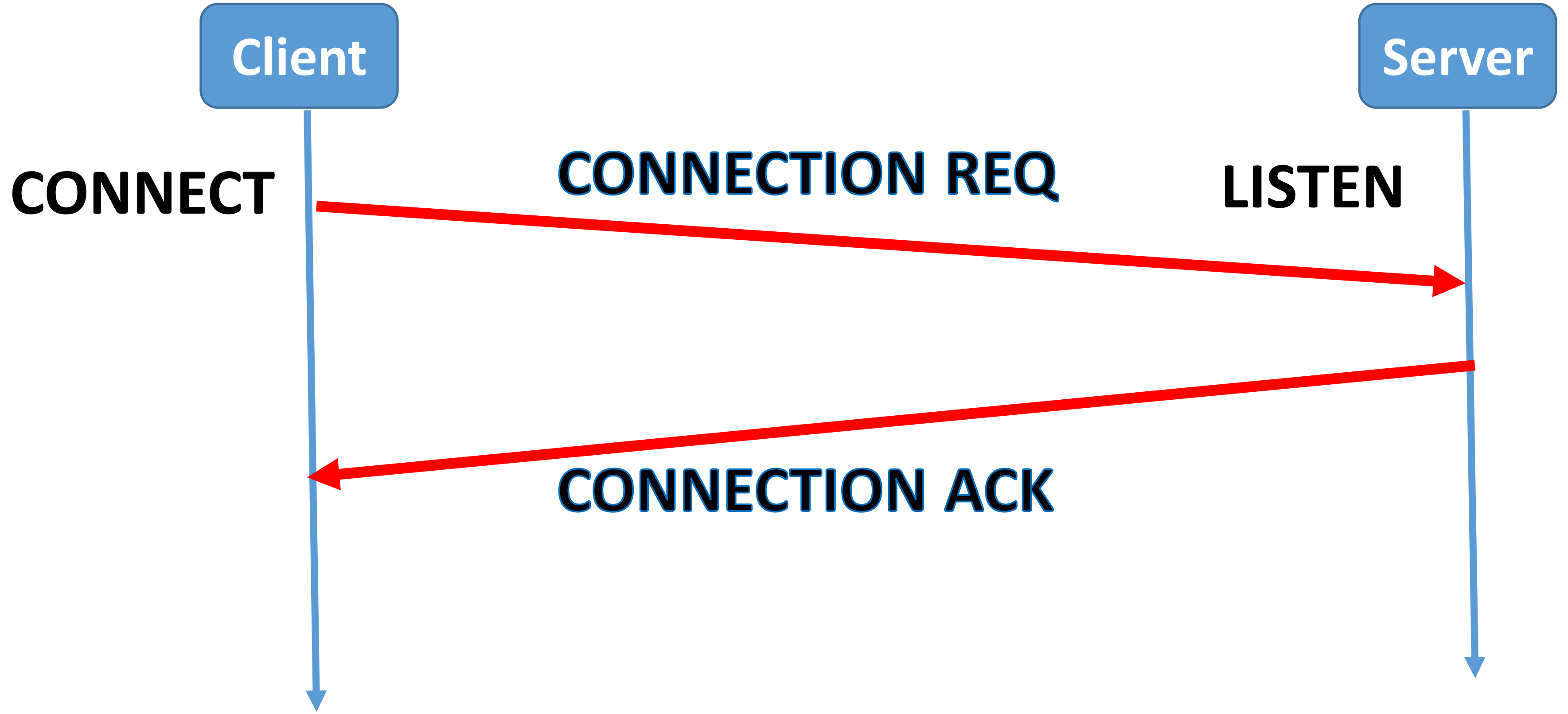
Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Segment, Packet (or Datagram) and Frame



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

Connection Establishment



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

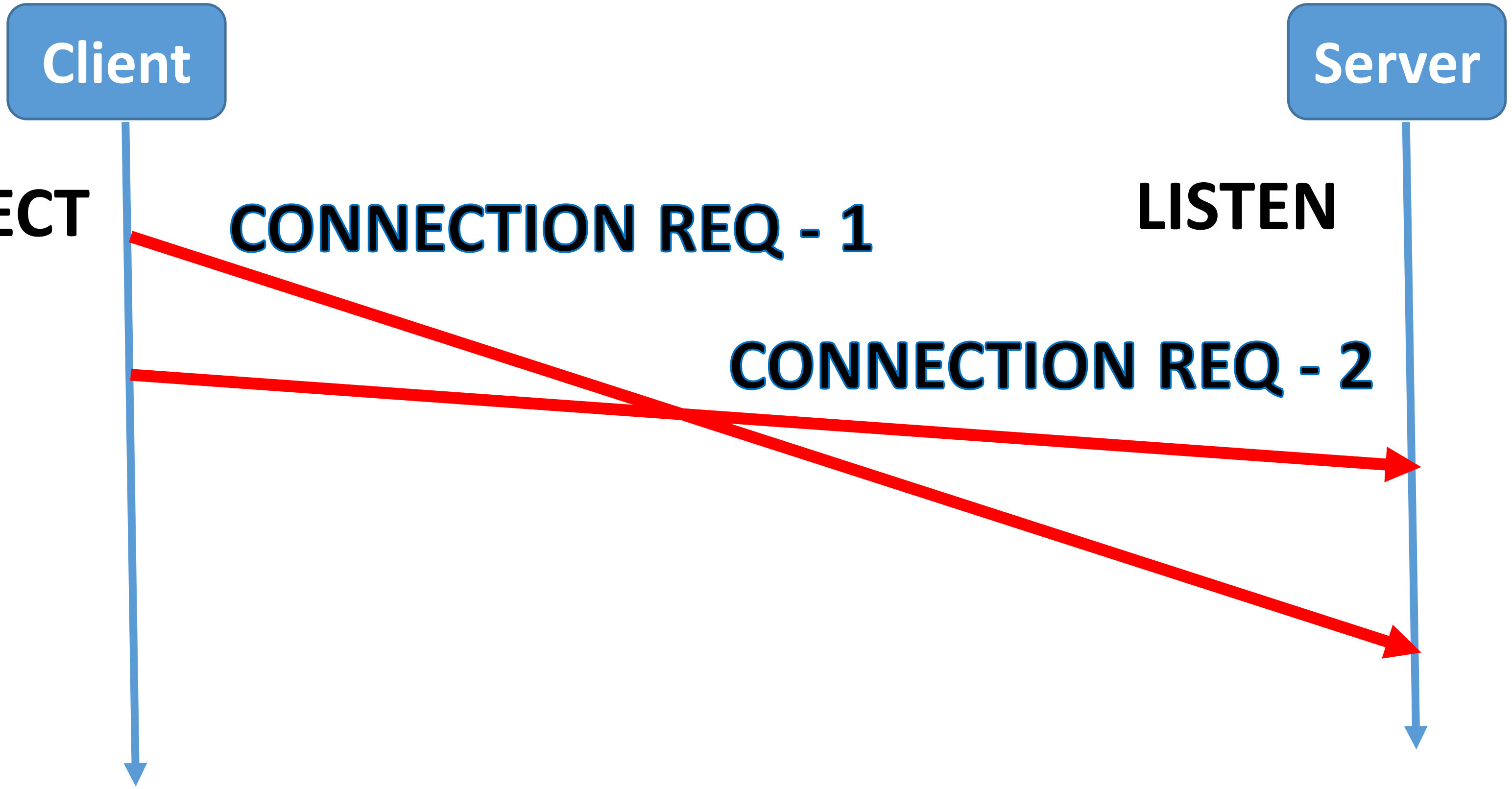
Connection Establishment

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

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

- **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?

- **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**
- 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.
- **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**

- **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**

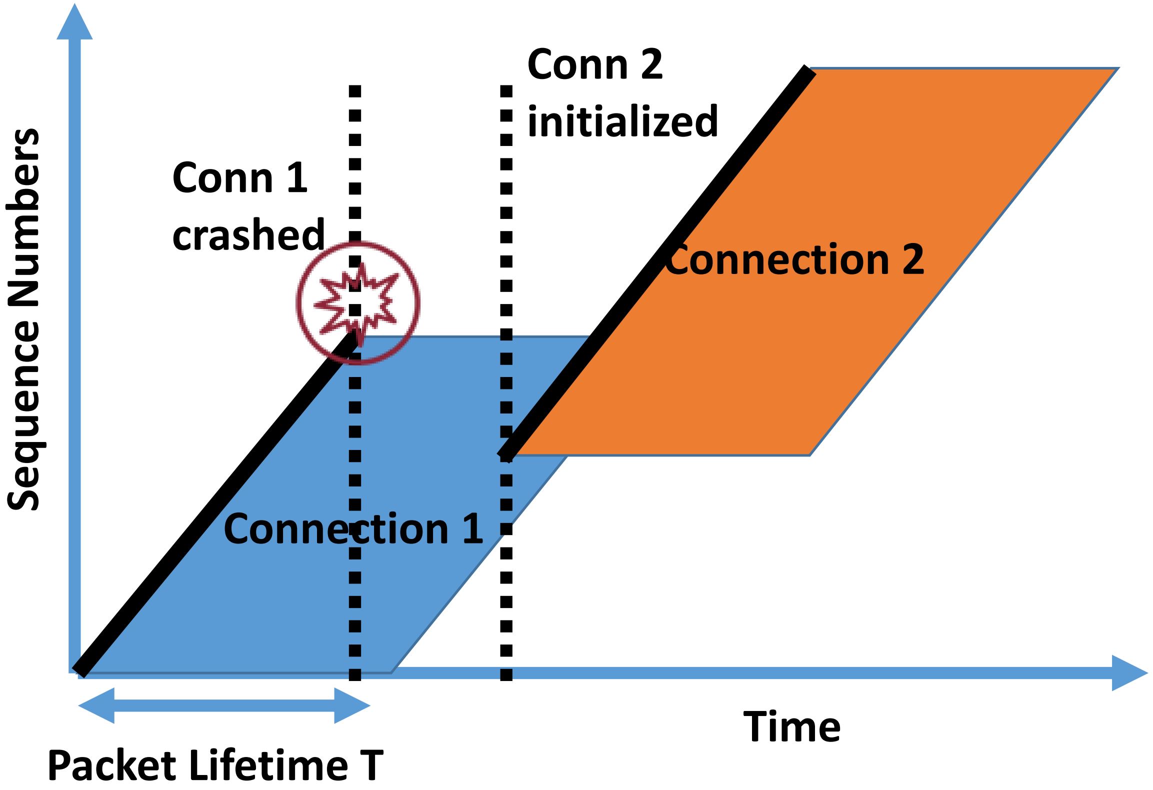
Sequence Number Adjustment

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

  - **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)**

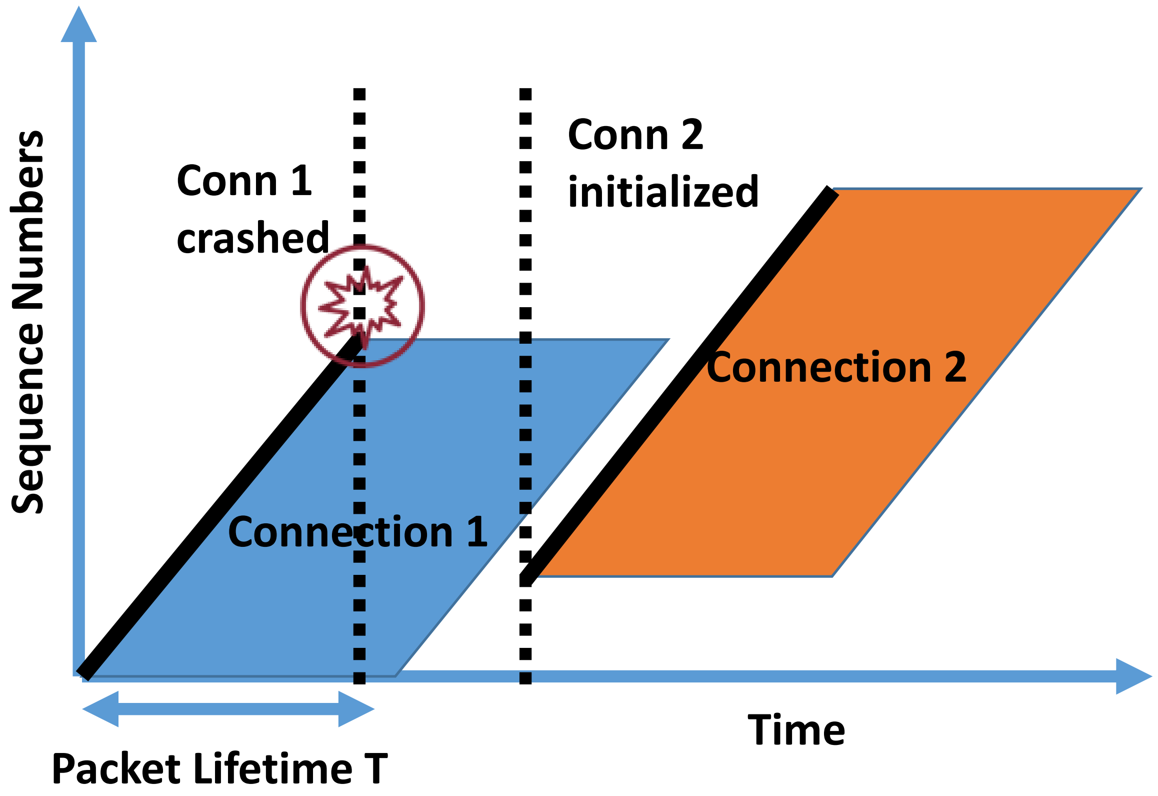
  - **The valid range of sequence numbers must be positively synchronized between the sender and the receiver, whenever a connection is used (three way handshaking followed by the flow control mechanism – once connection is established, only send the data with expected sequence numbers)**

Why Initial Sequence Number is Important

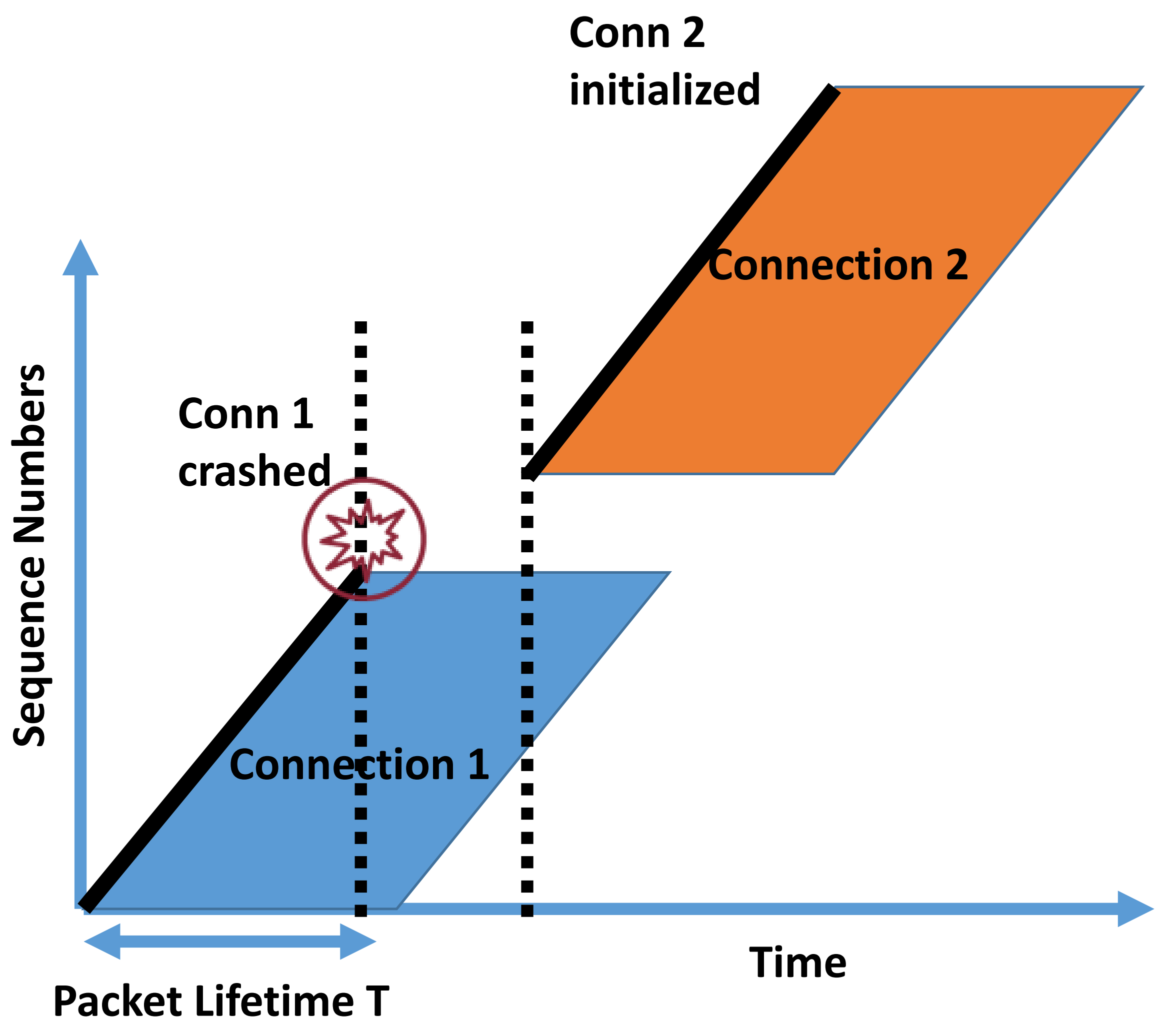


- A Delayed duplicate packet of connection 1 can create a confusion for connection 2

What We Ideally Want? Or …



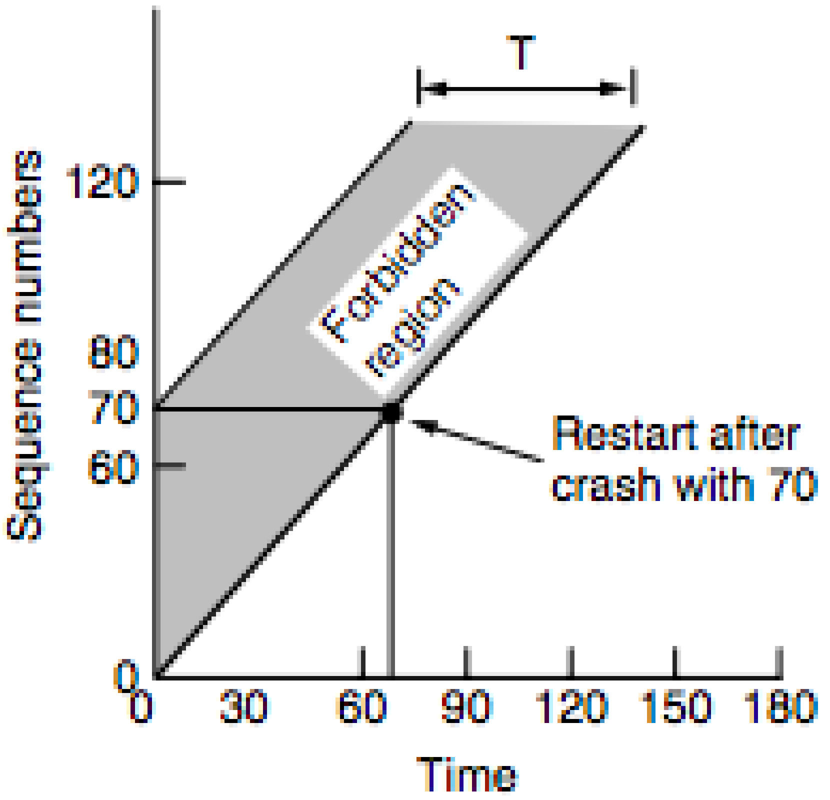
What We Ideally Want? Either …



Connection Establishment – Handling Delayed Duplicates

- 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.

