draft-ietf-tcpm-rack WGLC plan

{ycheng, ncardwell}@google.com
tcpm IETF 4/29/30

Protocol: Lost-retransmit bug in pseudocode

7.1. Transmitting a data packet

Upon transmitting a new packet or retransmitting an old packet, record the time in Packet.xmit_ts and set Packet.lost to FALSE. Upon retransmitting a packet, set Packet.retransmitted to TRUE. ~~RACK does not care if the transmission is triggered by an ACK, new application data, an RTO, or any other means.~~

```
RACK_transmit_data(Packet):
    Packet.time_ts = Now()
    Packet.lost = FALSE

RACK_retransmit_data(Packet)
    Packet.retransmitted = TRUE
    RACK_transmit_data(Packet)
```

Retransmitted packets reset Packet.lost upon next (re)transmit

MAY apply RACK in RTO recovery

The text describing how RACK applies to RTO recovery isn't clear. We propose adding text describing a new "MAY" optimization.

Overview Section:

In addition to fast recovery, RACK improves the accuracy of timeout (RTO) recovery as well. Without RACK, when an RTO expires traditional TCP senders mark all the unacknowledged packets as lost. For example, consider a simple case where one packet was sent with an RTO of 1 second. Then the application writes more data and the second packet is sent right before the RTO of the first packet expires. When the RTO expires, it'll mark both packets lost. But in this case the second packet is likely still in flight. Since RACK checks the transmission time of each packet, it reduces the risk of spuriously marking packets as lost.

New subsection to cover RACK-based RTO recovery:

```
RACK_detect_loss_RTO():
    RACK.reo_wnd = RACK_update_reo_wnd()
    For each packet, Packet, not acknowledged yet:
        If Packet.lost is TRUE AND Packet.retransmitted is FALSE:
            Continue /* Packet lost but not yet retransmitted */

        If RACK_sent_after(RACK.xmit_ts, RACK.end_seq, Packet.xmit_ts, Packet.end_seq) AND
           Packet.xmit_ts + RACK.rtt + RACK.reo_wnd <= Now():
                Packet.lost = TRUE
```

Protocol: reordering timer

For timely loss detection, the sender ~~MAY~~ SHOULD install a "reordering settling" timer. The timer expires as soon as ~~any~~ every unacknowledged packet within the reordering window is deemed lost.

Rationale:
SHOULD: data shows the reordering timer improves performance
every: less frequent timer expiration

Writing: major additions / removals

- Make TLP more integrated instead of an appendix/add-on -- mention TLP in abstract, intro., and overview

- Overview / Design sections
  - More detail about RACK/TLP protocol in Overview
  - Explain how RACK works for RTO recovery

- A new table of contents

- Remove Section 9: Experiments and Performance Evaluations

Writing: further clarifications needed

- The authors by no means intend to encourage more packet reordering
- TLP, RACK, SACK, DSACK, Limited-Transmit (LT) relationships
  - RACK requires SACK, TLP requires RACK
  - DSACK, LT are RECOMMENDED but not required
- TCP Segmentation Offload handling
- RACK timer cancellation
- RACK reorder window initiation and DUPTHRESH interaction

Writing: cut unsubstantiated or unclear text

- “Therefore their main constraint on speed is reordering, and there is pressure to relax that constraint. If RACK becomes widely deployed, the underlying networks **may** introduce more reordering for higher throughput.”

- “[RFC4653][RFC5827][RFC5681][RFC6675][RFC7765][FACK][THIN-STREAM], we've found that together they do not perform well…. And under these conditions none of them can detect lost retransmissions well. Also, these algorithms, including RFCs, rarely address the interactions with other algorithms. “

- “Using a threshold for counting duplicate acknowledgments (i.e., DupThresh) alone is no longer reliable because of today's prevalent reordering patterns. “

- “Today's prevalent lost retransmissions also cause problems with packet-counting approaches“

- “While RACK can be a supplemental loss detection mechanism on top of these algorithms, this is not necessary, because RACK implicitly subsumes most of them.”

- ...

Thanks to many reviewers (not a complete list):

Mirja Kuehlewind, Gorry Fairhurst, Theresa Enghardt, Ilpo Järvinen, Praveen Balasubramanian, Martin Duke, Yi Huang, TCPM-chairs@