Transport Layer

Outline
Reliable Byte-Stream (TCP) - continued
  Sliding Window Revisited
  Flow Control
  Adaptive Timeout

Housekeeping

• Turn in homework
• UDP Length field
  – Still a mystery
  – One speculation: it’s there to make the header length a multiple of 4
  – “Jumbo packet” extension requires setting it to 0 and calculating the UDP data size based on the IP packet length

TCP Overview

• Connection-oriented
• Byte-stream
  – app writes bytes
  – TCP sends segments
  – app reads bytes

• Full duplex
• Flow control: keep sender from overrunning receiver
• Congestion control: keep sender from overrunning network

Data Link Versus Transport

• Potentially dynamic connection to different hosts
  – need explicit connection establishment and termination
• Potentially different and varying RTT
  – need adaptive timeout mechanism
• Potentially long delay in network
  – need to be prepared for arrival of very old packets
• Potentially different capacity at destinations
  – need to accommodate different node capacity
• Potentially different and varying network capacity
  – Discover network capacity
  – need to be prepared for network congestion
Sliding Window Revisited

- **Sending side**
  - $\text{LastByteAcked} \leq \text{LastByteSent}$
  - $\text{LastByteSent} \leq \text{LastByteWritten}$
  - buffer bytes between $\text{LastByteAcked}$ and $\text{LastByteWritten}$

- **Receiving side**
  - $\text{LastByteRead} < \text{NextByteExpected}$
  - $\text{NextByteExpected} \leq \text{LastByteRcvd} + 1$
  - buffer bytes between $\text{NextByteRead}$ and $\text{LastByteRcvd}$

Flow Control

- Send buffer size: $\text{MaxSendBuffer}$
- Receive buffer size: $\text{MaxRcvBuffer}$
- Receiving side
  - $\text{LastByteRcvd} - \text{LastByteRead} \leq \text{MaxRcvBuffer}$
  - $\text{AdvertisedWindow} = \text{MaxRcvBuffer} - (\text{NextByteExpected} - \text{NextByteRead})$
- Sending side
  - $\text{LastByteSent} - \text{LastByteAcked} \leq \text{AdvertisedWindow}$
  - $\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSent} - \text{LastByteAcked})$
  - $\text{LastByteWritten} - \text{LastByteAcked} \leq \text{MaxSendBuffer}$
  - block sender if $(\text{LastByteWritten} - \text{LastByteAcked}) + y > \text{MaxSenderBuffer}$
- Always send ACK in response to arriving data segment
- Sender persists when $\text{AdvertisedWindow} = 0$

Protection Against Wrap Around

- 32-bit $\text{SequenceNum}$

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Time Until Wrap Around</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 (1.5 Mbps)</td>
<td>6.4 hours</td>
</tr>
<tr>
<td>Ethernet (10 Mbps)</td>
<td>57 minutes</td>
</tr>
<tr>
<td>T3 (45 Mbps)</td>
<td>13 minutes</td>
</tr>
<tr>
<td>FDDI (100 Mbps)</td>
<td>6 minutes</td>
</tr>
<tr>
<td>STS-3 (155 Mbps)</td>
<td>4 minutes</td>
</tr>
<tr>
<td>STS-12 (622 Mbps)</td>
<td>55 seconds</td>
</tr>
<tr>
<td>STS-24 (1.2 Gbps)</td>
<td>28 seconds</td>
</tr>
</tbody>
</table>

Keeping the Pipe Full

- Limits of 16-bit $\text{AdvertisedWindow}$: 100ms RTT

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Delay x Bandwidth Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 (1.5 Mbps)</td>
<td>18KB</td>
</tr>
<tr>
<td>Ethernet (10 Mbps)</td>
<td>122KB</td>
</tr>
<tr>
<td>T3 (45 Mbps)</td>
<td>549KB</td>
</tr>
<tr>
<td>FDDI (100 Mbps)</td>
<td>1.2MB</td>
</tr>
<tr>
<td>STS-3 (155 Mbps)</td>
<td>1.8MB</td>
</tr>
<tr>
<td>STS-12 (622 Mbps)</td>
<td>7.4MB</td>
</tr>
<tr>
<td>STS-24 (1.2 Gbps)</td>
<td>14.8MB</td>
</tr>
</tbody>
</table>
TCP Extensions – RFC 1323

- Implemented as header options
- Store timestamp in outgoing segments
- Extend sequence space with 32-bit timestamp (PAWS)
- Shift (scale) advertised window

Adaptive Retransmission (Original Algorithm)

- Measure SampleRTT for each segment/ ACK pair
- Compute weighted average of RTT
  - \( \text{EstRTT} = \alpha \times \text{EstRTT} + \beta \times \text{SampleRTT} \)
  - where \( \alpha + \beta = 1 \)
  - \( \alpha \) between 0.8 and 0.9
  - \( \beta \) between 0.1 and 0.2
- Set timeout based on EstRTT
  - \( \text{TimeOut} = 2 \times \text{EstRTT} \)

Karn/Partridge Algorithm

- Do not sample RTT when retransmitting
- Double timeout after each retransmission

Jacobson/ Karels Algorithm

- New Calculations for average RTT
- \( \text{Diff} = \text{SampleRTT} - \text{EstRTT} \)
- \( \text{EstRTT} = \text{EstRTT} + (\delta \times \text{Diff}) \)
- \( \text{Dev} = \text{Dev} + \delta( |\text{Diff}| - \text{Dev}) \)
  - where \( \delta \) is a factor between 0 and 1
- Consider variance when setting timeout value
- \( \text{TimeOut} = \mu \times \text{EstRTT} + \phi \times \text{Dev} \)
  - where \( \mu = 1 \) and \( \phi = 4 \)
- Notes
  - algorithm only as good as granularity of clock (500ms on Unix ??)
  - accurate timeout mechanism important to congestion control (later)