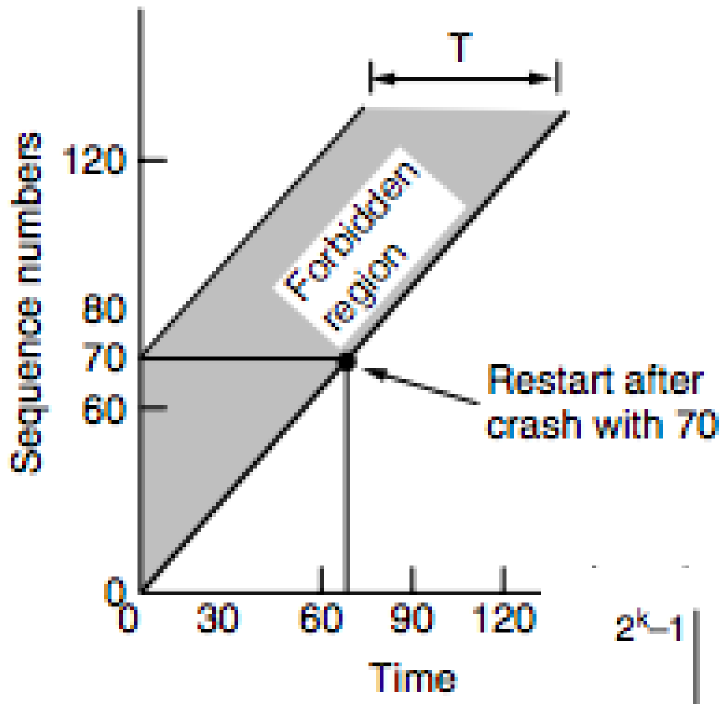
- For a crashed device, the transport entity remains idle for a duration T after recovery, to ensure that all packets from the previous connection are dead – **not a good solution**



- **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.**

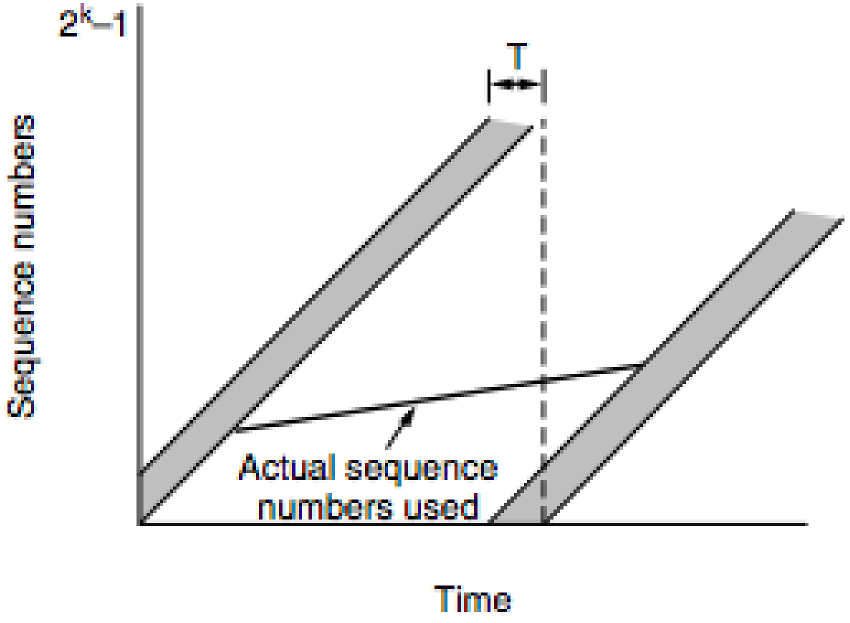
How do We Ensure that Packet Sequence Numbers are Out of the Forbidden Region

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



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

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

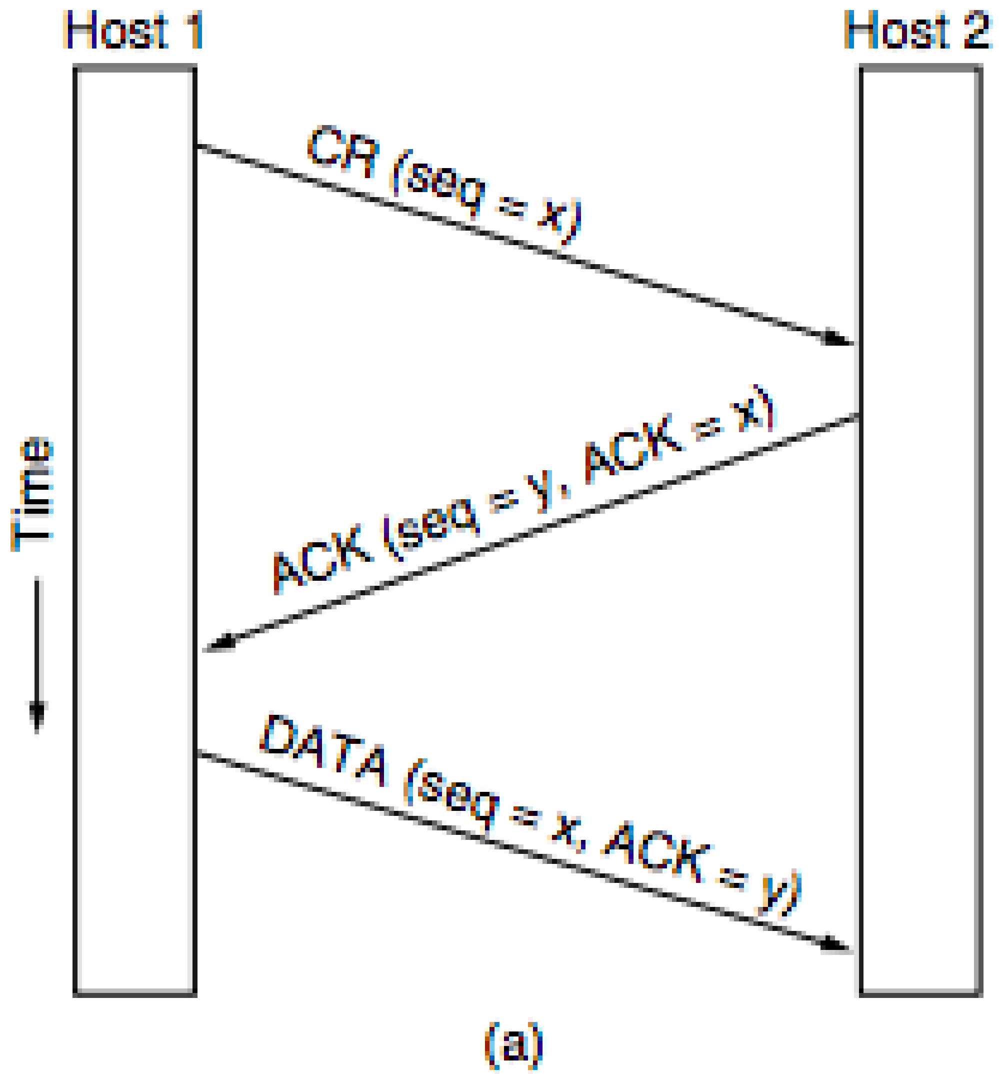


Adjusting the Sending Rate based on Sequence Numbers

- 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)

- **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

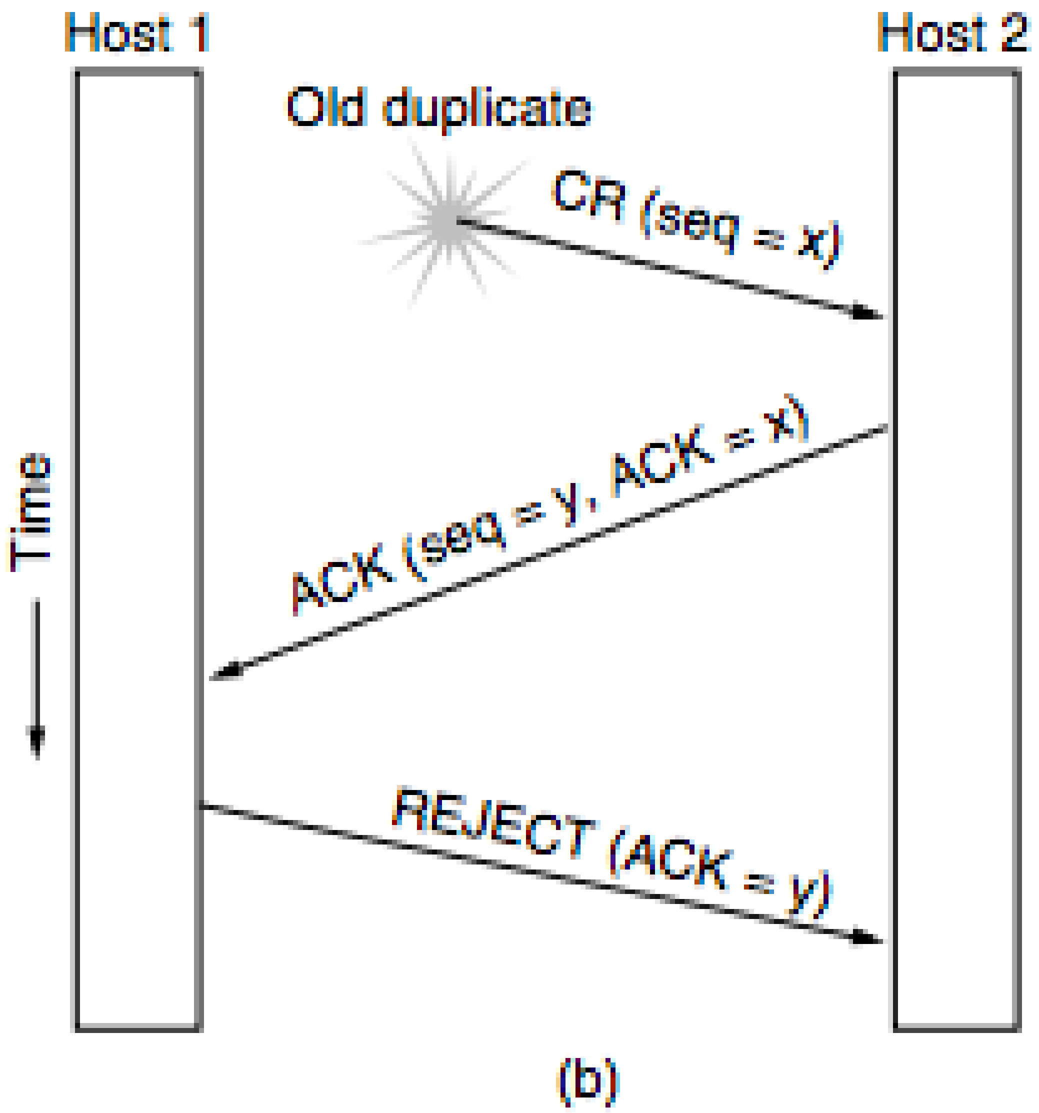


- By looking at the ACK, Host 1 ensures that Sequence number x does not belong to the forbidden region of any previously established connection

- 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

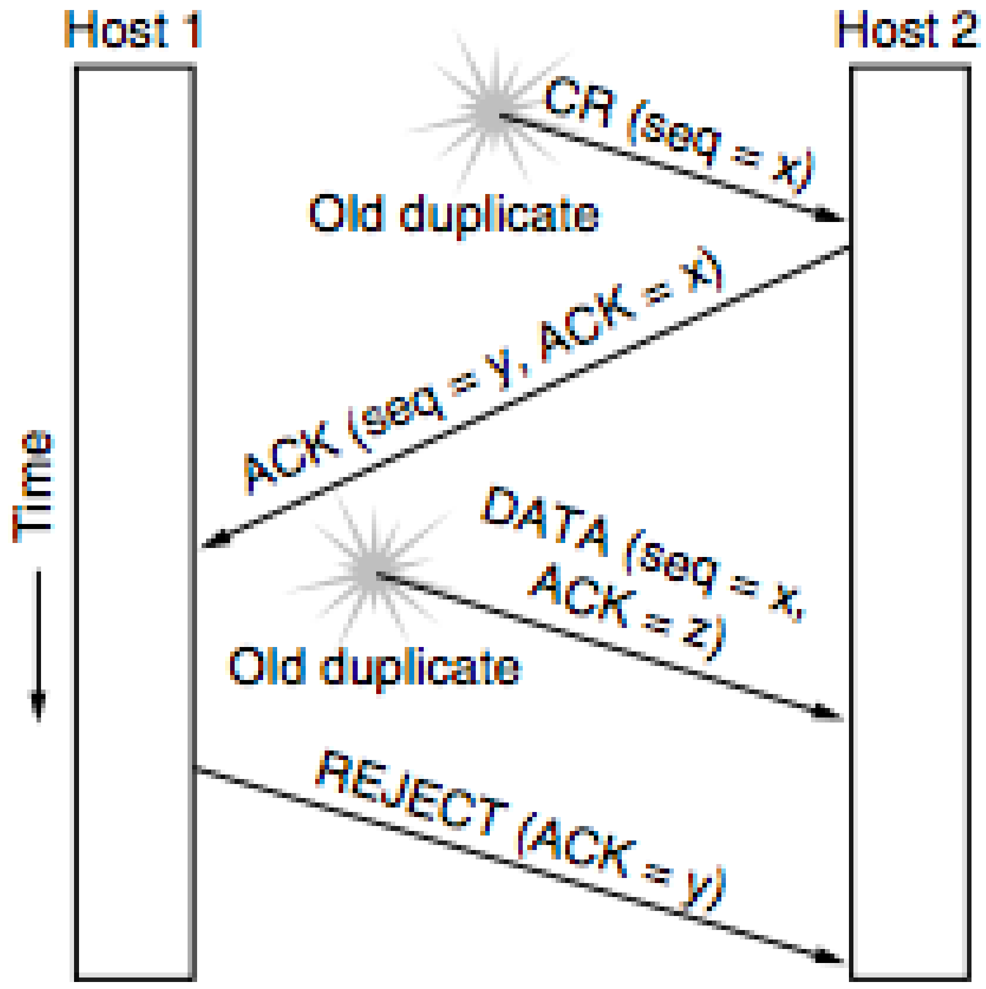
Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

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



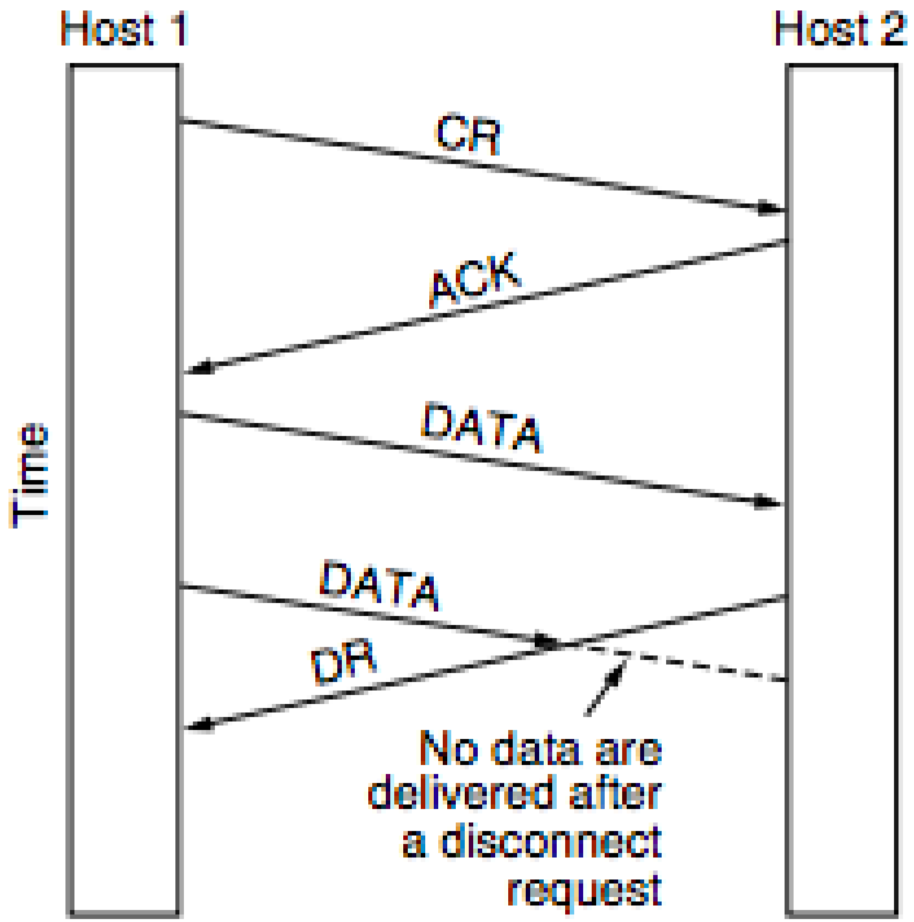
(c)

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

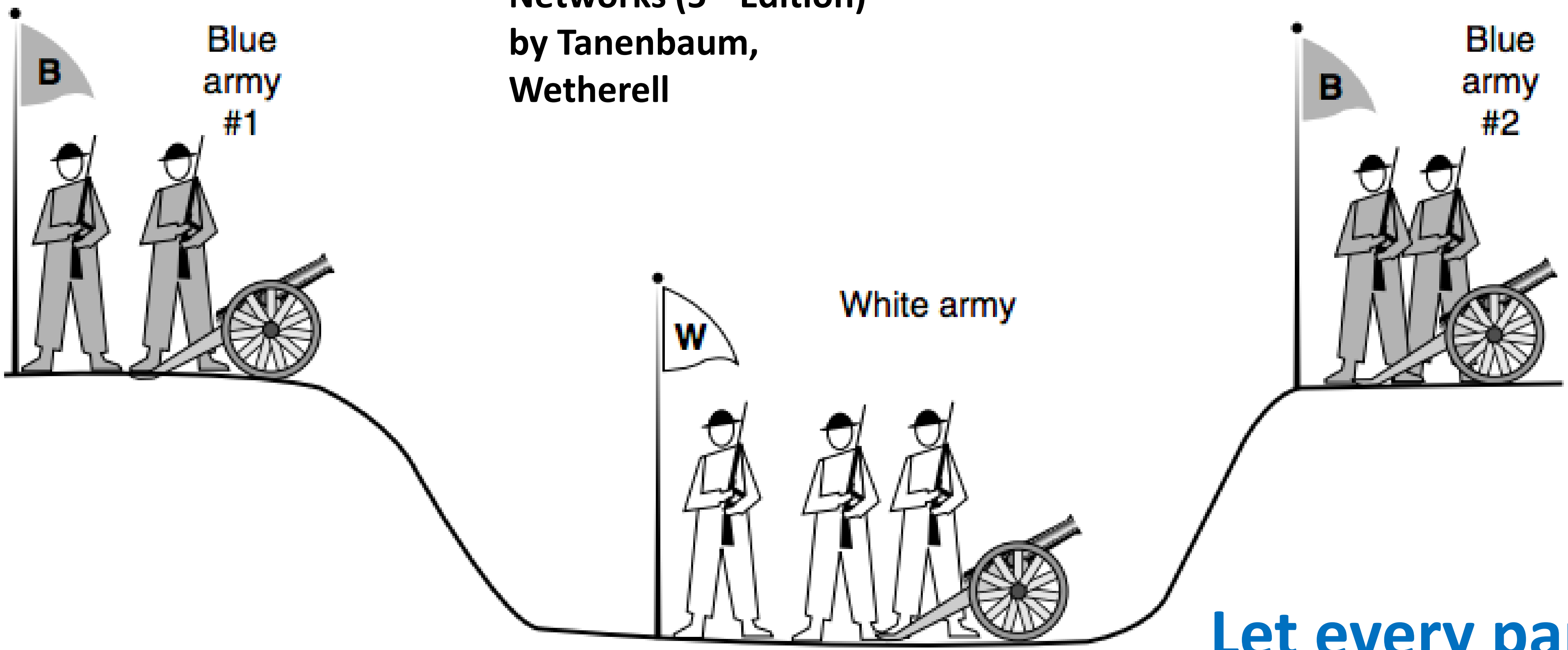
- Does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.

