

# Transport Protocols

Raj Jain  
Washington University  
Saint Louis, MO 63131  
Jain@cse.wustl.edu

These slides are available on-line at:

<http://www.cse.wustl.edu/~jain/cse473-05/>



- q TCP
  - q Key features
  - q Header format
  - q Mechanisms
  - q Implementation choices
  - q Slow start congestion avoidance

- q UDP

## TCP

- q Transmission Control Protocol
- q Key Services:
  - q Send: Please send when convenient
  - q Data stream push: Please send it all now, if possible.
  - q Urgent data signaling: Destination TCP! please give this urgent data to the user (Urgent data is delivered in sequence. Push at the source should be explicit if needed.)
  - q Note: Push has no effect on delivery. Urgent requests quick delivery

## TCP Header Format

Source Port	Dest Port	Seq No	Ack No	Data Offset	Resvd	Control	Window
16	16	32	32	4	6	6	16
Check-sum	Urgent	Options	Pad	Data			
16	16	x	y		← Size in bits		

## TCP Header

- q Source Port (16 bits): Identifies source user process
- q Destination Port (16 bits)  
20 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...
- q 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.
- q Ack number (32 bits): Next byte expected
- q Data offset (4 bits): Number of 32-bit words in the header
- q Reserved (6 bits)

## TCP Header (Cont)

- q Control (6 bits): Urgent pointer field significant,  
Ack field significant,  
Push function,  
Reset the connection,  
Synchronize the sequence numbers,  
No more data from sender



- q Window (16 bits):  
Will accept [Ack] to [Ack]+[window]-1

## TCP Header (Cont)

- q 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.
- q Urgent pointer (16 bits): Points to the byte following urgent data. Lets receiver know how much data it should deliver right away.
- q Options (variable):  
Max segment size (does not include TCP header, default 536 bytes), Window scale factor, Selective Ack permitted, Timestamp, No-Op, End-of-options

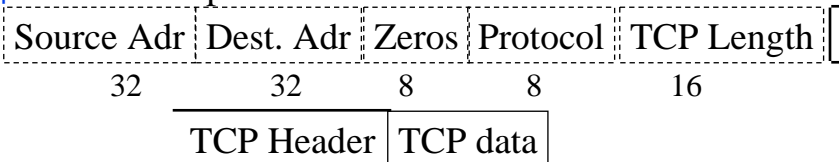
## TCP Options

Kind	Length	Meaning
0	1	End of Valid options in header
1	1	No-op
2	4	Maximum Segment Size
3	3	Window Scale Factor
8	10	Timestamp

- q End of Options: Stop looking for further option
- q No-op: Ignore this byte. Used to align the next option on a 4-byte word boundary
- q Max Segment Size (MSS): Does not include TCP header

## TCP Checksum

- q Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
- q Checksum field is filled with zeros initially
- q TCP length (in octet) is not transmitted but used in calculations.
- q Efficient implementation in RFC1071.



## 1's Complement

**2's Complement:** -ve of a number is 1+complement

- q 1 = 0001                    -1 = 1111
- q 2 = 0010                    -2 = 1110
- q 3 = 0011                    -3 = 1101

**1's complement:** -ve of a number is it's complement

- q 1 = 0001                    -1 = 1110
- q 2 = 0010                    -2 = 1101
- q 3 = 0011                    -3 = 1100

**2's Complement sum:** Add with carry

**1's complement sum:** Add. Add the carry back to the sum

- q 8+9 = 1000 + 1001 = 1 0001 => 0001 + 1 = 0010

**Complement of 1's complement sum:** 1101

**Why:** 1's complement sum is independent of the Endian-ness of the machines.

Little Endian = LSB is the left most bit.

