TCP (Part III: Misc)

Overview

- TCP Persist and Keepalive timers
- Silly window syndrome
- Path MTU
- Window Scale Factor
- Timestamp option
- T/TCP: TCP for transactions

Ref: Chap 22, 23, 24; RFC 1323

TCP Persist Timer

- Receiver flow control can set window to zero
- Receiver later sends “window update acks”
- But TCP does not transmit acks reliably => update acks may be lost and source may be stuck at a zero window value
- TCP uses persist timer to query the receiver periodically to find if the window has been increased.
- Persist timer always bounded between 5s and 60s. It does exponential backoff like other timers too.
**Silly Window Syndrome**
- A) The system operates at a small window (sends segments which are not MSS-sized) even if the receiver grants a large window.
- B) Receiver advertises small windows.
- Solution: batching
  - Receiver must not advertise small windows
  - Sender waits until segment full before sending (extension of Nagle’s algo).
  - It can transmit everything if it is not waiting for any ACK (or if Nagle’s algo has been disabled).

**TCP Keepalive timer**
- Optional timer.
- Not part of TCP spec, but found in most implementations.
  - Not necessary, because “connection” defined by endpoints.
  - Connection can be “up” as long as source/destination “up”.
- Typical use: to detect idle clients or half-open connections and de-allocate server resources tied up to them. Eg: telnet, ftp.

**Path MTU discovery**
- Assume MSS = Min (local MTU - headers, destination MSS). Set DF bit.
- If ICMP error, reduce segment size and retransmit.
- Since routes change dynamically, a larger value can be tried again after a time interval (RFC 1191 recommends 10 min, but Solaris uses 30 s).
Gigabit Networks

- "Higher Bandwidth Networks"
- Propagation latency unchanged.
- Increasing bandwidth from 1.5Mb/s to 45 Mb/s (factor of 29) decreases file transfer time of 1MB by a factor of 25.
- But, increasing from 1 Gb/s to 2 Gb/s gives an improvement of only 10%!
- Transfer time = propagation time + transmission time + queueing/processing.
- Design networks to minimize delay (queueing, processing, reduce retransmission latency)

Window Scaling Option

- Long Fat Pipe Networks (LFN): Satellite links
- Need very large window sizes.
- Normally, Max window = $2^{16} = 64$ KBytes
- Window scale: Window = $W \times 2^{\text{Scale}}$

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

- Max window = $2^{16} \times 2^{255}$
- Option sent only in SYN and SYN+Ack segments.
- RFC 1323

Timestamp option

- For LFNs, need accurate and more frequent RTT estimates.
- Timestamp option:
  - Place a timestamp value in any segment.
  - Receiver echoes timestamp value in ack
  - If acks are delayed, the timestamp value returned corresponds to the earliest segment being acked.
  - Segments lost/retransmitted => RTT overestimated
**PAWS: Protection against wrapped sequence numbers**
- Largest receiver window = $2^{30} = 1$ GB
- "Lost" segment may reappear before MSL, and the sequence numbers may have wrapped around
- The receiver considers the timestamp as an extension of the sequence number => discard out-of-sequence segment based on both seq # and timestamp.
- Reqt: timestamp values need to be monotonically increasing, and need to increase by at least one per window

**T/TCP: Transaction Oriented TCP**
- Three-way handshake ⇒ Long delays for transaction-oriented applications.
- T/TCP extension avoids 3-way handshakes
- Request/reply data sent with connection messages
- Server caches a connection count (CC) per-client to detect duplicate requests and avoid replaying transaction
- TIME_WAIT is shortened by setting it to $8\times RTO$
- Latency = RTT + server processing time (SPT)

**Summary**
- Persist and keepalive timers, silly window avoidance
- Enhancements for LFNs: window scale option, timestamp option, PAWS
- T/TCP extension to TCP for transactions