RACK: a time-based fast loss recovery

Draft-ietf-tcpm-rack-03 updates

Yuchung Cheng
Neal Cardwell
Nandita Dukkipati
Priyaranjan Jha
Google

tcpm IETF 101, March 2018
What's RACK (Recent ACK)?

Time-based loss inferences instead packet or sequence counting

Conceptually...
- Every sent packet has a timer
- All timers are constantly adjusted based on most recent RTT sample
- A packet is retransmitted after RTT + reoWnd

- RACK is about implementing this w/ one timer per connection and ACK events

Expect ACK of P1 by then … wait RTT/4 in case P1 was reordered
Tail Loss Probe (TLP)

- **Problem:**
  - Tail drops are common on request/response traffic
  - Tail drops lead to timeouts, which are often 10x longer than fast recovery
  - 70% of losses on Google.com recovered via timeouts before TLP was deployed

- **Goal:**
  - Reduce tail latency of request/response transactions

- **Approach:**
  - Convert RTOs to fast recovery
  - Solicit a DUPACK by retransmitting the last packet in 2 SRTTs
  - Requires RACK to trigger fast recovery

After 2 SRTTs... send TLP to get SACK to start RACK recovery of a tail loss
Updates since IETF 100

- What’s new in [draft-ietf-tcpm-rack-03](https://datatracker.ietf.org/doc/draft-ietf-tcpm-rack-03):
  - Dynamic reordering window
  - DUPACK-threshold mode
  - Fast implementation example
  - Congestion control interactions
  - Cosmetic changes

- Deployment
  - RACK/TLP has now entirely replaced [RFC6675](https://tools.ietf.org/html/rfc6675) recovery in the latest Google/YouTube server TCP
    - Previously both [RFC6675](https://tools.ietf.org/html/rfc6675) and RACK/TLP were enabled to detect losses
Dynamic reordering window

- Previous RACK: \( reo_{\text{wnd}} = \text{min}_{\text{RTT}}/4 \)
  - Spurious loss recoveries when reordering degree > \( reo_{\text{wnd}} \)

- Initial idea: precisely measure reordering degree in time
  - Complex
  - Requires remembering per-packet timestamp after the packet is ACKed and deallocated in the stack

- New idea: dynamically adapt \( reo_{\text{wnd}} \) using Duplicate SACK (DSACK; [RFC2883](https://tools.ietf.org/html/rfc2883))
  - DSACK signals [spurious retransmission](https://tools.ietf.org/html/rfc2883) and implies \( reo_{\text{wnd}} \) is too small
  - Increase \( reo_{\text{wnd}} \) on DSACKs
  - Decrease \( reo_{\text{wnd}} \) gradually if no DSACKs
  - DSACK is supported by Linux, iOS, MacOS, and Windows
Dynamic reordering window details

Init: \( \text{reo}_\text{wnd} = \frac{\text{min}_\text{RTT}}{4} \)

For every round trip that receives some ACKs with DSACK option
  \( \text{reo}_\text{wnd} += \frac{\text{min}_\text{RTT}}{4} \)

After 16 loss recoveries without observing more DSACK options, reset state
  \( \text{reo}_\text{wnd} = \frac{\text{min}_\text{RTT}}{4} \)

Temporarily set \( \text{reo}_\text{wnd} = 0 \) during loss recovery for prompt repair

Always cap \( \text{reo}_\text{wnd} \) by SRTT (smoothed RTT from RFC6298)
Dynamic reordering window

Old: static reo_wnd (draft -02)

New: adaptive reo_wnd (draft -03)
DUPACK-threshold emulation mode

DUPACK-threshold is useful with ultra-low RTTs (when RACK timer tick slower than RTT)

New: RACK support for DUPACK-threshold

If #DUPACKs >= 3, Then reo_wnd = 0

Subtle differences between RFC6675 and RACK:

1. RFC6675: a packet is lost when >=3 packets are SACKed and have higher sequence
2. RACK: a packet is lost when >=3 packets are SACKed and at least one has higher sequence
3. Example: send 10 packets, and packets 3, 5, 7 are SACKed
   RFC6675: packets 1, 2 lost
   RACK: packets 1, 2, 4, 6 lost
Interaction with congestion control

Potential burst interaction with Reno congestion control (i.e. RFC5681)

a. On a single ACK, RACK could mark a large number of packets lost
b. Inflight (aka pipe) drops suddenly
c. TCP retransmission bursts (cwnd - inflight) == (ssthresh - inflight)
d. Causes more drops

Recommendation

e. Use Proportional Rate Reduction [RFC6937] to pace via packet conservation or slow start
   i. Also helpful: a rate-based TCP pacing mechanism (e.g. Linux fq/pacing)
Conclusion

The development of RACK is near the end

1. Linux/FreeBSD/Windows support RACK
2. Authors consider draft-03 as complete and ready for final review

Questions? Concerns?