- What can be a protocol for this?
  - **Host 1: “I am done”**
  - **Host 2: ”I am done too”**

- **Does this protocol work good always?**

The Two Army Problem

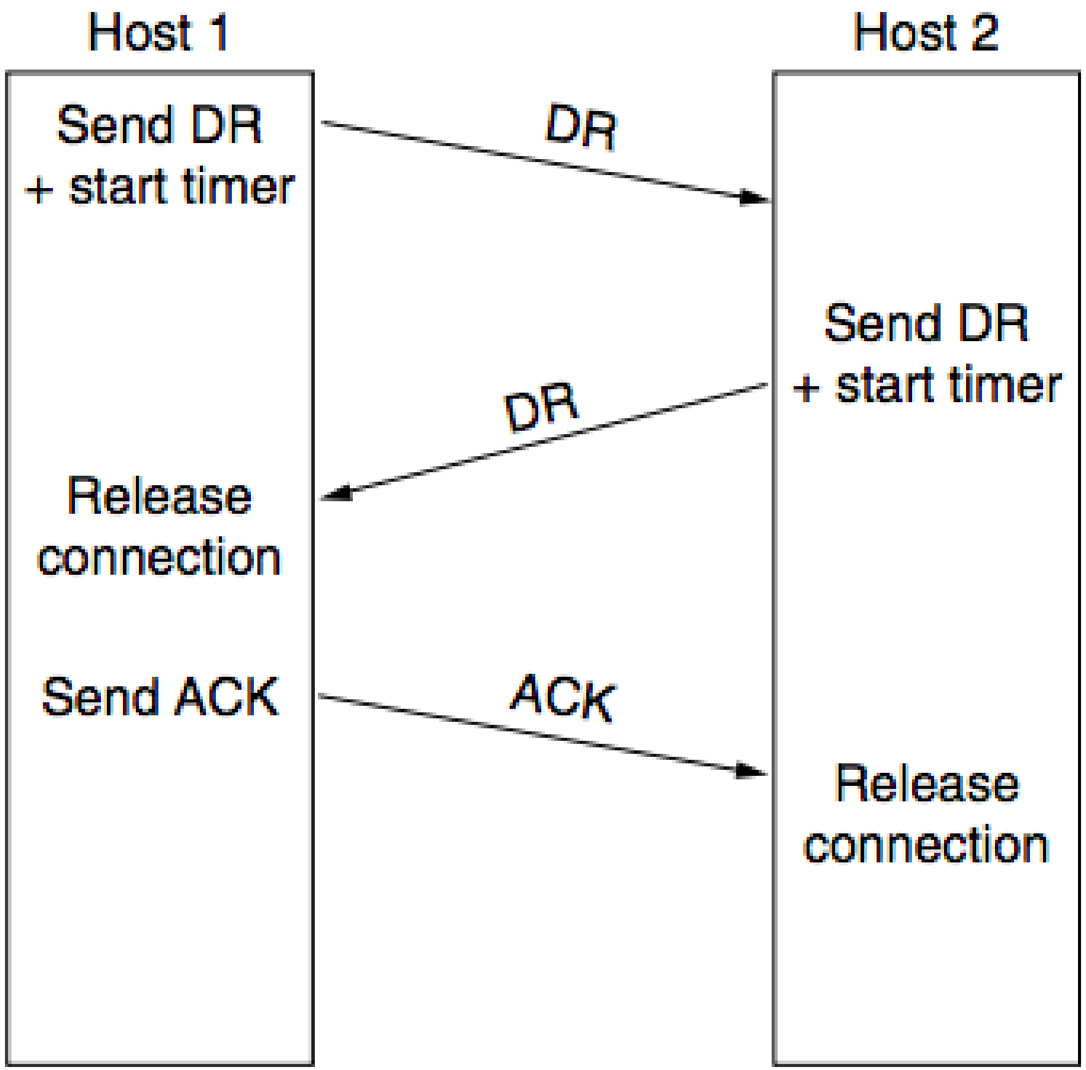
Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell



No protocol exists to solve this

Let every party take independent decisions

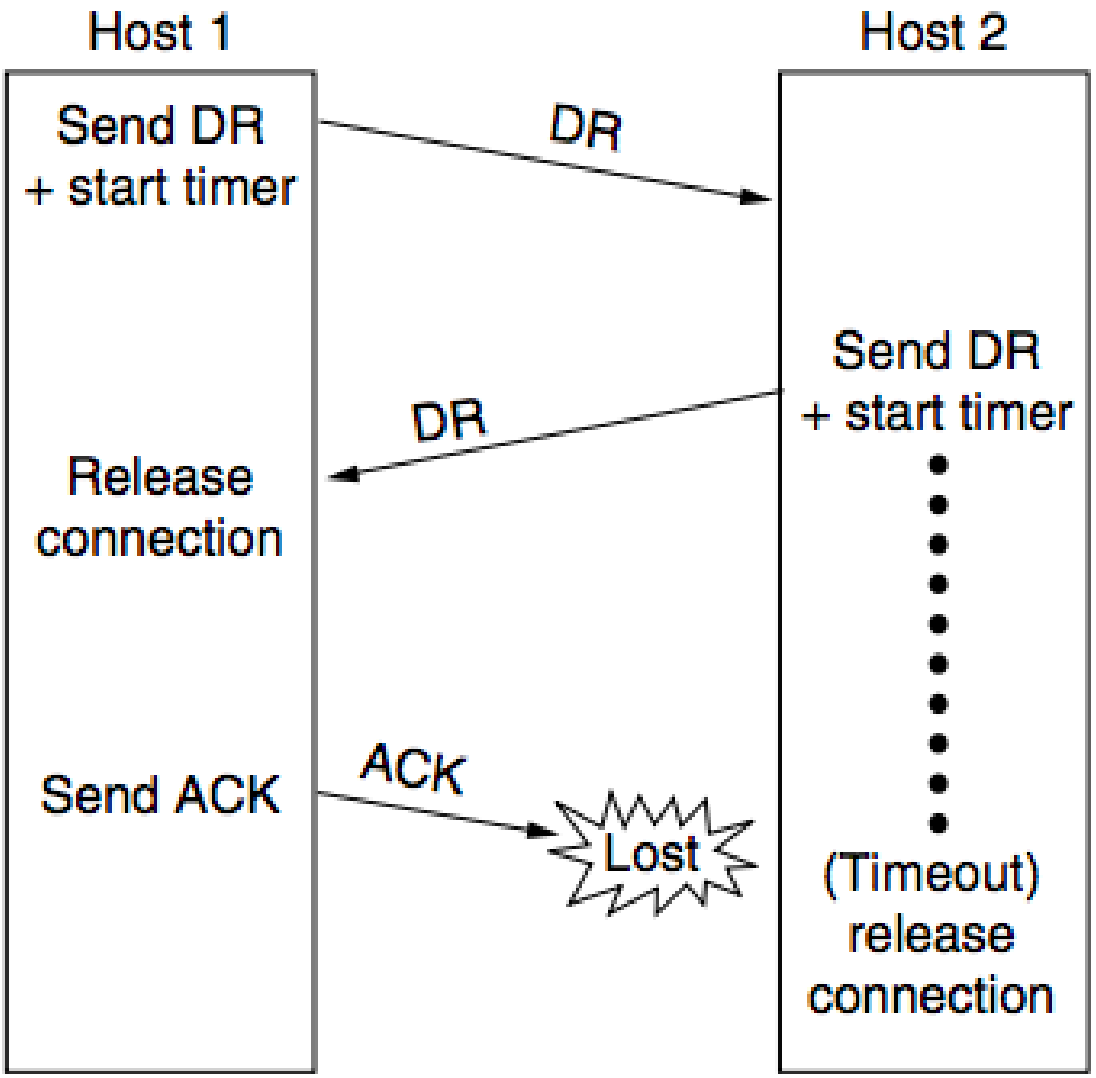
Connection Release



(a)

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

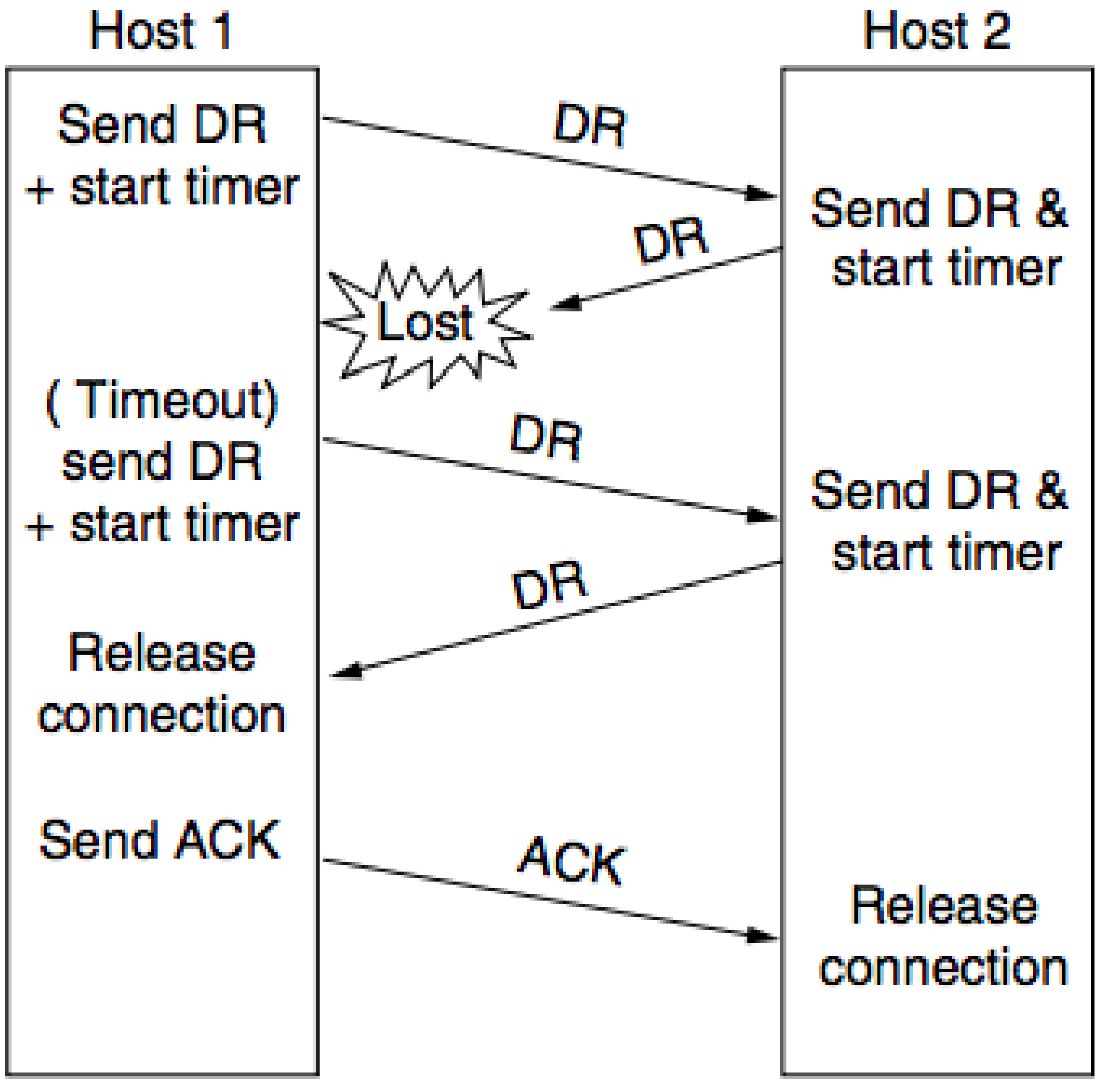
Connection Release – Final ACK Lost



(b)

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

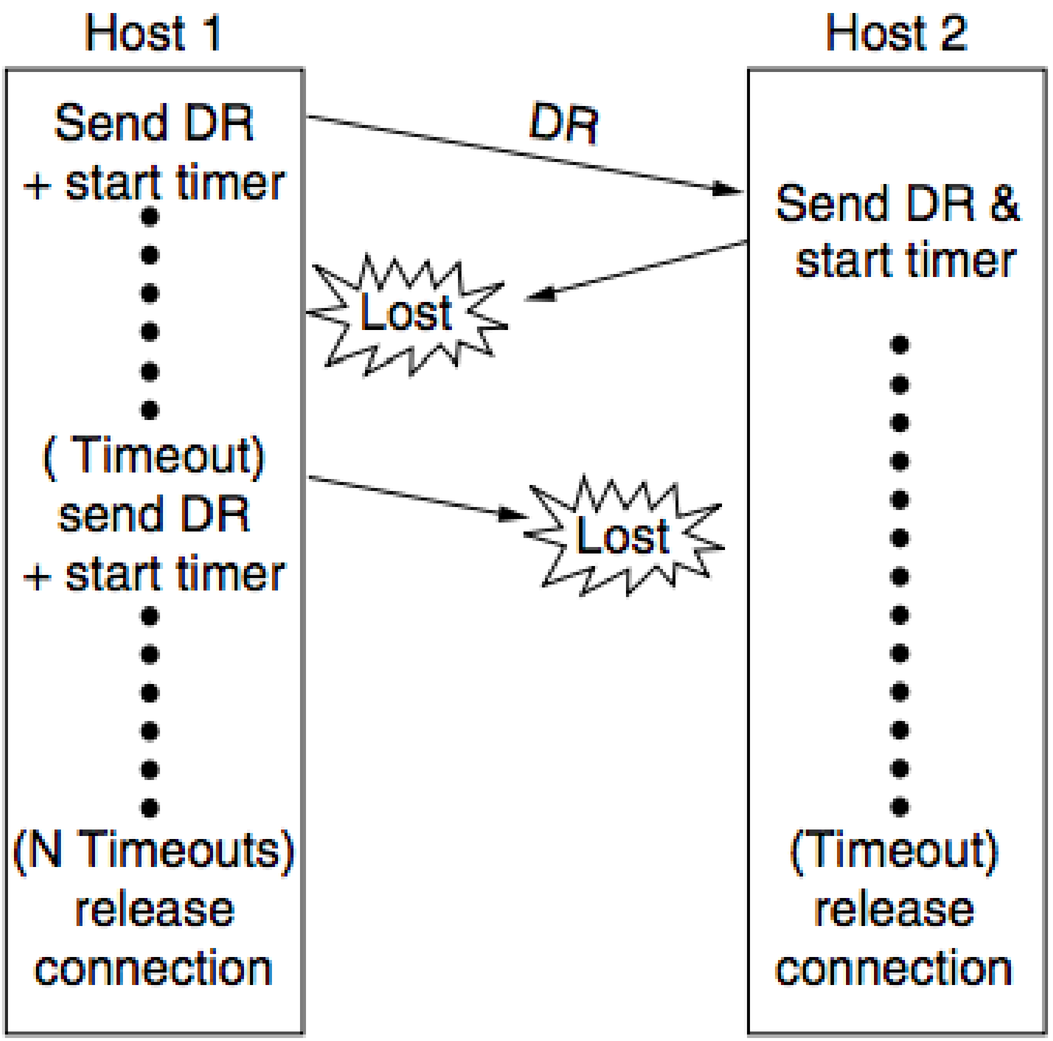
Connection Release – Response Lost



(c)

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

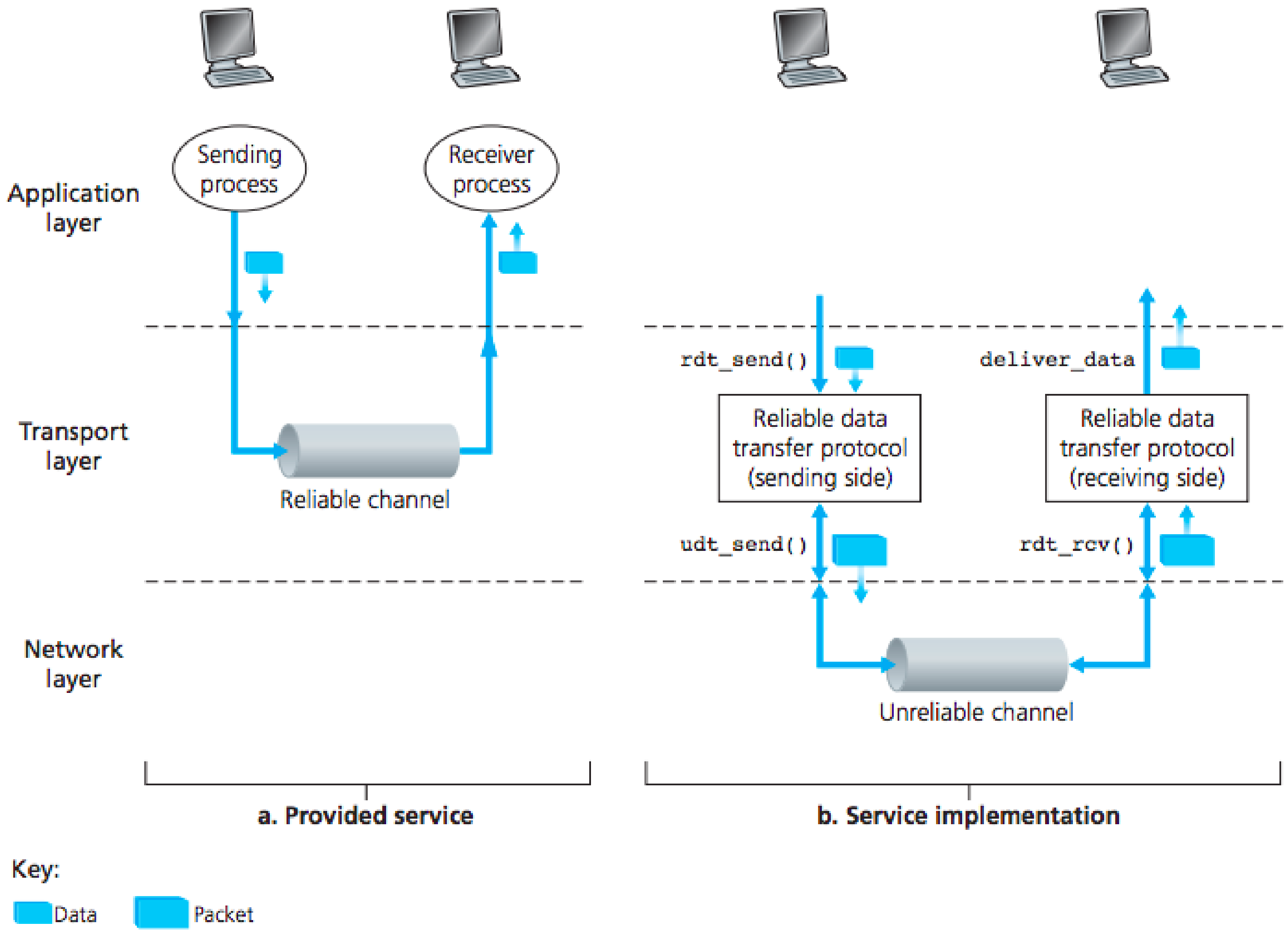
Connection Release – Response Lost and Subsequent DRs Lost



(d)

