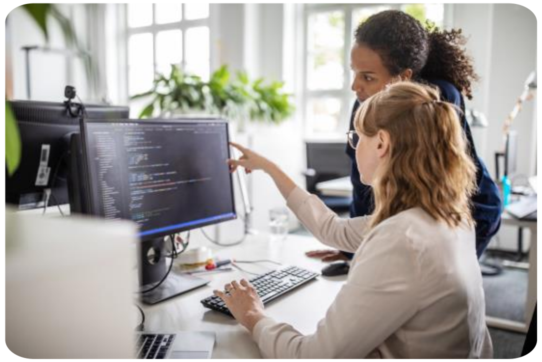
Connection Oriented Transport: TCP

Transmission Control Protcol

agenda

- The **TCP** Connection
- TCP Segment **Structure**
- Round-Trip Time **Estimation** and **Timeout**
- Reliable **Data** Transfer
- **Flow** Control



Transmission Control Protocol (TCP)

Fundamental communication protocol in computer networking. It operates at the transport layer of the **OSI** model and plays a crucial role in ensuring reliable and orderly data transmission over the internet and other interconnected networks. It works hand in hand with the **Internet Protocol (IP)** to form the basis of the widely used **TCP/IP** protocol suite

Connection Oriented Protocol

(TCP)

Connection Oriented Protocol

- Connection Oriented Protocol is one that establishes and maintains a **logical** connection b/w two devices.
- This connection ensures that data is transmitted **reliably** and in the correct **order** b/w sender and receiver.

The TCP Connection

TCP IS CONNECTION-ORIENTED

APPLICATION PROCESS **“HANDSHAKE”** WITH EACH OTHER.

MUST SEND SOME **PRE-LIMINARY SEGMENTS** TO EACH OTHER

This Photo by Unknown Author is licensed under CC BY-SA-NC

Connection Oriented Transport: TCP

- Full-Duplex Service
  - Data can flow from one process to another and vice versa simultaneously.
- Point to Point
  - The transfer of data from one sender to many receivers in a single send operation is not possible.

Establishment of connection

- Client/Server Process
  - clientSocket.connect((serverName, serverPort)) > *port identifies process on the server*
- Three Way Hand-Shake
- Connection Establishment
- Maximum Segment Size (MSS) - Largest Link Layer frame

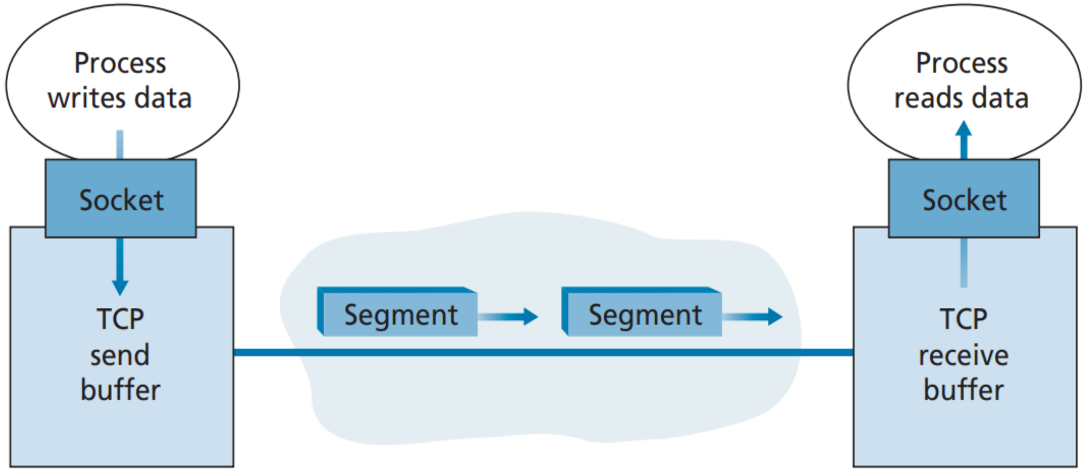
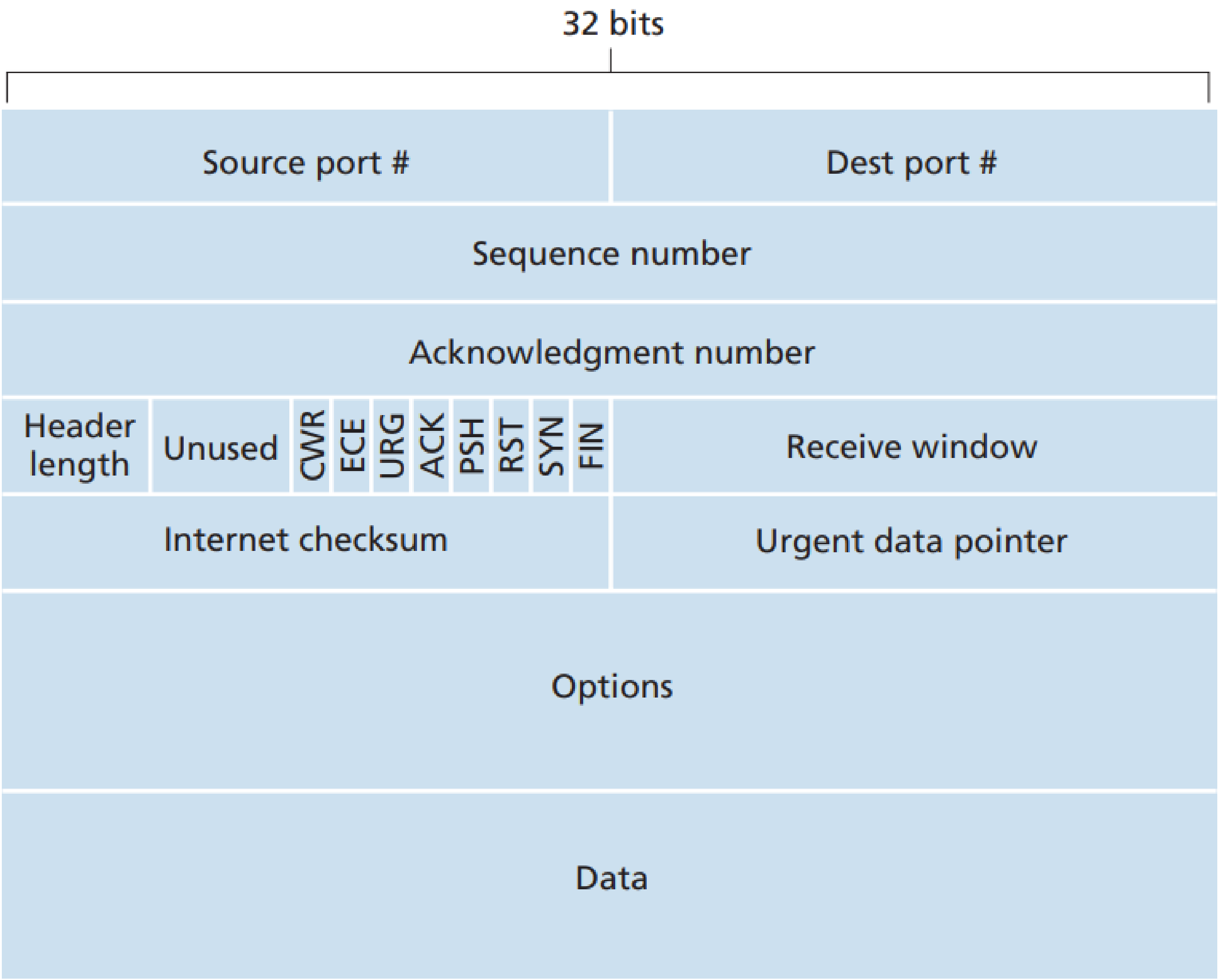


