TCP and SCTP RTO Restart

draft-hurtig-tcpm-rtorestart-02

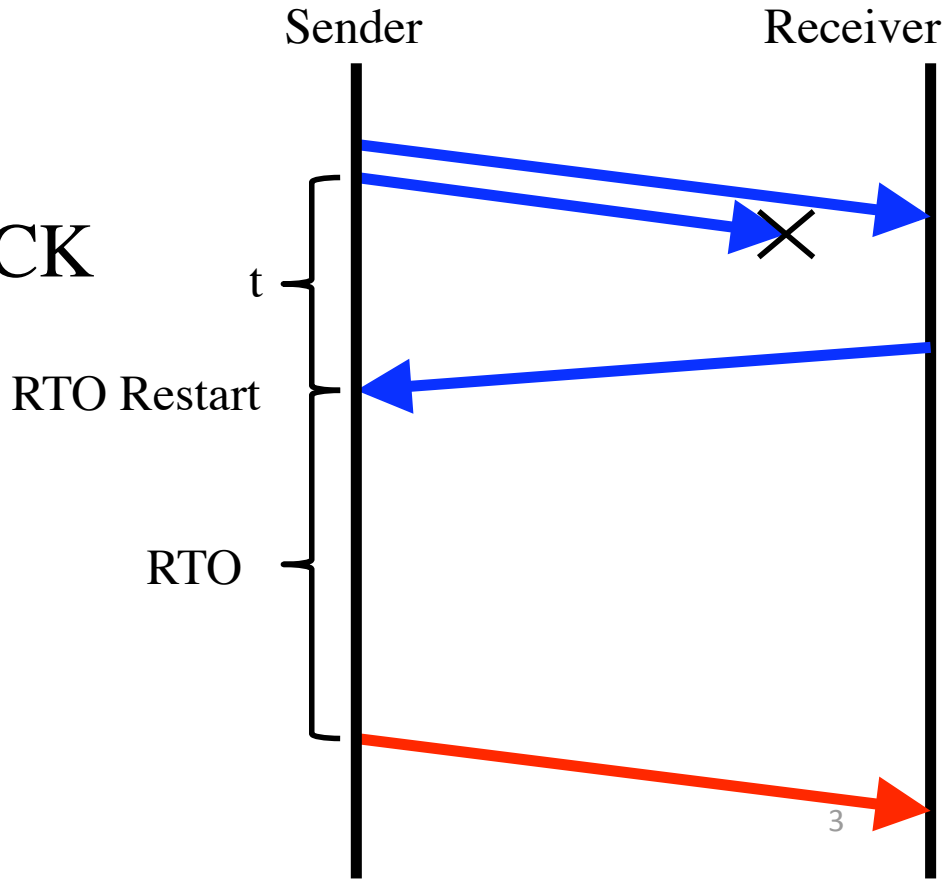
Michael Welzl
michawe@ifi.uio.no

Motivation

- In some cases TCP/SCTP must use RTO for loss recovery
  - e.g., if a connection has 2 outstanding packets and 1 is lost
- Some solutions exist, but they are not always applicable
  - Limited Transmit (RFC 3042)
    - requires: unsent data, no ack loss
  - Early Retransmit (RFC 5827)
    - requires: 2 outstanding segments, no ack loss, no reordering

Motivation

- Thus, some flows have to use RTO for loss recovery
- However, the effective RTO often becomes RTO = RTO + t
  - Where t ≈ RTT [+delACK]
- The reason is that the timer is restarted on each incoming ACK (RFC 6298, RFC 4960)



Impact

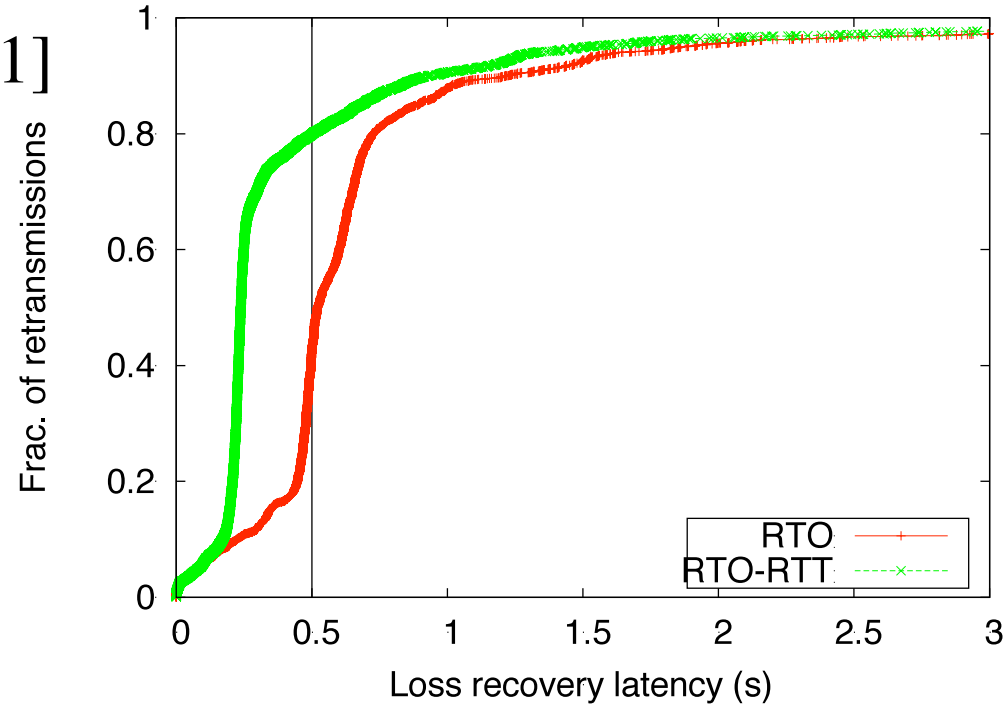
- Standard approach no problem when congestion window is large
- Actually, it can be beneficial
  - lower risk for spurious RTOs
  - gives FR more time to detect loss
    - smaller congestion window reduction using FR
- This is not the case for short-lived/thin flows
  - congestion window low anyhow