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

Ensure Reliability at the Transport Layer



Source: Computer Networks, Kurose, Ross

Error Control and Flow Control

- **These features are used in both Data Link Layer and Transport Layer – Why?**

- **Flow control and error control at the transport layer is essential**

- **Flow control and error control at the data link layer improves performance**

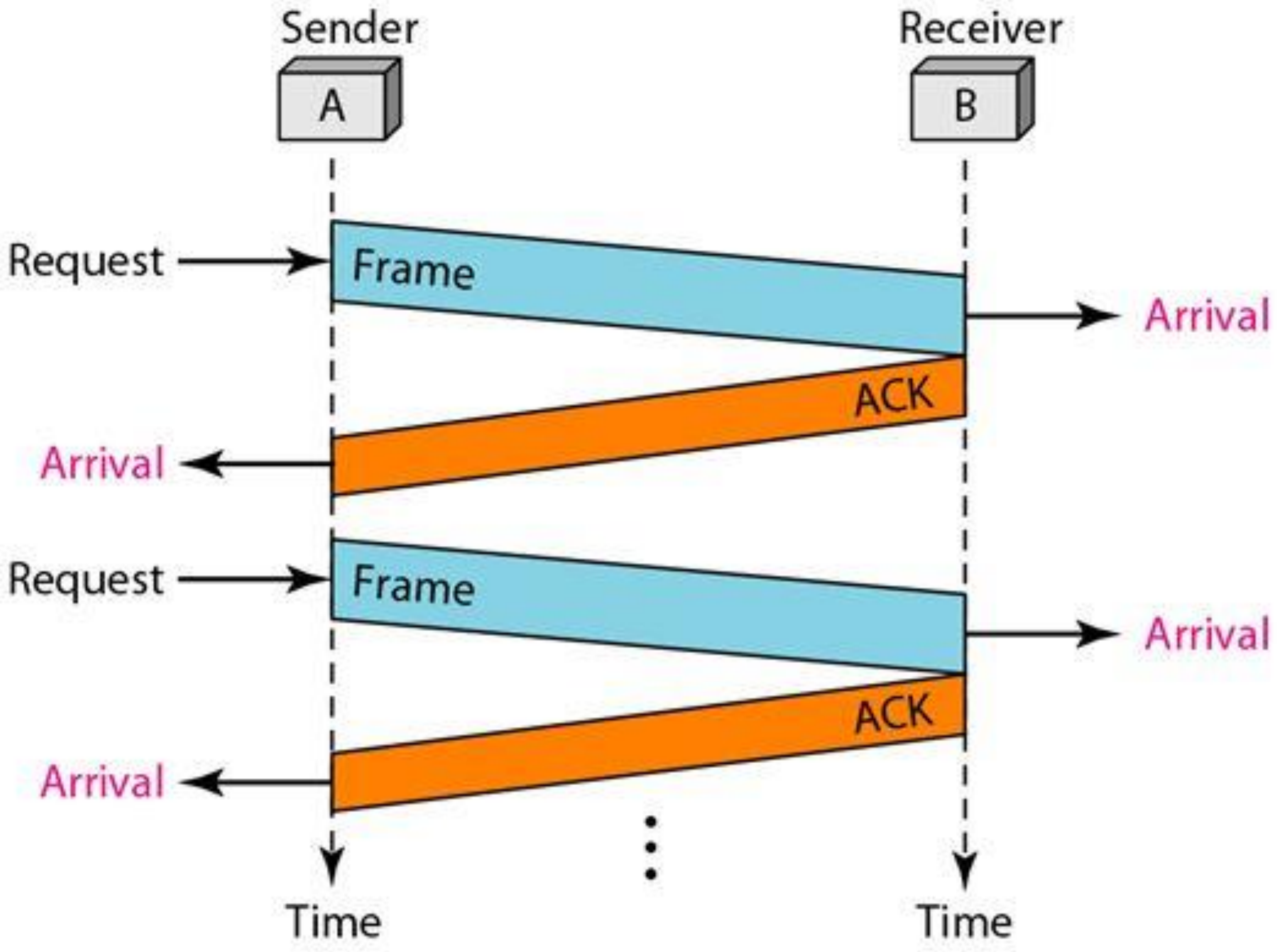
END-TO-END ARGUMENTS IN SYSTEM DESIGN

J.H. Saltzer, D.P. Reed and D.D. Clark*

M.I.T. Laboratory for Computer Science

Flow Control Algorithms

- **Stop and Wait Flow Control (Error Free Channel):**



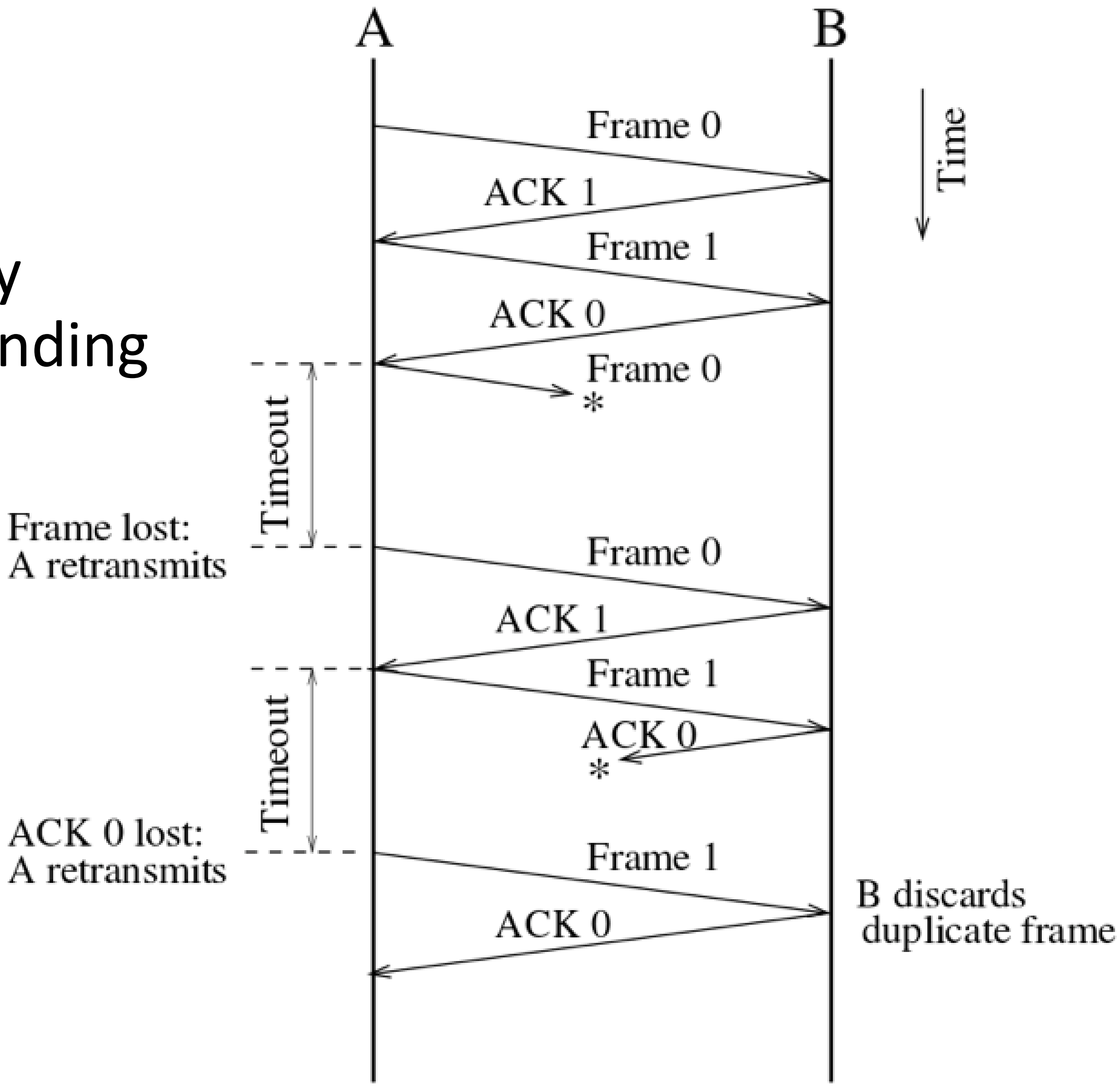
Flow Control Algorithms

- **Stop and Wait (Noisy Channel):**

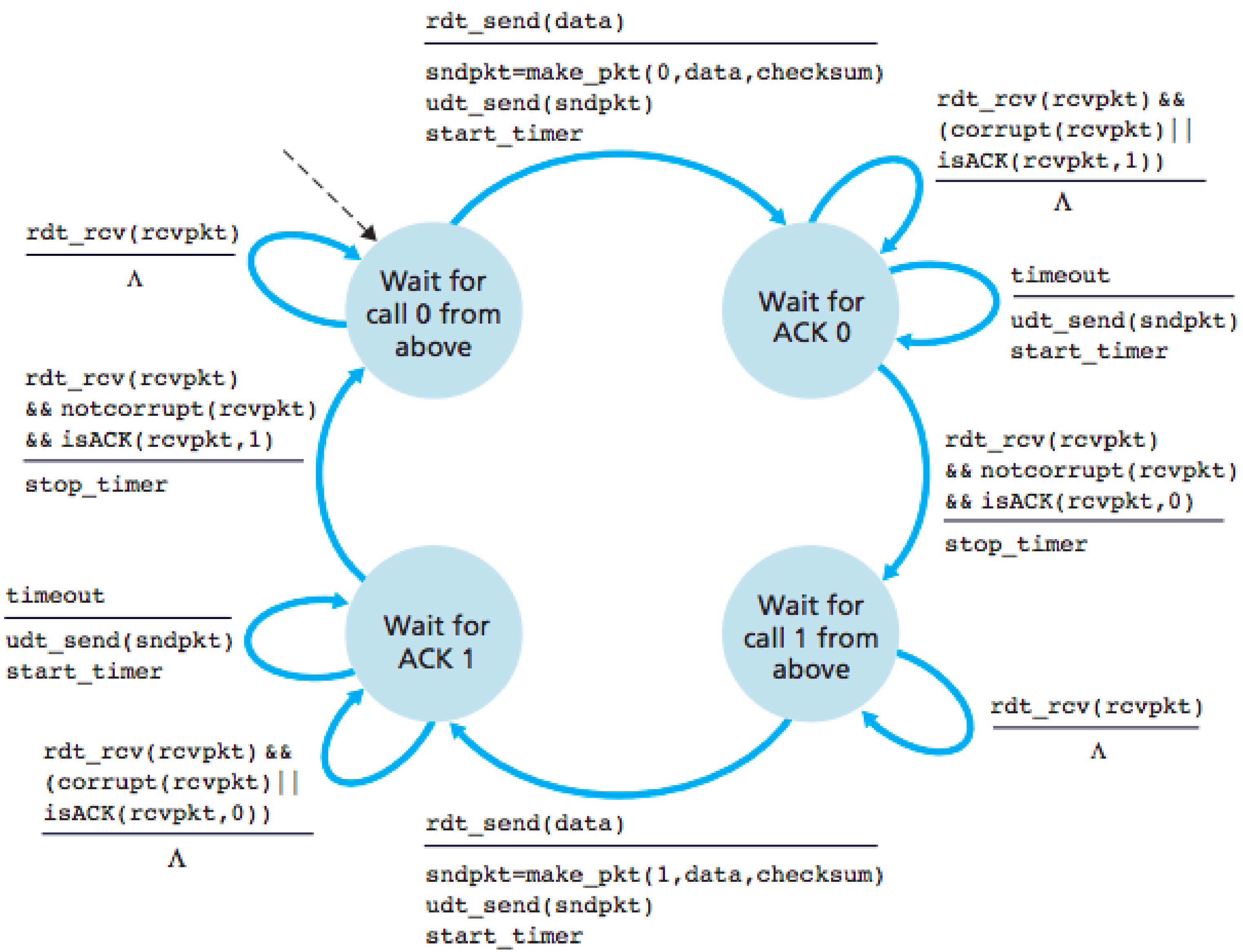
- Use sequence numbers to individually identify each frame and the corresponding acknowledgement