Figure 3.28 ♦ TCP send and receive buffers

TCP Segment Structure

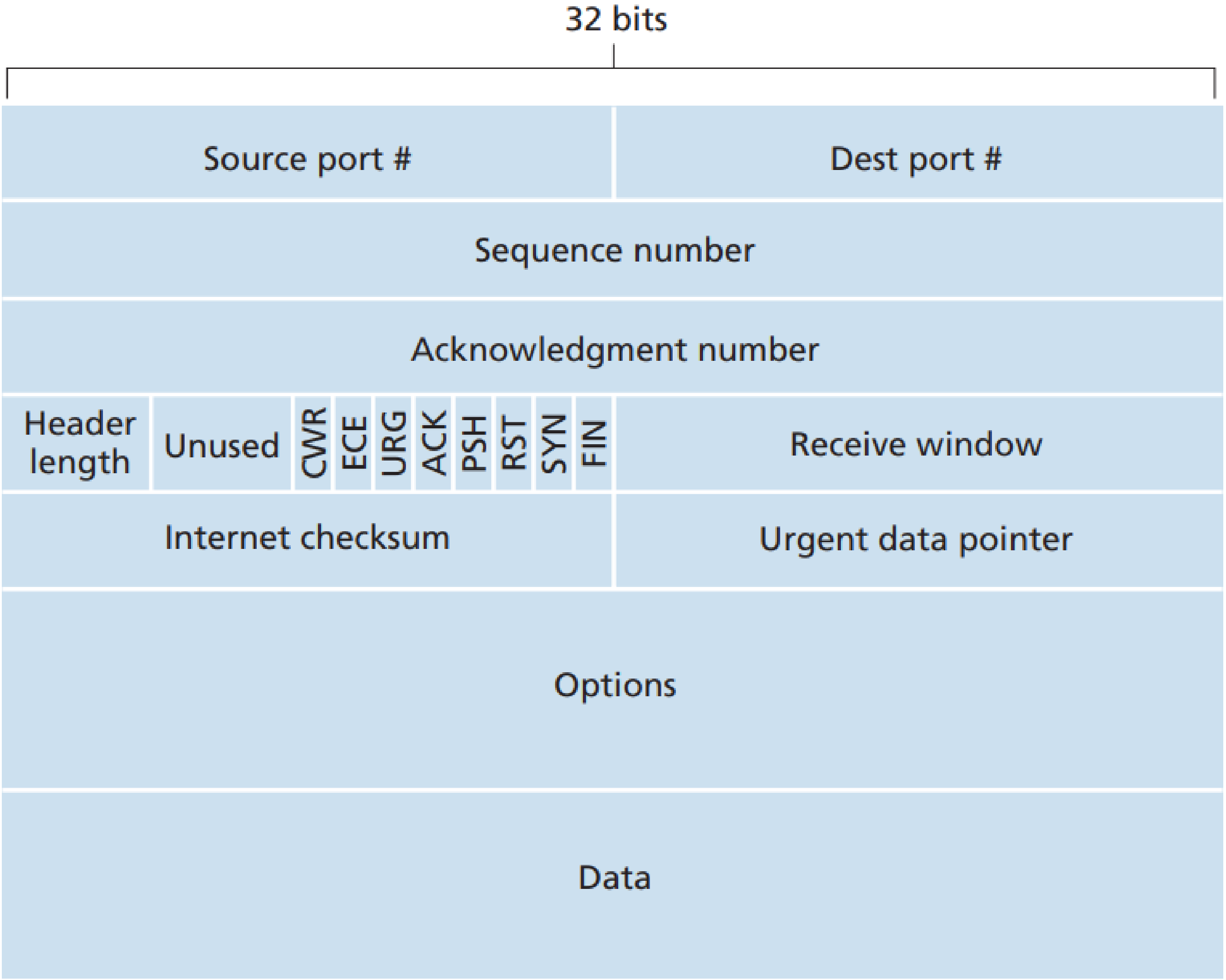
TCP Segment Structure

- Header field
- Data field
- Source & Destination Ports
- Checksum field(integrity)
- Sequence number field
- Acknowledgment number field



TCP Segment Structure

- Receive window field
- Header length field(32bit)
- Options field (MSS)
- Flag field(6 bits)
  - ACK, RST, SYN, FIN, PSH,URG



Establishment of connection

- Sequence number for a segment
- Acknowledgement Number

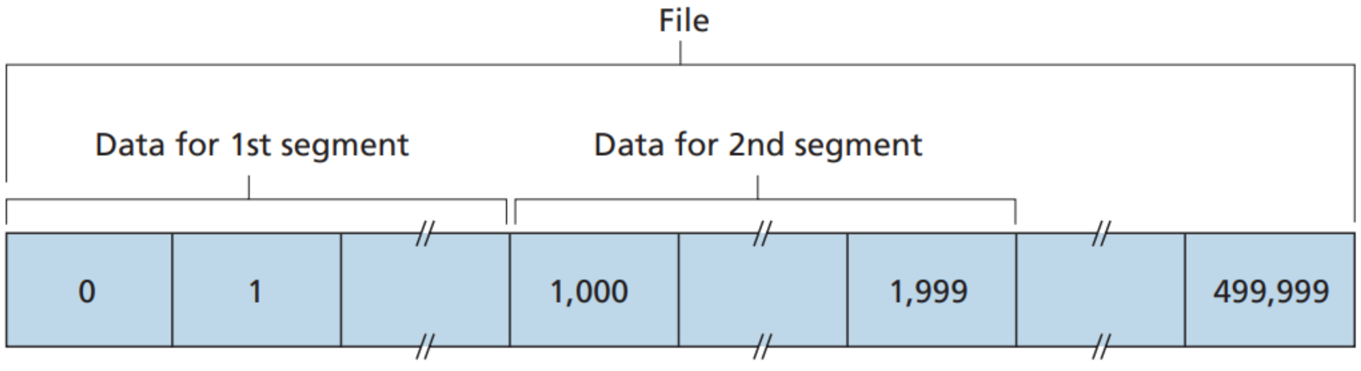


Figure 3.30 ♦ Dividing file data into TCP segments

TCP Implementation

There are basically two choices:

(1) the receiver immediately discards out-of-order segments

(2) the receiver keeps the out-of-order bytes and waits for the missing bytes to fill in the gaps

Telnet: A Case Study

- **Telnet**, popular application-layer protocol used for remote login.
- Unlike the bulk data transfer, Telnet is an interactive application.
- **Piggybacked**

