Multipath TCP R. Barik
Internet-Draft University of Oslo
Intended status: Standards Track S. Ferlin
Expires: December 29, 2016 Simula Research Laboratory
M. Welzl
University of Oslo
June 27, 2016

A Linked Slow-Start Algorithm for MPTCP
draft-barik-mptcp-lisa-01

Abstract

This document describes the LISA (Linked Slow-Start Algorithm) for Multipath TCP (MPTCP). Currently during slow-start, subflows behave like independent TCP flows making MPTCP unfair to cross-traffic and causing more congestion at the bottleneck. This also yields more losses among the MPTCP subflows. LISA couples the initial windows (IW) of MPTCP subflows during the initial slow-start phase to remove this adverse behavior.

Table of Contents

1. Introduction 3
  1.1. Definitions 3
2. MPTCP Slow-Start Problem Description 4
  2.1. Example of current MPTCP slow-start problem 4
3. Linked Slow-Start Algorithm 4
  3.1. Description of LISA 4
  3.2. Algorithm 5
4. Implementation Status 6
5. Acknowledgements 6
6. IANA Considerations 6
7. Security Considerations 6
8. Change History 7
9. References 7
  9.1. Normative References 7
  9.2. Informative References 7
Authors' Addresses 8

1. Introduction

The current MPTCP implementation provides multiple congestion control algorithms, which aim to provide fairness to TCP flows at the shared bottlenecks. However, in RFC 6356 [RFC6356], the subflows' slow-start phase remains unchanged to RFC 5681 [RFC5681], and all the subflows at this stage behave like independent TCP flows. Following the development of IW as per [RFC6928], each MPTCP subflow can start with IW = 10. With an increasing number of subflows, the subflows' collective behavior during the initial slow-start phase can temporarily be very aggressive towards a concurrent regular TCP flow at the shared bottleneck.

According to [UIT02], most of the TCP sessions in the Internet consist of short flows, e.g., HTTP requests, where TCP will likely never leave slow-start. Therefore, the slow-start behavior becomes of critical importance for the overall performance.

To mitigate the adverse effect during initial slow-start, we introduce LISA, the "Linked Slow-Start Algorithm". LISA shares the congestion window MPTCP subflows in slow start whenever a new subflow joins.

1.1. Definitions

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 RFC 2119 [RFC2119].

Acronyms used in this document:

IW -- Initial Window

RTT -- Round Trip Time

CWND -- Congestion Window

Inflight -- MPTCP subflow's inflight data

old_subflow.CWND -- Congestion Window of the subflow having largest sending rate

new_subflow.CWND -- New incoming subflow's Congestion Window

Ignore_ACKs -- a boolean variable indicating whether ACKs should be ignored

ACKs_To_Ignore -- the number of ACKs for which old_subflow.CWND stops increasing during slow-start

compound CWND -- sum of CWND of the subflows in slow-start

2. MPTCP Slow-Start Problem Description

Since it takes 1 RTT for the sender to receive any feedback on a given TCP connection, sending an additional segment after every ACK is rather aggressive. Therefore, in slow-start, all subflows independently doubling their CWND as in regular TCP results in MPTCP also doubling its compound CWND. The MPTCP aggregate only diverges from this behavior when the number of subflows changes. Coupling of CWND is therefore not necessary in slow-start except when a new subflow joins.

2.1. Example of current MPTCP slow-start problem

We illustrate the problematic MPTCP slow-start behavior with an example: Consider an MPTCP connection consisting of 2 subflows. The first subflow starts with IW = 10, and after 2 RTTs the CWND becomes 40 and a new subflow joins, again with IW = 10. Then, the compound CWND becomes 40+10 = 50. With an increasing number of subflows, the compound CWND in MPTCP becomes larger than that of a concurrent TCP flow.

For example, MPTCP with eight subflows (as recommended in [DCMPTCP11] for datacenters) will have a compound CWND of 110 (40+7*10). As a result, MPTCP would behave unfairly to a concurrent TCP flow sharing the bottleneck. This aggressive behavior of MPTCP also affects the performance of MPTCP. If multiple subflows share a bottleneck, each of them doubling their rate every RTT, will cause excessive losses at the bottleneck. This makes MPTCP enter the congestion avoidance phase earlier and thereby increases the completion time of the transfer.

This problem, and the improvement attained with LISA, are documented in detail in [lisa].