- **What can be a maximum size of the sequence number in Stop and Wait?**

- **Automatic Repeat Request (ARQ)**



Stop and Wait ARQ – Sender Implementation

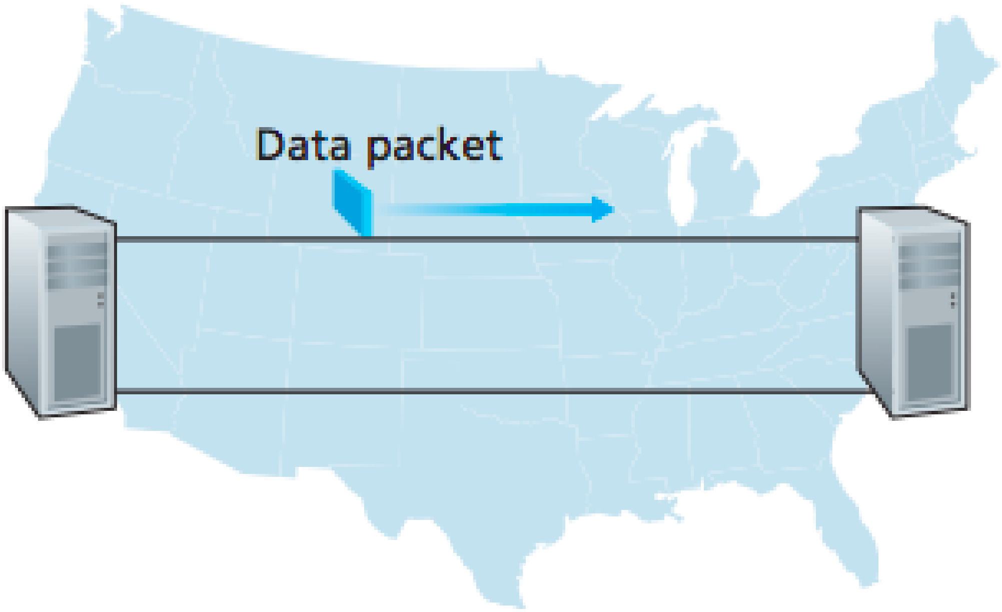


Source: Computer Networks, Kurose, Ross

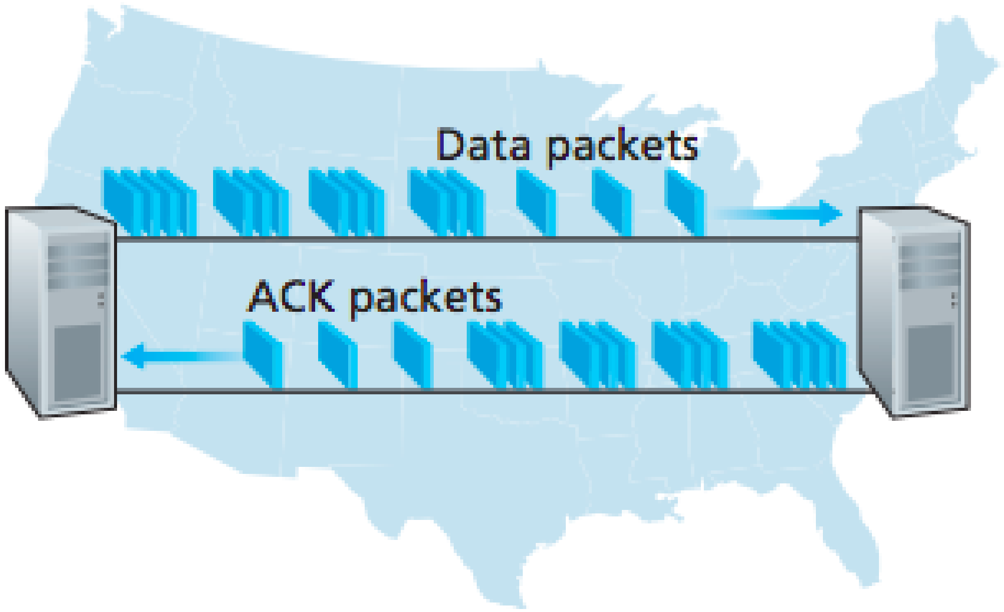
Problem with Stop and Wait

- 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

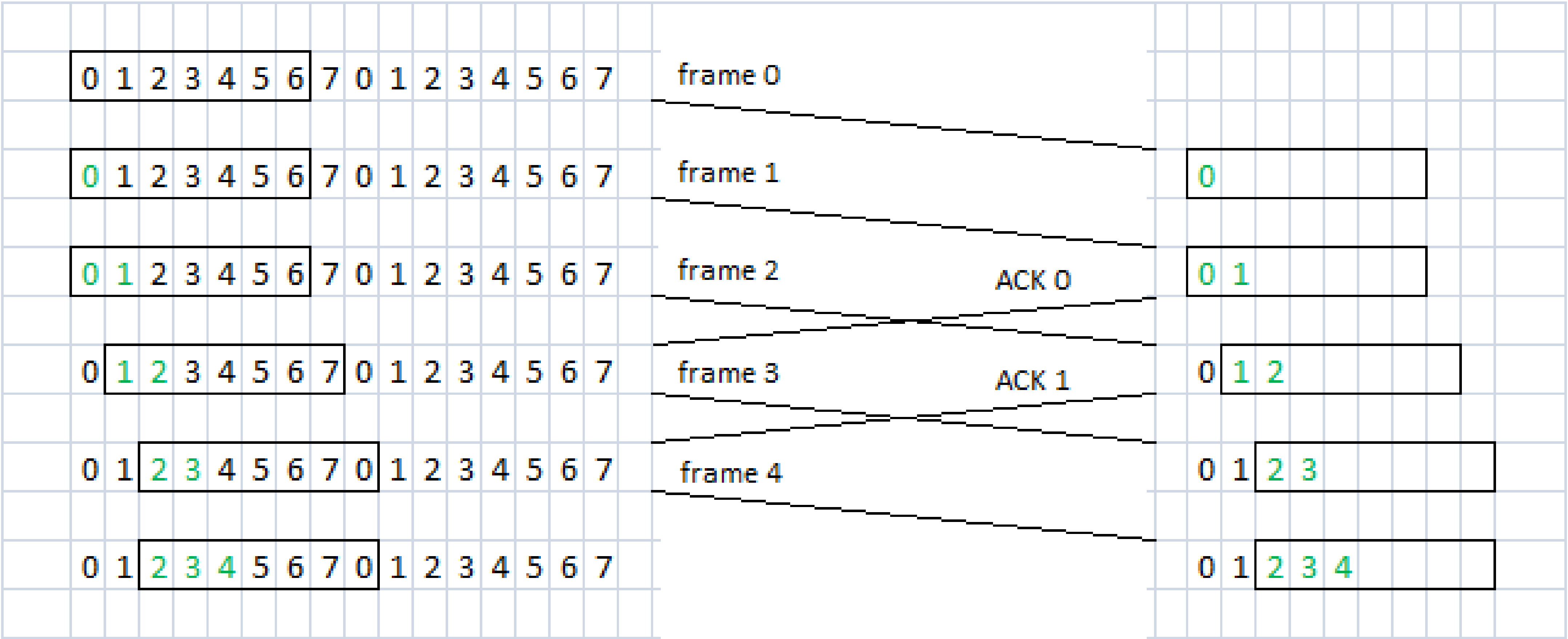
Sliding Window Protocols

- 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

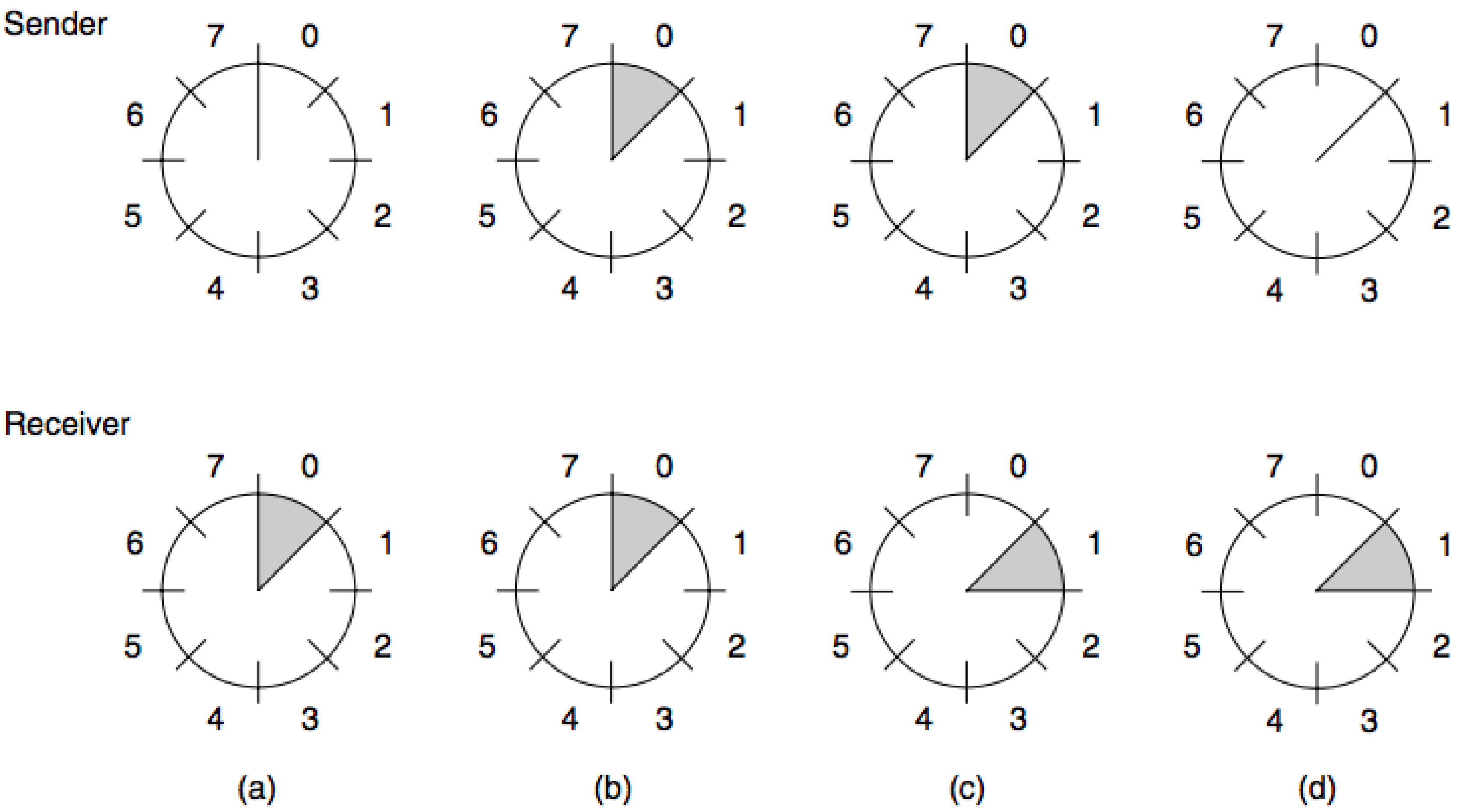


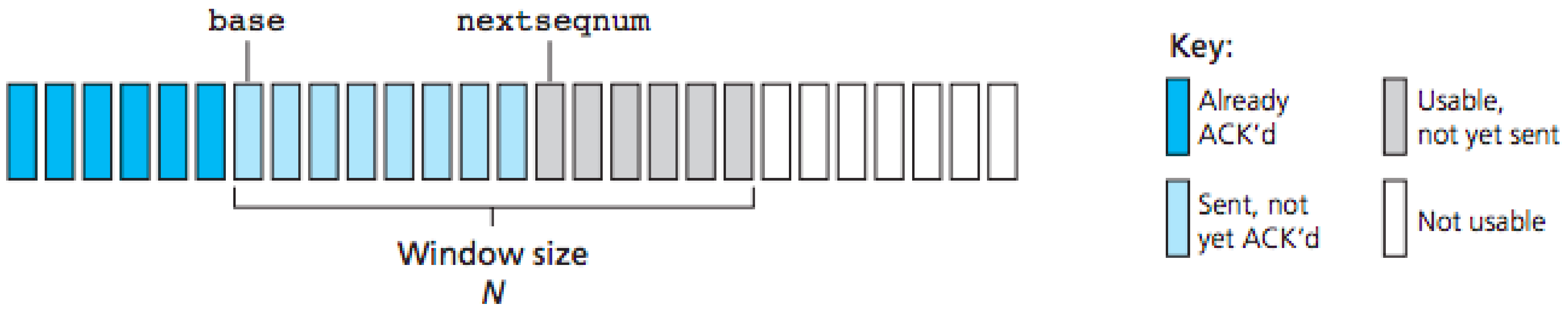
Figure 3-15. A sliding window of size 1, with a 3-bit sequence number. (a) Initially. (b) After the first frame has been sent. (c) After the first frame has been received. (d) After the first acknowledgement has been received.

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

Sliding Window Protocols in Noisy Channels

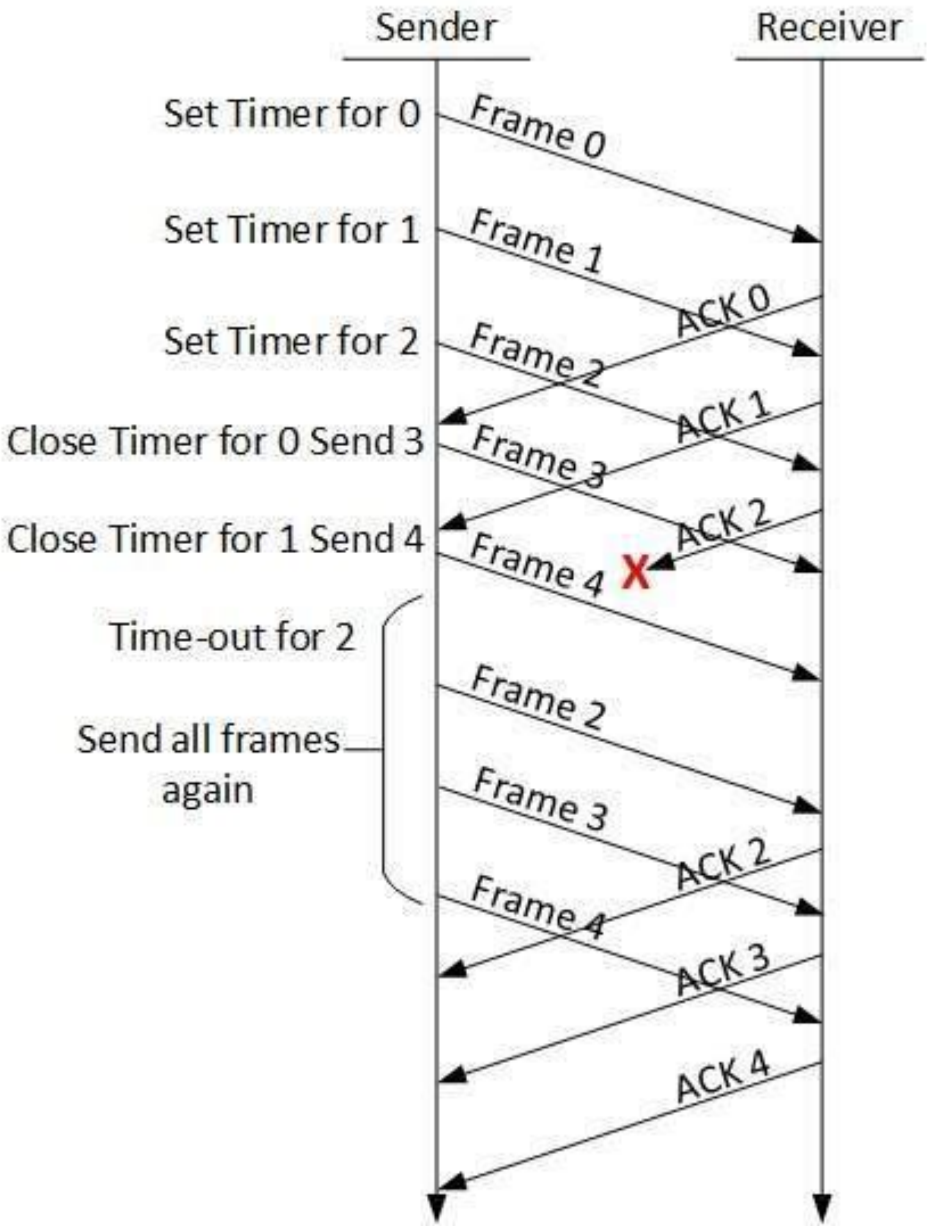
- **A timeout occurs if a segment (or the acknowledgment) gets lost**
- **How does the flow and error control protocol handle a timeout?**
- **Go Back N ARQ:** If segment N is lost, all the segments from segment 0 (start of the sliding window) to segment N are retransmitted
- **Selective Repeat (SR) 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 – Sender Window Control



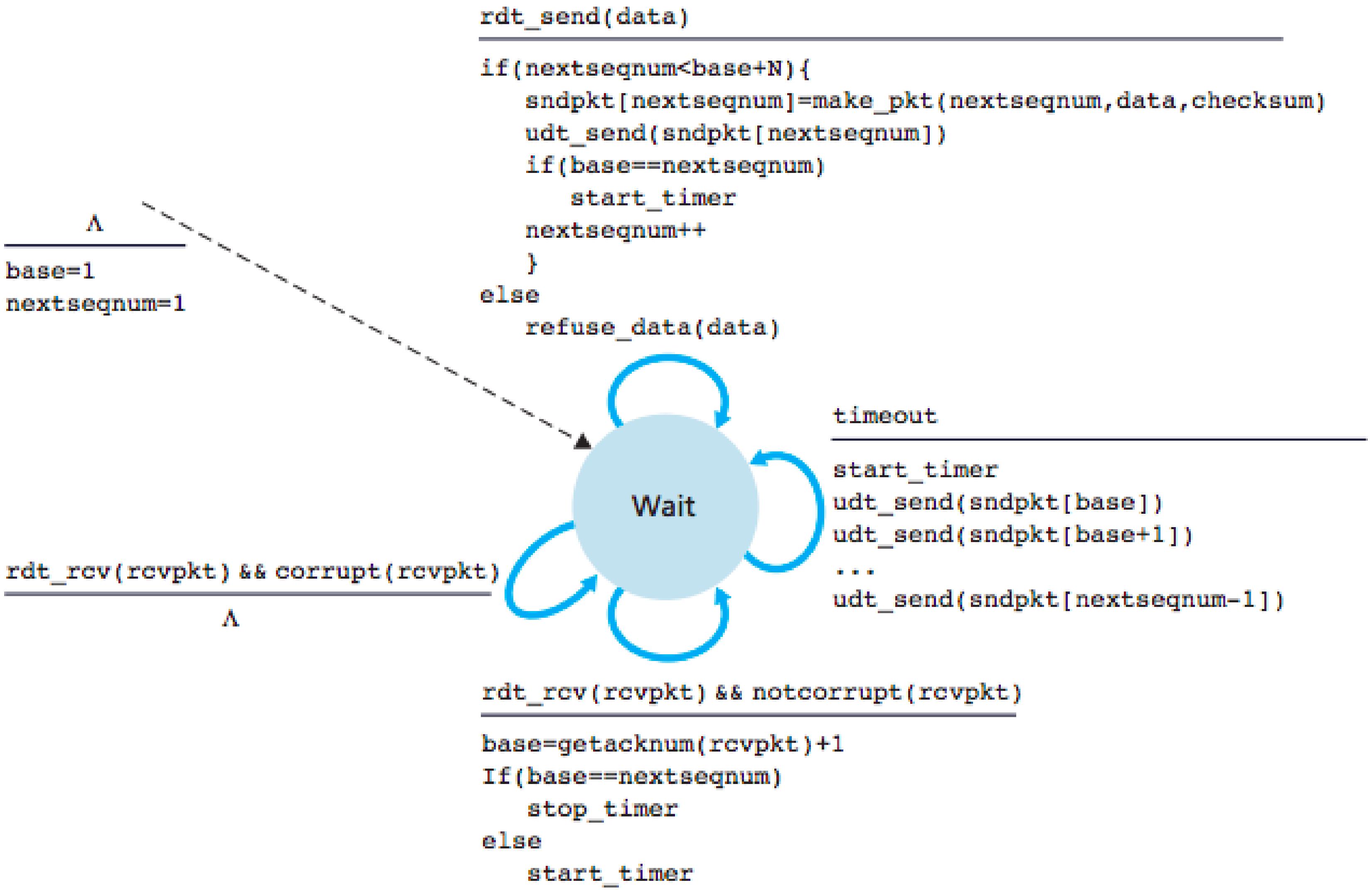
Source: Computer Networks, Kurose, Ross

Go Back N ARQ



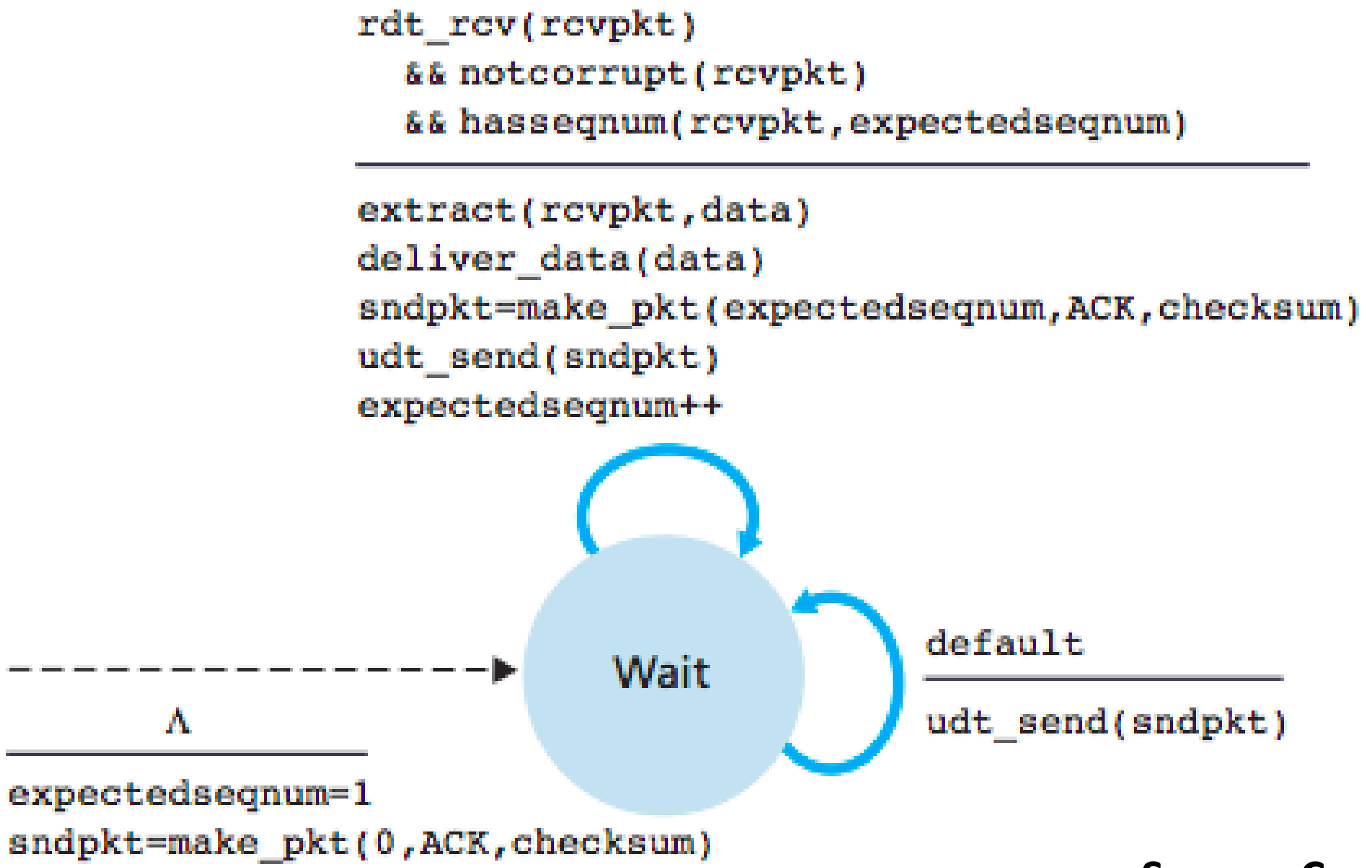
Source
https://www.tutorialspoint.com/data_communication_computer_network/data_link_control_and_protocols.htm

Go Back N ARQ – Sender



Source: Computer Networks, Kurose, Ross

Go Back N ARQ – Receiver



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

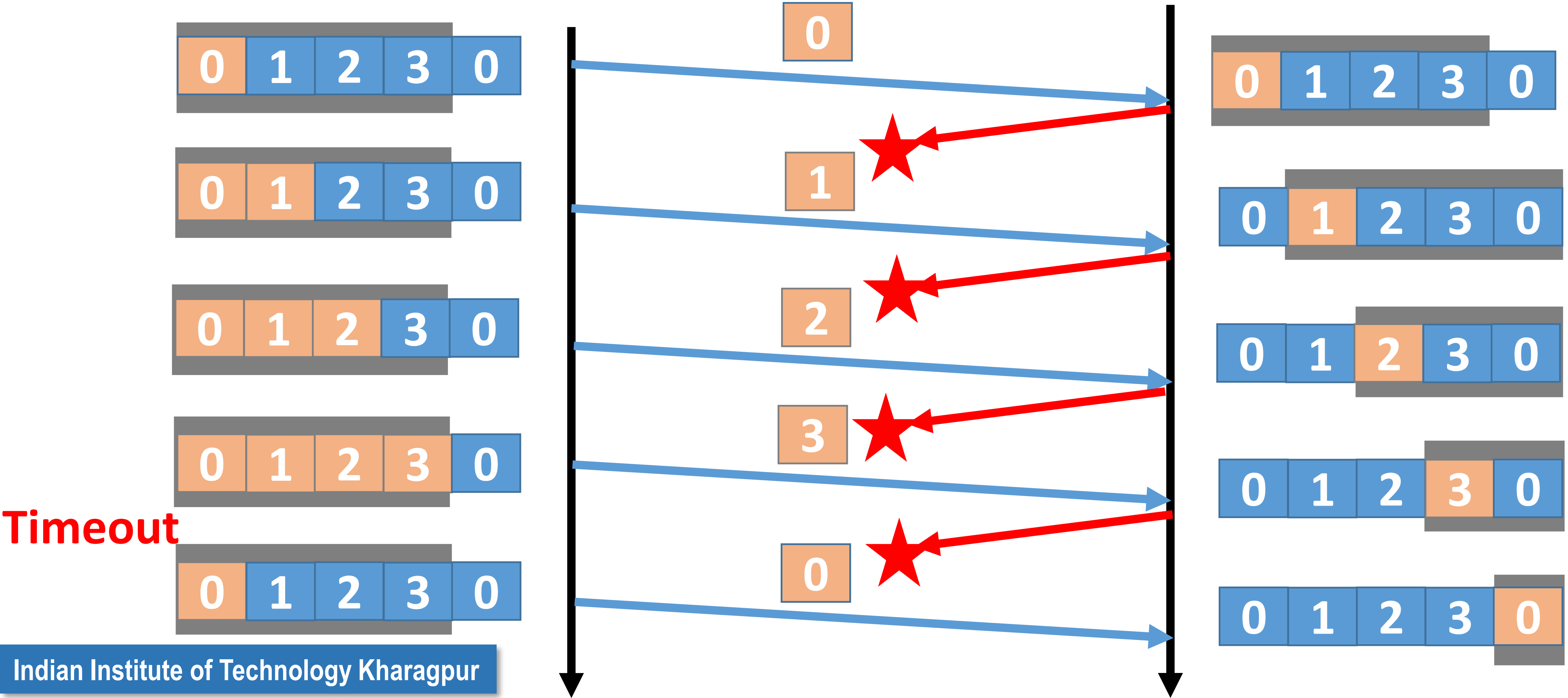
- **Maximum Sequence Number (MAX_SEQ): MAX_SEQ+1** distinct sequence numbers are there
  - **0,1,…,MAX_SEQ**

