

Multipath TCP
INTERNET-DRAFT
Intended Status: Standard Track
Expires: August 21, 2013

Ramin Khalili
T-Labs/TU-Berlin
Nicolas Gast
Miroslav Popovic
Jean-Yves Le Boudec
EPFL-LCA2
February 17, 2013

<Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP>
draft-khalili-mptcp-congestion-control-00

Abstract

This document describes the mechanism of OLIA, the "opportunistic linked increases algorithm". OLIA is a congestion control algorithm for MPTCP. The current congestion control algorithm of MPTCP, LIA, forces a tradeoff between optimal congestion balancing and responsiveness. OLIA's design departs from this tradeoff and provide these properties simultaneously. Hence, it solves the identified performance problems with LIA while retaining non-flappiness and responsiveness behavior of LIA [5].

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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 August 22, 2013.

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
1.1	Requirements Language	4
1.2	Terminology	4
2	The set of best paths and paths with maximum windows	5
3	Opportunistic Linked-Increases Algorithm	6
4	Practical considerations	8
5	Discussion	9
6	References	10
6.1	Normative References	10
6.2	Informative References	10
	Authors' Addresses	11

1 Introduction

The current MPTCP implementation uses a congestion control algorithm called LIA, the "Linked-Increases" algorithm [4]. However, MPTCP with LIA suffers from important performance issues (see [5] and [6]): in some scenarios upgrading TCP users to MPTCP can result in a significant drop in the aggregate throughput in the network without any benefit for anybody. We also show in [5] that MPTCP users could be excessively aggressive toward TCP users.

The design of LIA forces a tradeoff between optimal congestion balancing and responsiveness. Hence, to provide good responsiveness, LIA's current implementation must depart from optimal congestion balancing that leads to the identified problems. In this draft, we introduce OLIA, the "opportunistic linked increases algorithm", as an alternative to LIA. Contrary to LIA, OLIA's design is not based on a trade-off between responsiveness and optimal congestion balancing; it can provide both simultaneously.

Similarly to LIA, OLIA couples the additive increases and uses unmodified TCP behavior in the case of a loss. The difference between LIA and OLIA is in the increase part. OLIA's increase part, Equation (1), has two terms:

- The first term is an adaptation of the increase term of Kelly and Voice's algorithm [8]. This term is essential to provide optimal resource pooling.
- The second term guarantees responsiveness and non-flappiness of OLIA. By measuring the number of transmitted bytes since the last loss, it reacts to events within the current window and adapts to changes faster than the first term.

By adapting the window increases as a function of RTTs, OLIA also compensates for different RTTs. As OLIA is rooted on the optimal algorithm of [8], it provides fairness and optimal congestion balancing. Because of the second term, it is responsive and non-flappy.

OLIA is implemented in the Linux kernel and is now a part of MPTCP's implementation. In [5], we study the performance of MPTCP with OLIA over a testbed, by simulations and by theoretical analysis. We prove theoretically that OLIA is Pareto-optimal and that it satisfies the design goals of MPTCP described in [4]. Hence, it can provide optimal congestion balancing and fairness in the network. Our measurements and simulations indicate that MPTCP with OLIA is as responsive and non-flappy as MPTCP with LIA and that it solves the identified problems with LIA.

The rest of the document provides a description of OLIA. For an analysis of its performance, we refer to [5].

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 RFC 2119 [1].

1.2 Terminology

Regular TCP: The standard version of TCP that operates between a single pair of IP addresses and ports [2].

Multipath TCP: A modified version of the regular TCP that allows a user to spread its traffic across multiple paths.

MPTCP: The proposal for multipath TCP specified in [3].

LIA: The Linked-Increases Algorithm of MPTCP (the congestion control of MPTCP) [4].

OLIA: The Opportunistic Linked-Increases Algorithm for MPTCP proposed in [5].

best_paths: The set of paths that are presumably the best paths for the MPTCP connection.

max_w_paths: The set of paths with largest congestion windows.

collected_paths: The set of paths that are presumably the best paths but do not have largest congestion window (i.e. the paths of best_paths that are not in max_w_paths).

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

MSS_r: The Maximum Segment Size that specifies the largest amount of data can be transmitted by a TCP packet on the path r.

2 The set of best paths and paths with maximum windows

A MPTCP connection has access to one or more paths. Let r be one of these paths. We denote by l_{1r} the number of bytes that were successfully transmitted over path r between the last two losses seen on r , and by l_{2r} the number of bytes that are successfully transmitted over r after the last loss. We denote by $l_r = \max\{l_{1r}, l_{2r}\}$ the smoothed estimation of number of bytes transmitted on path r between last two losses.

l_{1r} and l_{2r} can be measured by using information that is already available to a regular TCP user:

- For each ACK on r : $l_{2r} \leftarrow l_{2r} + (\text{number of bytes that are acknowledged by ACK}),$

- For each loss on r : $l_{1r} \leftarrow l_{2r}$ and $l_{2r} \leftarrow 0.$

l_{1r} and l_{2r} are initially set to zero when the connection is established. If no losses have been observed on r until now, then $l_{1r}=0$ and l_{2r} is the total number of bytes transmitted on r .

Let rtt_r be the round-trip time observed on path r (e.g. the smoothed round-trip time used by regular TCP). We denote by $best_paths$ the set of paths r that have the maximum value of $l_r * l_r / rtt_r$ and by max_w_paths the set of paths with largest congestion window. Also, let $collected_paths$ be the set of paths that are in $best_paths$ but not in max_w_paths . Note that l_{1r} , l_{2r} , l_r , $best_paths$, max_w_paths and $collected_paths$ are all functions of time.

$best_paths$ represents the set of paths that are presumably the best paths for the user: $1/l_r$ can be considered as an estimate of byte loss probability on path r , and hence the rate that path r can provide to a TCP user can be estimated by $(2 * l_r)^{1/2} / rtt_r$. $collected_paths$ is the set of "collected" paths: the paths that are presumably the best paths for the user but that are not yet fully used.

3 Opportunistic Linked-Increases Algorithm

In this section, we introduce OLIA. OLIA is a window-based congestion-control algorithm. It couples the increase of congestion windows and uses unmodified TCP behavior in the case of a loss. OLIA is an alternative for LIA, the current congestion control algorithm of MPTCP.

The algorithm only applies to the increase part of the congestion avoidance phase. The fast retransmit and fast recovery algorithms, as well as the multiplicative decrease of the congestion avoidance phase, are the same as in TCP [2]. We also use a similar slow start algorithm as in TCP, with the modification that we set the ssthresh (slow start threshold) to be 1 MSS if multiple paths are established. In the case of a single path flow, we use the same minimum ssthresh as in TCP (i.e. 2 MSS). The purpose of this modification is to avoid transmitting unnecessary traffic over congested paths when multiple paths are available to a user.

For a path r , we denote by w_r the congestion windows on this path (also called subflow). We denote by MSS_r be the maximum segment size on the path r . We assume that w_r is maintained in bytes.

Our proposed "Opportunistic Linked-Increases Algorithm" (OLIA) must:

- For each ACK on path r , increase w_r by

$$\left(\frac{w_r / rtt_r^2}{(\text{SUM } (w_p / rtt_p))^2} + \frac{\alpha_r}{w_r} \right) \quad (1)$$

multiplied by $MSS_r * \text{bytes_acked}$.

The summation in the denominator of the first term is over all the paths available to the user.

α_r is calculated as follows:

- If r is in `collected_paths`, then

$$\alpha_r = \frac{1/\text{number_of_paths}}{|\text{collected_paths}|}$$

- If r is in `max_w_paths` and if `collected_paths` is not empty, then

$$\alpha_r = - \frac{1/\text{number_of_paths}}{|\text{max_w_paths}|}$$

- Otherwise, $\alpha_r=0$.

$|\text{collected_paths}|$ and $|\text{max_w_paths}|$ are the number of paths in collected_paths and in max_w_paths . Note that the sum of all α_r is equal to 0.

The first term in (1) is an adaptation of Kelly and Voice's increase term [8] and provides the optimal resource pooling (Kelly and Voice's algorithm is based on scalable TCP; the first term in (1) is a TCP compatible version of their algorithm that compensates also for different RTTs). The second term, with α_r , guarantees responsiveness and non-flappiness of our algorithm.

By definition of α_r , if all the best paths have the largest window size, then $\alpha_r=0$ for any r . This is because we already use the capacity available to the user by using all the best path.