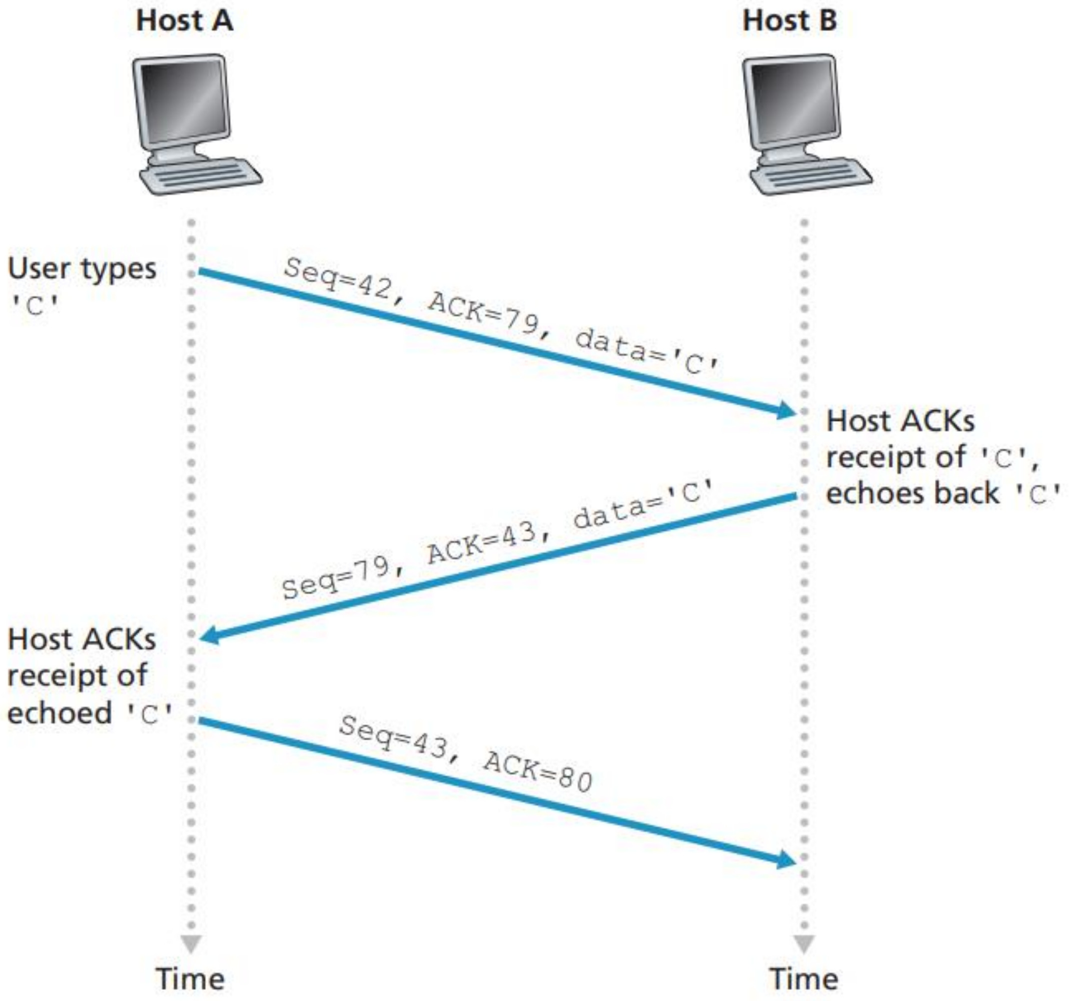
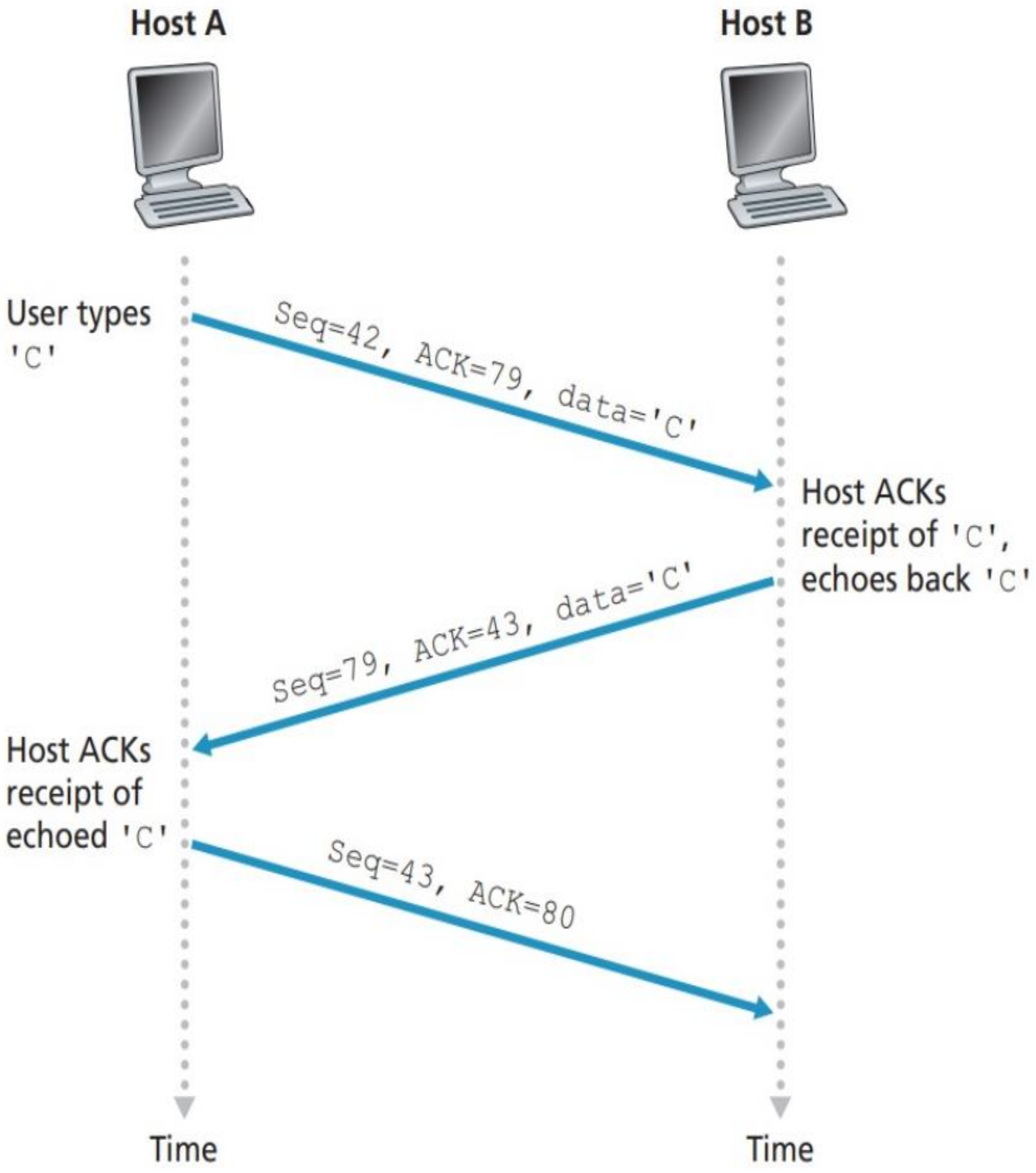


Figure 3.31 ♦ Sequence and acknowledgment numbers for a simple Telnet application over TCP

