Network Working Group
Internet-Draft
Intended status: Informational
Expires: October 28, 2020

P. Balasubramanian
Y. Huang
M. Olson
Microsoft

April 26, 2020

HyStart++: Modified Slow Start for TCP
draft-balasubramanian-tcpm-hystartplusplus-03

Abstract

This informational memo describes HyStart++, a simple modification to the slow start phase of TCP congestion control algorithms. HyStart++ combines the use of one variant of HyStart and Limited Slow Start (LSS) to prevent overshooting of the ideal sending rate, while also mitigating poor performance which can result from false positives when HyStart is used alone. This memo also describes the details of the current implementation in the Windows operating system.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 28, 2020.

Copyright Notice

Table of Contents

1. Introduction

[RFC0793] and [RFC5681] describe the slow start mechanism for TCP. The slow start algorithm is used when the congestion window (cwnd) is less than the slow start threshold (ssthresh). During slow start, in absence of packet loss signals, TCP sender increases cwnd exponentially to probe the network capacity. Such a fast growth can lead to overshooting the ideal sending rate and cause significant packet loss. TCP has several mechanisms for loss recovery, but they are only effective for moderate loss. When these techniques are unable to recover lost packets, a last-resort retransmission timeout (RTO) is used to trigger packet recovery. In most operating systems, the minimum RTO is set to a large value (200 ms or 300ms) to prevent spurious timeouts. This results in a long idle time which drastically impairs flow completion times.

HyStart++ adds delay increase as a signal to exit slow start before any packet loss occurs. This is one of two algorithms specified in [HyStart]. After the HyStart delay algorithm finds an exit point, LSS is used for further congestion window increases until the first packet loss occurs.

This document describes HyStart++ as implemented in the Microsoft Windows operating system. HyStart++ is widely deployed on the public Internet. Precise documentation of running code enables follow-up IETF Experimental or Standards Track RFCs. It also enables other implementations and sharing of results for various workloads.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Definitions

SENDER MAXIMUM SEGMENT SIZE (SMSS): The SMSS is the size of the largest segment that the sender can transmit. This value can be based on the maximum transmission unit of the network, the path MTU discovery [RFC1191, RFC4821] algorithm, RMSS (see next item), or other factors. The size does not include the TCP/IP headers and options.

RECEIVER MAXIMUM SEGMENT SIZE (RMSS): The RMSS is the size of the largest segment the receiver is willing to accept. This is the value specified in the MSS option sent by the receiver during connection startup. Or, if the MSS option is not used, it is 536 bytes [RFC1122]. The size does not include the TCP/IP headers and options.

CONGESTION WINDOW (cwnd): A TCP state variable that limits the amount of data a TCP can send. At any given time, a TCP MUST NOT send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd.

4. HyStart++ Algorithm

4.1. Use of HyStart Delay Increase and Limited Slow Start

[HyStart] specifies two algorithms (a "Delay Increase" algorithm and an "Inter-Packet Arrival" algorithm) to be run in parallel to detect that the sending rate has reached capacity. In practice, the Inter-Packet Arrival algorithm does not perform well and is not able to detect congestion early, primarily due to ACK compression. The idea of the Delay Increase algorithm is to look for RTT spikes, which

suggest that the bottleneck buffer is filling up.

After the HyStart "Delay Increase" algorithm triggers an exit from slow start, LSS (described in [RFC3742]) is used to increase Cwnd until the first packet loss occurs. LSS is used because the HyStart exit is often premature as a result of RTT fluctuations or transient queue buildup. LSS grows the cwnd fast but much slower than traditional slow start. LSS helps avoid massive packet losses and subsequent time spent in loss recovery or retransmission timeout.

4.2. Algorithm Details

We assume that Appropriate Byte Counting (as described in [RFC3465]) is in use and L is the cwnd increase limit. The choice of value of L is up to the implementation.