If there is any best path with a small window size, i.e. if collected_paths is not empty, then α_r is positive for all r in collected_paths and negative for all r in max_w_paths . Hence, our algorithm increases windows faster on the paths that are presumably best but that have small windows. The increase will be slower on the paths with maximum windows. In this case, OLIA re-forwards traffic from fully used paths (i.e. paths in max_w_paths) to paths that have free capacity available to the users (i.e. paths in collected_paths).

In [4], three goals have been proposed for the design of a practical multipath congestion control algorithm : (1) Improve throughput: a multipath TCP user should perform at least as well as a TCP user that uses the best path available to it. (2) Do no harm: a multipath TCP user should never take up more capacity from any of its paths than a TCP user. And (3) balance congestion: a multipath TCP algorithm should balance congestion in the network, subject to meeting the first two goals.

Our theoretical results in [5] show that OLIA fully satisfies these three goals. LIA, however, fails to fully satisfy the goal (3) as discussed in [4] and [5]. Moreover, in [5], we show through measurements and by simulation that our algorithm is as responsive and non-flappy as LIA and that it can solve the identified problems with LIA.

4 Practical considerations

Calculation of alpha requires performing costly floating point operation whenever an ACK received over path r. In practice, however, we can integrate calculation of alpha and Equation (1) together. Our algorithm can be therefore simplified as the following.

For each ACK on the path r:

- If r is in collected_paths, increase w_r by

$$\frac{w_r/r_{tt_r}^2}{(\sum (w_p/r_{tt_p}))^2} + \frac{1}{w_r * \text{number_of_paths} * |\text{collected_paths}|} \quad (2)$$

multiplied by MSS_r * bytes_acked.

- If r is in max_w_paths and if collected_paths is not empty, increase w_r by

$$\frac{w_r/r_{tt_r}^2}{(\sum (w_r/r_{tt_r}))^2} - \frac{1}{w_r * \text{number_of_paths} * |\text{max_w_paths}|} \quad (3)$$

multiplied by MSS_r * bytes_acked.

- Otherwise, increase w_r by

$$\frac{(w_r/r_{tt_r}^2)}{(\sum (w_r/r_{tt_r}))^2} \quad (4)$$

multiplied by MSS_r * bytes_acked.

Therefore, to compute the increase, we only need to determine the sets collected_paths and max_w_paths when an ACK is received on the path r. We can further simplify the algorithm by updating the sets collected_paths and max_w_paths only once per round-trip time or whenever there is a drop on the path.

We can see from above that in some cases (i.e. when r is max_w_paths and collected_paths is not empty) the increase could be negative. This is a property of our algorithm as in this case OLIA re-forwards traffic from paths in max_w_paths to paths in collected_paths. It is easy to show that using our algorithm, $w_r \geq 1$ for any path r.

5 Discussion

Our results in [5] show that the identified problems with current MPTCP implementation are not due to the nature of a window-based multipath protocol, but rather to the design of LIA. OLIA shows that it is possible to build an alternative to LIA that mitigates these problems and that is as responsive and non-flappy as LIA.

Our proposed algorithm can provide similar resource pooling as Kelly and Voice's algorithm [8] and fully satisfies the design goals of MPTCP described in [4]. Hence, it can provide optimal congestion balancing and fairness in the network. Moreover, it is as responsive and non-flappy as LIA [5].

We therefore believe that IETF should revisit the congestion control part of MPTCP and that an alternative algorithm, such as OLIA, should be considered.

6 References

6.1 Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [3] Ford, A., Raiciu, C., Greenhalgh, A., and M. Handley, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.
- [4] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, October 2011.

6.2 Informative References

- [5] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec. "MPTCP is not Pareto-optimality: Performance issues and a possible solution", ACM CoNext 2012.
- [6] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec. "Performance Issues with MPTCP", draft-khalili-mptcp-performance-issues-02.
- [7] M. Handley, D. Wischik, C. Raiciu, "Coupled Congestion Control for MPTCP", presented at 77th IETF meeting, Anaheim, California.
<<http://www.ietf.org/proceedings/77/slides/mptcp-9.pdf>>.
- [8] Kelly, F. and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control", ACM SIGCOMM CCR vol. 35 num. 2, pp. 5-12, 2005.

Authors' Addresses

Ramin Khalili
T-Labs/TU-Berlin
TEL 3, Ernst-Reuter-Platz 7
10587 Berlin
Germany

Phone: +49 30 8353 58276
EMail: ramin@net.t-labs.tu-berlin.de

Nicolas Gast
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 1254
EMail: nicolas.gast@epfl.ch

Miroslav Popovic
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 6466
EMail: miroslav.popovic@epfl.ch

Jean-Yves Le Boudec
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 6631
EMail: jean-yves.leboudec@epfl.ch

Multipath TCP
INTERNET-DRAFT
Intended Status: Standard Track
Expires: August 21, 2013

Ramin Khalili
T-Labs/TU-Berlin
Nicolas Gast
Miroslav Popovic
Jean-Yves Le Boudec
EPFL-LCA2
February 17, 2013

<Performance Issues with MPTCP>
draft-khalili-mptcp-performance-issues-02

Abstract

We show, by measurements over a testbed and by mathematical analysis, that the current MPTCP suffers from two problems: (P1) Upgrading some TCP users to MPTCP can reduce the throughput of others without any benefit to the upgraded users; and (P2) MPTCP users can be excessively aggressive towards TCP users. We attribute these problems to the "Linked Increases" Algorithm (LIA) of MPTCP [4], and more specifically, to an excessive amount of traffic transmitted over congested paths. Our results show that these problems are important and can be mitigated. We believe that the design of the congestion control of MPTCP should be improved.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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 August 22, 2013.

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1	Introduction	3
1.1	Requirements Language	4
1.2	Terminology	4
2	MPTCP's LIA	5
3	Testbed Setup	6
4	Scenario A: MPTCP can penalize regular TCP users	7
5	Scenario B: MPTCP can penalize other MPTCP users	10
6	Scenario C: MPTCP is excessively aggressive towards TCP users	12
7	Can the suboptimality of MPTCP with LIA be fixed?	14
8	Conclusion	17
9	References	17
9.1	Normative References	17
9.2	Informative References	17
	Authors' Addresses	19

1 Introduction

Regular TCP uses a window-based congestion-control mechanism to adjust the transmission rate of users [2]. It always provides a Pareto-optimal allocation of resources: it is impossible to increase the throughput of one user without decreasing the throughput of another or without increasing the congestion cost [5]. It also guarantees a fair allocation of bandwidth among the users but favors the connections with lower RTTs [6].

Various mechanisms have been used to build a multipath transport protocol compatible with the regular TCP. Inspired by utility maximization frameworks, [7, 8] propose a family of algorithms. These algorithms tend to use only the best paths available to users and are optimal in static settings where paths have similar RTTs. In practice, however, they suffer from several problems [9]. First, they might fail to quickly detect free capacity as they do not probe paths with high loss probabilities sufficiently. Second, they exhibit flappiness: When there are multiple good paths available to a user, the user will randomly flip its traffic between these paths. This is not desirable, specifically, when the achieved rate depends on RTTs, as with regular TCP.

Because of the issues just mentioned, the congestion control part of MPTCP does not follow the algorithms in [7, 8]. Instead, it follows an ad-hoc design based on the following goals [4]. (1) Improve throughput: a multipath TCP user should perform at least as well as a TCP user that uses the best path available to it. (2) Do no harm: a multipath TCP user should never take up more capacity from any of its paths than a regular TCP user. And (3) balance congestion: a multipath TCP algorithm should balance congestion in the network, subject to meeting the first two goals.

MPTCP compensates for different RTTs and solves many problems of multipath transport [10, 11]: It can effectively use the available bandwidth, it improves throughput and fairness compared to independent regular TCP flows in many scenarios, and it solves the flappiness problem.

We show, however, by measurements over our testbed and mathematical analysis, that MPTCP still suffers from the following problems:

(P1) Upgrading some regular TCP users to MPTCP can reduce the throughput of other users without any benefit to the upgraded users. This is a symptom of non-Pareto optimality.

(P2) MPTCP users can be excessively aggressive towards TCP users.

We attribute these problems to the "Linked Increases" Algorithm (LIA) of MPTCP [4] and specially to an excessive amount of traffic transmitted over congested paths.

