MPTCP is not Pareto-Optimal: Performance Issues and a Possible Solution[*]

Ramin Khalili[†], Nicolas Gast, Miroslav Popovic, Utkarsh Upadhyay, Jean-Yves Le Boudec
EPFL, IC-LCA2, Switzerland
firstname.lastname@epfl.ch

ABSTRACT

MPTCP has been proposed recently as a mechanism for supporting transparently multiple connections to the application layer. It is under discussion at the IETF. We show, however, 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, which is a symptom of not being Pareto-optimal; and (P2) MPTCP users could be excessively aggressive towards TCP users. We attribute these problems to the linked-increases algorithm (LIA) of MPTCP and, more specifically, to an excessive amount of traffic transmitted over congested paths.

The design of LIA forces a tradeoff between optimal resource pooling and responsiveness. We revisit the problem and show that it is possible to provide these two properties simultaneously. We implement the resulting algorithm, called opportunistic linked increases algorithm (OLIA), in the Linux kernel, and we study its performance over our testbed, by simulations and by theoretical analysis. We prove that OLIA is Pareto-optimal and satisfies the design goals of MPTCP. Hence it can avoid the problems P1 and P2. Our measurements and simulations indicate that MPTCP with OLIA is as responsive and non-flappy as MPTCP with LIA, and that it solves problems P1 and P2.

Categories and Subject Descriptors

C.2 [**Computer-communication Networks**]: Network Protocols.; C.4 [**Performance of Systems**]: Design studies, Modeling techniques.

Keywords

Multipath TCP, Congestion control algorithm, Protocol design, Performance evaluation.

*This research has received funding from the EU 7th Framework Programme (FP7/2007-2013) under grant agreement n. 257740 (Network of Excellence "TREND").

†Ramin Khalili is currently affiliated with T-Labs/TU Berlin, ramin@net.t-labs.tu-berlin.de.

1. INTRODUCTION

The regular TCP uses a window-based congestion-control mechanism to adjust the transmission rate of users [1]. 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 [2]. It also guarantees a fair allocation of bandwidth among the users, but favors the connections with lower RTT [3].

Various mechanisms were used to build a multipath transport protocol compatible with the regular TCP. Authors of [4–6] propose a family of algorithms inspired by utility maximization frameworks. 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 [7–9]. First, they sometimes 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 TCP.

MultiPath TCP (MPTCP) is a concrete proposal for multipath transport; it is under discussion at the IETF [10]. Because of the issues aforementioned, its congestion control part does not follow the algorithms in [4–6]. Instead, it follows an ad-hoc design based on the following goals [10]: (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 [7, 9]: It can effectively use the available bandwidth; compared to independent TCP flows, it improves throughput and fairness in many scenarios; and it solves the flappiness problem. Through analysis and by using measurements over a testbed, we show, however, 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 [10] and specifically to an excessive amount of traffic transmitted over congested paths. These problems indicate that MPTCP fails to fully satisfy its design goals, especially goal 3.

The design of LIA forces a tradeoff 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. We revisit the design and show that it is possible to simultaneously provide both properties. We introduce OLIA, the "opportunistic linked increases algorithm", as an alternative to LIA. Based on utility maximization frameworks, we prove that OLIA is Pareto-optimal. Hence it can avoid the problems P1 and P2. Furthermore, its construction makes it as responsive and non-flappy as LIA.

OLIA is a window-based congestion-control mechanism. Similarly to LIA, it couples the additive increases and uses unmodified TCP behavior in the case of a loss. OLIA's increase part, Equation (5), has two terms:

- The first term is an adaptation of the increase term of Kelly and Voice's algorithm [4]. This term is essential to provide Pareto-optimality.
- The second term guarantees responsiveness and non-flappiness of OLIA. By measuring the number of transmitted bits 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.

