

# Chapter 6

## The Transport Layer

Raj Jain

Professor of CIS

The Ohio State University

Columbus, OH 43210

Jain@ACM.Org

<http://www.cis.ohio-state.edu/~jain/>

The Ohio State University

Raj Jain

12-1



- 6.4 TCP and UDP
  - Key features
  - Header format
  - Mechanisms
  - Implementation choices
  - Slow start congestion avoidance
  - TCP vs ISO TP4
  - UDP

The Ohio State University

Raj Jain

12-2

## Transport Control Protocol (TCP)

- ❑ Key Feature: Stream oriented. Not block oriented.
- ❑ Key Services:
  - ❑ Send: Please send when convenient
  - ❑ Data stream push: Please send it all now
  - ❑ Urgent data signalling: Destination TCP! please give this urgent data to the user

## TCP Header Format

Source Port	Dest Port	Seq No	Ack No	Data Offset	Resvd	Flags	Window
16	16	32	32	4	6	6	16

Check-sum	Urgent	Options	Pad
16	16	x	y

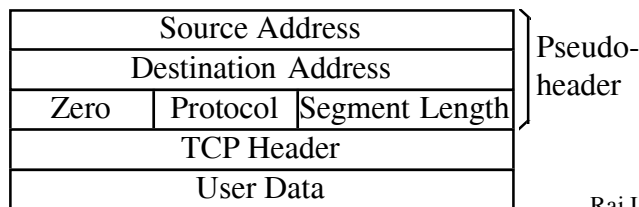
← Size in bits

## TCP Header

- ❑ Source Port (16 bits): Identifies source user process
- ❑ Destination Port (16 bits)
- ❑ Sequence Number (32 bits): Sequence number of the first byte in the segment. If syn is present, this is the initial sequence number (ISN) and the first data byte is ISN+1.
- ❑ Ack number (32 bits): Next byte expected
- ❑ Data offset (4 bits): Number of 32-bit words in the header
- ❑ Reserved (6 bits)
- ❑ Flags (6 bits): Urgent pointer field significant, ack field significant, push function, reset the connection, synchronize the sequence numbers, no more data from sender

## TCP Header (Cont)

- ❑ Window (16 bits): Will accept [Ack] to [Ack]+[window]
- ❑ Checksum (16 bits): covers the segment plus a pseudo header  
Includes the following fields from IP header: source and dest adr, protocol, segment length. Protects from IP misdelivery.
- ❑ Urgent pointer (16 bits): Points to the byte following urgent data. Lets receiver know how much urgent data is coming.
- ❑ Options (variable): Max TPDU size (Default 536 bytes)  
Window scale, SACK permitted



## TCP Service Requests

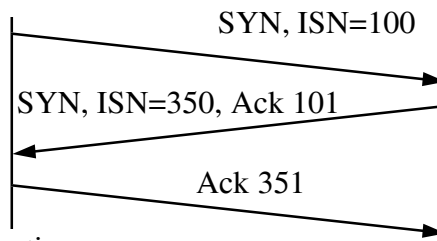
- ❑ Unspecified passive open:  
Listen for connection requests from any user
- ❑ Full passive open:  
Listen for connection requests from specified user
- ❑ Active open: Request connection
- ❑ Active open with data: Request connection and transmit data
- ❑ Send: Send data
- ❑ Allocate: Issue incremental allocation for receive data
- ❑ Close: Close the connection gracefully
- ❑ Abort: Close the connection abruptly
- ❑ Status: Report connection status

## TCP Service Responses

- ❑ Open ID: Informs the name assigned to the pending request
- ❑ Open Failure: Your open request failed
- ❑ Open Success: Your open request succeeded
- ❑ Deliver: Reports arrival of data
- ❑ Closing: Remote TCP has issued a close request
- ❑ Terminate: Connection has been terminated
- ❑ Status Response: Here is the connection status
- ❑ Error: Reports service request or internal error

## TCP Mechanisms

- ❑ Connection Establishment
  - ❑ Three way handshake
  - ❑ SYN flag set  $\Rightarrow$  Request for connection



- ❑ Connection Termination
  - ❑ Close with FIN flag set
  - ❑ Abort

## Data Transfer

- ❑ Stream: Every byte is numbered modulo  $2^{32}$ .
- ❑ Header contains the sequence number of the first byte
- ❑ Flow control: Credit = number of bytes
- ❑ Data transmitted at intervals determined by TCP
  - Push  $\Rightarrow$  Send now
- ❑ Urgent: Send this data in ordinary data stream with urgent pointer
- ❑ If TPDU not intended for this connection is received, the “reset” flag is set in the outgoing segment

## Implementation Policies (Choices)

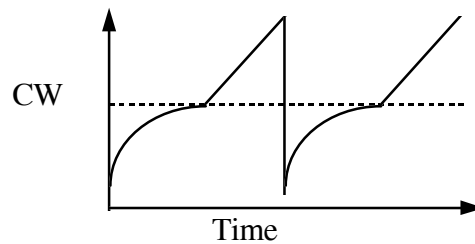
- ❑ Send Policy: Too little  $\Rightarrow$  More overhead. Too large  $\Rightarrow$  Delay  
Push  $\Rightarrow$  Send now.
- ❑ Delivery Policy: May store or deliver each in-order segment.  
Push  $\Rightarrow$  Send now.
- ❑ Accept Policy: May or May not discard out-of-order segments
- ❑ Retransmit Policy: First only
  - Retransmit all
  - Retransmit individual
  - (maintain separate timer for each segment)
- ❑ Ack Policy: Immediate (no piggybacking)  
Cumulative (wait for outgoing data or timeout)

## Slow Start Flow Control

- ❑ Window = Flow Control Avoids receiver overrun
- ❑ Need congestion control to avoid network overrun
- ❑ The sender maintains two windows: Credits from the receiver  
Congestion window from the network  
Congestion window is always less than the receiver window
- ❑ Starts with a congestion window of 1 segment (one max segment size) Do not disturb existing connections too much.
- ❑ Increase CW by 1 every time an ack is received

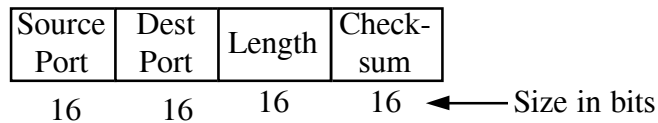
## Slow Start (Cont)

- If packets lost, remember slow start threshold to  $CW/2$   
Set  $CW$  to 1  
Increment by 1 per ack until SS threshold  
Increment by  $1/CW$  per ack afterwards



## User Datagram Protocol (UDP)

- Connectionless end-to-end service
- No flow control. No error recovery (no acks)
- Provides port addressing
- Error detection (Checksum) optional. Applies to pseudoheader (same as TCP) and UDP segment. If not used, it is set to zero.
- Used by network management



## Summary



- TCP header format and services
- TCP Streams, credit flow control, 3-way handshake
- Slow-start congestion avoidance
- UDP is connectionless and simple. No flow/error control.

## Homework

- Read section 6.4
- Solve problems 18, 19, 20, 21