These problems indicate that LIA fails to fully satisfy its design goals, especially goal number 3. The design of LIA forces a trade off between optimal resource pooling and responsiveness, it cannot provide both at the same time. Hence, to provide good responsiveness, LIA's current implementation must depart from Pareto-optimality, which leads to problems (P1) and (P2).

This document provides a number of examples and scenarios (Sections 4 to 6) in which MPTCP with LIA exhibits problems (P1) and (P2). Our results show that the identified problems with LIA are important. Moreover, we show in Section 7 that these problems are not due to the nature of a window-based multipath protocol, but rather to the design of LIA; it is possible to build an alternative to LIA that mitigates these problems and is as responsive and non-flappy as LIA. Hence, we believe that the design of the congestion control of MPTCP should be improved.

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 RFC 2119 [1].

1.2 Terminology

Regular TCP: The standard version of TCP that operates between a single pair of IP addresses and ports [2].

Multipath TCP: A modified version of the regular TCP that allows a user to spread its traffic across multiple paths.

MPTCP: The proposal for multipath TCP specified in [3].

LIA: The "Linked Increases" Algorithm of MPTCP (the congestion control of MPTCP) [4].

OLIA: The Opportunistic "Linked Increases" Algorithm for MPTCP proposed in [12].

RTT: The Round-Trip Time seen by a user.

MSS: The Maximum Segment Size that specifies the largest amount of data can be transmitted by a TCP packet.

AP: Access Point.

2 MPTCP's LIA

The design of the congestion control algorithm of MPTCP has been described in details by Mark Handly et al. at the 77th IETF meeting in Anaheim [13].

The actual implementation of the congestion control algorithm, called "Linked Increases" Algorithm (LIA), increases the window size w_r by a quantity proportional to w_r / w_{tot} for each ACK received on path r (see [13, slide 3]). w_{tot} is the sum of the congestion windows over all paths available to the user.

LIA is one of a family of multipath congestion control algorithms that can be indexed by a parameter $0 < \phi < 2$ (see [13, slide 9]). These algorithms results in transmitting a traffic proportional to $(1/p_r)^{1/\phi}$ a path that has a probability loss p_r :

- * If ϕ is very close to zero, then only path with smallest loss rate will be used (e.g. if $p_2 > p_1$ then $w_2 = 0$ and $w_1 > 0$). It correspond to the fully coupled algorithm of [7] and is flappy [9].

- * $\phi = 2$ corresponds to having uncoupled TCP flows on each of the path. It is very responsive and non-flappy, but does not balance congestion in the network.

- * $\phi = 1$ correspond to LIA and is described in RFC6356 [4]. It provides a compromise between congestion balancing and responsiveness.

3 Testbed Setup

To investigate the behavior of MPTCP in practice, three testbed topologies are created representing scenarios in Sections 4, 5, and 6. We use server-client PCs that run MPTCP enabled Linux kernels. We use MPTCP for the Linux kernel 3.0.0 released in February 2012. In all our scenarios, laptop PCs are used as routers. We install "Click Modular Router" software [14] on the routers to implement topologies with different characteristics. Iperf is used to create multiple connections.

In our scenarios, we are able to implement links with configurable bandwidth and delay. We are also able to set the parameters of the RED queues following the structure in [15]. For a 10 Mbps link, we set the packet loss probability equal to 0 up to a queue size of 25 Maximum Segment Size (MSS). Then it grows linearly to the value 0.1 at 50 MSS. It again increases linearly up to 1 for 100 MSS. The parameters are proportionally adapted when the link capacity changes.

To verify that the problems observed are caused by the congestion-control algorithm of MPTCP and not by some unknown problems in our testbed, we perform a mathematical analysis of MPTCP. This analysis is based on the fix point analysis of MPTCP. As we will see, our mathematical results confirm our measurement results. The details of these mathematical analyses are available in [12].

4 Scenario A: MPTCP can penalize regular TCP users

Consider a network with two types of users. There are N_1 users of type1, each with a high-speed private connection. These users access different files on a media streaming server. The server has a network connection with capacity limit of N_1C_1 Mbps. Type1 users can activate a second connection through a shared AP by using MPTCP. There are also N_2 users of type2; they are connected to the shared AP. They download their contents from the Internet. The shared AP has a capacity of N_2C_2 Mbps.

We implement this scenario in a testbed similarly to Figure 1. Within router-PCs R1 and R2, we implement links with capacities N_1C_1 and N_2C_2 and RTTs of 150 ms (including queuing delay), modeling the bottleneck at the server side and the shared AP, respectively. High speed connections are used to implement private connections of type1 users.

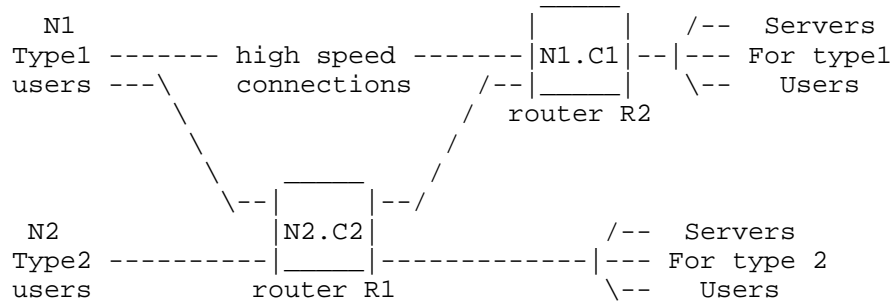


Figure 1: Testbed implementation of Scenario A: router R1 implements the bottleneck at the server side and router R2 implements the shared AP bottleneck.

When type1 users use only their private connection, each type1 user receives a rate of C_1 and each type2 user receives a rate of C_2 . By upgrading type1 users to MPTCP, we observe, through measurement and by mathematical analysis, that the throughput of type2 users significantly decreases. However, type1 users receive the same throughput as before, because the bottleneck for their connections is at the server side. We report the throughput received by users after upgrading type1 users to MPTCP on Table 1 for $C_1=C_2=1$ Mbps. For each case, we take 5 measurements. In each case, the confidence intervals are very small (less than 0.01Mbps).

In [12 Section3.2], we provide a mathematical analysis of MPTCP that confirm our measurements: The predicted rate of type1 users is always 1 and the predicted rate for type2 users is, respectively, 0.74 when $N_1=N_2=10$ and 0.50 when $N_1=30$ and $N_2=10$.

		Type1 users are single path (measurement)	Type1 users are multipath		
			MPTCP (meas.)	optimal algorithm	
				with p. cost (theory)	w/out p. cost (theory)
N1=10	type1	0.98	0.96	1	1
N2=10	type2	0.98	0.70	0.94	1
N1=30	type1	0.98	0.98	1	1
N2=10	type2	0.98	0.44	0.8	1

meas.=measurements, p.=probing, w/out=without, Values are in Mbps.

Table 1: Throughput obtained by type1 and type2 users in Scenario A: upgrading type1 users to MPTCP decrease the throughput of type2 with no benefit for type1 users. The problem is much less critical using optimal algorithm with probing cost.

We observe that MPTCP exhibits problem (P1): upgrading type1 users to MPTCP penalizes type2 users without any gain for type1 users. As the number of type1 users increases, the throughput of type2 users decreases, but the throughput of type1 users does not change as it is limited by the capacity of the streaming server. For $N1=N2$, type2 users see a decrease of 30%. When $N1=3N2$, this decrease is 55%.

We compare the performance of MPTCP with two theoretical baselines. They serve as references to see how far from the optimum MPTCP with LIA is. We show in Section 7 that it is possible to replace LIA by an alternative that keeps the same non-flappiness and responsiveness and performs closer to the optimum.

The first baseline is an algorithm that provides theoretical optimal resource pooling in the network (as discussed in [7] and several other theoretical papers). We refer to it as "optimal algorithm without probing cost".

In practice, however, the value of the congestion windows are bounded below by 1 MSS. Hence, with a window-based congestion-control algorithm, a minimum probing traffic of 1 MSS per RTT will be sent over a path. We introduce a second theoretical baseline, called "optimal algorithm with probing cost"; it provides optimal resource pooling in the network given that a minimum probing traffic of 1 MSS per RTT is sent over each path.

We show the performance of these optimal algorithms in Table 1. Using an optimal algorithm with probing cost, the entire capacity of the shared AP is allocated to type2 users. Hence, all the users in the network receives a throughput of 1 Mbps. By using an optimal algorithm with probing cost, type1 users will send only 1MSS per RTT over the shared AP. Hence, we observe a decrease on the throughput of type2 users. However, the decrease is much less than what we observe using MPTCP. The performance of our proposed alternative to LIA is shown in Section 7, Table 4.

