Proportional Rate Reduction for TCP

draft-ietf-tcpm-prr-rfc6937bis-10

IETF 120 tcpm Jul 2024

Matt Mathis <ietf@mattmathis.net>
Nandita Dukkipati <nanditad@google.com>
Yuchung Cheng <ycheng@google.com>
Neal Cardwell <ncardwell@google.com> [presenting]

RFC 6937: Proportional Rate Reduction for TCP

- Published in 2013 as an experimental RFC
  - At the time, only implemented by Linux; implemented without RFC 8985 (RACK-TLP)

- A "**mini** congestion control" during fast recovery
  - Send packets at the ratio of CC's cwnd reduction
    - Reno: 0.5, Cubic: 0.7
  - If in-flight data drops below ssthresh
    - Mode SSRB: slow-start
    - Mode CRB: packet conservation
  - Implementation has to pick SSRB or CRB option

- Interesting fact: for a flow that mostly operates in fast recovery, its cwnd is mostly controlled by RFC 6937 instead of Cubic/CC
  - e.g. policed video streaming flows using Cubic

RFC 6937-bis

10 years after RFC 6937, in 2021

- PRR is default-enabled in Linux, FreeBSD, NetFlix-BSD/RACK, MS Windows TCP
- tcpm voted to revise and publish as a standard RFC

Current draft is: draft-ietf-tcpm-prr-rfc6937bis

Revisions since RFC 6937 that were previously described at IETF meetings:

- Algorithm refinements
  - Merge SSRB and CRB modes: automatically choose the mode based on if the last ACK indicates further losses // bis-01
  - Force a fast retransmit on entering recovery to maintain ACK clock // bis-02
  - Non-SACK support // bis-03
  - Streamline the sending process if experienced higher network reordering previously // bis-04

- Editorial clarifications
  - Do not slow start on ACKs that trigger RFC 6675 “last resort” retransmission as it may indicate further losses
  - Relationships with RFC 6675 (pipe), RFC 8985 (RACK-TLP)
  - Removed experiments section

RFC 6937-bis: changes between 08 and 10

Most changes were in response to comments from Markku Kojo on TCPM list (many thanks for the comments!)

Here are the specific changes since the last PRR presentation, IETF 119 in Mar 2024, version 08:

text diff between version 08 and version 10

The following slides summarize the main substantive differences between revisions 08 and 10

Starting PRR

- Clarified PRR triggers initialization based on start of **congestion control** reduction, not **loss recovery**
  - Since congestion control **might** reduce ssthresh for each round trip with new losses in recovery

New text:

```
At the beginning of a congestion control response episode initiated by the congestion control
algorithm, a TCP data sender using PRR MUST initialize the PRR state.

The timing of the start of a congestion control response episode is entirely up to the
congestion control algorithm, and (for example) could correspond to the start of a fast recovery
episode, or a once-per-round-trip reduction when lost retransmits or lost original transmissions
are detected after fast recovery is already in progress.
```

RecoverFS Initialization

- Refined the initialization of RecoverFS
  - No longer uses pipe, since pipe can be incorrect with spurious loss detection decisions
  - Incorporates data SACKed before entering recovery
  - Incorporates data cumulatively ACKed upon entering Fast Recovery (a rare corner case)

New text:

```
RecoverFS: The "recovery flight size", the number of bytes the sender estimates are in flight in
the network upon entering fast recovery. PRR uses RecoverFS to compute a smooth sending rate.
Upon entering fast recovery, PRR initializes RecoverFS to the sender's best estimate of the
number of bytes outstanding in the network; for connections with SACK this is typically "pipe"
as specified in RFC 6675. RecoverFS remains constant during a given fast recovery episode.
...

PRR initialization ...
   RecoverFS = snd.nxt - snd.una
   // Bytes SACKed before entering recovery will not be
   // marked as delivered during recovery:
   RecoverFS -= (bytes SACKed in scoreboard) - (bytes newly SACKed)
   // Include the (rare) case of cumulatively ACKed bytes:
   RecoverFS += (bytes newly cumulatively acknowledged)
```

DeliveredData

- Fixed bugs in the definition of DeliveredData (reverted to definition from RFC 6937)
- Removed buggy pseudocode for computing DeliveredData (just using the prose definition below)