We implement OLIA in the Linux kernel and study its performance over our testbed, by simulations and by theoretical analysis. Using a fluid model of OLIA based on differential inclusion, we prove that OLIA is Pareto-optimal (Theorem 3) and that it satisfies the design goals of MPTCP (Corollary 2). Our measurements and simulations indicate that MPTCP with OLIA is as responsive and non-flappy as MPTCP with LIA. Moreover, it solves problems P1 and P2. Hence, we believe that IETF should revisit the congestion control part of MPTCP and that an alternative algorithm, such as OLIA, should be considered.

In the next section, we briefly introduce MPTCP and related work. In Section 3, we provide a number of examples and scenarios in which MPTCP with LIA exhibits problems P1 and P2. In Section 4, we introduce OLIA and detail its Linux implementation. In Section 5, we prove that OLIA is Pareto-optimal and satisfies MPTCP's design goals. In Section 6, we study the performance of OLIA through measurements and by simulations.

2. MPTCP AND RELATED WORK

Multipath TCP (MPTCP) is a set of extensions to the regular TCP, which allows users to spread their traffic across potentially disjoint paths [10]. MPTCP discovers the number of paths available to a user, establishes the paths, and distributes traffic across these paths through creation of separate subflows [11, 12].

MPTCP's congestion control algorithm forces a tradeoff between optimal resource pooling and responsiveness [8]. The idea behind the algorithm is to transmit over a path r at a rate proportional to $p_r^{-1/\varepsilon}$, where p_r is the loss probability over this link and $\varepsilon \in [0, 2]$ is a design parameter. The choice ε=0 corresponds to the fully coupled algorithm of [4–6]: the traffic is sent only over the best paths, it is Pareto-optimal but is flappy. The choice ε=2 corresponds to using uncoupled TCP flows on each path: it is very responsive and non-flappy, but does not balance congestion. MPTCP's implementation uses ε=1 to provide a compromise between optimal resource pooling and responsiveness. This algorithm is called "linked increases" algorithm (LIA) [10].

Let w_r and rtt_r be the window size and the estimated round-trip time on path $r \in \mathcal{R}_u$. $\mathcal{R}_u$ is the set of all paths available to user u. LIA works as follows:

- For each ACK on subflow r, increase w_r by
$$\min\left(\frac{\max_{i\in\mathcal{R}_u} w_i/\text{rtt}_i^2}{(\sum_{i\in\mathcal{R}_u} w_i/\text{rtt}_i)^2}, \frac{1}{w_r}\right). \quad (1)$$
- For each loss on subflow r, decrease w_r by $w_r/2$.

LIA increases by at most $1/w_r$ to be at most as aggressive as regular TCP on any of its paths. When the RTTs are similar, this minimum can be neglected as the first term $(\max_i w_i/\text{rtt}_i^2)/(\sum_i w_i/\text{rtt}_i)^2$ would always be less than $1/w_r$. In this case, a fixed point analysis provides a simple loss-throughput formula for LIA [9]: LIA allocates to a path r a window w_r proportional to the inverse of the loss probability $1/p_r$ and such that the total rate $\sum_{p\in\mathcal{R}_u} w_p/\text{rtt}_p$ equals the rate that a regular TCP user would get on the best path, *i.e.* $\max_{p\in\mathcal{R}_u}\sqrt{2/p_p}/\text{rtt}_p$. Thus, the window size for the flow on a path r is given by
$$w_r = \frac{1}{p_r}\cdot\frac{\max_{p\in\mathcal{R}_u}\sqrt{2/p_p}/\text{rtt}_p}{\sum_{p\in\mathcal{R}_u} 1/(\text{rtt}_p p_p)}. \quad (2)$$

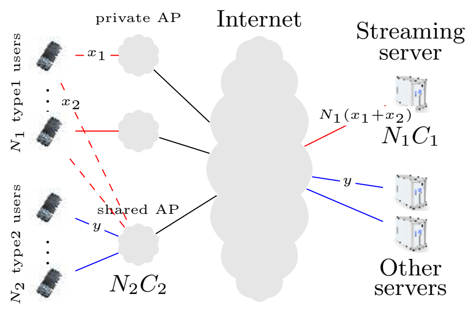
Besides MPTCP and algorithms in [4–6], a few other algorithms have been proposed to implement multipath protocols. In [13], an opportunistic multipath scheduler measures the path conditions on time scales up to several seconds. [14] uses a mechanism to detect shared bottlenecks and to avoid the use of multiple subflows on the same bottleneck. [15] proposes to use uncoupled TCP flows with a weight depending on the congestion level. These mechanisms are complex, their robustness is not clear, and they need explicit information about congestion in the network. Our proposed algorithm, OLIA, differs from these works as it is implemented, proven to be Pareto optimal, and relies only on information that is available to regular TCP. It also differs from [4–6] as it is not flappy and has a better responsiveness.