This performance problem of MPTCP can be explained by the fact that LIA does not fully balance congestion. For $N1=N2$, we observe through measurements that $p1=0.009$ and $p2=0.02$ (the probability of losses at routers R1 and R2). For $N1=3N2$, the value of $p1$ remains almost the same and $p2$ increases to $p2=0.05$. LIA excessively increases congestion on the shared AP, which is not in compliance with goal 3. In [12], we propose an alternative to LIA. Using this algorithm, we have $p1=0.009$ and $p2=0.012$ for $N1=10$ and 0.018 for $N1=30$. Hence, it is possible to provide a better congestion balancing in the network.

Because of some practical issues, we did not study larger number of connections in our testbed. However, we have mathematical results (using LIA's loss-throughput formula [12]) as well as simulation results (using a flow-level simulator) that confirm our observation for larger number of connections.

5 Scenario B: MPTCP can penalize other MPTCP users

Consider a multi-homing scenario as follows. We have four Internet Service Providers (ISPs), X, Y, Z, and T. Y is a local ISP in a small city, which connects to the Internet through Z. X, Z, and T are nation-wide service providers and are connected to each other through high speed links. X provides Internet services to users in the city and is a competitor of Y. They have access capacity limits of CX, CY, CZ, and CT. Z and T are hosts of different video streaming servers.

There are two types of users: Blue users download contents from servers in Z and Red users download from servers in ISP T. To increase their reliability, Blue users use multi-homing and are connected to both ISPs X and Y. Red users can connect either only to Y or to both X and Y.

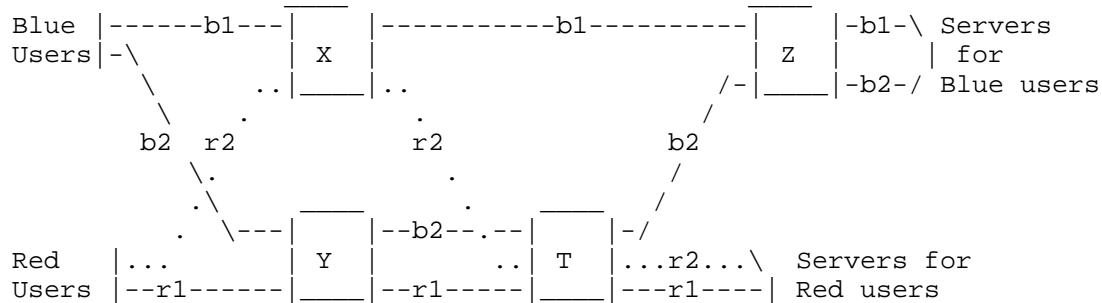


Figure 2. Testbed implementation of Scenario B. Blue users transmit over path b1 and b2. Red users use path r1, but can upgrade to MPTCP by establishing a second connection through path r2.

We implement the scenario in a testbed similar to Figure 2. b1 and b2 are the paths available to Blue users. Red users use the path r1, but can upgrade to MPTCP by establishing a second connection through path r2. The measurement results are reported in Table 2 for a setting with CX=27, CT=36, and CY=CZ=100, all in Mbps, where we have 15 Red and 15 Blue users. RTTs are around 150 ms (including queuing delay) over all paths. We also show the performance of theoretically optimal algorithms with and without probing cost.

We observe that when Red users only connect to ISP Y, the aggregate throughput of users is close to the cut-set bound, 63 Mbps. However, Blue users get a higher share of the network bandwidth. Now, let's consider that Red users upgrade to MPTCP by establishing a second connection through X (showed by pointed-line in Figure 2). Our results in Table 1 show that Red users do not receive any higher throughput. However, the average rate of Blue users drops by 20%,

which results in a drop of 13% in aggregate throughput.

	Red users are single-path			Red users are multipath		
	Blue users use			Blue and Red users use		
	MPTCP	optimal algorithm		MPTCP	optimal algorithm	
		with p.	w/out p.		with p.	w/out p.
		cost	cost		cost	cost
	(meas.)	(theory)	(theory)	(meas.)	(theory)	(theory)
Red users	1.5	2.1	2.1	1.4	2.04	2.1
Blue users	2.5	2.1	2.1	2.0	2.04	2.1

meas.=measurements, p.=probing, w/out=without, Values are in Mbps.

Table 2: Throughput received by users before and after upgrading Red users to MPTCP. We have 15 Red and 15 Blue users. By upgrading Red users to MPTCP, the aggregate throughput of users decreases by 13% with no benefit for Red users.

In [12 Section 3.3], we also provide a mathematical analysis of MPTCP. Our mathematical results predict that by upgrading the Red user to MPTCP the rate of Blue users will be reduced by 21%. This results in 14% decrease in the aggregate throughput. Hence, our mathematical results confirm our observations from the measurement. Similar behavior is predicted for other values of CX and CT [12 Figure 4a].

Using an optimal algorithm without probing cost, Red users transmit only over path r1 and Blue users split their traffic over paths b1 and b2 to equalize the rate of blue and red users. Upgrading Red users to multipath does not change the allocation. Hence, we observe no decrease in the aggregate throughput and the rate of each user. By using an optimal algorithm with probing cost, the rate of Blue and Red users decreases by 3% when we upgrade the Red users to multipath users since red users are forced to send 1 MSS per RTT over path r2. This decrease is much less than what we observe using MPTCP with LIA. The performance of our proposed alternative to LIA is shown in Section 7, Table 5.

Similarly to Scenario A, the problem can be attributed to the excessive amount of traffic sent over the congested paths. This illustrates that MPTCP fails to balance the congestion in the network.

6 Scenario C: MPTCP is excessively aggressive towards TCP users

Consider a scenario with N_1 multipath users, N_2 single-path users, and two APs with capacities N_1C_1 and N_2C_2 Mbps. Multipath users connect to both APs and they share AP2 with single-path users. The users download their contents from the Internet. This scenario is implemented in a testbed similar to Figure 3.

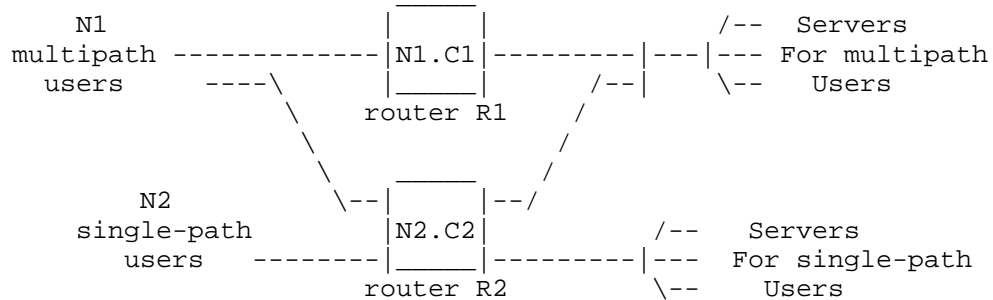


Figure 3: Testbed implementation of Scenario C: routers R1 and R2 implement AP1 and AP2 with capacities N_1C_1 and N_2C_2 Mbps.

If the allocation of rates is proportionally fair, multipath users will use AP2 only if $C_1 < C_2$ and all users will receive the same throughput. When $C_1 > C_2$, a fair multipath user will not transmit over AP2. However, using MPTCP with LIA, multipath users receive a disproportionately larger share of bandwidth.

We implement the scenario in our testbed and measure the performance of MPTCP with LIA. We report the throughput received by single-path and multipath users in Table 3. We present the results for $N_2=10$ and two values of $N_1=10$ and 30, where $C_1=C_2=1$ Mbps. RTTs are around 150 ms (including queuing delay). We also present the performance of optimal (proportionally fair) algorithms.

As $C_1=C_2$, for any fairness criterion, multipath users should not use AP2. Our results show that, MPTCP users are disproportionately aggressive and exhibit problem (P2). Hence, single-path users receive a much smaller share than they should. For $N_1=N_2$, single-path users see a decrease of about 30% in their received throughput compared to a fair allocation. When $N_1=3N_2$, this decrease is around 55%.

