IIT KHARAGPUR

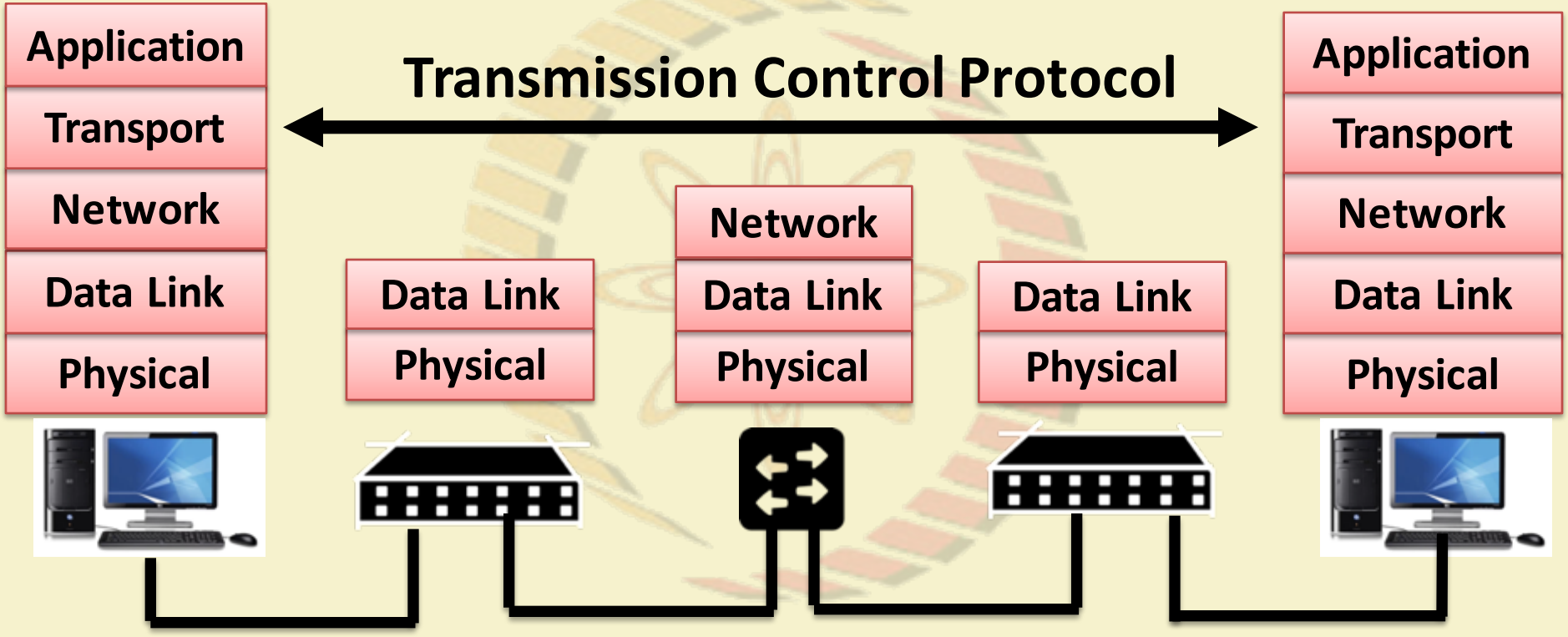
NPTEL ONLINE
CERTIFICATION COURSES

COMPUTER NETWORKS AND INTERNET PROTOCOLS

SOUMYA K GHOSH
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

SANDIP CHAKRABORTY
COMPUTER SCIENCE AND ENGINEERING,
IIT KHARAGPUR

Transmission Control Protocol I (Primitives)



Transmission Control Protocol (TCP)

- TCP was specifically designed to provide a reliable, end-to-end byte stream over an unreliable **internetwork**.

- **Internetwork** – different parts may have widely different topologies, bandwidths, delays, packet sizes and other parameters

Transmission Control Protocol (TCP)

- **TCP** dynamically adapts to properties of the internetwork and is robust in the face of many kinds of failures.

- RFC 793 (September 1981) – Base protocol
  - RFC 1122 (clarifications and bug fixes), RFC 1323 (High performance), RFC 2018 (SACK), RFC 2581 (Congestion Control), RFC 3168 (Explicit Congestion Notification)

TCP Service Model

- All TCP connections are full-diplex and point-to-point. TCP does not support multicasting or broadcasting.

