Multipath TCP
Internet-Draft
Intended status: Standards Track
Expires: January 24, 2016

A. Walid
Bell Labs
Q. Peng
Caltech
J. Hwang
Samsung Electronics
S. Low
Caltech

July 23, 2015

Balanced Linked Adaptation Congestion Control Algorithm for MPTCP
draft-walid-mptcp-congestion-control-03

Abstract

This document describes the mechanism of Balia, the "Balanced linked adaptation", which is a congestion control algorithm for Multipath TCP (MPTCP). The recent proposals, LIA and OLIA, suffer from either unfriendliness to Single Path TCP (SPTCP) or unresponsiveness to network changes under certain conditions. The tradeoff between friendliness and responsiveness is inevitable, but Balia judiciously balances this tradeoff based on a new design framework that allows one to systematically explore the design space. Balia has been implemented in the Linux kernel and also included in the UCLouvain’s MPTCP implementation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Various congestion control algorithms have been proposed as extensions of TCP NewReno to MPTCP. A straightforward extension is to run TCP NewReno on each subpath, e.g., [HONDA09]. This algorithm, however, can be highly unfriendly when it shares a path with a SPTCP user. This motivates the Coupled algorithm which is fair because it has the same underlying utility function as TCP NewReno, e.g., [KELLY05], [HAN04]. It is found in [RFC6356], however, that the Coupled algorithm responds slowly in a dynamic network environment.

The current default congestion control algorithm for MPTCP, called LIA (Linked-Increases Algorithm), is more responsive than the Coupled algorithm. However, it has been reported that LIA can sometimes be excessively aggressive toward SPTCP users without any benefit to multipath users [KHALILI12]. Recently, OLIA (Opportunistic Linked-Increases Algorithm) [KHALILI12] was proposed as a variant of Coupled algorithm [KELLY05] which is as friendly as the Coupled algorithm. We have found, however, that OLIA can be unresponsive to changes in network conditions in some scenarios (e.g., when the paths used by a user have similar round trip times (RTTs)) [PENG14].

In this draft, we introduce Balia, the "Balanced linked adaptation", which is a window-based congestion control algorithm for MPTCP. The main design goal of Balia is to systematically tradeoff different properties such as TCP friendliness and responsiveness by developing structural understanding of MPTCP algorithms in a new design framework. For instance, it is widely suspected that there is a tradeoff between friendliness and responsiveness and it is proved in this framework that this tradeoff is indeed inevitable. By parameterizing different structural properties, Balia generalizes existing algorithms and explicitly balances the tradeoff. We also prove mathematically that Balia has a unique equilibrium point, and that it is asymptotically stable. Therefore, Balia can provide balanced performance in terms of friendliness and responsiveness.

In [PENG14], we compare the performance of several MPTCP algorithms over a testbed, including Balia, OLIA and LIA. Our experimental results show that Balia is friendlier than LIA and more responsive than OLIA. It also solves LIA’s problem identified by [KHALILI12]. Balia has been implemented in the Linux kernel and also included in the UCLouvain’s MPTCP implementation.

1.1. Requirements Language

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].

1.2. Terminology

Regular/Singlepath TCP (SPTCP): The standard version of TCP [RFC5681] that uses a single pair of IP address and ports per connection.

Multipath TCP (MPTCP): A modified version of the regular TCP that simultaneously uses multiple paths between hosts.

LIA: The Linked-Increases Algorithm for MPTCP [RFC6356].

OLIA: The Opportunistic Linked-Increases Algorithm for MPTCP [KHALILI12].

Balia: The Balanced linked adaptation algorithm for MPTCP [PENG14].

AIMD: The Additive Increase Multiplicative Decrease algorithm used in TCP congestion avoidance.

w_r: The congestion window on a path r.

rtt_r: The Round-Trip Time (RTT) on a path r.

2. Balanced Linked Adaptation Algorithm

Balia is a generalized MPTCP algorithm that strikes a good balance between friendliness and responsiveness. The algorithm only applies to the AIMD part of the congestion avoidance phase. The other parts such as slow start, fast retransmit/recovery algorithms are the same as in TCP [RFC5681]. The minimum ssthresh is set to 1 MSS instead of 2 when more than 1 path is available.