- **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)**

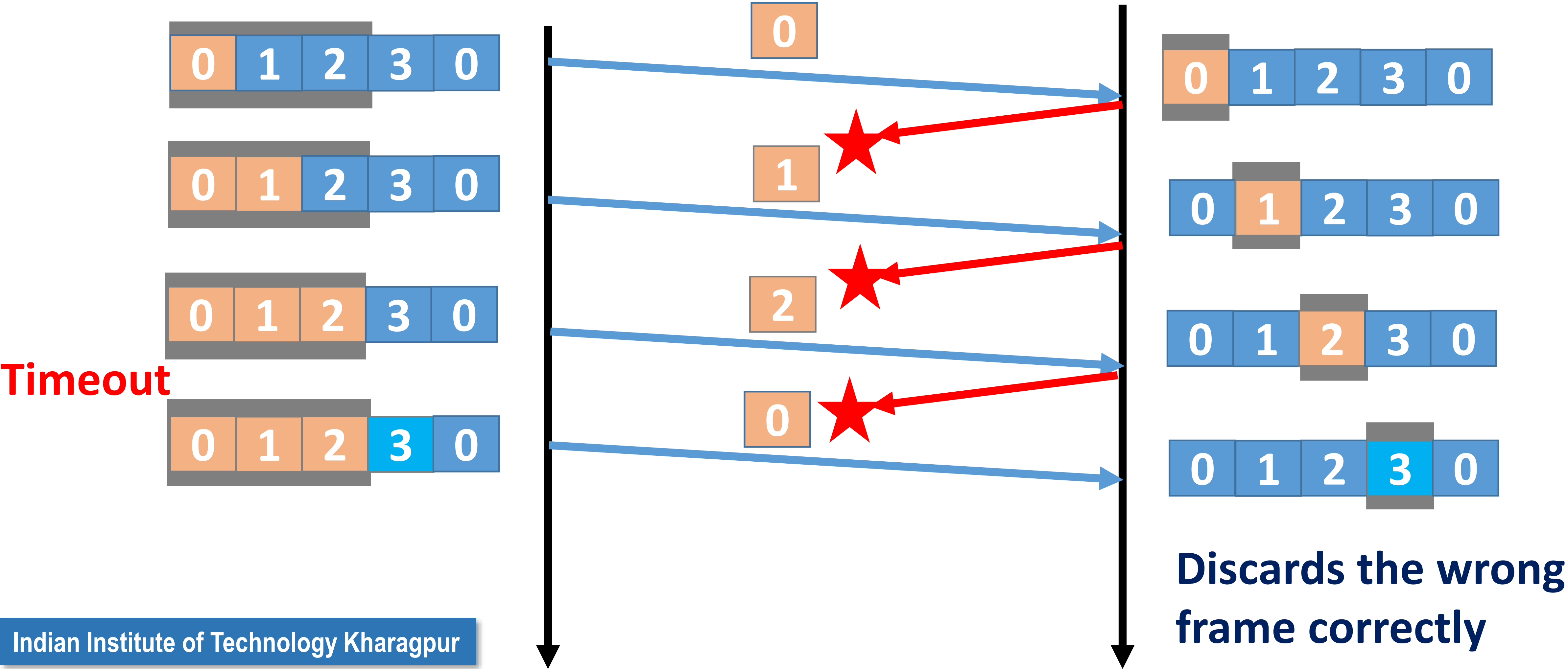
Go Back N ARQ – A Bound on Window Size

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

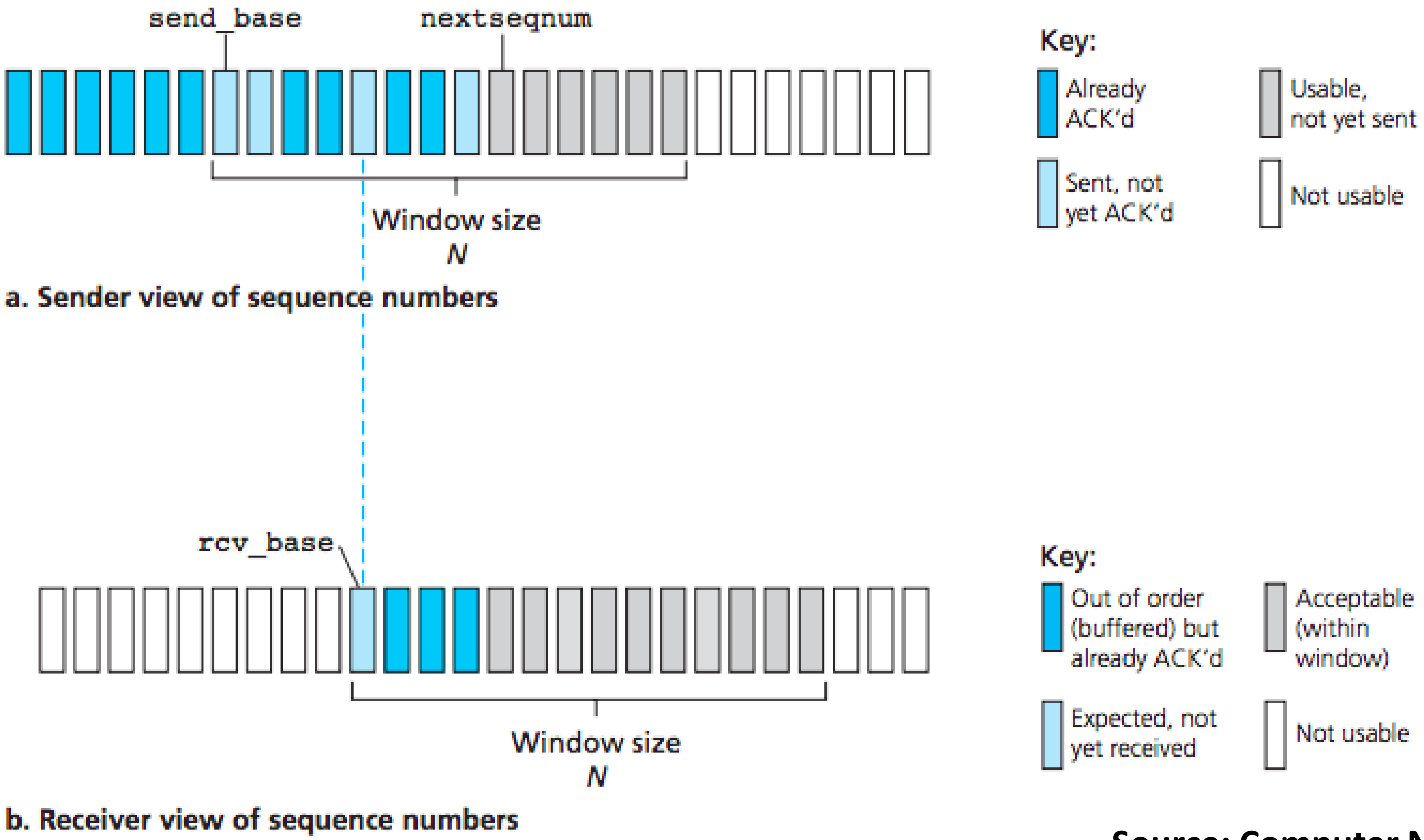


Go Back N ARQ – A Bound on Window Size

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

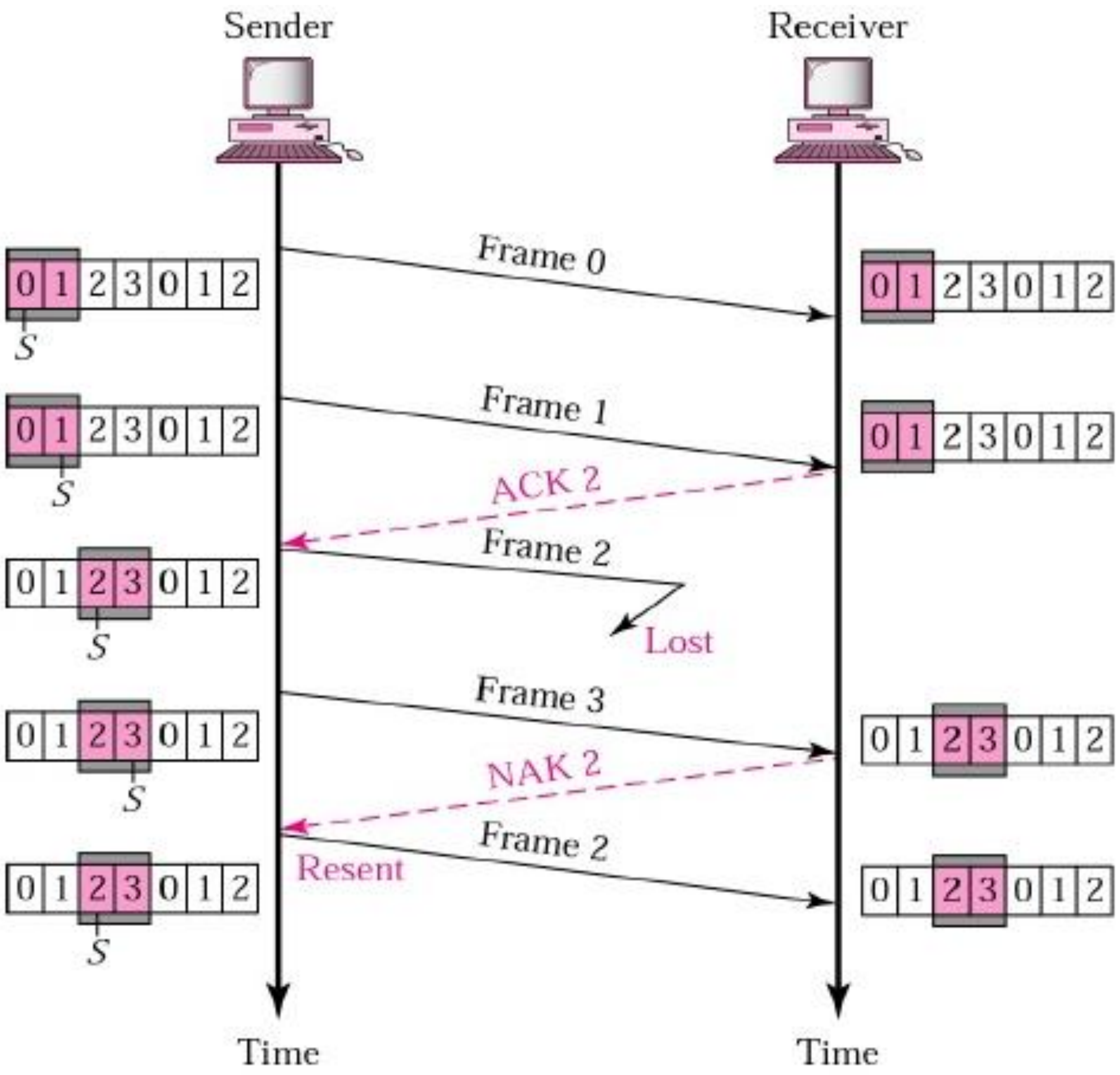


Selective Repeat (SR) – Window Control



Source: Computer Networks, Kurose, Ross

Selective Repeat ARQ



Selective Repeat – A Bound on Window Size

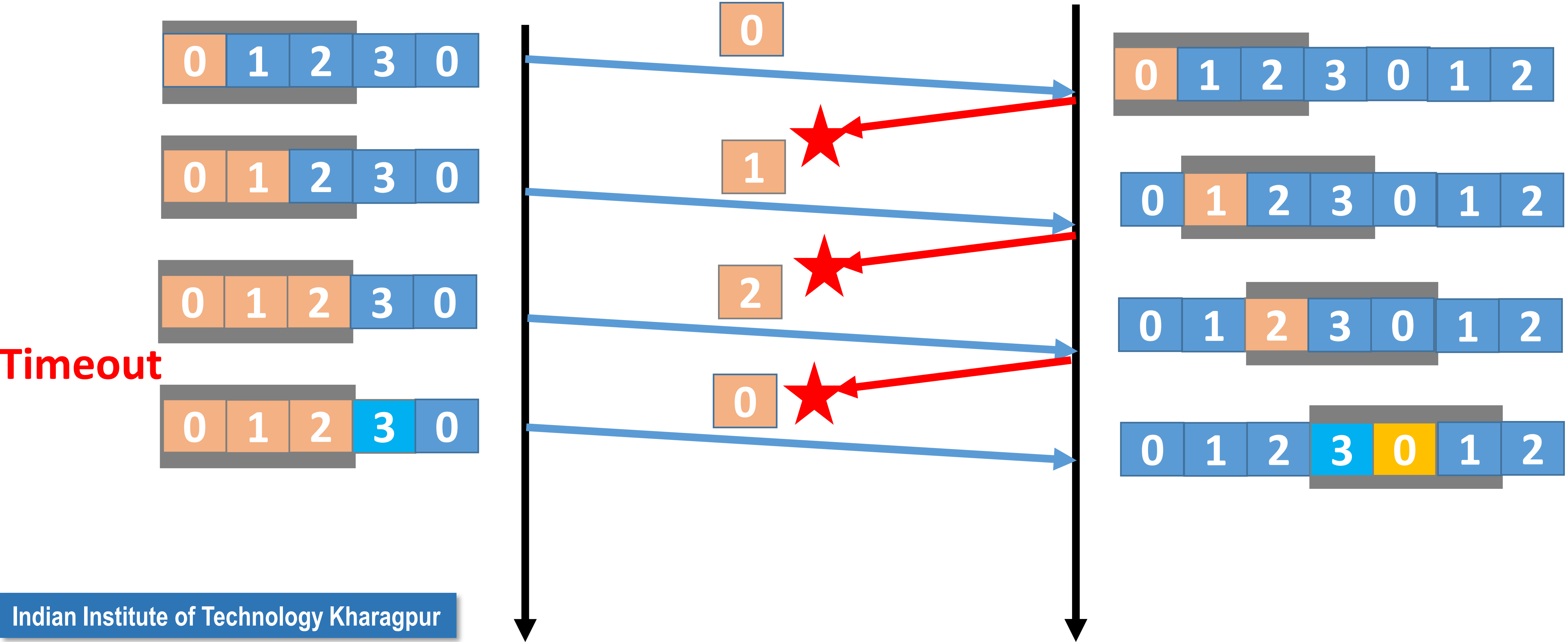
- **Maximum Sequence Number (MAX_SEQ): MAX_SEQ+1** distinct sequence numbers are there
  - **0,1,…,MAX_SEQ**

- **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

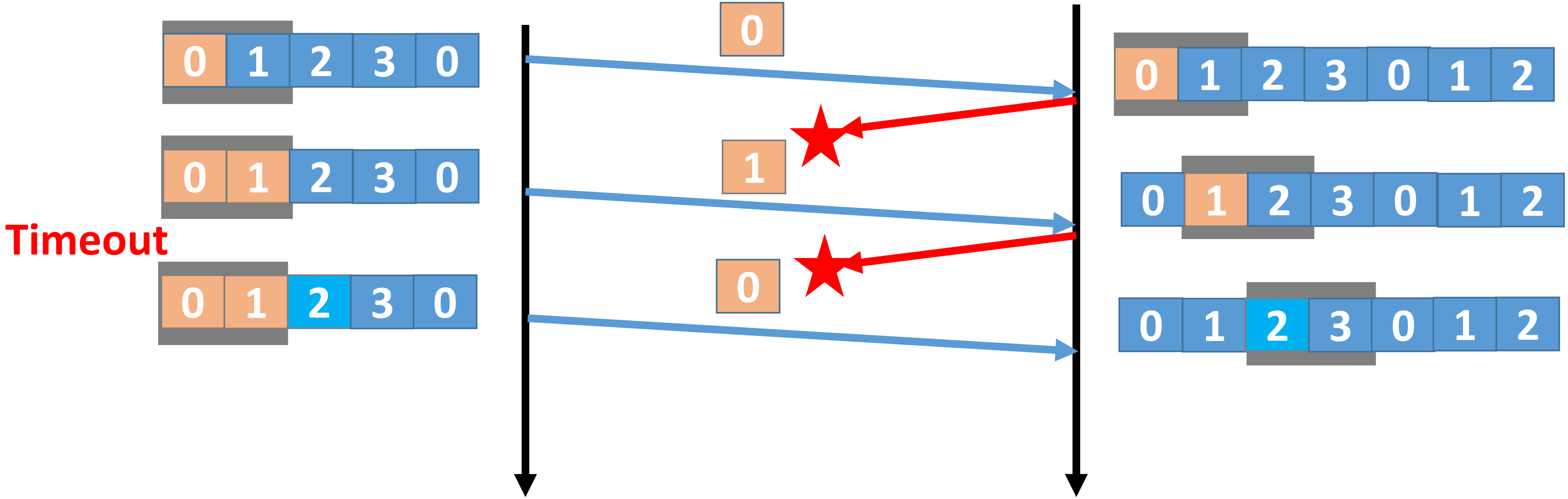
Selective Repeat – A Bound on Window Size

- **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

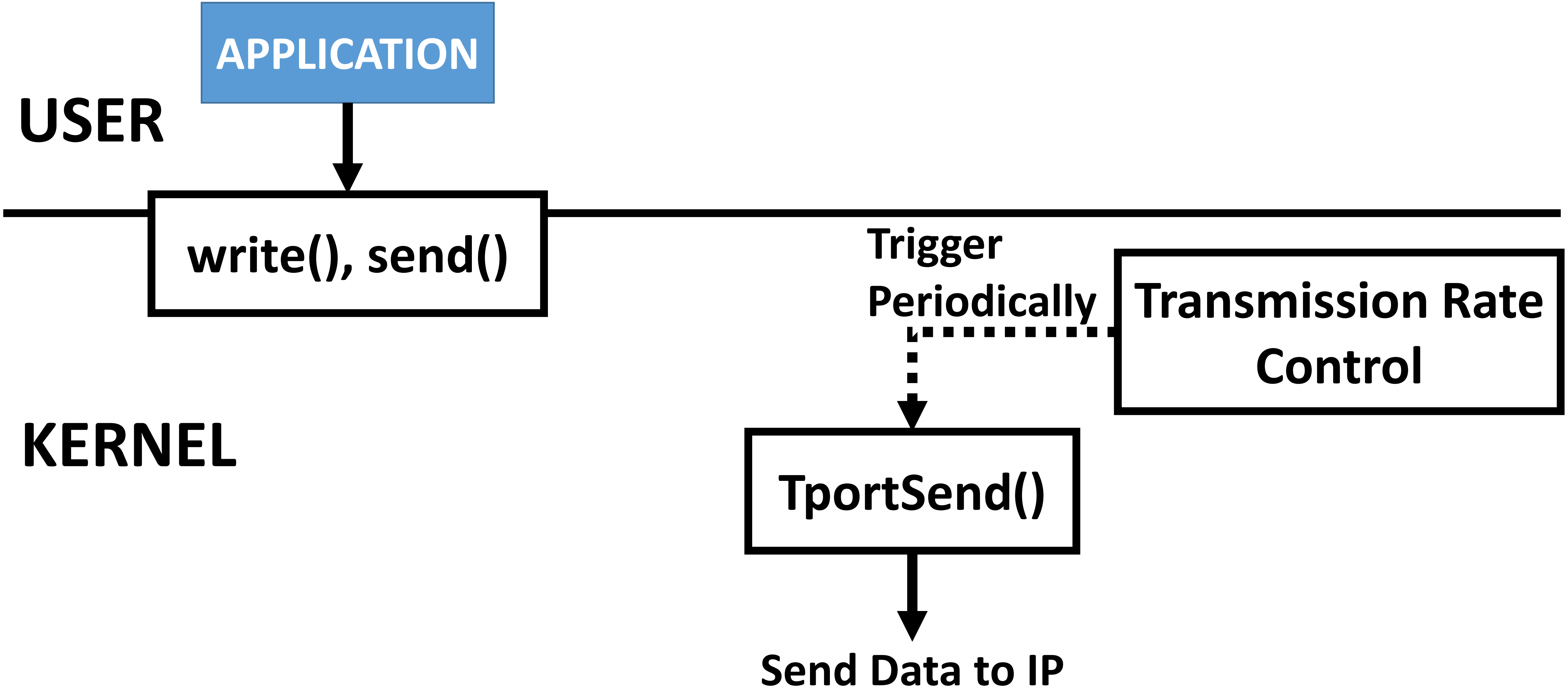
- So, **w = 2BD + 1** gives the maximum link utilization – **this is an important concept to decide the window size for a window based flow control mechanism**