These measurements are confirmed by our mathematical analysis as shown in [12 Section 3.4]. The predicted rate of type1 users is 1.3 for $N_1=N_2=10$ and is 1.17 when $N_1=30$ and $N_2=10$. For type2 users, the predicted rate is 0.7 when $N_1=N_2=10$ and 0.48 when $N_1=30$ and $N_2=10$.

		multipath users use MPTCP (measurement)	multipath users use optimal algorithm with p. cost (theory)	w/out p. cost (theory)
N1=10	multipath users	1.3	1.04	1
N2=10	single-path users	0.68	0.94	1
N1=30	multipath users	1.19	1.04	1
N2=10	single-path users	0.38	0.8	1

p.=probing, w/out=without, Values are in Mbps.

Table3: Throughput obtained by single-path and multipath users in Scenario C: MPTCP is excessively aggressive toward TCP users and performs far from how an optimal algorithm would perform.

An optimal algorithm with probing cost provide a proportional fairness among the users. By using an optimal algorithm with probing cost, single-path users receive a rate less than what a proportionally fair algorithm will provide them. However, as we observe, the problem is much less critical compared to the case we use MPTCP. The performance of our proposed alternative to LIA is shown in Section 7, Table 6.

These results clearly show that MPTCP suffers from fairness issues. The problem occurs because LIA fails to fully satisfy goal 3. As in Scenarios A and B, MPTCP sends an excessive amount of traffic over the congested paths.

7 Can the suboptimality of MPTCP with LIA be fixed?

We have shown in Section 4, 5, and 6 that MPTCP with LIA performs far behind an optimal algorithm. The question is, "is it possible to modify the congestion control algorithm of MPTCP to perform closer to the optimum". To answer this question, we implement a new congestion control algorithm for MPTCP, called Opportunistic "Linked Increases" Algorithm (OLIA). OLIA is described in [13] and its performance is analyzed in [12]. In this section, we show that in Scenarios A, B and C OLIA performs close to an optimal algorithm with probing cost. Moreover, as shown in [12, Sections 4.3 and 6.2], OLIA keeps the same non-flappiness and responsiveness as LIA.

Contrary to LIA, OLIA's design is not based on a trade-off between responsiveness and optimal resource pooling. OLIA couples the additive increases and uses unmodified TCP behavior in the case of a loss. The difference between LIA and OLIA is in the increase part. OLIA's increase part has two terms:

- * The first term is an adaptation of the increase term of the optimal algorithm in [7]. This term is essential to provide congestion balancing and fairness.
- * The second term guarantees responsiveness and non-flappiness of OLIA. By measuring the number of transmitted bytes since the last loss, it reacts to events within the current window and adapts to changes faster than the first term.

Because OLIA is rooted in the optimal algorithm of [7], it can provide fairness and congestion balancing. Because of the second term, it is responsive and non-flappy.

We implemented OLIA by modifying the congestion control part of the MPTCP implementation based on the Linux Kernel 3.0.0. For conciseness, we do not describe OLIA in this paper and refer to [12] for details about the algorithm and its implementation.

We study the performance of MPTCP with OLIA through measurements in Scenarios A, B, and C. The results are reported in Tables 4, 5 and 6. We compare the performance of our algorithm with MPTCP with LIA and with optimal algorithms. We observe that in all cases, MPTCP with OLIA provide a significant improvement over MPTCP with LIA. Moreover, it performs close to an optimal algorithm with probing cost.

		Type1 users are single path (measurement)	Type1 users are multipath			
			MPTCP w. OLIA [with LIA] (measurement)	optimal algorithm		
				with p. cost (theory)	w/out p. cost (theory)	
N1=10	type1	0.98	0.98 [0.96]	1	1	
N2=10	type2	0.98	0.86 [0.70]	0.94	1	
N1=30	type1	0.98	0.98 [0.98]	1	1	
N2=10	type2	0.98	0.75 [0.44]	0.8	1	

p.=probing, w.=with, w/out=without, Values are in Mbps.

Table 4. Performance of MPTCP with OLIA compared to MPTCP with LIA in scenario A. We show the throughput obtained by users before and after upgrading type1 users to MPTCP. The values in brackets are the values for MPTCP with LIA (taken from table 1).

		Red users are single-path			Red users are multipath		
		Blue users use			Blue and Red users use		
		MPTCP with OLIA [with LIA] (meas.)	optimal algorithm with p. cost (theory)	w/out p. cost (theory)	MPTCP with OLIA [with LIA] (meas.)	optimal algorithm with p. cost (theory)	w/out p. cost (theory)
Red users		1.8 [1.5]	2.1	2.1	1.7 [1.4]	2.04	2.1
Blue users		2.2 [2.5]	2.1	2.1	2.2 [2.0]	2.04	2.1

meas.=measurements, p.=probing, w/out=without, Values are in Mbps.

Table 5. Performance of MPTCP with OLIA compared to MPTCP with LIA in scenario B. We show the throughput received by users before and after upgrading Red users to MPTCP. The values in brackets are the values for MPTCP with LIA (taken from Table 2).

		multipath users use MPTCP with OLIA [with LIA] (measurement)		multipath users use optimal algorithm with probing cost (theory)		w/out probing cost (theory)	
N1=10	multipath users	1.11	[1.30]	1.04		1	
N2=10	single-path users	0.88	[0.68]	0.94		1	
N1=10	multipath users	1.1	[1.19]	1.04		1	
N2=10	single-path users	0.72	[0.38]	0.8		1	

meas.=measurements, w/out=without, Values are in Mbps.

Table 6. Performance of MPTCP with OLIA compared to MPTCP with LIA in scenario C. We show the throughput obtained by single-path and multipath users. The values in brackets are the values for MPTCP with LIA (taken from Table 3).

The results show that it is possible to perform close to an optimal algorithm with probing cost by using a TCP-like algorithm. Moreover, we show in [12, Section 4.3 and Section 6.2] that MPTCP with OLIA is as responsive and non-flappy as MPTCP with LIA. This shows that it is possible to build a practical multi-path congestion control that works close to an optimal algorithm with probing cost.

8 Conclusion

We have shown that MPTCP with LIA suffers from important performance issues. Moreover, it is possible to build an alternative to LIA that performs close to an optimal algorithm with probing cost while being as responsive and non-flappy as LIA. Hence, we believe that IETF should revisit the congestion control part of MPTCP and that an alternative algorithm, such as OLIA [12], should be considered.

9 References

9.1 Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, September 2009.
- [3] Ford, A., Raiciu, C., Greenhalgh, A., and M. Handley, "Architectural Guidelines for Multipath TCP Development", RFC 6182, March 2011.
- [4] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, October 2011.

9.2 Informative References

- [5] F.P. Kelly. Mathematical modelling of the internet. Mathematics unlimited-2001 and beyond.
- [6] F.P. Kelly, A.K. Maulloo, and D.K.H. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research society, 49, 1998.
- [7] Kelly, F. and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control", ACM SIGCOMM CCR vol. 35 num. 2, pp. 5-12, 2005.
- [8] H. Han, S. Shakkottai, CV Hollot, R. Srikant, and D. Towsley. "Multi-path tcp: a joint congestion control and routing scheme to exploit path diversity in the internet", Trans. on Net., 14, 2006.
- [9] D. Wischik, M. Handly, and C. Raiciu. "Control of multipath tcp and optimization of multipath routing in the

internet", NetCOOP, 2009.

- [10] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handly. "Design, implementation and evaluation of congestion control for multipath tcp", Usenix NSDI, 2011.
- [11] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handly. "Improving datacenter performance and robustness with multipath tcp", ACM Sigcomm, 2011.
- [12] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec. "MPTCP is not Pareto-optimality: Performance issues and a possible solution", ACM CoNext 2012.
- [13] M. Handly, D. Wischik, C. Raiciu, "Coupled Congestion Control for MPTCP", presented at 77th IETF meeting, Anaheim, California.
<<http://www.ietf.org/proceedings/77/slides/mptcp-9.pdf>>.
- [14] B. Chen J. Jannotti E. Kohler, R. Morris and M. F. Kaashoek. "The click modular router", Trans. Comput. Syst., 18, August 2000.
- [15] S. Floyd and V. Jacobson. "Random early detection gateways for congestion avoidance", Trans. on Net., 1, August, 1993.
- [16] R. Khalili, N. Gast, M. Popovic, J.-Y. Le Boudec, "Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP", draft-khalili-mptcp-congestion-control-00.