Round Trip Time(RTT) and Timeout

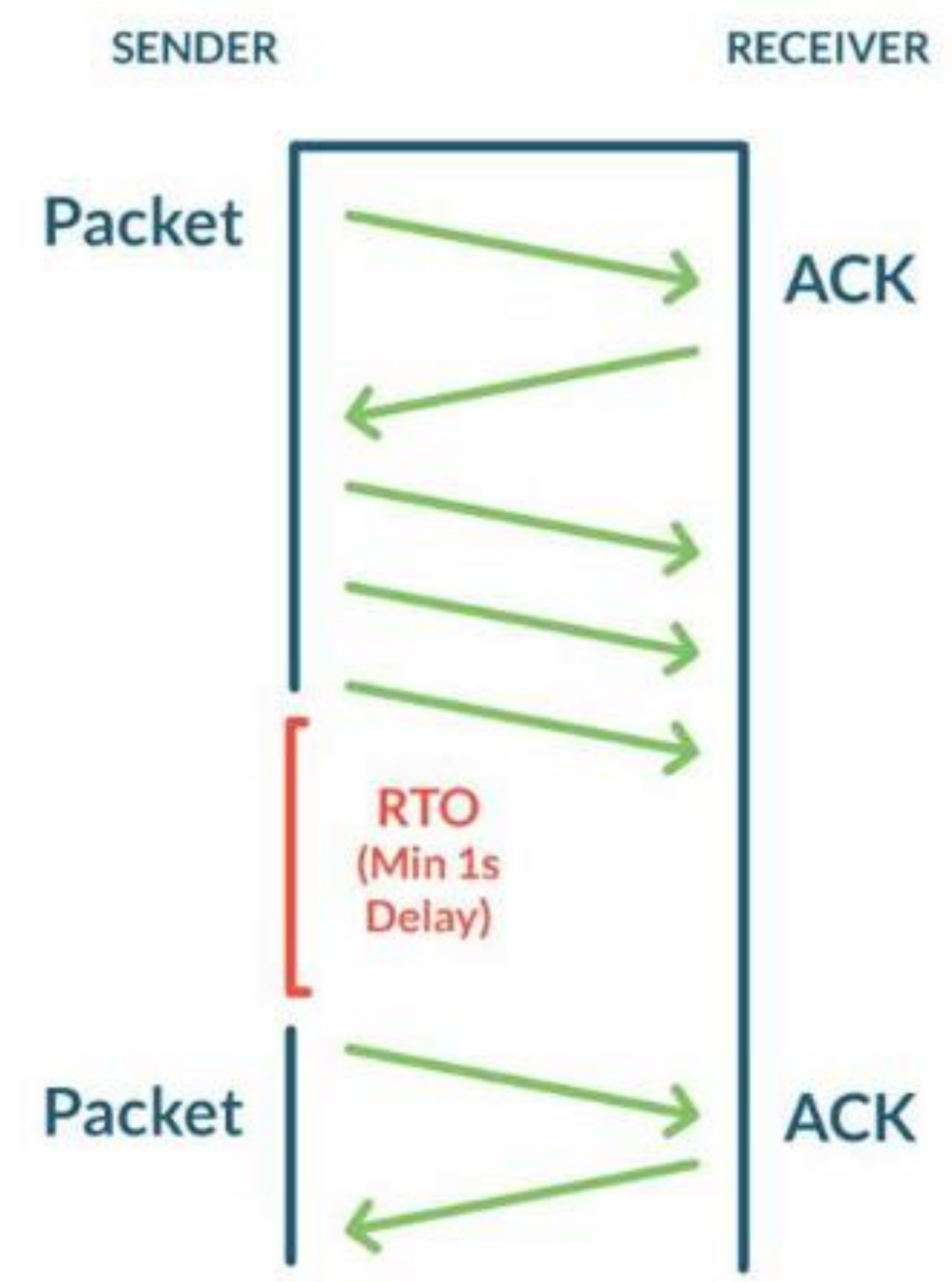
Round Trip Time

- Amount of time between sent time and ack message time



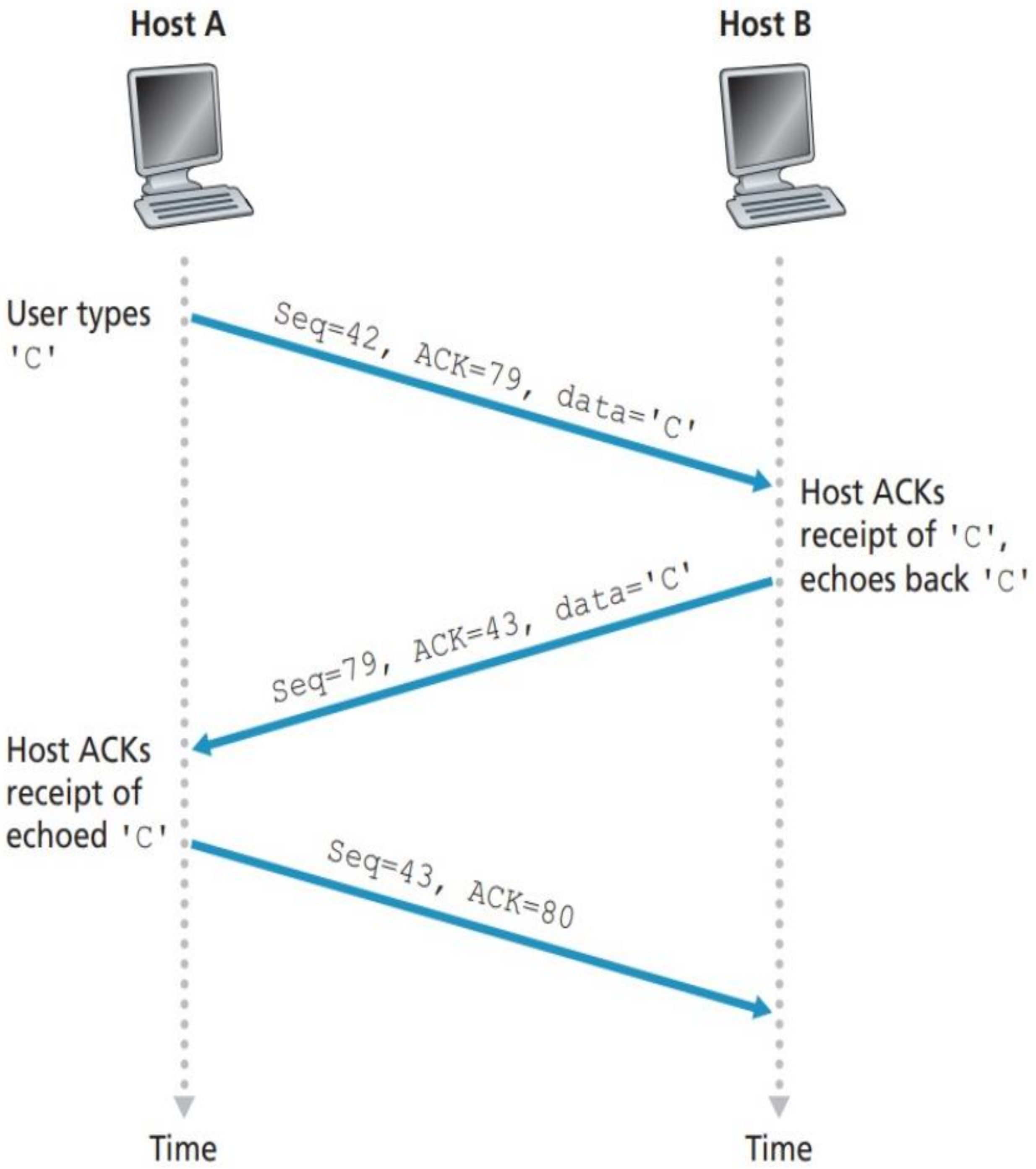
How to set TCP timeout value.

- Longer than RTT
  - Issue: RTT varies
- Too short than RTT:
  - Unnecessary timeout,
- Too long than RTT:
  - Slow reaction to segment loss



How to measure RTT.

- **Sample RTT:** measured time from segment transmission until Ack message received
  - Ignore retransmissions
- **Estimated RTT:** average several sample RTTs, not just single RTT



Estimated RTT.