Implication of BDP on Protocol Design Choice

- Consider the link bandwidth = 1Mbps, Delay = 1ms
- Consider a network, where segment size is 1 KB (1024 bytes)
- Which protocol is better for flow control?
  - (a) stop and wait,
  - (b) Go back N,
  - (c) Selective Repeat

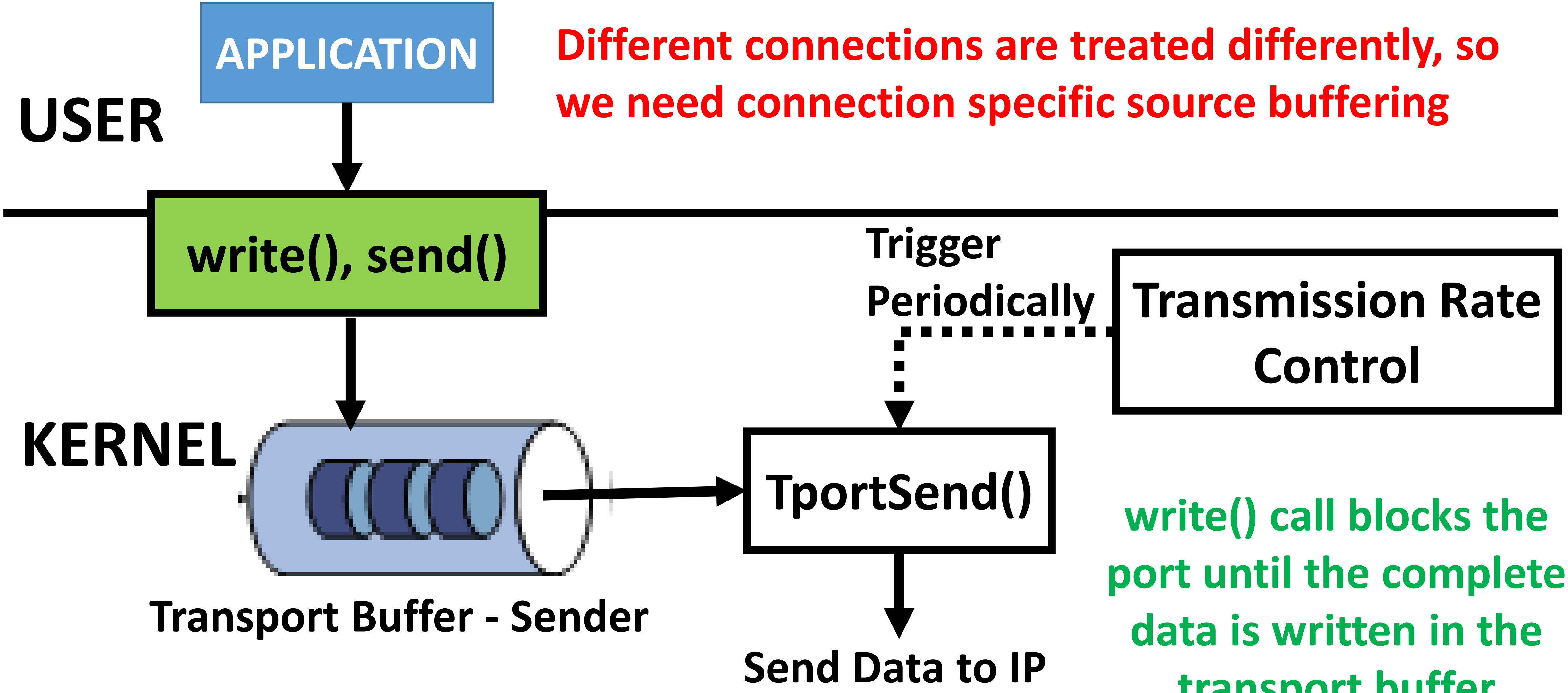
- **BDP = 1 Mbps x 1ms = 1 Kb (1024 bits)**
- **The segment size is eight times larger than the BDP -> the link can not hold an entire segment completely**
- **Sliding window protocols do not improve performance**
- **Stop and Wait is better – less complexity**

Application Transport Interfacing – Sender Side



Function names are hypothetical

Application Transport Interfacing – Sender Side



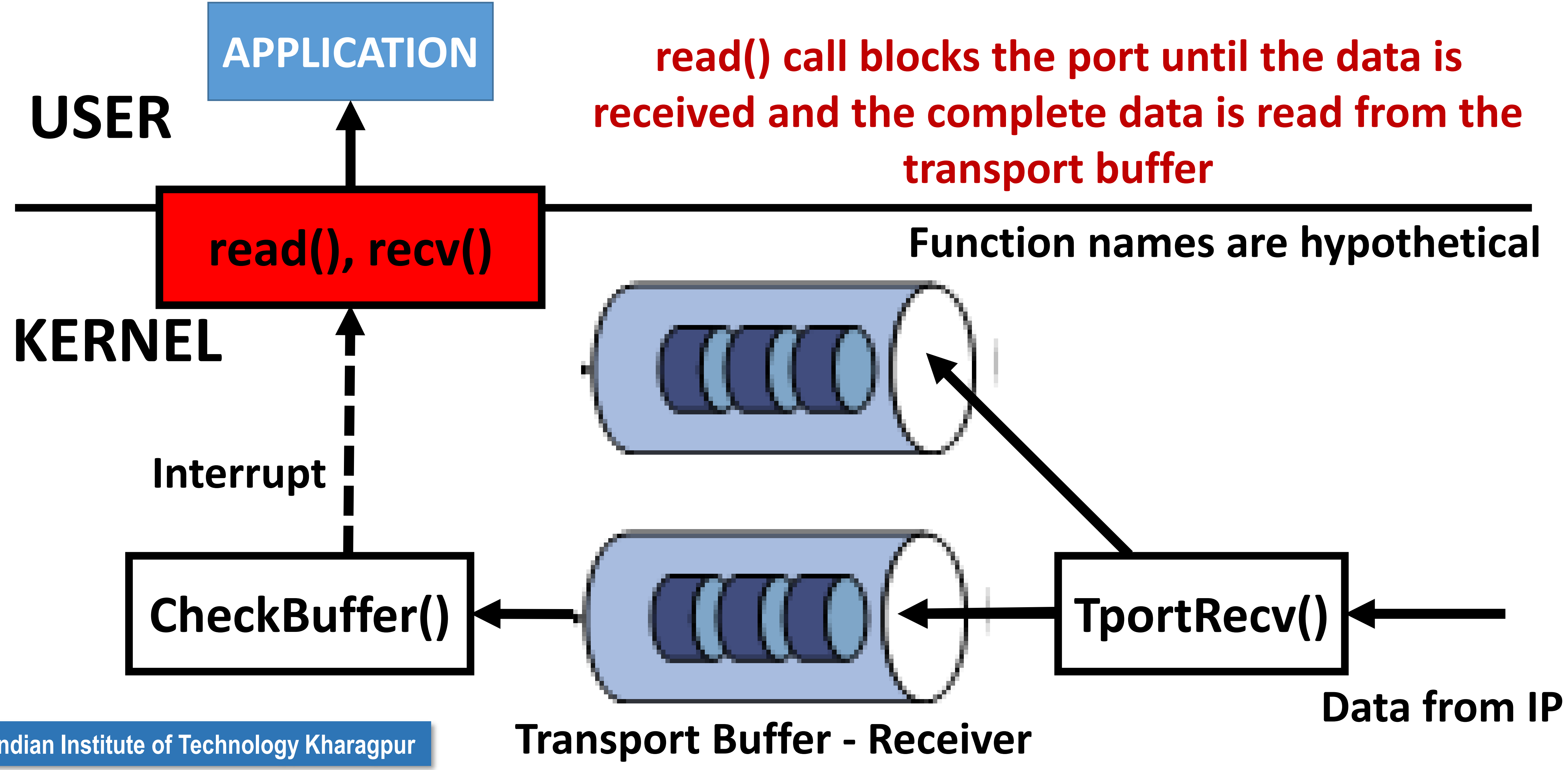
Different connections are treated differently, so we need connection specific source buffering

write() call blocks the port until the complete data is written in the transport buffer

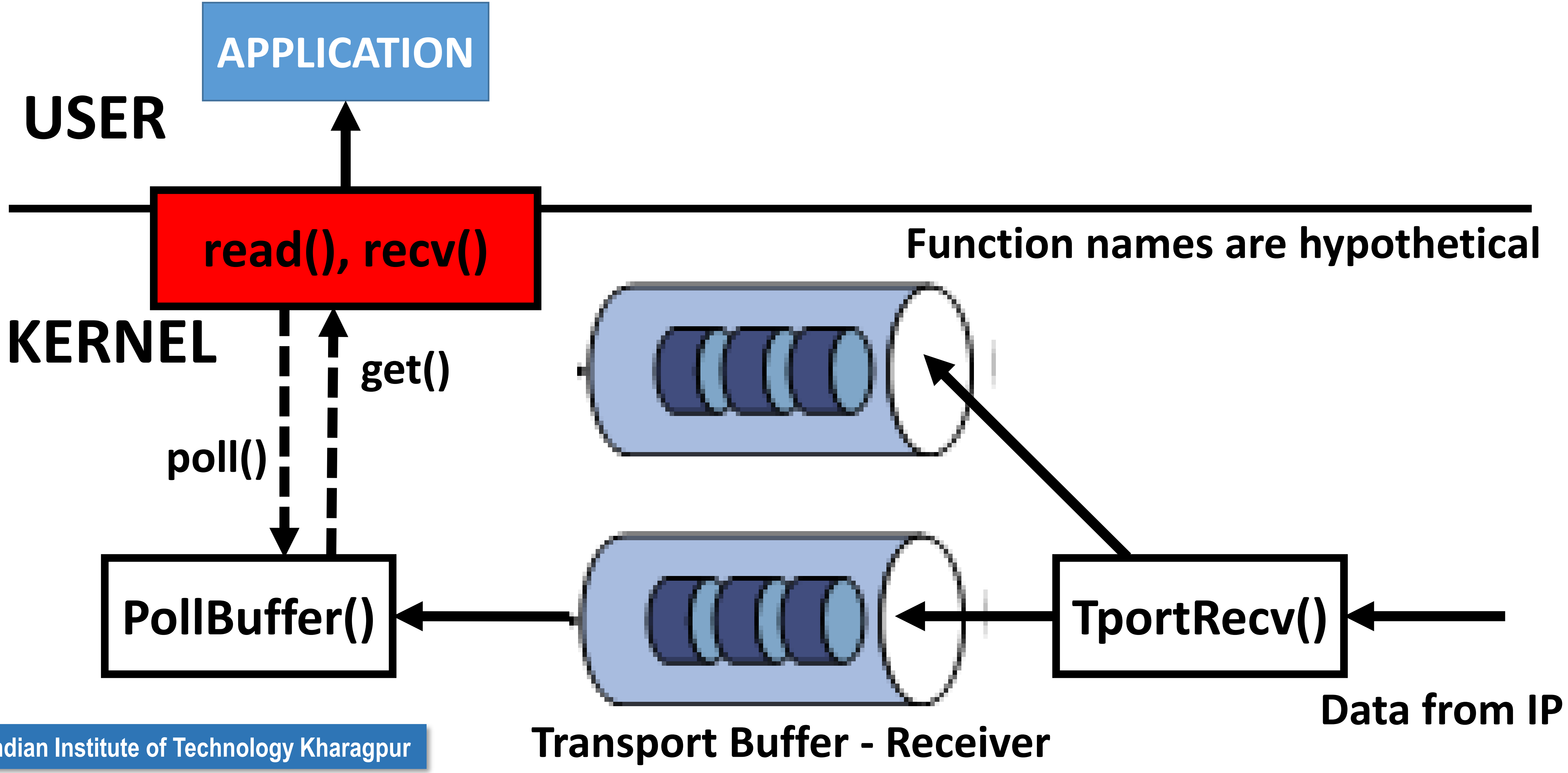
Function names are hypothetical

Application Transport Interfacing – Receiver Side

read() call blocks the port until the data is received and the complete data is read from the transport buffer



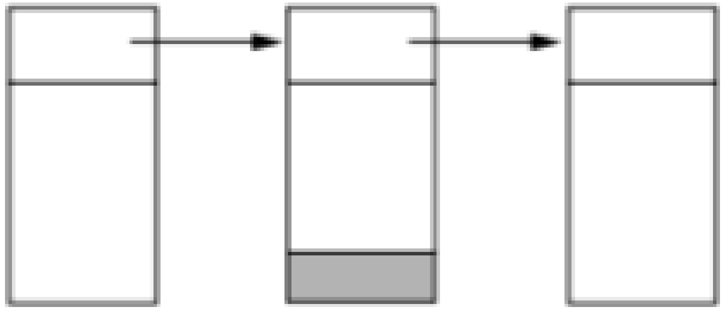
Application Transport Interfacing – Receiver Side (Alternate Implementation)



Organizing Transport Buffer Pool

- If most segments are nearly the same size, organize the buffer as a pool of identically sized buffers (one segment per buffer)

- For variable segment size – **chained fixed sized buffer** (buffer size = maximum segment size`

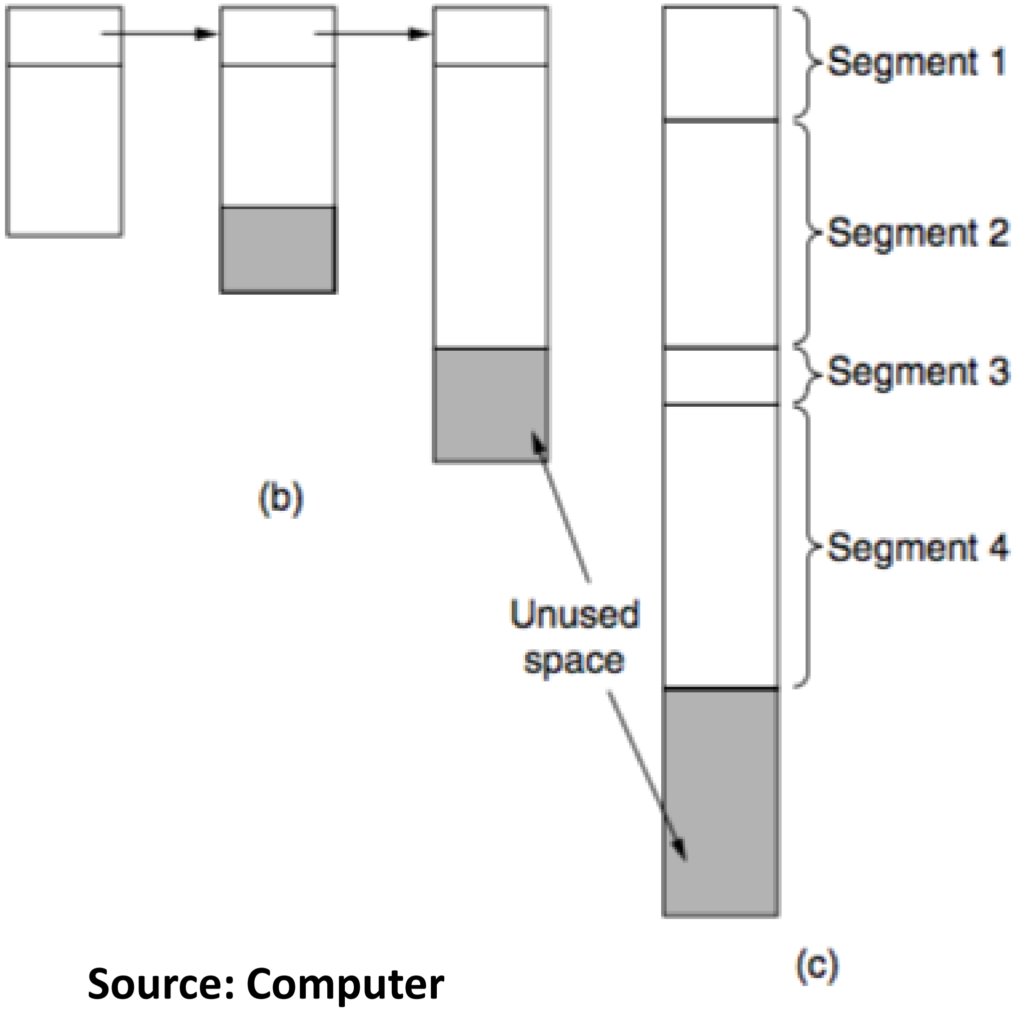


- Space would be wasted if segment sizes are widely varied
- Small buffer size – multiple buffers to store a single segment – added complexity in implementation

Organizing Transport Buffer Pool

- **Variable size buffers (b)**
  - Advantage: better memory utilization
  - Disadvantage: Complicated implementation

- Single large **circular buffer** for every connection (c)
  - Good use of memory only when connections are heavily loaded

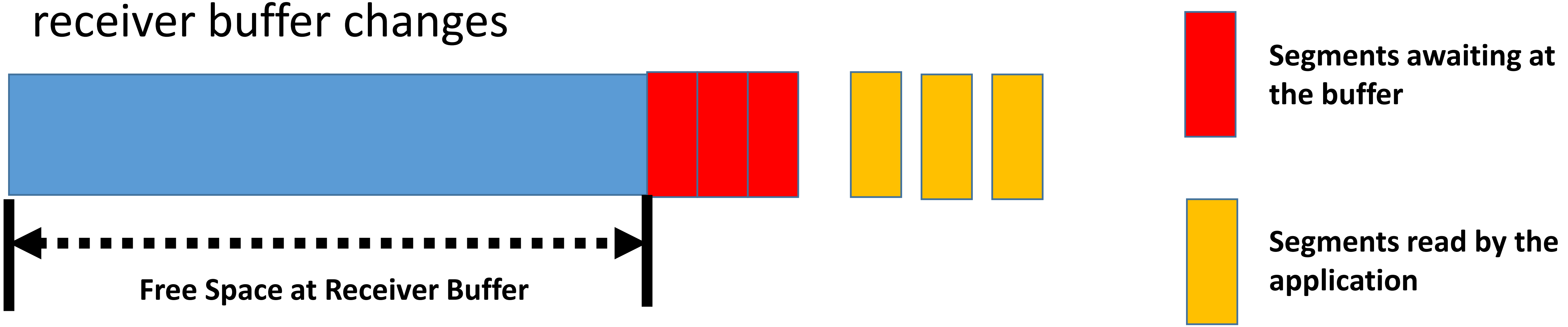


Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Dynamic Buffer Management for Window Based Flow Control

- Sender and receiver needs to dynamically adjust buffer allocations

- Based on the rate difference between the **receive rate by the transport entity** and the **receive rate by the application**, the available size of the receiver buffer changes



- Sender should not send more data compared to receiver buffer space – dynamically adjust the window size based on availability of receiver buffer space