3. PERFORMANCE PROBLEMS OF MPTCP

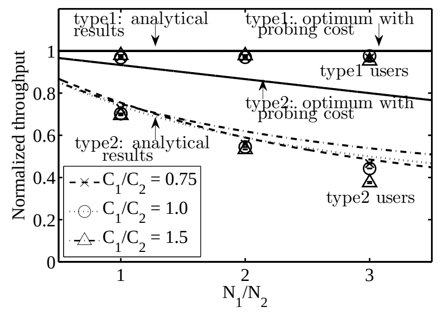
In this section, we investigate the behavior of MPTCP with LIA in three different scenarios: scenarios A, B, and C. Using scenarios A and B, we show that upgrading some regular TCP users to MPTCP could reduce the throughput of other users in the network without any benefit to the upgraded users (problem P1). In Scenario C, we discuss the aggressiveness of MPTCP users that compete with regular TCP users (problem P2). Our conclusions are based on analytical results and measurements over a testbed.

3.1 Testbed Setup

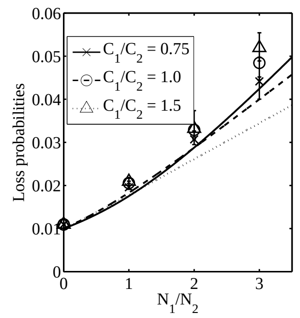
To investigate the behavior of the algorithms, we create three testbed topologies that represent our scenarios.



(a) Scenario A



(b) Normalized throughput of users: $(x_1 + x_2)/C_1$ and y/C_2.



(c) Loss prob. p_2 at the shared AP.

Figure 1: Scenario A: type1 users are all downloading through the same streaming server and have access to both a private high speed access point and a shared access point. Type2 users have access only to the shared access point. The performance of MPTCP with LIA obtained by measurement (points) or numerical analysis (lines) is shown on figures (b) and (c). We observe that it is not Pareto-optimal, penalizes type2 users, and its performance is far from the theoretical optimum with probing cost. It also fails to balance the congestion.

Server-client PCs run MPTCP (with LIA or OLIA) enabled Linux kernels. In all scenarios laptop PCs are used as routers. We install "Click Modular Router" software [16] to emulate topologies with different characteristics. We emulate links with configurable bandwidth and delay with RED queuing (drop-tail queuing is also studied in htsim simulation, see Section 6.2). Figure 2 represents the testbed configuration of the scenario described in Figure 1(a).

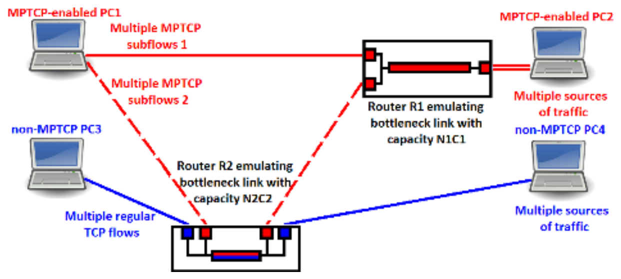

MPTCP-enabled PC1
Multiple MPTCP subflows 1
MPTCP-enabled PC2
Multiple MPTCP subflows 2
Router R1 emulating bottleneck link with capacity N1C1
Multiple sources of traffic
non-MPTCP PC3
non-MPTCP PC4
Router R2 emulating bottleneck link with capacity N2C2
Multiple regular TCP flows
Multiple sources of traffic

Figure 2: Testbed implementation of scenario A: router R_1 emulates the bottleneck at the server side and router R_2 the shared AP bottleneck. Iperf is used to emulate multiple connections. The red PCs use MPTCP and the blue PCs use regular TCP.