- Exponential weighted moving average(**EWMA**)
- Influence of past sample decreases exponentially fast

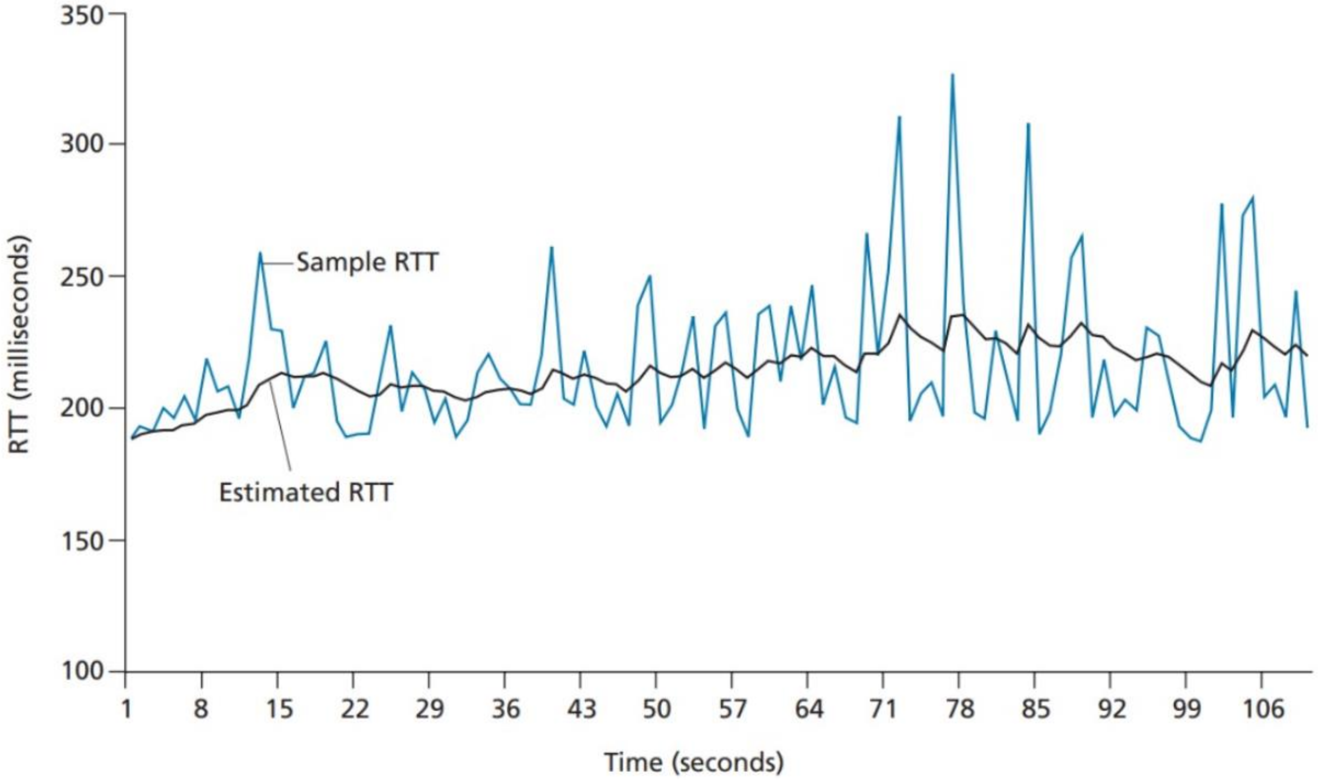


Figure 3.32 ♦ RTT samples and RTT estimates

Estimated RTT - cont

$$\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$$

- Typical value for α is : 0.125

$$\text{EstimatedRTT} = 0.875 \cdot \text{EstimatedRTT} + 0.125 \cdot \text{SampleRTT}$$

Deviation RTT.