Each source s has a set of paths r. As a special case, the set can be a singleton in which case Balia reduces to TCP Reno (see below). Each path r maintains a congestion window w_r and measures its round-trip time rtt_r. The window adaptation of Balia is as follows:

- For each ACK on path r, increase w_r by:

$$\frac{x_r}{rtt_r * (\mathrm{SUM}(x_k))^2} * \left(\frac{1 + alpha_r}{2} \right) * \left(\frac{4 + alpha_r}{5} \right)$$

- For each packet loss on path r, decrease w_r by:

$$\frac{w_r}{2} \; * \; \min \{ alpha_r, \; 1.5 \}$$

where x_r = w_r / rtt_r and alpha_r = max { x_k } / x_r.

Note that Balia’s decrement algorithm multiplies the MD algorithm of TCP Reno by a factor in the range of [1, 1.5].

If a Balia user uses only a single path, then alpha_r = 1, in which case both the increment and the decrement algorithms of Balia reduce to those of TCP Reno. Hence Balia reduces to TCP Reno on single paths.

3. Theoretical justification

In [PENG14], we have developed a unified model of MPTCP algorithms and characterized the design space. This provides a framework to systematically design MPTCP algorithms and analyze their behavior in a large network at design time. For instance, it has allowed us to identify designs that guarantee the existence, uniqueness and stability of network equilibrium. Balia is designed using this framework to achieve specific design goals. In this section, we will focus on how Balia balances three often conflicting design goals, TCP friendliness, responsiveness and window oscillation.

TCP friendliness characterizes how much more throughput a MPTCP flow will get when it competes with an SPTCP flow. A MPTCP flow is said to be "TCP friendly" if it does not dominate the available bandwidth when it shares the same network with a SPTCP flow.

Responsiveness characterizes how fast the MPTCP algorithm reacts to changes in network conditions.

The window oscillation property characterizes how severely the window size fluctuates around the equilibrium point. It is an inherent property of AIMD-like algorithms.

In [PENG14], it is proved mathematically that there is an inevitable tradeoff between TCP friendliness and responsiveness, and between responsiveness and window oscillation. Thus, it is theoretically impossible to maximize the performance in all three metrics simultaneously.

Our design philosophy is to allow window oscillation up to an acceptable level in order to improve both friendliness and responsiveness. This is achieved by explicitly parameterizing these properties and systematically choosing these parameters.

4. Implementation considerations

To enable Balia to operate in a wide spectrum of applications scenarios, i.e., with wide range of w_r and rtt_r, we need to re-write the Balia’s additive increase (AI) formula in an equivalent form which allows easier implementation in the Linux kernel with fixed point operations, and avoids integer-overflow problems. Note that in an extreme case, the sending rate on a path, x_r, may increase to 2^30 (w_r/rtt_r) or more. In such a case, the term (SUM(x_k))^2 in the current formula can easily cause 64-bit integer overflow. In addition, there can be also a significant rounding error when we do a fixed-point division by a large number.

Therefore, to mitigate the above issues, we rewrite Balia’s additive increase (AI) formula as follows:

$$\frac{x\_r}{rtt\_r * (SUM(x\_k))^2} * \left(\frac{x\_r + \max\{x\_k\}}{2 * x\_r} \right) * \left(\frac{4 * x\_r + \max\{x\_k\}}{5 * x\_r} \right)$$

which can be simplified to:

$$\frac{(x\_r + \max\{x\_k\}) * (4 * x\_r + \max\{x\_k\})}{w\_r * (SUM(x\_k))^2 * 10}$$

Now the term x_r is computed in bytes/sec and may increase up to about 2^52. Thus we need to scale the sending rate through bit-shift operations so that max{x_k} does not exceed a cerntain value, e.g., 2^25, to safely calculate the term (SUM(x_k))^2. At this point, if max{x_k} is a very large number, a small x_r can be sometimes 0 after the right shift operation. This may hurt the accuracy of the calculation. But we have seen that the overall rounding error is not significant since max{x_k} is the dominant term in the formula while x_r would be neglible in such cases.