Dynamic Buffer Management for Window Based Flow Control

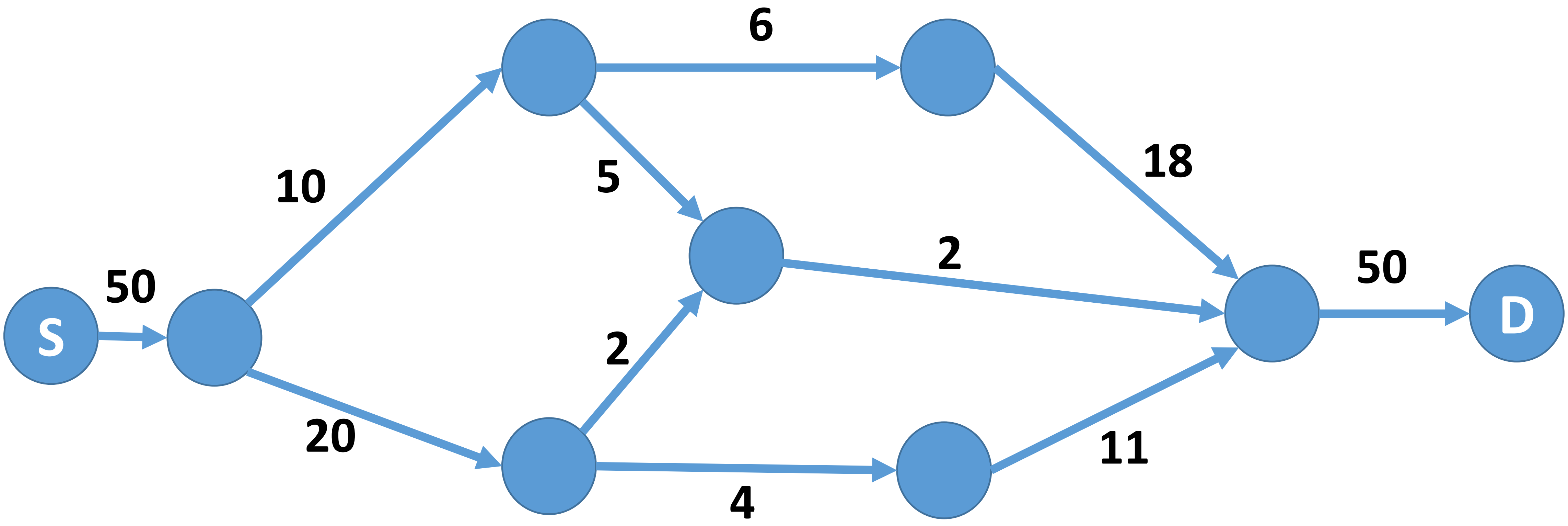
- Receiver forwards available buffer space through ACK

	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

Ensure that the ACKs are flowing in the network continously

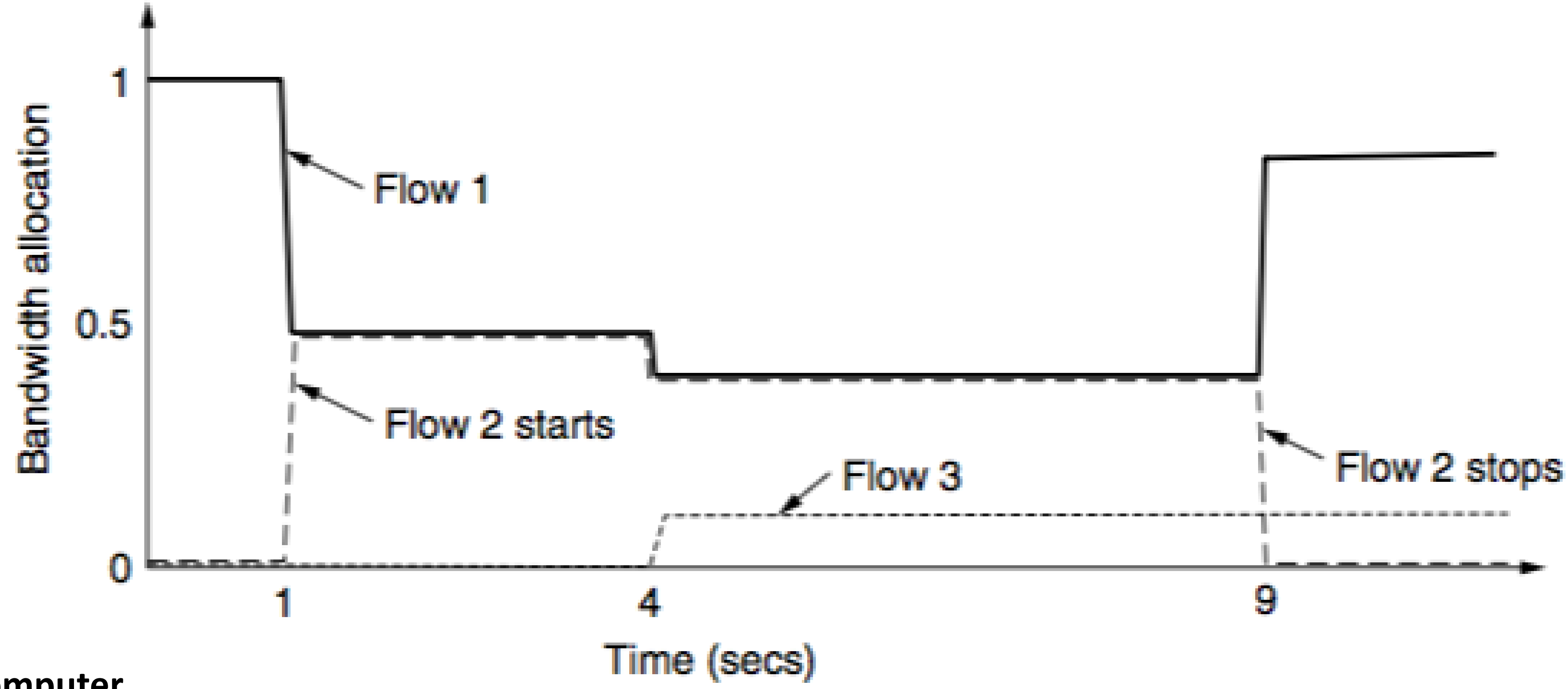
Congestion Control in the Network

- Consider a centralized network scenario – how can you maintain optimal flow rates?



Apply Max Flow Min Cut Theorem !

But this is hard in a real network …



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

Changing Bandwidth Allocation over Time

Congestion Control in the Network

- Flows enter and exit network dynamically – so applying an algorithm for congestion control is difficult

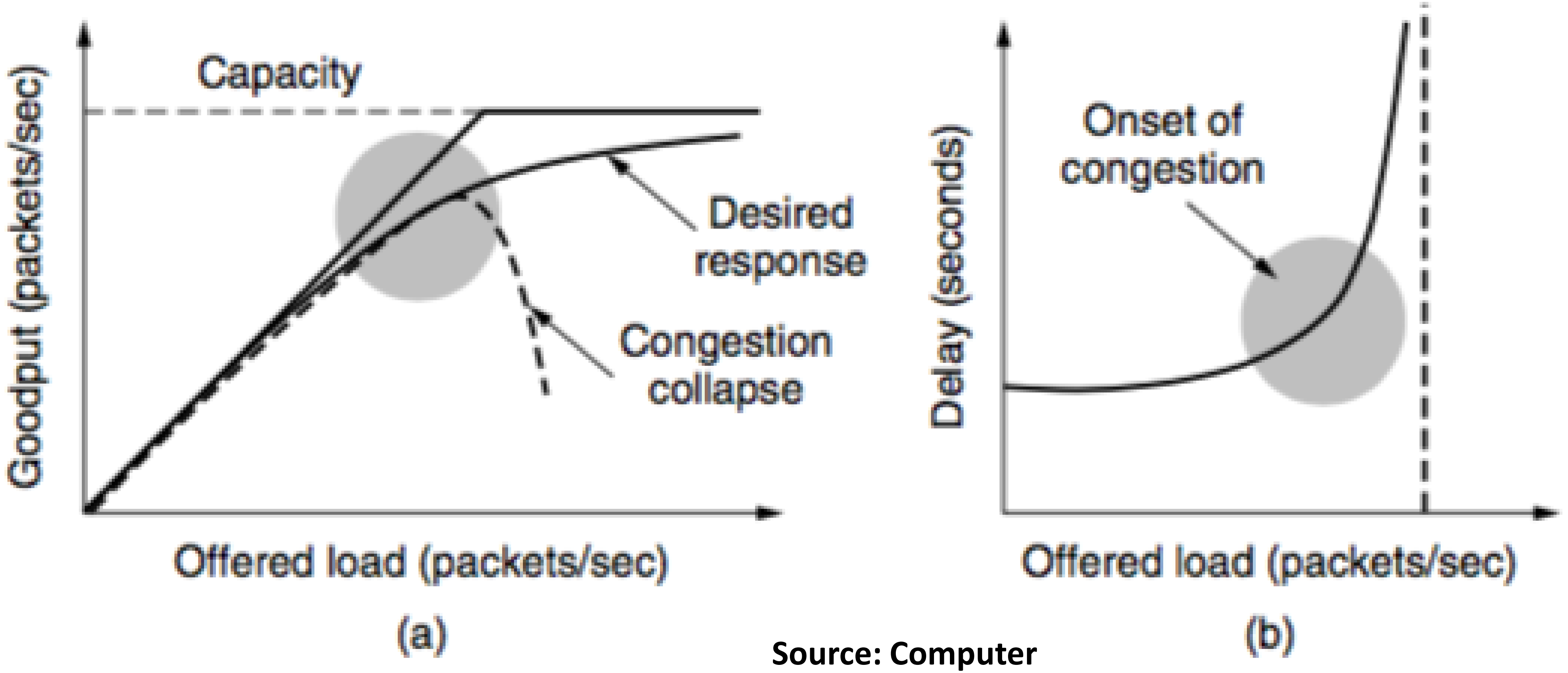
- **Congestion avoidance:** Regulate the sending rate based on what the network can support

Sending Rate = minimum (network rate, Receiver rate)

Gradually increase the network rate and observe the effect on flow rates (packet loss)

Comes from flow control – receiver advertised window size for a sliding window flow control

Network Congestion – Impact over Goodput and Delay

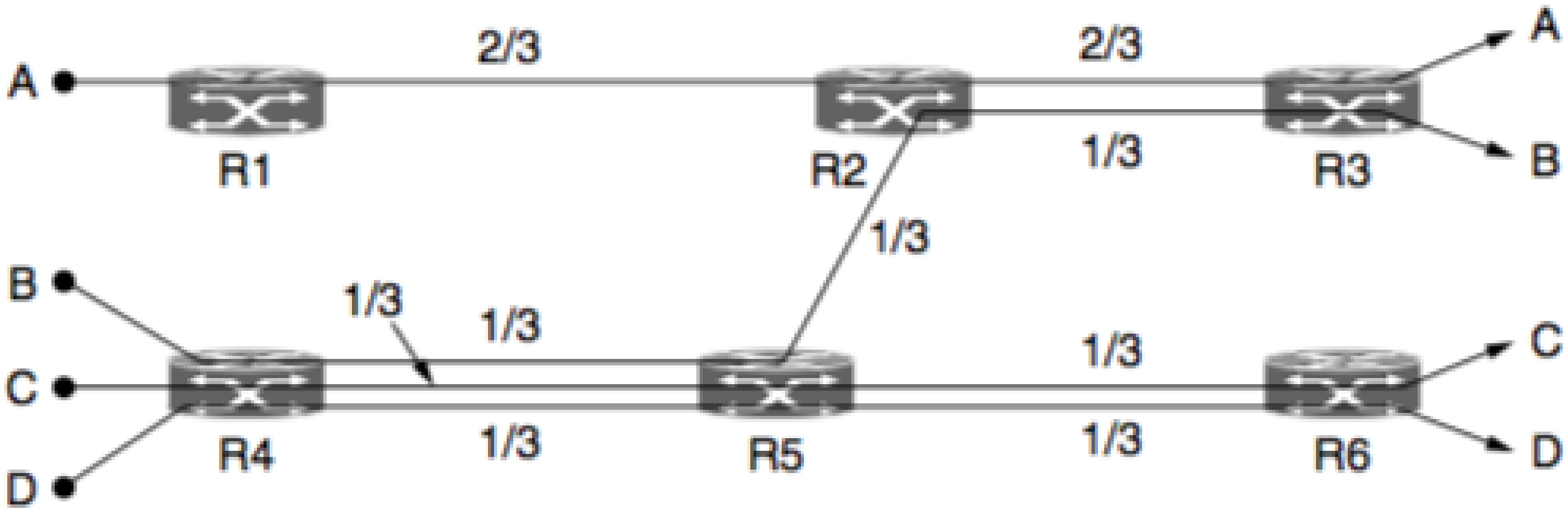


Source: Computer Networks (5th Edition) by Tanenbaum, Wetherell

Congestion Control and Fairness

- Ensure that the rate of all the flows in the network is controlled in a **fair way**
- A bad congestion control algorithm may affect fairness - Some flows can get starved
- Hard fairness in a decentralized network is difficult to implement
- **Max-Min Fairness**: An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation.

Max-Min Fairness – An Example



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

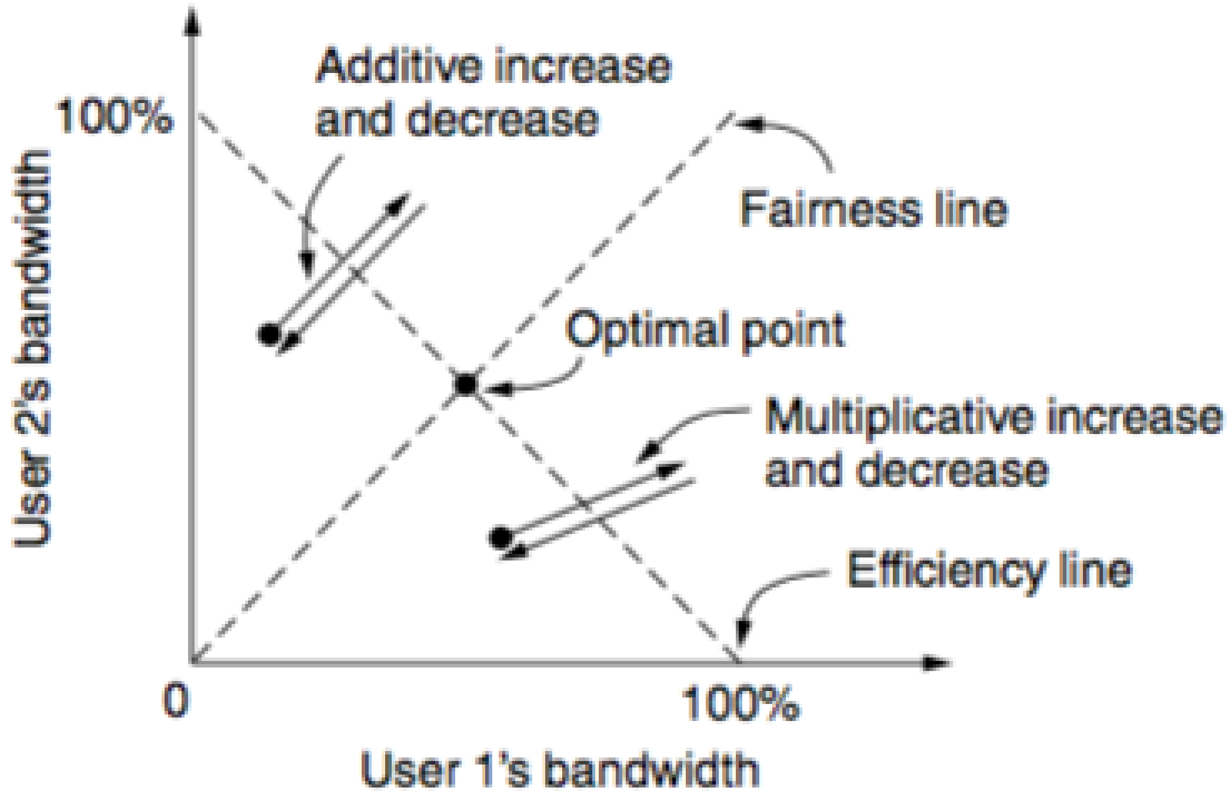
AIMD – Efficient and Fair Operating Point for Congestion Control

- **Additive Increase Multiplicative Decrease (AIMD)** – Chiu and Jain (1989)

- Let *w(t)* be the sending rate. *a (a > 0)* is the additive increase factor, and *b (0<b<1)* is the multiplicative decrease factor

$$w(t+1) = \begin{cases} w(t) + a & \text{if congestion is not detected} \\ w(t) \times b & \text{if congestion is detected} \end{cases}$$

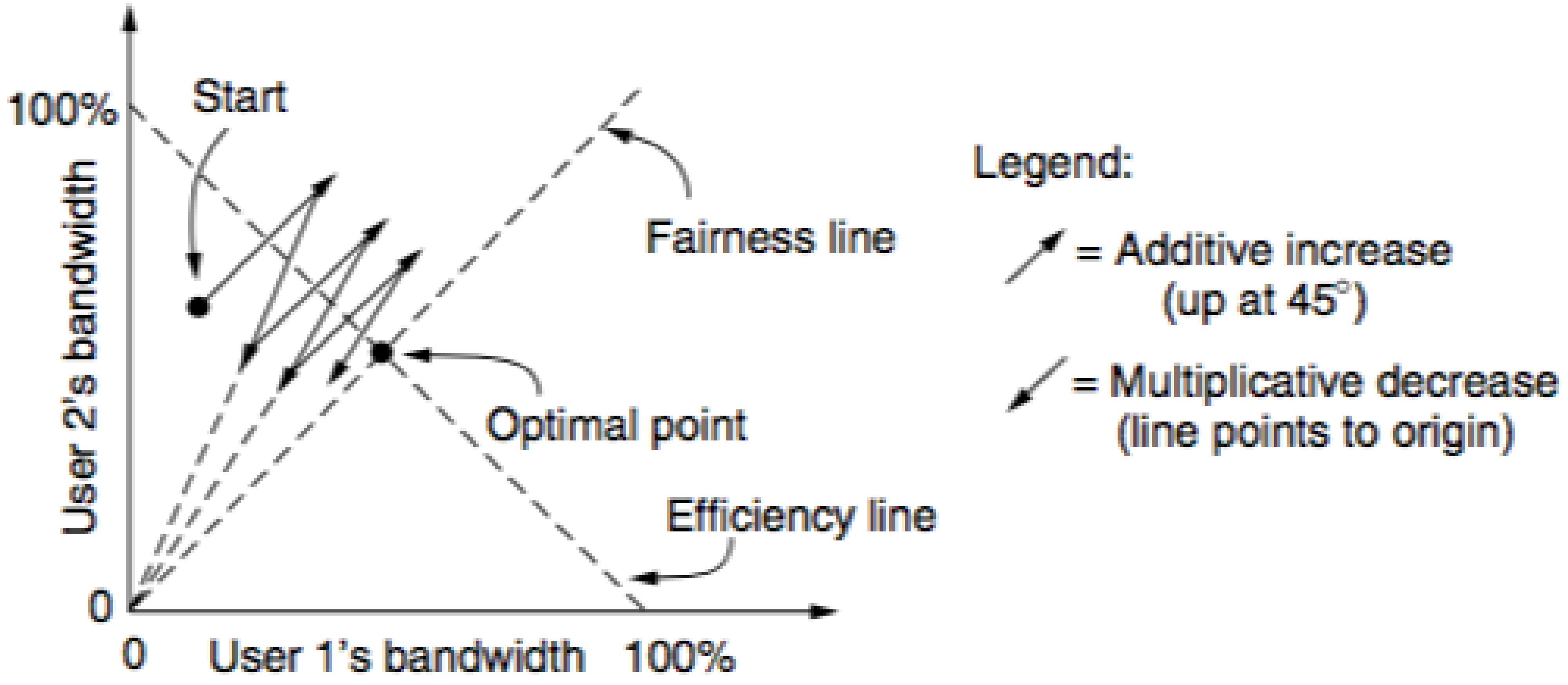
AIMD – Design Rationale (Two Flows Example)



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

- AIAD – Oscillate across the efficiency line
- MIMD – Oscillate across the efficiency line (different slope from AIAD)

AIMD – Design Rationale (Two Flows Example)



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

- The path converges towards the optimal point
- Used by TCP - Adjust the size of the sliding window to control the rates

Let us look TCP design details …