- Deviation of SampleRTT from EstimatedRTT

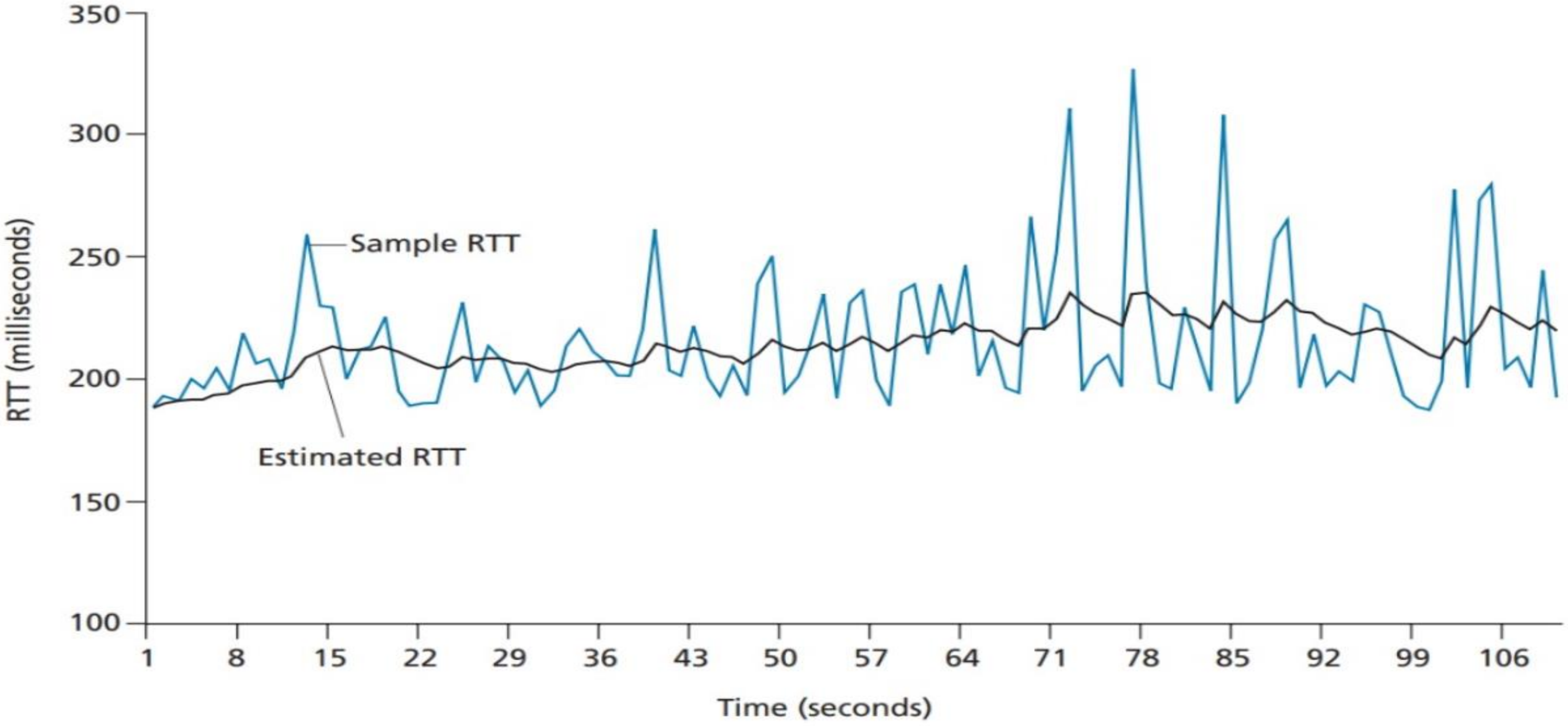


Figure 3.32 ♦ RTT samples and RTT estimates

Deviation RTT. - cont

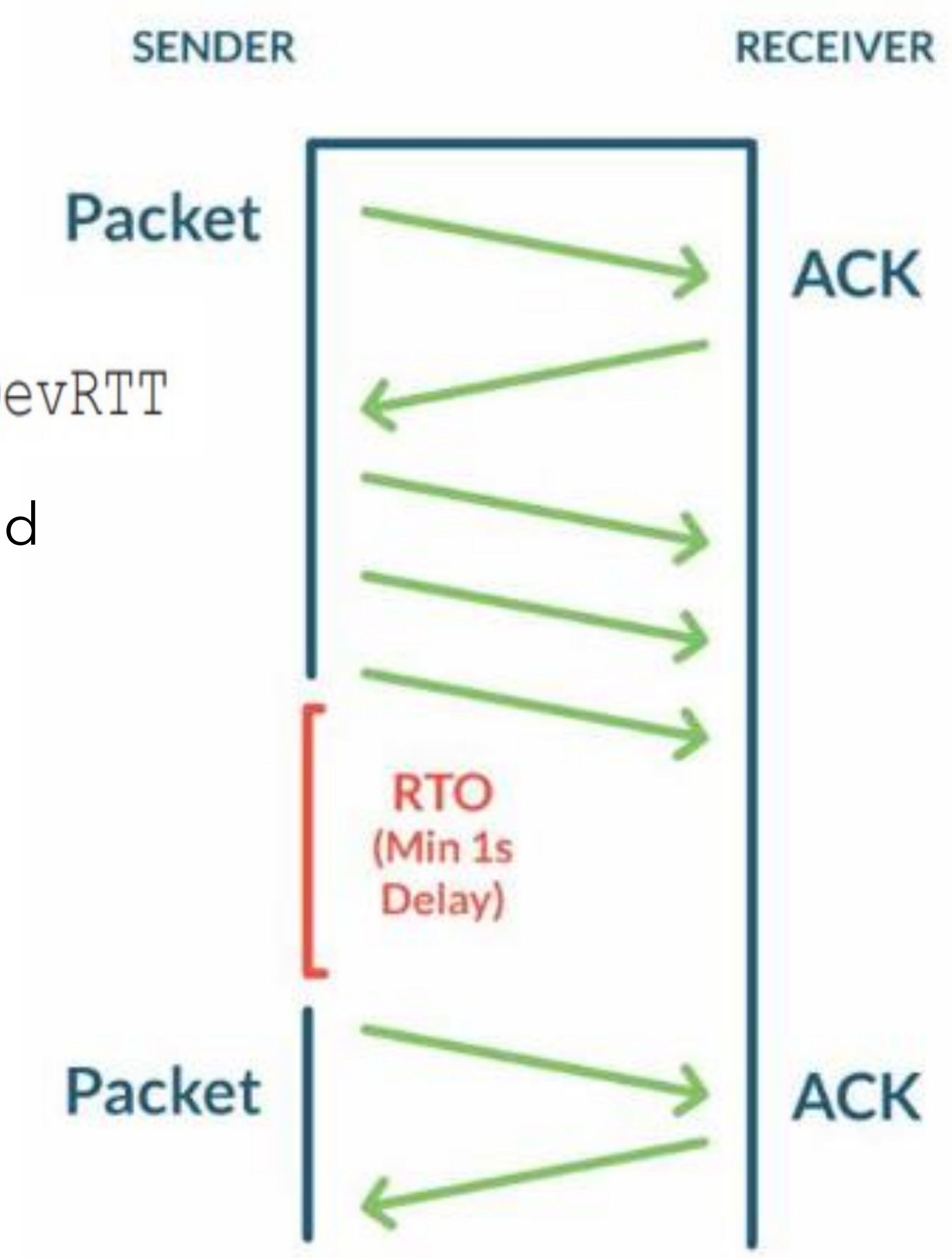
$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot |\ \text{SampleRTT} - \text{EstimatedRTT}\ |$$

- The recommended value of β is 0.25.

Timeout Interval

$$\texttt{TimeoutInterval} = \texttt{EstimatedRTT} + 4 \cdot \texttt{DevRTT}$$

- An initial T*imeoutInterval* value of 1 second is recommended.
- When timeout occurs, value of T*imeoutInterval* is doubled to avoid premature timeout and is again updated through the formula when the acknowledge message is received



Reliable Data Transfer

- What is **Reliable Data Transfer** in TCP?
- Reason to use this?
- Three **Events** and **Actions** on it
- Three **Scenarios**

What is Reliable Data Transfer

- A Service which ensures that data received on the receiver side via TCP is…

1

Uncorrupted

2

In Sequence

3

Without Duplications

4

Without Gaps

Reason to use this

- Sending Data only To IP is Un-reliable
  - Sender is not confirmed that receiver received the data or not
  - Data that is send can be lost or Corrupted
- Acknowledges
  - A Verification from receiver side to sender side when data is received

data received on the receiver side via TCP is...

- Re-Transmissions
  - When data is send to receiver. That data is marked as **Not-Acknowledge**
  - If the Timer out … then the **Not-Acknowledge** data is re-send to receiver

Re-Transmissions Triggered

- Sending Data only To IP is Un-reliable
  - Sender is not confirmed that receiver received the data or not
  - Data that is send can be lost or Corrupted
- Acknowledges
  - A Verification from receiver side to sender side when data is received

data received on the receiver side via TCP is...

- Re-Transmissions
  - When data is send to receiver. That data is marked as **Not-Acknowledge**
  - If the Timer out … then the **Not-Acknowledge** data is re-send to receiver

Three Events

1

Data Received

When Sender received Data from Application

2

data received on the receiver side via TCP is...

Timer Timeout

Data send but not acknowledge

3

Acknowledge Received

Receiver send Acknowledge message to Sender

Actions on Three Events

- Data Received
  - Create Segment with Sequence Number (= NextSeqNum)
  - Send Segment (to receiver)
  - Increment NextSeqNum with data-length
  - Start Time if not running
- Timer Timeout
  - Re-transmit a Not-Yet-Acknowledge Segment
  - Start Timer
- Acknowledge Received

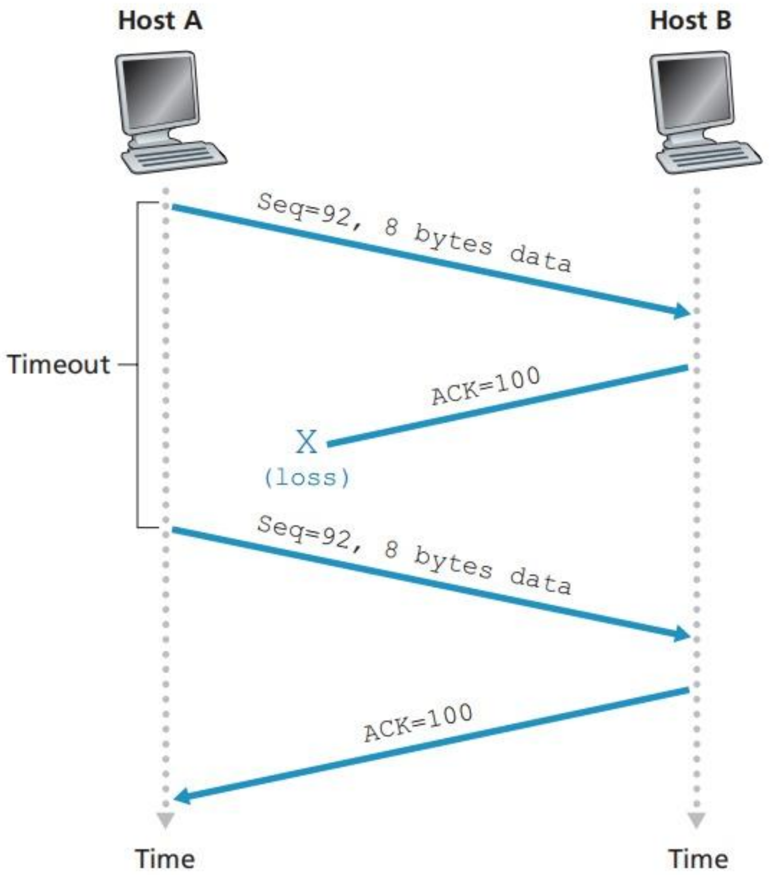
data received on the receiver side via TCP is...

```
If( y > Send Base ) {
     Send Base = y
     if(Not-Yet-Acknowledge Segment Present){
         Start Timmer
     }
}
```