5. Experimental results

In this section, we summarize our experimental results that illustrate the weaknesses of the current algorithms (LIA and OLIA). We evaluate the MPTCP algorithms using the UCLouvain’s MPTCP implementation [MPLKI]. The network parameters such as network bandwidth and one-way delay are implemented by Dummynet [DUMMYNET]. Iperf is used to generate traffic and measure the throughput.

5.1. Khalili’s scenario

In [KHALILI12], it has been revealed that LIA can be unfriendly to SPTCP users even when its own MPTCP throughput is saturated. That is, the throughputs of SPTCP flows are significantly degraded without any benefit to MPTCP flows. To reproduce this scenario, we create a testbed as shown in Figure 1. In this scenario, N1 type1 users can be either single-path or multipath while N2 type2 users are always single-path.

```
                                                    +----+
          N1                                        |    |   Server
         Type1 ---------------------------------| C1 |-- for type1
         flows --\                            /--|    |   flows
                  \                          /   +----+
                   \                        /    Router1
                    \   +----+   +----+   /
                     \--|    |   |    |--/
          N2            | C2 |---|    |
         Type2 ---------|    |   |    |------------------ Server
         flows          +----+   +----+                    for type2
                                                            flows
                        Router2  Router3
```

Figure 1: Testbed topology for Khalili’s scenario. The Router1 emulates the server-side bottleneck for type1 users and the Router2 emulates the shared bottleneck.

The aggregate throughputs of these users are shown in Table 1 for the case when all users are SPTCP and the case when all type1 users are upgraded to MPTCP users using different algorithms. We observe that upgrading type1 users to MPTCP decreases type2 users’ throughput without any benefit to type1 users if LIA is used; the type2 users are worse off by 19% when N1=N2=5 and by 25% when N1=15 and N2=5. Both OLIA and Balia are more friendly than LIA to SPTCP (type2) users.

C1=C2=10Mbps

		Type1 users are single-path	Type1 users are multipath		
			LIA	OLIA	Balia
N1=5	type1	9.47	9.26	9.25	9.25
N2=5	type2	9.29	7.55	8.13	8.32
N1=15	type1	9.39	8.96	8.93	9.02
N2=5	type2	9.29	6.94	7.41	7.98

Values are in Mbps.

Table 1: Throughput obtained by type1 and type2 users: Upgrading type1 users to MPTCP decreases type2 users’ throughput without any benefit to type1 users.

5.2. Responsiveness

To demonstrate the dynamic performance of MPTCP algorithms, we implement a testbed topology as shown in Figure 2. One-way delay of each single-path is about 10ms. In this scenario, a MPTCP flow is long lived while 5 SPTCP flows start at 40s and end at 80s. Table 2 shows the convergence time, which is defined as the first time the congestion window on the second path via Router2 reaches the average congestion window after the SPTCP users have left.

```
                        Router1               Router3
                       +-------+ 20Mbps,10ms +-------+ 40Mbps
     1 MPTCP ----------|       |-------------|       |-------- Server
       flow  --\       |       |       /-----|       |
     (0-200s)   \      +-------+      /      +-------+
                 \                   /
                  \             20Mbps,10ms
                   \   +-------+   /
                    \--|       |--/
     5 SPTCP ----------|       |
       flows           +-------+
     (40-80s)           Router2
```

Figure 2: Testbed topology for the responsiveness scenario.

	Coupled	OLIA	LIA	Balia
Convergence time	94.36	58.5	17.75	14.73

Values are in seconds.

Table 2: Responsiveness: Convergence time of MPTCP user after SPTCP users have left the network.

We observe that in this scenario Balia and LIA are quite responsive while both Coupled and OLIA algorithms take an excessively long time to recover. Note that in this scenario, the increment/decrement algorithms of Coupled and those of OLIA are similar, and therefore they behave in a similar way. For both algorithms, the excessively slow recovery of the congestion window on the second path is due to the design that increases the window roughly by w_r / (SUM(w_k))^2 on each ACK assuming the RTTs are similar. After the SPTCP users have left, w_2 is small while w_1 is large, so that w_2 / (w_1 + w_2)^2 is very small. It therefore takes a long time for w_2 to increase to its steady state value. In general, under the Coupled algorithm, a route with a large throughput can greatly suppress the throughput on another route even though the other route is underutilized.