TCP and SCTP RTO Restart

- To allow retransmissions after exactly RTO seconds, the timer is restarted as:
  - RTO = RTO - t
- The modified restart is only used when
  - the number of outstanding segments < 4;
  - and there is no unsent data ready for transmission.
- Thus, only flows incapable of FR can use the modified RTO restart

Faster Recovery Needed?

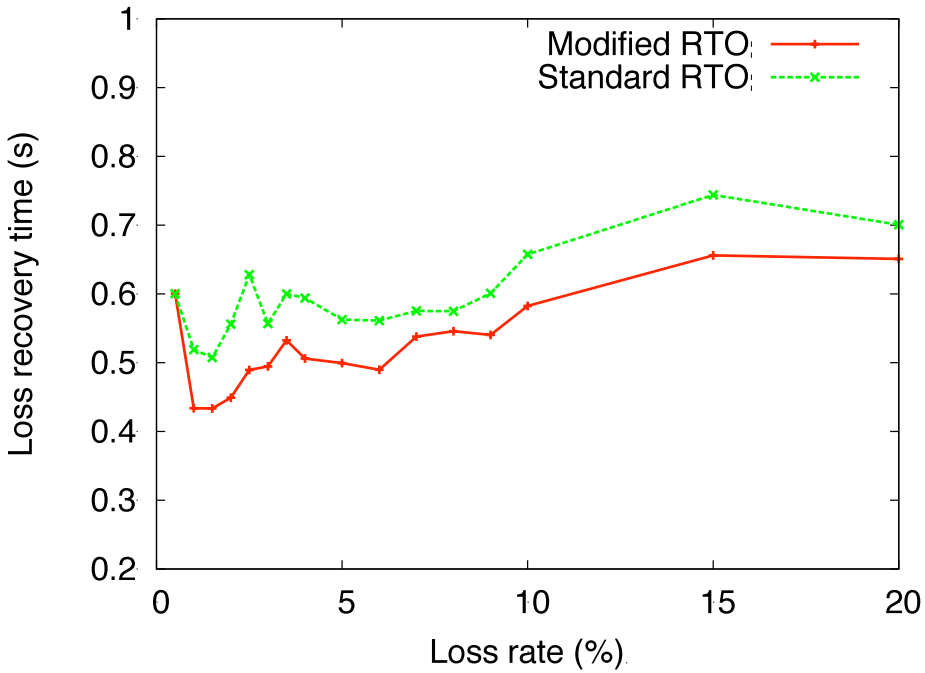
- One extra RTT could lead to performance problems for short-lived (e.g. web) and thin streams
  - Thin streams are flows that only use a fraction of the available bandwidth (e.g. signaling, online games, chat, VoIP, …)
  - IETF 78: http://www.ietf.org/proceedings/78/slides/iccrg-4.pdf
- Example: Anarchy Online [1]
  - Approx. 1% packet loss
  - Most loss recovered using RTOs
  - Maximum tolerable latency about 500 msec [2]



[1] A. Petlund, P. Halvorsen, P. F. Hansen, T. Lindgren, R. Casais, C. Griwodz "Network Traffic from Anarchy Online: Analysis, Statistics and Applications", In Proc of ACM MMSys, February 2012.
[2] M. Claypool and K. Claypool, "Latency and Player Actions in Online Games", In Communications of the ACM, November 2006.

Performance

- Initial simulations
  - Ns-3 (with real Linux TCP)
  - Short-lived flows
  - Multiple clients served by one host
  - Large set of bw's and delays
- Results show that
  - Loss recovery times are reduced with approximately 1 RTT on average
  - The amount of spurious RTOs is slightly higher than for regular TCP (<1% more)
- New experiments underway
  - Congestion losses
  - New RTO management alg.
  - To investigate burst situations more thoroughly



Results from 200 concurrent flows with 100 ms RTT

Changes between -01 and -02

- Smaller text changes
- No longer a requirement to store the transmission time of each segment
  - Sufficient to “remember” only the last four

Open issues and possible solutions

- Increased aggressiveness
  - Might trigger spurious RTOs when bursts are sent
- Possible mitigations
  - Careful version of the algorithm
    - Disables modified restart during bursty transmission
  - noRestart approach (suggested by Mark Allman)
    - Don't restart the timer if no data is available for transmission and less than four segments is outstanding
    - Same effect as modified restart for small windows
    - More conservative for larger windows