Three Scenarios

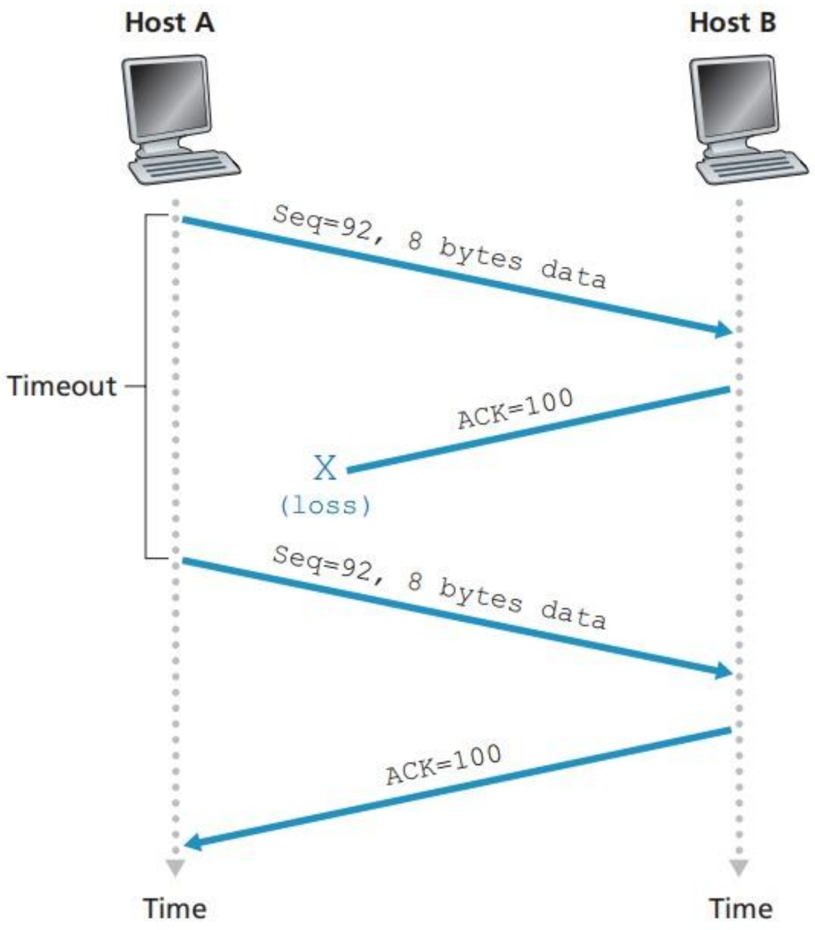
Retransmission due to a lost acknowledgement

- Host A send data to Host B
- Data is Send , But acknowledge is not Received in terminal
- Data again send
- Acknowledgement Received



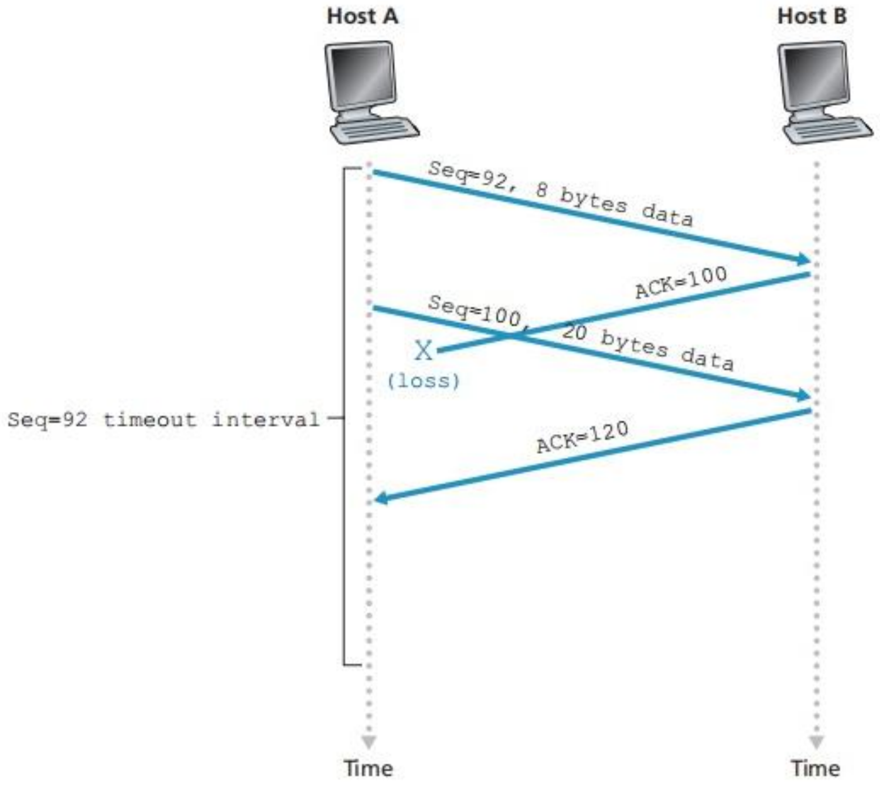
Segment 100 not retransmitted

- Two Data Segment sends 92 and 100
- Acknowledge not received in time interval
- Smallest Not-Yet acknowledge send
- Segment 100 not send



A cumulative acknowledgement avoid retransmission of the first segment

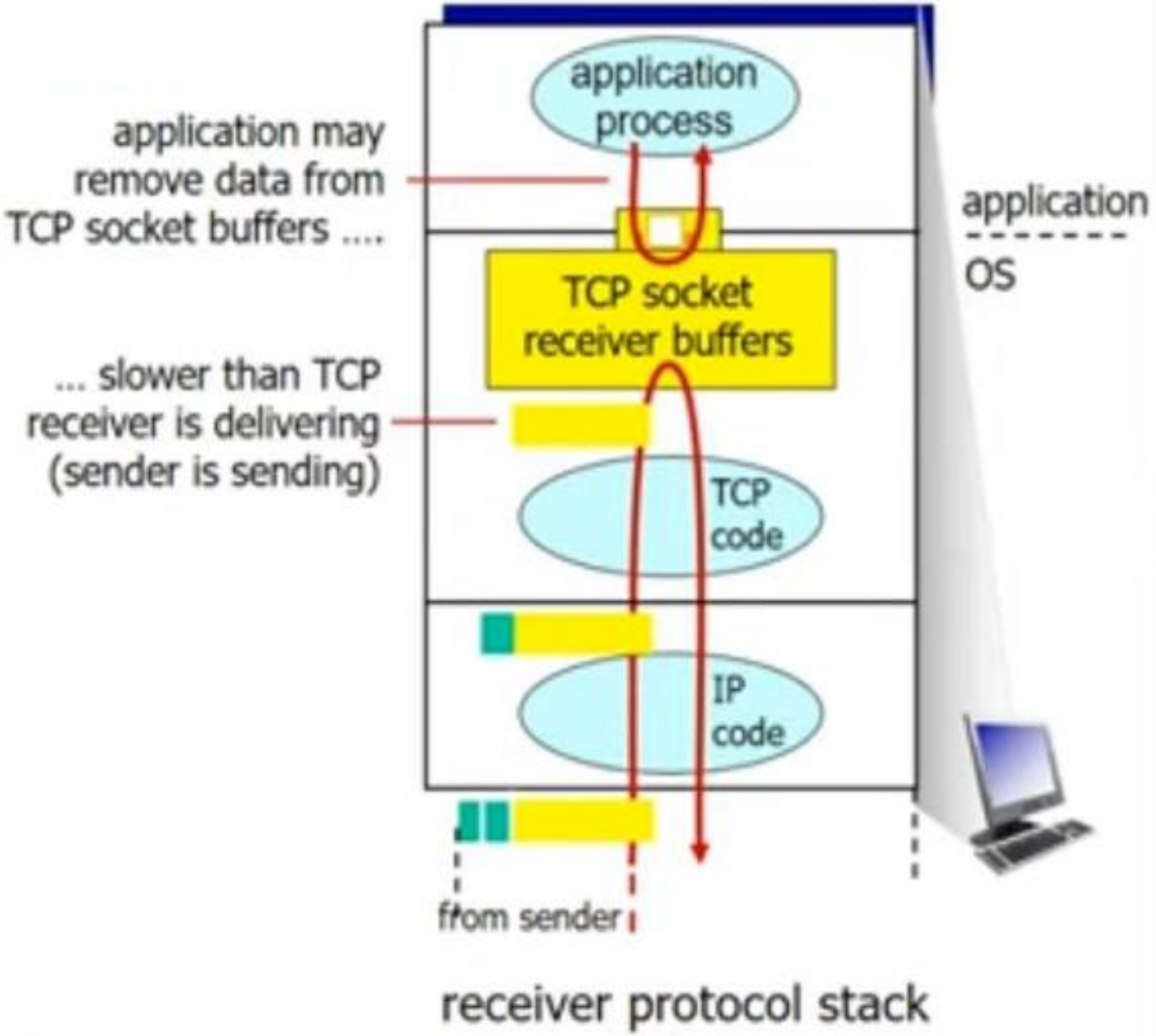
- Segment 92 is send and Timer start
- Acknowledge not received
- Segment 100 is send within Timer
- Acknowledge 120 received
- segment 92 will not be resend



