Transport Protocols

Raj Jain
Professor of CIS

Raj Jain is now at
Washington University in Saint Louis
Jain@cse.wustl.edu
http://www.cse.wustl.edu/~jain/
Overview

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

- UDP
TCP

- Transmission Control Protocol

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

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest Port</th>
<th>Seq No</th>
<th>Ack No</th>
<th>Data Offset</th>
<th>Resvd</th>
<th>Control</th>
<th>Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>32</td>
<td>32</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Checksum</th>
<th>Urgent</th>
<th>Options</th>
<th>Pad</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>x</td>
<td>y</td>
<td></td>
</tr>
</tbody>
</table>

*Size in bits*
TCP Header

- Source Port (16 bits): Identifies source user process
  20 = FTP, 23 = Telnet, 53 = DNS, 80 = HTTP, ...

- 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)
TCP Header (Cont)

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

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

- 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 data it should deliver right away.

- 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

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>End of Valid options in header</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>No-op</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Maximum Segment Size</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Window Scale Factor</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>Timestamp</td>
</tr>
</tbody>
</table>

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

- 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.
- Checksum field is filled with zeros initially
- TCP length (in octet) is not transmitted but used in calculations.
- Efficient implementation in RFC1071.

<table>
<thead>
<tr>
<th>Source Adr</th>
<th>Dest. Adr</th>
<th>Zeros</th>
<th>Protocol</th>
<th>TCP Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>8</td>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

TCP Header | TCP data
TCP Service Requests

- Unspecified passive open:
  Listen for connection requests from any user (port)
- Full passive open:
  Listen for connection requests from specified port
- 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 ⇒ Request for connection

  \[\text{SYN, ISN} = 100\]
  \[\text{SYN, ISN} = 350, \text{Ack} 101\]
  \[\text{Ack} 351\]

- **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 ⇒ More overhead. Too large ⇒ Delay
  - Push ⇒ Send now, if possible.

- **Delivery Policy:**
  - May store or deliver each in-order segment.
  - Urgent ⇒ Deliver now, if possible.

- **Accept Policy:**
  - May or May not discard out-of-order segments
Implementation Policies (Cont)

- 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 (CWND) of 1 segment (one max segment size)
  ⇒ Do not disturb existing connections too much.
- Increase CWND by 1 every time an ack is received
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
Slow Start (Cont)

- At the beginning, SSThresh = Receiver window
- After a long idle period (exceeding one round-trip time), reset the congestion window to one.
- Exponential growth phase is also known as “Slow start” phase
- The linear growth phase is known as “congestion avoidance phase”
Fast Retransmit and Recovery

- If 3 duplicate acks are received for the same packet, assume that the next packet has been lost. Retransmit it right away. Retransmit only one packet.

- Helps if a single packet is lost.
  
  Does not help if multiple packets lost.

- Ref: Stevens, Internet draft
FRR (Cont)

- Upon receiving the third duplicate Ack:
  - Set SSThresh to 1/2 of current CWND
  - Retransmit the missing segment
  - Set CWND to SSthresh+3

- For each successive duplicate Ack:
  - Increment CWND by 1 MSS
  - New packets are transmitted if allowed by CWND

- Upon receiving the next (non-duplicate) Ack:
  - Set CWND to SSthresh ⇒ Enter linear growth phase

- Receiver caches out-of-order data.
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 pseudo-header (same as TCP) and UDP segment. If not used, it is set to zero.
- Used by network management

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Dest Port</th>
<th>Length</th>
<th>Checksum</th>
<th>Size in bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>
TCP provides reliable full-duplex connections.
TCP Streams, credit flow control
Slow-start, Fast retransmit/recovery
UDP is connectionless and simple. No flow/error control.
Homework

- Read RFCs 0768 (UDP) 0793 (TCP), 2001 (Slow start and FRR)
  All RFCs can be found on ftp://ftp.isi.edu/in-notes/
- Read Sections 17.2, 17.3, and 17.4 of Stallings’ sixth edition
- Submit answer to Exercise 17.19