- Uses **Sockets** to define an end-to-end connection (Source IP, Source Port, Source Initial Sequence Number, Destination IP, Destination Port, Destination Initial Sequence Number)

TCP Service Model

- **Unix Model of Socket Implementation:**
  - A single daemon process, called **Internet Daemon (inetd)** runs all the times at different **well known ports**, and wait for the first incoming connection
  - When a first incoming connection comes, *inetd* forks a new process and starts the corresponding daemon (for example *httpd* at port 80, *ftpd* at port 21 etc.)

TCP Service Model

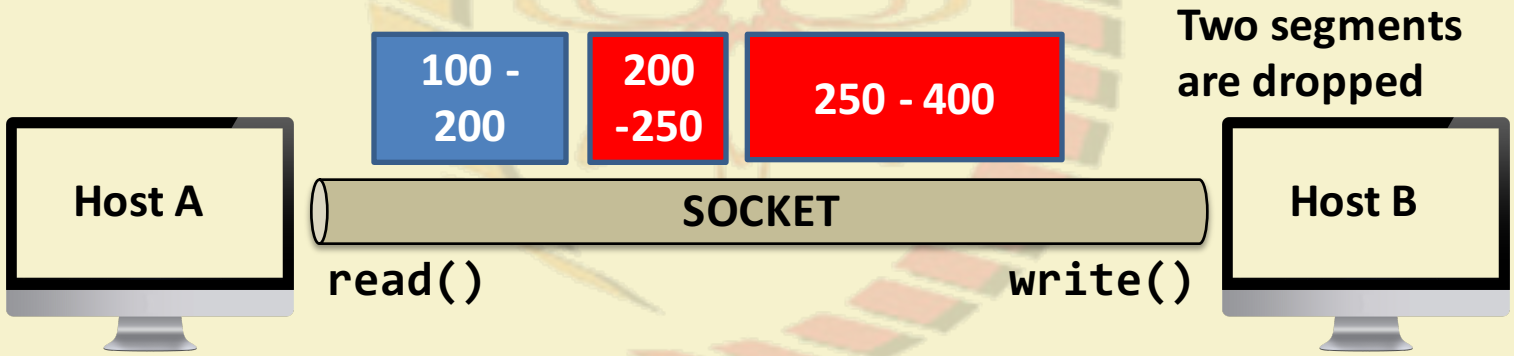
- A TCP connection is a **byte stream**, not a message stream

- Message boundaries are not preserved end-to-end

TCP Service Model

- A TCP connection is a **byte stream**, not a message stream

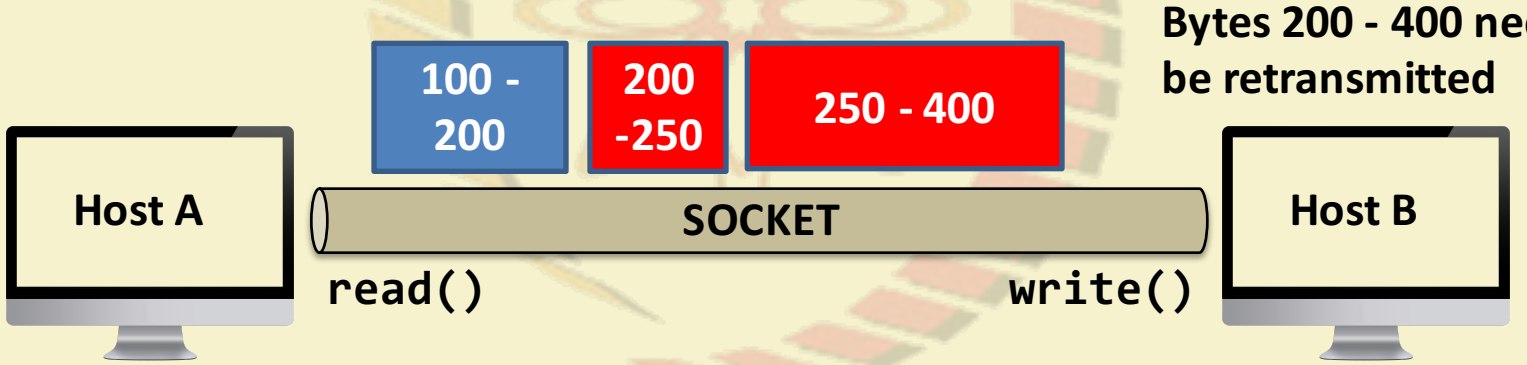
- Message boundaries are not preserved end-to-end