Flow Control

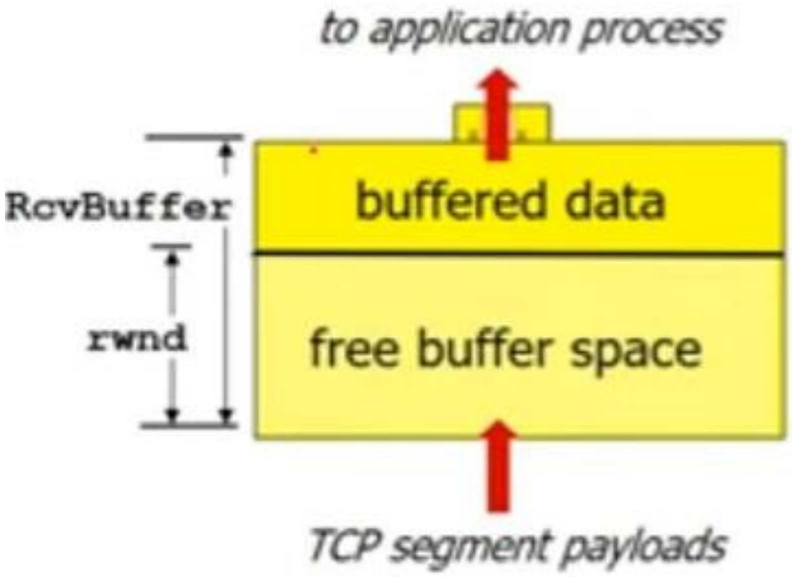
Flow Control

- Receiver controls sender, so sender won't overflow receiver's buffer by transmitting too much, too fast.


application process
application may remove data from TCP socket buffers ....
TCP socket receiver buffers
... slower than TCP receiver is delivering (sender is sending)
TCP code
IP code
application
OS
from sender
receiver protocol stack

Flow Control

- Receiver informs free buffer space by including rwnd value in TCP header of receiver-to-sender segments
  - RcvBuffer size set via socket options(default 4096 bytes)
  - Many operating systems auto adjust RcvBuffer



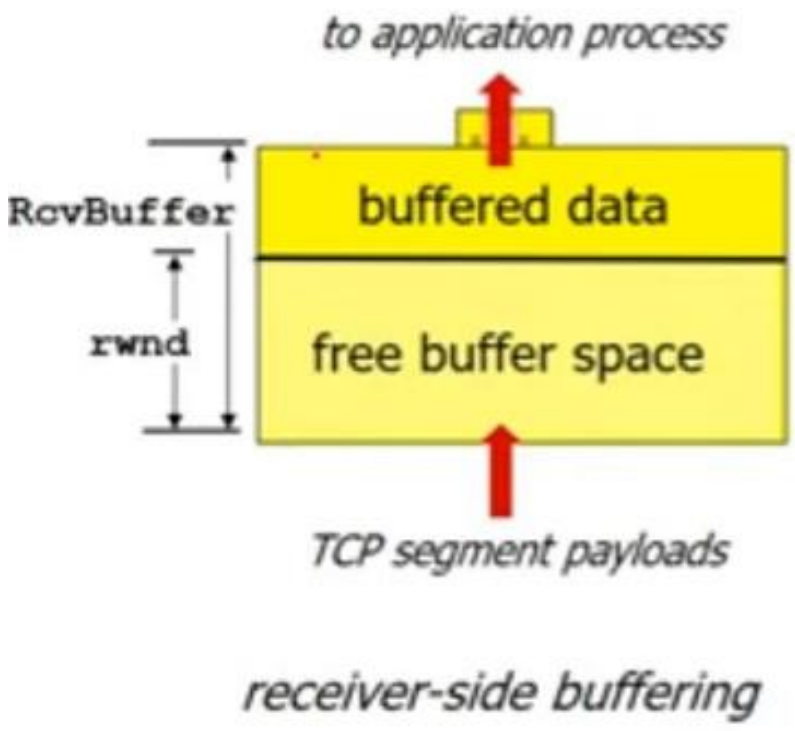
receiver-side buffering

Flow Control

- Sender limits amount of unacknowledged data to receiver's rwnd value.

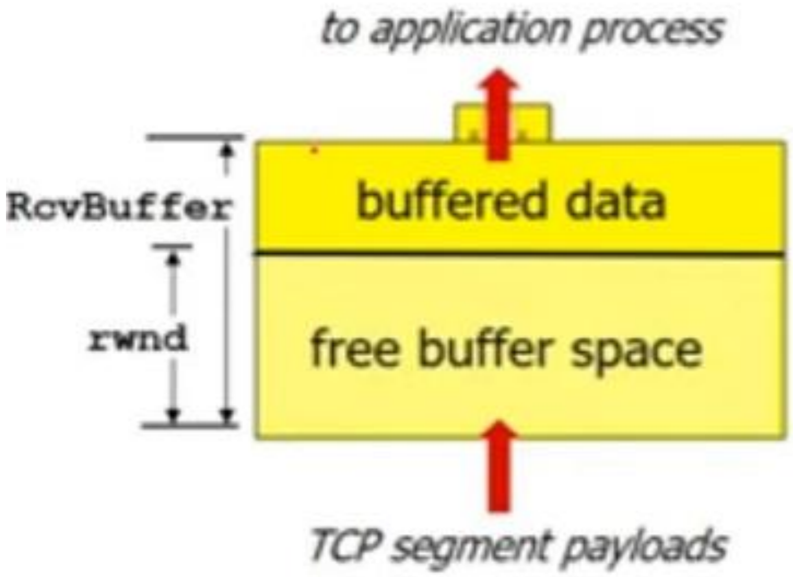
  LastByteSent – LastByteAck <= rwnd

- Gaurantees receive buffer will not overflow



Flow Control – handling issue

- As the sender gets blocked if the rnwd is full when, it will unblock?
- Sender keeps on sending 1 byte for acknowledgement untill it is acknowledged.



receiver-side buffering

Thank you

- Presentation by Group 6
  - **S**alman Naeem
  - **N**uman Ahmad
  - **F**arjad Waseem