Authors' Addresses

Ramin Khalili
T-Labs/TU-Berlin
TEL 3, Ernst-Reuter-Platz 7
10587 Berlin
Germany

Phone: +49 30 8353 58276
EMail: ramin@net.t-labs.tu-berlin.de

Nicolas Gast
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 1254
EMail: nicolas.gast@epfl.ch

Miroslav Popovic
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 6466
EMail: miroslav.popovic@epfl.ch

Jean-Yves Le Boudec
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 6631
EMail: jean-yves.leboudec@epfl.ch

Network Working Group
Internet Draft
Intended status: Experimental
Expires: August 2014

A. Shpiner
Technion - Israel Institute of Technology
R. Tse
C. Schelp
PMC-Sierra
T. Mizrahi
Marvell
February 11, 2014

Multi-Path Time Synchronization
draft-shpiner-multi-path-synchronization-03.txt

Abstract

Clock synchronization protocols are very widely used in IP-based networks. The Network Time Protocol (NTP) has been commonly deployed for many years, and the last few years have seen an increasingly rapid deployment of the Precision Time Protocol (PTP). As time-sensitive applications evolve, clock accuracy requirements are becoming increasingly stringent, requiring the time synchronization protocols to provide high accuracy. Slave Diversity is a recently introduced approach, where the master and slave clocks (also known as server and client) are connected through multiple network paths, and the slave combines the information received through all paths to obtain a higher clock accuracy compared to the conventional one-path approach. This document describes a multi-path approach to PTP and NTP over IP networks, allowing the protocols to run concurrently over multiple communication paths between the master and slave clocks. The multi-path approach can significantly contribute to clock accuracy, security and fault protection. The Multi-Path Precision Time Protocol (MPPTP) and Multi-Path Network Time Protocol (MPNTP) define an additional layer that extends the existing PTP and NTP without the need to modify these protocols. MPPTP and MPNTP also allow backward compatibility with nodes that do not support the multi-path extension.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on August 11, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Conventions Used in this Document	5
2.1. Abbreviations	5
2.2. Terminology	5
3. Multiple Paths in IP Networks	5
3.1. Load Balancing	5
3.2. Using Multiple Paths Concurrently	5
3.3. Two-Way Paths	6
4. Solution Overview	6
4.1. Path Configuration and Identification	6
4.2. Combining	7
5. Multi-Path Time Synchronization Protocols over IP Networks ...	7
5.1. Single-Ended Multi-Path Synchronization	8
5.1.1. Single-Ended MPPTP Synchronization Message Exchange	8
5.1.2. Single-Ended MPNTP Synchronization Message Exchange	9
5.2. Dual-Ended Multi-Path Synchronization	10
5.2.1. Dual-Ended MPPTP Synchronization Message Exchange .	10

5.2.2. Dual-Ended MPNTP Synchronization Message Exchange .	11
5.3. Using Traceroute for Path Discovery	12
5.4. Using Unicast Discovery for MPPTP	12
6. Combining Algorithm	13
6.1. Averaging	13
6.2. Switching / Dynamic Algorithm	13
6.3. NTP-like Filtering-Clustering-Combining Algorithm	13
7. Security Considerations	14
8. IANA Considerations	14
9. Acknowledgments	14
10. References	14
10.1. Normative References	14
10.2. Informative References	15

1. Introduction

The two most common time synchronization protocols in IP networks are the Network Time Protocol [NTP], and the Precision Time Protocol (PTP), defined in the IEEE 1588 standard [IEEE1588].

The accuracy of the time synchronization protocols directly depends on the stability and the symmetry of propagation delays on both directions between the master and slave clocks. Depending on the nature of the underlying network, time synchronization protocol packets can be subject to variable network latency or path asymmetry (e.g. [ASYMMETRY], [ASYMMETRY2]). As time sensitive applications evolve, accuracy requirements are becoming increasingly stringent.

Using a single network path in a clock synchronization protocol closely ties the slave clock accuracy to the behavior of the specific path, which may suffer from temporal congestion, faults or malicious attacks. Relying on multiple clock servers as in NTP solves these problems, but requires active maintenance of multiple accurate sources in the network, which is not always possible. The usage of Transparent Clocks (TC) in PTP solves the congestion problem by eliminating the queueing time from the delay calculations, but requires the intermediate routers and switches to support the TC functionality, which is not always the case.

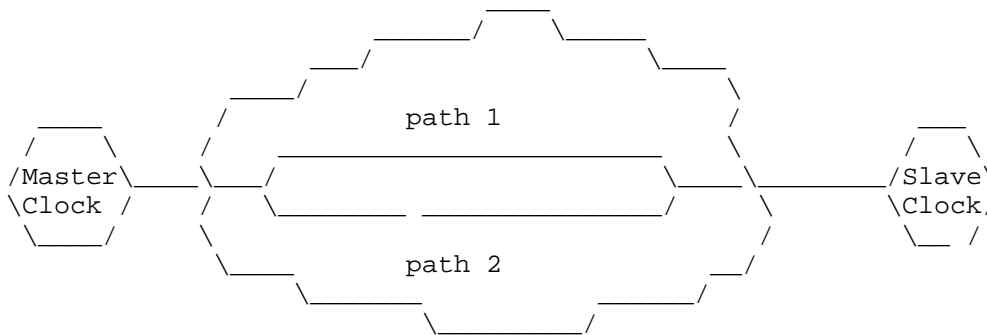


Figure 1 Multi-Path Connection

Since master and slave clocks are often connected through more than one path in the network, as shown in Figure 1, [SLAVEDIV] suggested that a time synchronization protocol can be run over multiple paths, providing several advantages. First, it can significantly increase the clock accuracy as shown in [SLAVEDIV]. Second, this approach provides additional security, allowing mitigating man-in-the-middle attacks against the time synchronization protocol [DELAY-ATT]. Third, using multiple paths concurrently provides an inherent failure protection mechanism.

This document introduces Multi-Path PTP (MPPTP) and Multi-Path NTP (MPNTP), respectively. These extensions are defined at the network layer and do not require any changes in the PTP or in the NTP protocols.

MPPTP and MPNTP are defined over IP networks. As IP networks typically combine ECMP routing, this property is leveraged for the multiple paths used in MPPTP and MPNTP. The key property of the multi-path extension is that clocks in the network can use more than one IP address. Each {master IP, slave IP} address pair defines a path. Depending on the network topology and configuration, the IP combination pairs can form multiple diverse paths used by the multi-path synchronization protocols.

This document introduces two variants for each of the two multi-path protocols; a variant that requires both master and slave nodes to support the multi-path protocol, referred to as the dual-ended variant, and a backward compatible variant that allows a multi-path

clock to connect to a conventional single-path clock, referred to as the single-ended variant.

2. Conventions Used in this Document

2.1. Abbreviations

ECMP	Equal Cost Multiple Path
LAN	Local Area Network
MPNTP	Multi-Path Network Time Protocol
MPPTP	Multi-Path Precision Time Protocol
NTP	Network Time Protocol
PTP	Precision Time Protocol

2.2. Terminology

In the NTP terminology, a time synchronization protocol is run between a client and a server, while PTP uses the terms master and slave. Throughout this document, the sections that refer to both PTP and NTP generically use the terms master and slave.

3. Multiple Paths in IP Networks

3.1. Load Balancing

Traffic sent across IP networks is often load balanced across multiple paths. The load balancing decisions are typically based on packet header fields: source and destination addresses, Layer 4 ports, the Flow Label field in IPv6, etc. Three common load balancing criteria are per-destination, per-flow and per-packet. The per-destination load balancers take a load balancing decision based on the destination IP address. Per-flow load balancers use various fields in the packet header, e.g., IP addresses and Layer 4 ports, for the load balancing decision. Per-packet load balancers use flow-blind techniques such as round-robin without basing the choice on the packet content.

3.2. Using Multiple Paths Concurrently

To utilize the diverse paths that traverse per-destination load-balancers or per-flow load-balancers, the packet transmitter can vary

the IP addresses in the packet header. The analysis in [PARIS2] shows that a significant majority of the flows on the internet traverse per-destination or per-flow load-balancing. It presents statistics that 72% of the flows traverse per-destination load balancing and 39% of the flows traverse per-flow load-balancing, while only a negligible part of the flows traverse per-packet load balancing. These statistics show that the vast majority of the traffic on the internet is load balanced based on packet header fields.