5.3. NorNet experiment

To show that Balia works well on real Internet environments, we create two virtual machine hosts A and B over the NorNet Core, a country-wide (Norway) multi-homed research testbed [NORNET]. Host A is connected to Internet via ISP(Internet service provider)-1 while host B is connected via ISP-2 and ISP-3.

Considering a scenario where host B downloads a file from host A via two interfaces, we measure the througputs of both SPTCP and MPTCP with Reno and Balia respectively, as shown in Table 3. There are two logical paths between the hosts, (ISP-1 to ISP-2) and (ISP-1 to ISP-3), so we measure the bandwidth of each single-path with SPTCP and both of the two paths with MPTCP. The measurement is repeated 30 times for each case. In Table 3, it is observed that MPTCP with Balia aggregates the bandwidths of the two paths well.

	SPTCP(Reno)
	A(1)->B(2)
		SPTCP(Reno)
		A(1)->B(3)
			MPTCP(Balia)
			A(1)->B(2,3)

Avg. throughput	3.976	3.823	7.508
Max. throughput	4.08	3.83	7.69
Min. throughput	3.93	3.82	7.2

Values are in Mbps.

Table 3: Throughputs of SPTCP and MPTCP over the NorNet Core. Numbers in parenthesis refer to the ISP number.

6. Conclusion

In [PENG14], we have developed a model for MPTCP and identified designs that guarantee the existence, uniqueness and stability of the network equilibrium. We also characterize the design space and study the tradeoff among TCP friendliness, responsiveness, and window oscillation. Base on better understanding of the design space, our new congestion control algorithm for MPTCP, Balia, generalizes prior algorithms and strikes a good balance between friendliness and responsiveness. Balia has been implemented in the Linux kernel and tested in various scenarios.

7. References

7.1. 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>.

[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>.

7.2. Informative References

[HONDA09] Honda, M., Nishida, Y., Eggert, L., Sarolahti, P., and H. Tokuda, "Multipath congestion control for shared bottleneck", PFLDNeT Workshop, 2009.

[KELLY05] Kelly, F. and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control", ACM SIGCOMM Computer Communication Review, vol. 35, no. 2, pp. 5-12, 2005.

[HAN04] Han, H., Shakkottai, S., Hollot, C., Srikant, R., and D. Towsley, "Overlay tcp for multi-path routing and congestion control", IMA Workshop on Measurements and Modeling of the Internet, 2004.

[KHALILI12] Khalili, R., Gast, N., Popovic, M., Upadhyay, U., and J. Le Boudec, "MPTCP is not Pareto-optimality: Performance issues and a possible solution", ACM CoNext, 2012.

[MPLKI] UCL, Louvain-la-Neuve, Belgium, "MultiPath TCP-Linux kernel implementation", 2014, <http://multipath-tcp.org/>.

[PENG14] Peng, Q., Walid, A., Hwang, J., and S. Low, "Multipath TCP: Analysis, Design, and Implementation", IEEE/ACM Transactions on Networking, vol. PP, no. 99, 2014.

[DUMMYNET]
Carbone, M. and L. Rizzo, "Dummynet revisited", ACM SIGCOMM Computer Communication Review, vol. 40, no. 2, pp. 12-20, 2010.

[NORNET] Gran, E., Dreibholz, T., and A. Kvalbein, "NorNet Core - A multi-homed research testbed", Elsevier Computer Networks, vol. 61, pp. 75-87, 2014.

Authors’ Addresses

Anwar Walid
Bell Labs
600 Mountain Ave
New Providence, NJ, USA

Email: anwar@research.bell-labs.com

Qiuyu Peng
Caltech
Department of Electrical Engineering
Pasadena, CA, USA

Email: qpeng@caltech.edu

Jaehyun Hwang
Samsung Electronics
Advanced Technology Group, Networks Business
Suwon, Gyeonggi-do, Republic of Korea

Email: jhyun.hwang@samsung.com

Steven H. Low
Caltech
Department of Computing + Mathematical Sciences
Department of Electrical Engineering
Pasadena, CA, USA

Email: slow@caltech.edu