TCP Service Model

- A TCP connection is a **byte stream**, not a message stream

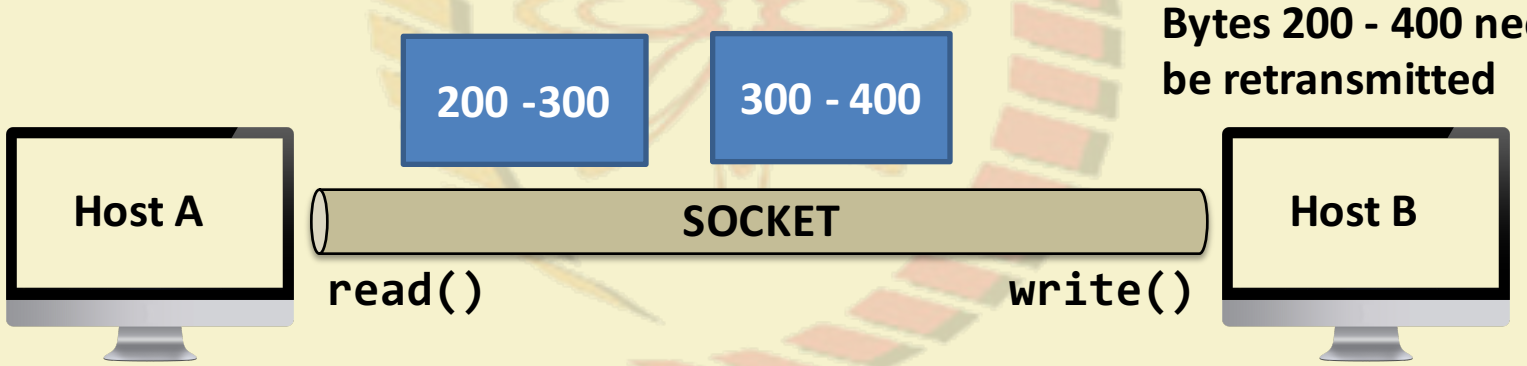
- Message boundaries are not preserved end-to-end



TCP Service Model

- A TCP connection is a **byte stream**, not a message stream

- Message boundaries are not preserved end-to-end

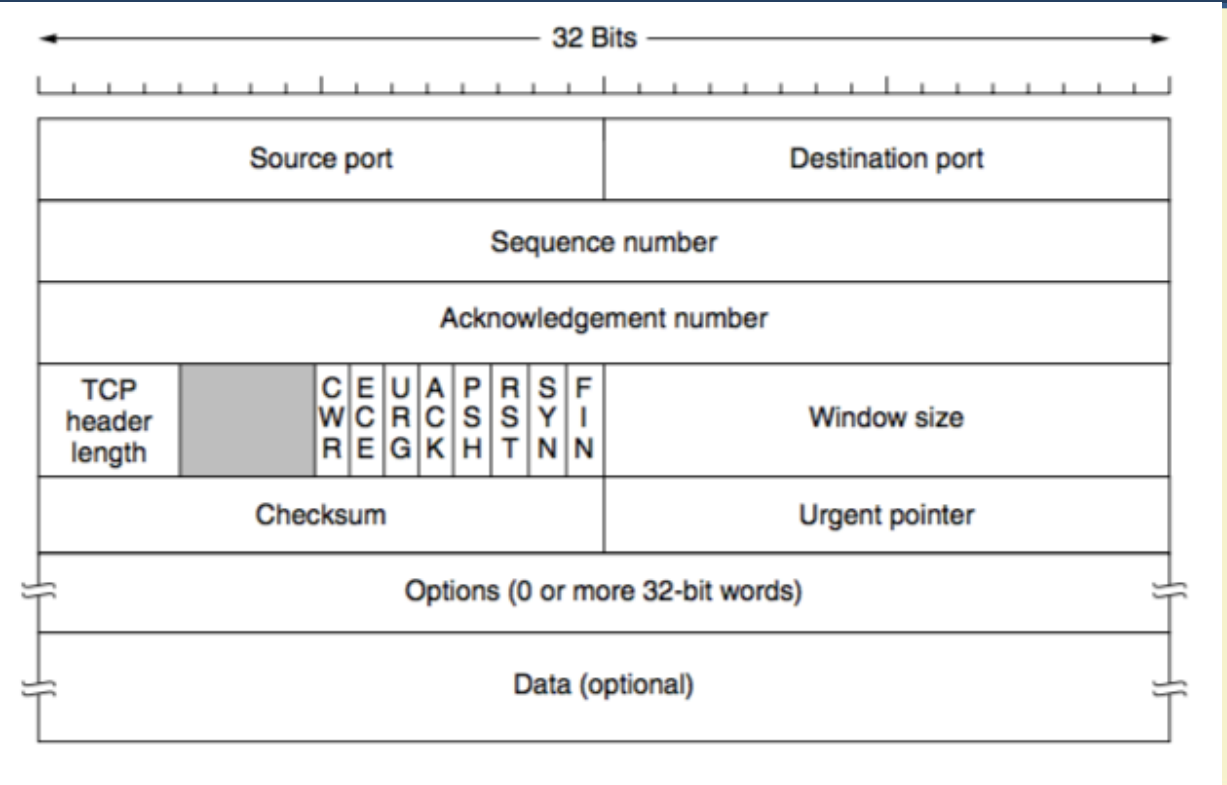


TCP Service Model

Example:

- The sending process does four 512 byte writes to a TCP stream – for `write()` call to the TCP socket
- These data may be delivered as – four 512 byte chunks, two 1024 byte chunks, one 2048 byte chunk or some other way
- There is no way for the receiver to detect the unit(s) in which the data were written by the sending process.

The TCP Protocol – The Header



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

TCP Sequence Number and Acknowledgement Number

- 32 bits sequence number and acknowledgement number
- Every byte on a TCP connection has its own 32 bit sequence number – a **byte stream** oriented connection
- TCP uses sliding window based flow control – the acknowledgement number contains next expected byte in order, which acknowledges the **cumulative bytes** that has been received by the receiver.
  - ACK number 31245 means that the receiver has correctly received up to 31244 bytes and expecting for byte 31245

TCP Segments

- The sending and receiving TCP entities exchange data in the form of **segments**.

- A TCP segment consists of a fixed 20 byte header (plus an optional part) followed by zero or more data bytes.

TCP Segments

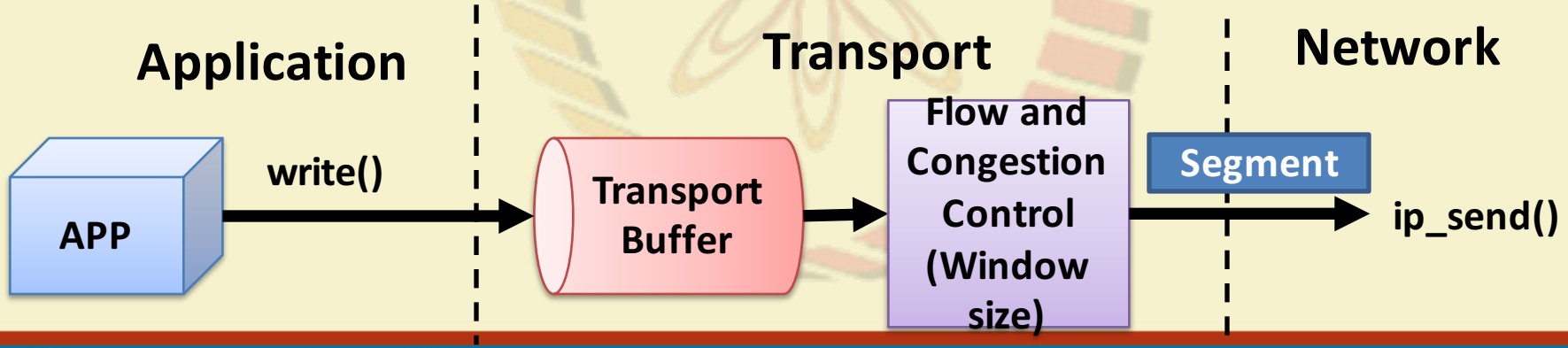
- TCP can accumulate data from several `write()` calls into one segment, or split data from one `write()` into multiple segments

- A segment size is restricted by two parameters
  - IP Payload (65515 bytes)
  - Maximum Transmission Unit (MTU) of the link

How a TCP Segment is Created

- `Write()` calls from the applications write data to the TCP sender buffer.

- Sender maintains a dynamic window size based on the flow and congestion control algorithm



How a TCP Segment is Created

- Modern implementations of TCP uses **path MTU discovery** to determine the MTU of the end-to-end path (uses ICMP protocol), and sets up the **Maximum Segment Size** (**MSS**) during connection establishment
  - May depend on other parameters (buffer implementation).

- Check the sender window after receiving an ACK. If the window size is less than MSS, construct a single segment; otherwise construct multiple segments, each equals to the MSS

Challenges in TCP Design

- Segments are constructed dynamically, so retransmissions do not guarantee the retransmission of the same data segment – a retransmission may contain additional data or less data

- Segments may arrive out-of-order. TCP receiver should handle out-of-order segments in a proper way, so that data wastage is minimized.

Window Size field in the TCP Segment Header

- Flow control in TCP is handled using a variable sized sliding window.
- The *window size* field tells how many bytes the receiver can receive based on the current free size at its buffer space.
- **What is meant by window size 0?**
- TCP Acknowledgement – combination of acknowledgement number and window size


thank you!