The approaches in this draft are based on varying the source and destination IP addresses in the packet header. Possible extensions have been considered that also vary the UDP ports. However some of the existing implementations of PTP and NTP use fixed UDP port values in both the source and destination UDP port fields, and thus do not allow this approach.

3.3. Two-Way Paths

A key property of IP networks is that packets forwarded from A to B do not necessarily traverse the same path as packets from B to A. Thus, we define a two-way path for a master-slave connection as a pair of one-way paths: the first from master to slave and the second from slave to master.

If possible, a traffic engineering approach can be used to verify that time synchronization traffic is always forwarded through bidirectional two-way paths, i.e., that each two-way path uses the same route on the forward and reverse directions, thus allowing propagation time symmetry. However, in the general case two-way paths do not necessarily use the same path for the forward and reverse directions.

4. Solution Overview

The multi-path time synchronization protocols we present are comprised of two building blocks; one is the path configuration and identification, and the other is the algorithm used by the slave to combine the information received from the various paths.

4.1. Path Configuration and Identification

The master and slave clocks must be able to determine the path of transmitted protocol packets, and to identify the path of incoming protocol packets. A path is determined by a {master IP, slave IP} address pair. The synchronization protocol message exchange is run independently through each path.

Each IP address pair defines a two-way path, and thus allows the clocks to bind a transmitted packet to a specific path, or to identify the path of an incoming packet.

If possible, the routing tables across the network should be configured with multiple traffic engineered paths between the pair of clocks. By carefully configuring the routers in such networks it is possible to create diverse paths for each of the IP address pairs between two clocks in the network. However, in public and provider networks the load balancing behavior is hidden from the end users. In this case the actual number of paths may be less than the number of IP address pairs, since some of the address pairs may share common paths.

4.2. Combining

Various methods can be used for combining the time information received from the different paths. This document surveys several combining methods in Section 5.4. The output of the combining algorithm is the accurate time offset.

5. Multi-Path Time Synchronization Protocols over IP Networks

This section presents two variants of MPPTP and MPNTP; single-ended multi-path time synchronization and dual-ended multi-path time synchronization. In the first variant, the multi-path protocol is run only by the slave and the master is not aware of its usage. In the second variant, all clocks must support the multi-path protocol.

The dual-ended protocol provides higher path diversity by using multiple IP addresses at both ends, the master and slave, while the single-ended protocol only uses multiple addresses at the slave. On the other hand, the dual-ended protocol can only be deployed when both the master and the slave support this protocol. Dual-ended and single-ended protocols can co-exist in the same network. Each slave selects the connection(s) it wants to make with the available masters. A dual-ended slave could switch to single-ended mode if it does not see any dual-ended masters available. A single-ended slave could connect to a single IP address of a dual-ended master.

Multi-path time synchronization, in both variants, requires clocks to use multiple IP addresses. If possible, the set of IP addresses for each clock should be chosen in a way that enables the establishment of paths that are the most different. It is applicable if the load balancing rules in the network are known. Using multiple IP addresses introduces a tradeoff. A large number of IP addresses allows a large number of diverse paths, providing the advantages of slave diversity

discussed in Section 1. On the other hand, a large number of IP addresses is more costly, requires the network topology to be more redundant, and exacts extra management overhead.

The descriptions in this section refer to the end-to-end scheme of PTP, but are similarly applicable to the peer-to-peer scheme. The MPNTP protocol described in this document refers to the NTP client-server mode, although the concepts described here can be extended to include the symmetric variant as well.

Multi-path synchronization protocols by nature require protocol messages to be sent as unicast. Specifically in PTP, the following messages must be sent as unicast in MPPTP: Sync, Delay_Req, Delay_Resp, PDelay_Req, PDelay_Resp, Follow_Up, and PDelay_Resp_Follow_Up. Note that [IEEE1588] allows these messages to be sent either as multicast or as unicast.

5.1. Single-Ended Multi-Path Synchronization

In the single-ended approach, only the slave is aware of the fact that multiple paths are used, while the master is agnostic to the usage of multiple paths. This approach allows a hybrid network, where some of the clocks are multi-path clocks, and others are conventional one-path clocks. A single-ended multi-path clock presents itself to the network as N independent clocks, using N IP addresses, as well as N clock identity values (in PTP). Thus, the usage of multiple slave identities by a slave clock is transparent from the master's point of view, such that it treats each of the identities as a separate slave clock.

5.1.1. Single-Ended MPPTP Synchronization Message Exchange

The single-ended MPPTP message exchange procedure is as follows.

- o Each single-ended MPPTP clock has a fixed set of N IP addresses and N corresponding clockIdentities. Each clock arbitrarily defines one of its IP addresses and clockIdentity values as the clock primary identity.
- o A single-ended MPPTP port sends Announce messages only from its primary identity, according to the BMC algorithm.
- o The BMC algorithm at each clock determines the master, based on the received Announce messages.

- o A single-ended MPPTP port that is in the 'slave' state uses unicast negotiation to request the master to transmit unicast messages to each of the N slave clock identities. The slave port periodically sends N Signaling messages to the master, using each of its N identities. The Signaling message includes the REQUEST_UNICAST_TRANSMISSION_TLV.
- o The master periodically sends unicast Sync messages from its primary identity, identified by the sourcePortIdentity and IP address, to each of the slave identities.
- o The slave, upon receiving a Sync message, identifies its path according to the destination IP address. The slave sends a Delay_Req unicast message to the primary identity of the master. The Delay_Req is sent using the slave identity corresponding to the path the Sync was received through. Note that the rate of Delay_Req messages may be lower than the Sync message rate, and thus a Sync message is not necessarily followed by a Delay_Req.
- o The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the IP address and sourcePortIdentity from the Delay_Req message.
- o Upon receiving the Delay_Resp message, the slave identifies the path using the destination IP address and the requestingPortIdentity. The slave can then compute the corresponding path delay and the offset from the master.
- o The slave combines the information from all negotiated paths.

5.1.2. Single-Ended MPNTP Synchronization Message Exchange

The single-ended MPNTP message exchange procedure is as follows.

- o A single-ended MPNTP client has N separate identities, i.e., N IP addresses. The assumption is that the server information, including its IP address is known to the NTP clients.
- o A single-ended MPNTP client initiates the NTP protocol with an NTP server N times, using each of its N identities.
- o The NTP protocol is maintained between the server and each of the N client identities.
- o The client sends NTP messages to the master using each of its N identities.

- o The server responds to the client's NTP messages using the IP address from the received NTP packet.
- o The client, upon receiving an NTP packet, uses the IP destination address to identify the path it came through, and uses the time information accordingly.
- o The client combines the information from all paths.

5.2. Dual-Ended Multi-Path Synchronization

In dual-ended multi-path synchronization each clock has N IP addresses. Time synchronization messages are exchanged between some of the combinations of {master IP, slave IP} addresses, allowing multiple paths between the master and slave. Note that the actual number of paths between the master and slave may be less than the number of chosen {master, slave} IP address pairs.

Once the multiple two-way connections are established, a separate synchronization protocol exchange instance is run through each of them.

5.2.1. Dual-Ended MPPTP Synchronization Message Exchange

The dual-ended MPPTP message exchange procedure is as follows.

- o Every clock has N IP addresses, but uses a single clockIdentity.
- o The BMC algorithm at each clock determines the master. The master is identified by its clockIdentity, allowing other clocks to know the multiple IP addresses it uses.
- o When a clock sends an Announce message, it sends it from each of its IP addresses with its clockIdentity.
- o A dual-ended MPPTP port that is in the 'slave' state uses unicast negotiation to request the master to transmit unicast messages to some or all of its N_s IP addresses. This negotiation is done individually between a slave IP address and the corresponding master IP address that the slave desires a connection with. The slave port periodically sends Signaling messages to the master, using some or all of its N_s IP addresses as source, to the corresponding master's N_m IP addresses. The Signaling message includes the REQUEST_UNICAST_TRANSMISSION_TLV.

- o The master periodically sends unicast Sync messages from each of its IP addresses to the corresponding slave IP addresses for which a unicast connection was negotiated.
- o The slave, upon receiving a Sync message, identifies its path according to the {source, destination} IP addresses. The slave sends a Delay_Req unicast message, swapping the source and destination IP addresses from the Sync message. Note that the rate of Delay_Req messages may be lower than the Sync message rate, and thus a Sync message is not necessarily followed by a Delay_Req.
- o The master, in response to a Delay_Req message from the slave, responds with a Delay_Resp message using the sourcePortIdentity from the Delay_Req message, and swapping the IP addresses from the Delay_Req.
- o Upon receiving the Delay_Resp message, the slave identifies the path using the {source, destination} IP address pair. The slave can then compute the corresponding path delay and the offset from the master.
- o The slave combines the information from all negotiated paths.

