Transport Layer V

Dmitri Loguinov
Texas A&M University

March 8, 2024
Chapter 3: Roadmap

3.1 Transport-layer services
3.2 Multiplexing and demultiplexing
3.3 Connectionless transport: UDP
3.4 Principles of reliable data transfer
3.5 Connection-oriented transport: TCP
  - Segment structure
  - Reliable data transfer
  - Flow control
  - Connection management
3.6 Principles of congestion control
3.7 TCP congestion control
TCP: Overview [RFCs: 793, 1122, 1323, 2001, 2018, 2581, 3390, 5681]

- Point-to-point (unicast):
  - One sender, one receiver
- Reliable, in-order *byte stream*:
  - Packet boundaries are not visible to the application
- Pipelined:
  - TCP congestion and flow control set window size
- Send & receive buffers

- Full duplex data:
  - Bi-directional data flow in same connection
- MSS: maximum segment size (excluding headers)
- Connection-oriented:
  - Handshaking (exchange of control msgs) initializes sender/receiver state before sending data
- Flow controlled:
  - Sender will not overwhelm receiver
Chapter 3: Roadmap

3.1 Transport-layer services
3.2 Multiplexing and demultiplexing
3.3 Connectionless transport: UDP
3.4 Principles of reliable data transfer
3.5 Connection-oriented transport: TCP
   - Segment structure
   - Reliable data transfer
   - Flow control
   - Connection management
3.6 Principles of congestion control
3.7 TCP congestion control
TCP Segment Structure

- **Sequence/ACK numbers**
  - Count *bytes*, not segments
  - ACKs piggybacked on data packets
- **Flags (U-A-P-R-S-F)**
  - Urgent data (not used)
  - ACK field is valid
  - PUSH (reduce latency)
  - RST (reset connection)
  - SYN (connection request)
  - FIN (connection close)
- **Hdr length in DWORDs (4-bit field)**
  - Normally 20 bytes, but longer if options are present

<table>
<thead>
<tr>
<th>source port #</th>
<th>dest port #</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequence number</td>
<td></td>
</tr>
<tr>
<td>acknowledgement number</td>
<td></td>
</tr>
<tr>
<td>hdr len</td>
<td>not used</td>
</tr>
<tr>
<td>receiver window</td>
<td></td>
</tr>
<tr>
<td>checksum</td>
<td>Urg data pointer</td>
</tr>
<tr>
<td>Options (variable length)</td>
<td></td>
</tr>
<tr>
<td>application data (variable length)</td>
<td></td>
</tr>
</tbody>
</table>
TCP Seq. #'s and ACKs

Seq. #'s:
- Sequence number of the first byte in segment’s data

ACKs:
- Seq # of next byte expected from sender
- Cumulative ACK

Q: how receiver handles out-of-order segments?
A: TCP spec doesn’t say, up to implementor

Simple telnet scenario

Host A
- Seq=43, ACK=80, data = ‘C’
- Seq=42, ACK=79, data = ‘C’

Host B
- Seq=79, ACK=43, data = ‘C’
- Seq=43, ACK=80

User types ‘C’

host ACKs receipt of ‘C’, echoes back ‘C’

host ACKs receipt of echoed ‘C’

time
**TCP Round Trip Time and Timeout**

**Q:** how to set TCP timeout value (RTO)?

- Want it slightly larger than the next RTT
  - But the RTT varies
- **Too short:** premature timeout
  - Unnecessary retransmissions
- **Too long:** slow reaction to segment loss
  - Protocol may stall, exhibit low performance

- **Idea:** dynamically measure RTT, average these samples, then add safety margin
- **SampleRTT:** measured time from segment transmission until ACK receipt
  - Ignore retransmissions, why?
- **SampleRTT will vary,** want estimated RTT “smoother”
  - Average several recent measurements, not just current SampleRTT
TCP Round Trip Time and Timeout

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

- Exponentially weighted moving average (EWMA)
  - Influence of past sample decreases exponentially fast
  - Typical value: \(\alpha = 1/8\)
- Task: derive a non-recursive formula for \(\text{EstimatedRTT}(n)\)
  - Assume \(\text{EstimatedRTT}(0) = \text{SampleRTT}(0)\)
  - Let \(Y(n) = \text{EstimatedRTT}(n)\) and \(y(n) = \text{SampleRTT}(n)\)

\[
Y(n) = (1 - \alpha)^n y(0) + \alpha \sum_{i=0}^{n-1} (1 - \alpha)^i y(n - i)
\]
Example RTT Estimation:

- Sampled RTT
- Estimated RTT

**Graph Details:**
- **X-axis:** Sample number
- **Y-axis:** RTT (ms)
- **Legend:**
  - Blue diamonds: Sampled RTT
  - Pink squares: Estimated RTT

---

*Example RTT Estimation:*

0 1 0 2 0 3 0 4 0 5 0 6 0 7 0 8 0 9 0

- Sampled RTT
- Estimated RTT
TCP Round Trip Time and Timeout

• Setting the timeout:
  - EstimatedRTT plus a “safety margin”
    - Larger variation in EstimatedRTT → larger safety margin
  - First estimate how much SampleRTT deviates from EstimatedRTT (typically, $\beta = 1/4$):

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

Then set retransmission timeout (RTO):

$$\text{RTO}(n) = \text{EstimatedRTT}(n) + 4\times \text{DevRTT}(n)$$
Example Timeout Estimation:

![Graph showing RTT (ms) vs sample number]

- Sampled RTT
- Estimated RTT
- Timeout

The graph illustrates the relationship between RTT (ms) and sample number, showing how timeout estimates are calculated.