New text:

```
DeliveredData: The total number of bytes that the current ACK indicates have been delivered to
the receiver. When there are no SACKed sequence ranges in the scoreboard before or after the
ACK, DeliveredData is the change in snd.una. With SACK, DeliveredData can be computed precisely
as the change in snd.una, plus the (signed) change in SACKed. In recovery without SACK,
DeliveredData is estimated to be 1 SMSS on receiving a duplicate acknowledgement, and on a
subsequent partial or full ACK DeliveredData is the change in snd.una, minus 1 SMSS for each
preceding duplicate ACK. Note that without SACK, a poorly-behaved receiver that returns
extraneous DUPACKs (as described in [Savage99]) could attempt to artificially inflate
DeliveredData. As a mitigation, if not using SACK then PRR disallows incrementing DeliveredData
when the total bytes delivered in a PRR episode would exceed the estimated data outstanding upon
entering recovery (RecoverFS).
```

Finishing PRR

- Clarified that PRR should set cwnd to ssthresh when finishing a PRR episode
  - Linux TCP PRR has done this from the first implementation (2011)
  - Mentioned the potential for bursts; RECOMMENDED pacing to deal with this

New text:

```
A PRR episode ends upon either completing fast recovery, or before initiating a new PRR episode
due to a new congestion control response episode.

On the completion of a PRR episode:
   cwnd = ssthresh

Note that this step that sets cwnd to ssthresh can potentially, in some scenarios, allow a burst
of back-to-back segments into the network. As with common scenarios that could allow bursts,
such as restarting from idle, it is RECOMMENDED that implementations use pacing to reduce the
burstiness of traffic.
```

PRR Examples

- Fixed diagrams in the 2 PRR examples to use the correct ssthresh of 10 rather than 11
  - Because the 2 segments sent as limited transmits should not be used to compute ssthresh
  - As specified in both RFC 5681, Sec 3.2 and RFC 6675 Sec 5

Figure 1: Old

```
RFC 6675
ack#   X  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
cwnd:    20 20 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
pipe:    19 19 18 18 17 16 15 14 13 12 11 10 10 10 10 10 10 10 10
sent:     N  N  R                          N  N  N  N  N  N  N  N

PRR
ack#   X  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
cwnd:    20 20 19 18 18 17 17 16 16 15 15 14 14 13 13 12 12 11 10
pipe:    19 19 18 18 17 17 16 16 15 15 14 14 13 13 12 12 11 11 10
sent:     N  N  R     N     N     N     N     N     N     N     N
```

New

```
RFC 6675
ack#   X  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
cwnd:    20 20 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
pipe:    19 19 18 18 17 16 15 14 13 12 11 10  9  9  9  9  9  9  9  9  9  9
sent:     N  N  R                          N  N  N  N  N  N  N  N  N  N

PRR
ack#   X  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
cwnd:    20 20 19 18 18 17 17 16 16 15 15 14 14 13 13 12 12 11 11 10 10 10
pipe:    19 19 18 18 17 17 16 16 15 15 14 14 13 13 12 12 11 11 10 10  9  9
sent:     N  N  R     N     N     N     N     N     N     N     N     N  N
```

Figure 2:

```
RFC 6675
ack#   X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  15 16 17 18 19
cwnd:                                           20 20 11 11 11
pipe:                                           19 19  4 10 10
sent:                                            N  N 7R  R  R

PRR
ack#   X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  15 16 17 18 19
cwnd:                                           20 20  5  5  5
pipe:                                           19 19  4  4  4
sent:                                            N  N  R  R  R
```

```
RFC 6675
ack#   X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  15 16 17 18 19
cwnd:                                           20 20 10 10 10
pipe:                                           19 19  4  9  9
sent:                                            N  N 6R  R  R

PRR
ack#   X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  15 16 17 18 19
cwnd:                                           20 20  5  5  5
pipe:                                           19 19  4  4  4
sent:                                            N  N  R  R  R
```

PRR interactions with some RFC 3168 AQMs

- Noted that problematic interactions have been found between:
  - (a) using PRR for for RFC 3168 cwnd reduction (as Linux TCP does)
  - (b) some ECN signals from at least one RFC 3168 AQM implementation

New text:

note that using PRR for for cwnd reductions for [RFC3168] ECN has been observed, with some ECN AQMs, to cause an excess cwnd reduction during ECN-triggered congestion episodes, as noted in [VCC].
...

[VCC]
Cronkite-Ratcliff, B., Bergman, A., Vargaftik, S., Ravi, M., McKeown, N., Abraham, I., and I. Keslassy, "Virtualized Congestion Control (Extended Version)", August 2016, <http://www.ee.technion.ac.il/~isaac/p/sigcomm16_vcc_extended.pdf>.