5.2.2. Dual-Ended MPNTP Synchronization Message Exchange

The MPNTP message exchange procedure is as follows.

- o Each NTP clock has a set of N IP addresses. The assumption is that the server information, including its multiple IP addresses is known to the NTP clients.
- o The MPNTP client chooses N_{svr} of the N server IP addresses and N_{c} of the N client IP addresses and initiates the $N_{\text{svr}} \times N_{\text{c}}$ instances of the protocol, one for each {server IP, client IP} pair, allowing the client to combine the information from the $N_{\text{s}} \times N_{\text{c}}$ paths.
(N_{svr} and N_{c} indicate the number of IP addresses of the server and client, respectively, which a client chooses to connect with)
- o The client sends NTP messages to the master using each of the source-destination address combinations.
- o The server responds to the client's NTP messages using the IP address combination from the received NTP packet.

- o Using the {source, destination} IP address pair in the received packets, the client identifies the path, and performs its computations for each of the paths accordingly.
- o The client combines the information from all paths.

5.3. Using Traceroute for Path Discovery

The protocols presented above use multiple IP addresses in a single clock to create multiple paths. However, although each two-way path is defined by a different {master, slave} address pair, some of the IP address pairs may traverse exactly the same network path, making them redundant. Traceroute-based path discovery can be used for filtering only the IP addresses that obtain diverse paths. 'Paris Traceroute' [PARIS] and 'TraceFlow' [TRACEFLOW] are examples of tools that discover the paths between two points in the network.

The Traceroute-based filtering can be implemented by both master and slave nodes, or it can be restricted to run only on slave nodes to reduce the overhead on the master. For networks that guarantee the path of the timing packets in the forward and reverse direction are the same, path discovery should only be performed at the slave.

5.4. Using Unicast Discovery for MPPTP

As presented above, MPPTP uses Announce messages and the BMC algorithm to discover the master. The unicast discovery option of PTP can be used as an alternative.

When using unicast discovery the MPPTP slave ports maintain a list of the IP addresses of the master. The slave port uses unicast negotiation to request unicast service from the master, as follows:

- o In single-ended MPPTP, the slave uses unicast negotiation from each of its identities to the master's (only) identity.
- o In dual-ended MPPTP, the slave uses unicast negotiation from its IP addresses, each to a corresponding master IP address to request unicast synchronization messages.

Afterwards, the message exchange continues as described in sections 5.1.1. and 5.2.1.

The unicast discovery option can be used in networks that do not support multicast or in networks in which the master clocks are known in advance. In particular, unicast discovery avoids multicasting Announce messages.

6. Combining Algorithm

Previous sections discussed the methods of creating the multiple paths and obtaining the time information required by the slave algorithm. This section discusses the algorithm used to combine this information into a single accurate time estimate. Note that the choice of the combining algorithm is local to the slave, and does not affect the interoperability of the protocol. Several combining methods are examined next.

6.1. Averaging

In the first method the slave performs an autonomous time computation for each of the master-slave paths, and obtains the combined time by simply averaging the separate instances. This method can be further enhanced by adding weights to each of the paths. For example, a reasonable weighting choice is to use an inverse of the round-trip delay between the peers. Another option is to use the inverse of the path delay variance, which is approximately the maximum likelihood estimator under certain assumptions [WEIGHT-MEAN].

6.2. Switching / Dynamic Algorithm

The switching and dynamic algorithms are presented in [SLAVEDIV]. The switching algorithm periodically chooses a primary path, and performs all time computations based on the protocol packets received through the primary path. The primary path is defined as the path with the minimal distance between the sampled delay and the average delay. The dynamic algorithm dynamically chooses between the result of the switching algorithm and the averaging.

6.3. NTP-like Filtering-Clustering-Combining Algorithm

NTP ([NTP], [NTP2]) provides an efficient algorithm of combining offset samples from multiple peers. The same approach can be used in MPPTP and MPNTP.

In the MPNTP, the selection and combining algorithms treat the offset samples from multiple paths as NTP treats samples from distinct peers. The rest of the selection and combining algorithms, as well as clock control logic is the same as in conventional NTP. In MPPTP, a similar approach to NTP can be adopted.

The combining algorithm [NTP3] contains three steps: filtering, selection and clustering.

In the filtering step, the best of the last n (usually $n=8$) samples of each peer is chosen. The choice criterion is the combination of a round trip delay estimate of the sample and the distance from the average offset of all n samples of a peer.

In the selection step the peers are divided into two groups: true-chimers and false tickers.

The clustering step chooses a subset of the true-chimers, whose peer jitter (the variance of peer offset samples) is smaller than the total select jitter of all selected peer offsets (the variance of the best offset of the selected peers).

The offset samples that passed through the three steps are combined by a weighted average into a single offset estimate. Detailed explanations are provided in [NTP2],[NTP3].

7. Security Considerations

The security aspects of time synchronization protocols are discussed in detail in [TICTOCSEC]. The methods describe in this document propose to run a time synchronization protocol through redundant paths, and thus allow to detect and mitigate man-in-the-middle attacks, as described in [DELAY-ATT].

8. IANA Considerations

There are no IANA actions required by this document.

RFC Editor: please delete this section before publication.

9. Acknowledgments

The authors gratefully acknowledge the useful comments provided by Peter Meyer and Doug Arnold, as well as other comments received from the TICTOC working group participants.

This document was prepared using 2-Word-v2.0.template.dot.

10. References

10.1. Normative References

- [IEEE1588] IEEE Instrumentation and Measurement Society, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588, 2008.

- [NTP] D. Mills, J. Martin, J. Burbank, W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", IETF, RFC 5905, 2010.

10.2. Informative References

- [ASSYMETRY] Yihua He and Michalis Faloutsos and Srikanth Krishnamurthy and Bradley Huffaker, "On routing asymmetry in the internet", IEEE Globecom, 2005.
- [ASSYMETRY2] Abhinav Pathak, Himabindu Pucha, Ying Zhang, Y. Charlie Hu, and Z. Morley Mao, "A measurement study of internet delay asymmetry", PAM'08, 2008.
- [DELAY-ATT] T. Mizrahi, "A Game Theoretic Analysis of Delay Attacks against Time Synchronization Protocols", ISPCS, 2012.
- [NTP2] Mills, D.L., "Internet time synchronization: the Network Time Protocol", IEEE Trans. Communications COM-39, 10 (October 1991), 1482-1493.
- [NTP3] Mills, D.L., "Improved algorithms for synchronizing computer network clocks", IEEE/ACM Trans. Networks 3, 3(June 1995), 245-254.
- [PARIS] Brice Augustin, Timur Friedman and Renata Teixeira, "Measuring Load-balanced Paths in the Internet", IMC, 2007.
- [PARIS2] B. Augustin, T. Friedman, and R. Teixeira, "Measuring Multipath Routing in the Internet", IEEE/ACM Transactions on Networking, 19(3), p. 830 - 840, June 2011.
- [SLAVEDIV] T. Mizrahi, "Slave Diversity: Using Multiple Paths to Improve the Accuracy of Clock Synchronization Protocols", ISPCS, 2012.
- [TICTOCSEC] T. Mizrahi, K. O'Donoghue, "Security Requirements of Time Protocols in Packet Switched Networks", IETF, draft-ietf-tictoc-security-requirements, work in progress, 2013.
- [TRACEFLOW] J. Narasimhan, B. V. Venkataswami, R. Groves and P. Hoose, "Traceflow", IETF, draft-janapath-intarea-traceflow, work in progress, 2012.

[WEIGHT-MEAN] http://en.wikipedia.org/wiki/Weighted_mean#Dealing_with_variance

Authors' Addresses

Alex Shpiner
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, 32000 Israel

Email: shalex@tx.technion.ac.il

Richard Tse
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: Richard.Tse@pmcs.com

Craig Schelp
PMC-Sierra
8555 Baxter Place
Burnaby, BC
Canada
V5A 4V7

Email: craig.schelp@pmcs.com

Tal Mizrahi
Marvell
6 Hamada St.
Yokneam, 20692 Israel

Email: talmi@marvell.com

