Updating Appropriate Byte Counting (RFC 3465)

TCPM, IETF 112
Nov 2021
RFC3465 Appropriate byte counting recap

• Addresses the infamous congestion control vulnerability to increase cwnd per ACK discovered by [1].

• $cwnd$ increases by 1000 by receiving 1000 ACKs each ack’ing one byte

• RFC3465 fixes this by increasing $cwnd$ based on the (new) bytes acknowledged instead of per ACK

• But a stretched ACK can increase $cwnd$ sharply leading to burst. Hence, a cap $L$ to limit burst

Problem statement

[RFC 3465] During slow-start, implementations MUST NOT use $L > 2 \times SMSS$

- Stretch / Compressed ACKs acknowledge more than 2 packets
- $cwnd$ increase is much slower than the amount traffic leaving the network
- Linux and other stacks don’t implement $L = 2 \times SMSS$
- Linux implements ABC without $L$ (since 2013) and senders are encouraged to use pacing to reduce ACK-induced bursts
Proposed solution

Remove the limit $L$

- Sender uses ACK info to learn about network capacity
- Separate $cwnd$ increase and sending rate
  - During slow-start, $cwnd$ increases by the amount of data that left the network, i.e.,
    \[
    cwnd + = DeliveredData^1
    \]
- Sending is controlled via pacing

DeliveredData

SACK is supported

When there are no SACKd sequence ranges,

• Change in \( snd \cdot una \)

When there are SACKd sequence ranges,

• Change in \( snd \cdot una + (\text{signed})\) change in SACKd
DeliveredData

SACK is not supported

When there are no dup ACKs,

• Change in $snd\cdot una$

When there are dup ACKs,

• 1 SMSS on a dup ACK

• On subsequent partial or full ACK, $((\text{change in } snd \cdot una) - (1 \text{ SMSS for each preceding dup ACK}))$
Should we fold these changes to 5681-bis?