3. Linked Slow-Start Algorithm

3.1. Description of LISA

The idea behind LISA is that each new subflow takes a 'packet credit' from an existing subflow in slow-start for its own IW. We design the mechanism such that a new subflow has 10 segments as the upper limit

[RFC6928] and 3 segments as the lower limit [RFC3390]. This is based on [RFC6928], [RFC3390] and the main reason behind it is to let these subflows compete reasonably with other flows. We also divide the CWND fairly in order to give all subflows an equal chance when competing with each other.

LISA first finds the subflow with the largest sending rate measured over the last RTT. Depending on the subflow's CWND, between 3 and 10 segments are taken from it as packet credit and used for the new subflow's IW. The packet credit is realized by reducing the CWND from the old subflow and halting its increase for ACKs_To_Ignore number of ACKs.

We clarify LISA with the example given in Section 2.1. After 2 RTTs, the old_subflow.CWND = 40 and a new_subflow joins the connection. Since old_subflow.CWND >= 20 (refer to Section 3.2), 10 packets can be taken by the new_subflow.CWND, resulting in old_subflow.CWND = 30 and new_subflow.CWND = 10. Hence, MPTCP's compound CWND, whose current size is 40, should ideally become 60+20 = 80 after 1 RTT (assuming a receiver without delayed ACKs). However, if 40 segments from old_subflow.CWND are already in flight, the compound CWND becomes in fact 70+20 = 90. Here, LISA keeps old_subflow.CWND from increasing for the next 10 ACKs. In comparison, MPTCP without LISA would have a compound CWND of 80+20=100 after 1 RTT.

3.2. Algorithm

Below, we describe the LISA algorithm. LISA is invoked before a new subflow sends its IW.

1. Before computing the new_subflow.CWND, Ignore_ACKs = False and ACKs_To_Ignore = 0.

2. Then, ignoring the new_subflow, the subflow in slow-start with the largest sending rate (old_subflow.CWND, measured over the last RTT) is selected.

3. If there is no such subflow, the IW of the new_subflow.CWND = 10 Otherwise, the following steps are executed:

```
if old_subflow.CWND >= 20 // take IW(10) packets

   old_subflow.CWND -= 10

   new_subflow.CWND = 10
```

```
            Ignore_ACKs = True

         else if old_subflow.CWND >= 6 // take half the packets

            new_subflow.CWND -= old_subflow.CWND / 2

            old_subflow.CWND -= new_subflow.CWND

            Ignore_ACKs = True

         else

            new_subflow.CWND = 3 // can't take from old_subflow

   4.  if Ignore_ACKs and Inflight > old_subflow.CWND

         // do not increase CWND when ACKs arrive

         ACKs_To_Ignore = Inflight - old_subflow.CWND
```

4. Implementation Status

LISA is implemented as a patch to the Linux kernel 3.14.33+ and within MPTCP's v0.89.5. It is meant for research and provided by the Unviersity of Oslo and Simula Research Laboratory, and available for download from http://heim.ifi.uio.no/runabk/lisa This code was used to produce the test results that are reported in [lisa].

5. Acknowledgements

This work was part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The authors also would like to thank David Hayes (UiO) for his comments. The views expressed are solely those of the authors.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

8. Change History

Changes made to this document:

00->01 :   Some minor text improvements and updated a reference.

9. References

9.1. Normative References

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

[RFC3390] Allman, M., Floyd, S., and C. Partridge, "Increasing TCP's Initial Window", RFC 3390, DOI 10.17487/RFC3390, October 2002, <http://www.rfc-editor.org/info/rfc3390>.

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

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <http://www.rfc-editor.org/info/rfc6356>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <http://www.rfc-editor.org/info/rfc6928>.

9.2. Informative References

[DCMPTCP11] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D., and M. Handley, "Improving datacenter performance and robustness with multipath TCP", ACM SIGCOMM p266-277, August 2011.

[UIT02] Brownlee, N. and K. Claffy, "Understanding internet traffic streams: Dragonflies and tortoises", IEEE Communications Magazine p110-117, 2002.

[lisa] Barik, R., Welzl, M., Ferlin, S., and O. Alay, "LISA: A Linked Slow-Start Algorithm for MPTCP", IEEE ICC 2016,

Kuala Lumpur, Malaysia , 2016.

Authors' Addresses

Runa Barik
University of Oslo
PO Box 1080 Blindern
Oslo N-0316
Norway

Email: runabk@ifi.uio.no

Simone Ferlin
Simula Research Laboratory
P.O.Box 134
Lysaker,   1325
Norway

Email: ferlin@simula.no

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo,   N-0316
Norway

Phone: +47 2285 2420
Email: michawe@ifi.uio.no