Big Endian = MSB is the left most bit

## TCP Service Requests

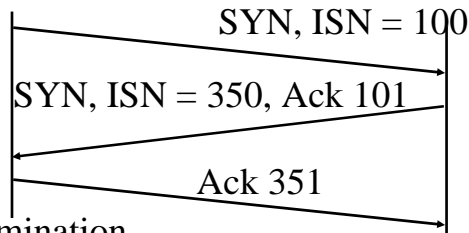
- q Unspecified passive open:  
Listen for connection requests from any user (port)
- q Full passive open:  
Listen for connection requests from specified port
- q Active open: Request connection
- q Active open with data: Request connection and transmit data
- q Send: Send data
- q Allocate: Issue incremental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

## TCP Service Responses

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

## TCP Mechanisms

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



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

## Data Transfer

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

## Implementation Policies (Choices)

- q Send Policy:
  - Too little  $\Rightarrow$  More overhead. Too large  $\Rightarrow$  Delay
  - Push  $\Rightarrow$  Send now, if possible.
- q Delivery Policy:
  - May store or deliver each in-order segment.
  - Urgent  $\Rightarrow$  Deliver now, if possible.
- q Accept Policy:
  - May or May not discard out-of-order segments

## Implementation Policies (Cont)

- q Retransmit Policy:
  - First only
  - Retransmit all
  - Retransmit individual
  - (maintain separate timer for each segment)
- q Ack Policy:
  - Immediate (no piggybacking)
  - Cumulative (wait for outgoing data or timeout)

## Slow Start Flow Control

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

Washington University in St. Louis

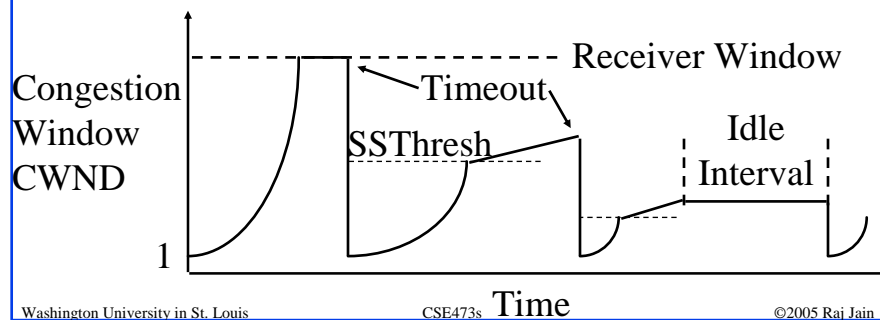
CSE473s

©2005 Raj Jain

16-17

## Slow Start (Cont)

- q If segments lost, remember slow start threshold (SSThresh) to  $CWND/2$ 
  - Set CWND to 1
  - Increment by 1 per ack until SSThresh
  - Increment by  $1/CWND$  per ack afterwards



Washington University in St. Louis

CSE473s

©2005 Raj Jain

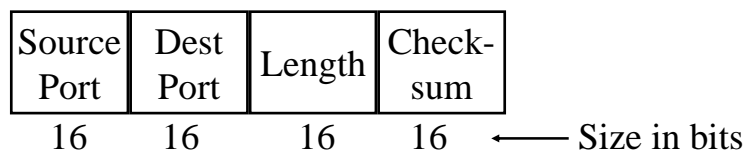
16-18

## Slow Start (Cont)

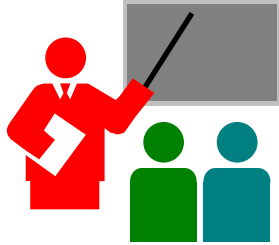
- q At the beginning,  $SSThresh = \text{Receiver window}$
- q After a long idle period (exceeding one round-trip time), reset the congestion window to one.
- q Exponential growth phase is also known as “Slow start” phase
- q The linear growth phase is known as “congestion avoidance phase”

## User Datagram Protocol (UDP)

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



## Summary



- q TCP provides reliable full-duplex connections.
- q TCP Streams, credit flow control
- q Slow-start
- q UDP is connectionless and simple.  
No flow/error control. Has error detection.

## Reading Assignment

- q Read Chapter 20 of Stallings' 7th edition

## Homework

- q Submit answer to Exercise 20.20  
This homework is worth 30 points.
- q **Exercise 20.20:** A TCP entity opens a connection and uses slow start. Approximately how many round-trip times are required before TCP can send N segments.
- q Hint: Write down what the CWND and total segments will be after 1 round trips, 2 round trips, 3 round trips, ...