Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 9, 2020

P. Balasubramanian
Y. Huang
M. Olson
Microsoft

July 8, 2019

HyStart++: Modified Slow Start for TCP
draft-balasubramanian-tcpm-hystartplusplus-00

Abstract

This experimental 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 value, 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 January 9, 2020.

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. HyStart++ Algorithm 3
3.1. Use of HyStart Delay Increase and Limited Slow Start . . 3
3.2. Algorithm Details 3
3.3. Constant used and tuning 4
4. Security Considerations 5
5. IANA Considerations 5
6. References 5
6.1. Normative References 5
6.2. Informative References 5
Authors' Addresses 6

1. Introduction

[RFC0793] and [RFC5681] describe the slow start mechanism for TCP. The slow start algorithm is used when congestion window (cwnd) is less than the slow start threshold (ssthresh). During slow start, a TCP increments cwnd by at most SMSS bytes. In the absence of packet loss signals, slow start effectively doubles the congestion window each round trip time.

While traditional TCP slow start can ramp up very quickly, it frequently overshoots the ideal sending rate and causes a lot of unnecessary packet drops. 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. A 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. HyStart++ Algorithm

3.1. 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.

3.2. Algorithm Details

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

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

```
Update the cwnd

   cwnd = cwnd + min (N, SMSS)

Keep track of minimum observed RTT.

   currentRoundMinRTT = min(currentRoundMinRTT, currRTT)

   where currRTT is the measured RTT based on the incoming ACK

For rounds where cwnd is greater than or equal to MIN_SSTHRESH,
check if delay increase triggers slow start exit

   if (cwnd is >= MIN_SSTHRESH)

      Eta = clamp(MIN_ETA, currentRoundMinRTT / 8, MAX_ETA)

      if (currentRoundMinRTT >= (lastRoundMinRTT + Eta))

         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 += N / K
```

HyStart++ ends when cwnd exceeds ssthresh or when congestion is observed.

3.3. Constant used and tuning

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

```
MIN_SSTHRESH = 16

MIN_ETA = 4 msec

MAX_ETA = 16 msec

LSS_DIVISOR = 0.25
```

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.

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.

4. Security Considerations

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

5. IANA Considerations

This document has no actions for IANA.

6. References

6.1. 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>.

[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>.

6.2. 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, March 2010, <https://doi.org/10.1016/j.comnet.2011.01.014>.

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