HyStart++: Modified Slow Start for TCP

draft-ietf-tcpm-hystartplusplus-03

TCPM, IETF 111
July 27, 2021

Praveen Balasubramanian, Yi Huang, Matt Olson
HyStart++ Recap

• Slow Start can overshoot ideal send rate & cause massive packet loss
  • Increased retransmissions
  • Time spent in recovery
  • Sometimes results in RTO (retransmission timeout)

• HyStart++ until draft-01
  • Simple modification to Slow Start
  • Only use Delay Increase algorithm from original HyStart
  • Compensate for premature slow start exit
  • Use max of Limited Slow Start (RFC3742) and Congestion Avoidance
  • Define tuning constants based on measurements and deployment experience
Jitter Problems

• Intra DC WAN transfers suffered latency spikes
  • Latency spike lasted 1-2 rounds but triggered HyStart exit
  • Source of spike not root caused, but later disappeared during testing

• Performance problems due to jitter
  • Raised as an issue in tcpm mailing list by Christian and others
  • Reported as an issue in [TCP HyStart Performance over a Satellite Network (wpi.edu)](http://wpi.edu)
Jitter Resiliency and Simplification

- Standard slow start (RFC 5681)
- Only use Delay Increase algorithm from original HyStart
- Upon exit from slow start, enter Conservative Slow Start (CSS)
- Under CSS increase cwnd as a fraction of standard slow start
- If measured RTT shrinks during CSS, exit was spurious, resume HyStart++
- Else enter congestion avoidance

- Rationale: Instead of trying to compensate for early exit, add detection for spurious exits to be able to resume slow start
Algorithm Details

• On each ACK in slow start
  • Update the cwnd per standard slow start
  • If taking an RTT sample, measure current round’s minRTT

• For each round in slow start (round approximates an RTT)
  • Remember last round’s minRTT
  • If cwnd >= (LOW_CWND * SMSS) and at least N_RTT_SAMPLE RTT samples taken
  • Check if currentRoundMinRTT is greater than lastRoundMinRTT + Threshold
  • If yes, set ssthresh = cwnd, cssBaselineRtt = currentRoundMinRTT, exit slow start and enter CSS

• CSS lasts at most CSS_ROUNDS rounds. On each ACK in CSS
  • Update the cwnd as “standard slow start cwnd” / CSS_GROWTH_DIVISOR

• For each round in CSS
  • If at least N_RTT_SAMPLE RTT samples taken
  • Check if currentRoundMinRTT is less than cssBaseLineRtt
  • If yes, declare exit as spurious and resume HyStart++
  • Else enter congestion avoidance

• Exit HyStart++ on first congestion signal
• SHOULD use on first slow start and MAY use after idle
Lab Measurements – no jitter
Lab Measurements – jitter

P90 Goodput in different scenarios

- New Hystart
- No Hystart
- Original Hystart

Goodput (Mbps)

0 10 20 30 40 50 60 70 80 90

100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
100ms, 100Mbits: 0.80, 0.50, 0.30, 0.20, 0.10
Lab Measurements – Varying CSS_ROUNDS
Status & Next Steps

• We made a rather significant change to the algorithm
• Currently flighting and doing A/B measurements
• Revaluate fixed threshold clamps

• Should we change the Intended Status to Experimental?