A round is chosen to be approximately the Round-Trip Time (RTT). Round can be approximated using sequence numbers as follows:

Define windowEnd as a sequence number initialize to SND.UNA

When windowEnd is ACKed, the current round ends and windowEnd is set to SND.NXT

At the start of each round during slow start:

lastRoundMinRTT = currentRoundMinRTT

currentRoundMinRTT = infinity

rttSampleCount = 0

For each arriving ACK in slow start, where N is the number of previously unacknowledged bytes acknowledged in the arriving ACK and w:

Update the cwnd

cwnd = cwnd + min (N, L * SMSS)

```
Keep track of minimum observed RTT

   currentRoundMinRTT = min(currentRoundMinRTT, currRTT)

   where currRTT is the measured RTT based on the incoming ACK

   rttSampleCount += 1

For rounds where cwnd is at or higher than LOW_CWND and
N_RTT_SAMPLE RTT samples have been obtained, check if delay
increase triggers slow start exit

   if (cwnd >= (LOW_CWND * SMSS) AND rttSampleCount >=
   N_RTT_SAMPLE)

      RttThresh = clamp(MIN_RTT_THRESH, lastRoundMinRTT / 8,
      MAX_RTT_THRESH)
```



```
      if (currentRoundMinRTT >= (lastRoundMinRTT + RttThresh))

         ssthresh = cwnd

         exit slow start and enter LSS
```

For each arriving ACK in LSS, where N is the number of previously unacknowledged bytes acknowledged in the arriving ACK:

```
   K = cwnd / (LSS_DIVISOR * ssthresh)

   cwnd = max(cwnd + (min (N, L * SMSS) / K), CA_cwnd())
```

CA_cwnd() denotes the cwnd that a congestion control algorithm would have increased to if congestion avoidance started instead of LSS. LSS grows cwnd very fast but for long-lived flows in high BDP networks, the congestion avoidance algorithm could increase cwnd much faster. For example, CUBIC congestion avoidance [RFC8312] in convex region can ramp up cwnd rapidly. Taking the max can help improve performance when exiting slow start prematurely.

HyStart++ ends when congestion is observed.

4.3. Constants used and tuning

The Windows operating system implementation of HyStart++ uses the following constants:

LOW_CWND = 16

MIN_RTT_THRESH = 4 msec

MAX_RTT_THRESH = 16 msec

LSS_DIVISOR = 0.25

N_RTT_SAMPLE = 8

An implementation MAY experiment with these constants and tune them for different network characteristics. Windows operating system implementation uses the same values for all connections. The maximum value of LSS_DIVISOR SHOULD NOT exceed 0.5 which is the value recommended in [RFC3742].

An implementation MAY choose to use HyStart++ for all slow starts including the ones post a retransmission timeout, or a long idle period. The Windows operating system implementation uses HyStart++ only for the initial slow start and uses traditional slow start for

subsequent ones. This is acceptable because subsequent slow starts will use the discovered ssthresh value to exit slow start.

5. Security Considerations

HyStart++ enhances slow start and inherits the general security considerations discussed in [RFC5681].

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

Neal Cardwell suggested the idea for using the maximum of cwnd value

computed by LSS and congestion avoidance after exiting slow start.

8. References

8.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3465] Allman, M., "TCP Congestion Control with Appropriate Byte Counting (ABC)", RFC 3465, DOI 10.17487/RFC3465, February 2003, <https://www.rfc-editor.org/info/rfc3465>.

[RFC3742] Floyd, S., "Limited Slow-Start for TCP with Large Congestion Windows", RFC 3742, DOI 10.17487/RFC3742, March 2004, <https://www.rfc-editor.org/info/rfc3742>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

8.2. Informative References

[HyStart] Ha, S. and I. Ree, "Hybrid Slow Start for High-Bandwidth and Long-Distance Networks", DOI 10.1145/1851182.1851192, International Workshop on Protocols for Fast Long-Distance Networks, 2008, <https://pdfs.semanticscholar.org/25e9/ef3f03315782c7f1cbcd31b587857adae7d1.pdf>.

[RFC8312] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and

R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC 8312, DOI 10.17487/RFC8312, February 2018, <https://www.rfc-editor.org/info/rfc8312>.

Authors' Addresses

Praveen Balasubramanian
Microsoft
One Microsoft Way
Redmond, WA 98052
USA

Phone: +1 425 538 2782
Email: pravb@microsoft.com

Yi Huang
Microsoft

Phone: +1 425 703 0447
Email: huanyi@microsoft.com

Matt Olson
Microsoft

Phone: +1 425 538 8598
Email: maolson@microsoft.com