Transparent TCP Timestamps

draft-scheffenegger-tcpm-timestamp-negotiation-01

Richard Scheffenegger
[rs@netapp.com]

Mirja Kühlewind
[mirja.kuehlewind@ikr.uni-stuttgart.de]
Agenda

Timestamps revisited
Problem statement
Use case examples
  One way delay (variation)
  TS+SACK synergy
    Early spurious retransmission detection
    Early lost retransmission detection
  TS integrity
Explicit signaling
TCP Timestamp Option

<table>
<thead>
<tr>
<th>Kind=8</th>
<th>10</th>
<th>TS Value (TSval)</th>
<th>TS Echo Reply (TSecr)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

- **Sender** sets current timestamp in **TSval**

- **Receiver** echos the opaque **TSval** field in **TSecr** of <ACK> and provides an own timestamp **TSval** on sending of the acknowledgement

- **Round-Trip Time** (specified in RFC1323):
  \[ RTT = \text{curr\_time}() - TSecr \]

- **Unless reordering / loss is detected**

- **Receiver**: **PAWS Test** (imposes some restrictions)
Challenges

- TCP Timestamp Option does not ensure certain resolution
  
  "The timestamp value to be sent in TSval is to be obtained from a (virtual) clock that we call the "timestamp clock". Its values must be at least approximately proportional to real time, in order to measure actual RTT."

- But in fact the receiver is just supposed to echo whatever is written in the TSval field

- Cases when more than one timestamp is available to echo (delayed ACK)

- Special treatment by receiver during loss / reorder events
Problem statement

- RFC1323 gives little guidance for timestamps

- New congestion control schemes (LEDBAT, TCP-RAPID, TCP-LP) require one-way delay (variation) as input
  - One-Way-Delay estimate: OWD = TSecr – TSval

- RFC1323 too restrictive to allow additional use

- Entire timestamps opaque to opposite host

Proposed Solution

Negotiate the sender and receiver TS capabilities
Use Case 1: OWD for Congestion Control

- One-way delay estimate
  \[ C(t) = T\text{Secr}(t) - T\text{Sval}(t) \]
- Increase of one-way delay is a sign for congestion
- Monitoring of one-way delay variation relative to an previous measurement
  \[ V(t) = C(t) - C(t-n) \]

Problems

- remote timestamp clock rate is unknown
  - can be learnt if clock rate is related to a real clock
  - network conditions don‘t change
  - whole T\text{Sval} field is used for a timestamp
- Delayed ACKs: OWD measurement includes delay outside the network
Use case 2: TS+SACK synergy

- Receivers echo TS of last in-sequence, unacked segment

Problems:
- Overly conservative if SACK is also enabled
- Delayed ACK behavior impacts sender RTTM calculation
Use case 3: Timestamp integrity

- Use of transparent TS value for CC is creating incentive for malicious receivers to meddle with TSecr value (ie. early versions of Linux BIC, CUBIC)

- Current approach:
  - Use a limited number of LSB bits in TSval to (secure) fingerprint the value (limited by TSval constraints)
  - Sender tracks RTTM independent of TSecr (per-segment state kept)

- Proposed solution:
  - Announce the number of opaque LSB bits in TSval
  - Exclude opaque bits in receiver-side calculations (ie. PAWS)
  - breaking strict monoton increasing values
    - only required for transparent part of Tsval
    - better fingerprinting possible (less constraint)
  - No per-segment state on sender side
Explicit signaling of TS capabilities

- Use TSecr in <SYN> to signal local capabilities
  - Update to RFC1323

- In <SYN,ACK> need to XOR received TSval and local capabilities
  - Minimal state required in sender during handshake
  - Interaction with TCP Cookies / TCPCT

- Enable direct mirroring of TSval when SACK is also negotiated (supported by both)

- Allows further research opportunities
Proposed TS capabilities

- **MSB**: always 1 to signify TS capabilities field
  - enable direct echo of TSval if SACK is also enabled
- **Ver(sion)**: must be 0
  - future use
- **Reserved**: must be 0
- **Mask**: # of LSBs for opaque use
  - secure hash
  - slow running TS clocks
- **S, Exp16, Frac16**: TS clock rate
  - range between ~16s ... 8ns (8ps with reduced precision)
Transparent TCP Timestamps

Thank you for your attention!

Questions?
Backup Slides
Early spurious retransmission detection

- Based on TSecr aka Eifel detection (RFC3522)
- Requires different timestamp for retransmitted segment than original segment
  - Doesn’t work if TS clock slower than ~RTT
  - Only works if first segment is delayed

- Senders using „slow“ TS clocks could use opaque masked least significant bits to differentiate retransmissions
Early lost retransmission detection

- SACK requires new segments to detect lost retransmissions
  - Unknown if SACKed segment is delayed original or retransmission

- Direct echo of TS would allow disambiguation