3.2 Scenario A: MPTCP is not Pareto-optimal and penalizes regular TCP users

Consider a network with two types of users as shown in Figure 1(a). There are N_1 users of type1, each with a high-speed private connection, accessing different files on a media streaming server. The server has a network connection with capacity limit of N_1C_1 Mbps. These users can activate a second connection through a shared access point (AP) by using MPTCP. There are also N_2 type2 users that have connections only through the shared AP. They download their contents from the Internet. The shared AP has a capacity of N_2C_2 Mbps.

Let x_1 be the rate that a type1 user receives over its private connection (by symmetry, every user of type1 will receive the same rate x_1). Similarly, let x_2 (resp. y) be the rate that a type1 (resp. type2) user receives over the shared connection. We denote by p_1 and p_2 the loss probability at the link connected to the streaming server and the shared AP, respectively (the loss probabilities at the Internet backbone and the private APs are negligible).

When type1 users use only their own private AP, we have $x_1{=}C_1$, $x_2{=}0$, and $y{=}C_2$. In this case the normalized throughput for both type1 and type2 users is 1. In the other case, assuming that all paths have RTT rtt, when all type1 users activate their public connections and use MPTCP with LIA to balance load between their connections, we have

$$\begin{array}{lll} (a) & N_1(x_1{+}x_2) = N_1C_1 & N_1x_2 + N_2y = N_2C_2 \\ (b) & x_1 + x_2 = \frac{1}{\text{rtt}}\sqrt{\frac{2}{p_1}} & x_2 = \frac{1}{2{+}p_2/p_1}\frac{1}{\text{rtt}}\sqrt{\frac{2}{p_1}} \\ (c) & y = \frac{1}{\text{rtt}}\sqrt{2/p_2} & \end{array}$$

where (a) are the capacity constraints at the two bottlenecks, (b) comes from the loss-throughput formula for LIA (Eq.(2)), and (c) follows the TCP loss-throughput formula [17]. This system has a unique solution (see [18], Appendix A). Figure 1(b) depicts the normalized throughput of type1 and type2 users, i.e. $(x_1 + x_2)/C_1$ and y/C_2. As shown in [18], Appendix A, these values depend only on the ratios C_1/C_2 and N_1/N_2.

A theoretically optimal algorithm (as discussed in [4, 5]) will allocate a normalized throughput of 1 to both type1 and type2 users. 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. In this paper, we introduce a theoretical baseline for window-based congestion-control algorithms, called *theoretical optimum 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. It serves as a reference to see how far from the optimum LIA is.

We measure the performance of LIA in Scenario A, by using the testbed, as shown in Figure 2. The measurements are taken for $N_2 = 10$ and three values of $N_1 = 10, 20, 30$. The capacities of R1 and R2 are N_1C_1 and N_2C_2 Mbps, where we set $C_2 = 1$Mbps and $C_1 = 0.75, 1, 1.5$ Mbps. All paths have similar RTTs (link delay plus queuing delay is around 150 ms over all paths). For each case, we took 5 measurements. The results are reported in Figure 1(b). Note that in all cases we present confidence intervals, but

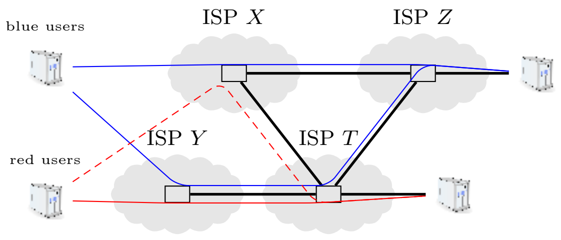


Figure 3: Scenario B. Thick lines represent peering agreements. Blue users are downloading from servers in ISP Z and Red users from servers in ISP T. Blue users use multi-homing and have access to ISPs X and Y. Initially, Red users have access only to ISP Y but upgrade to MPTCP and connect to both X and Y (by activating the dashed connection).

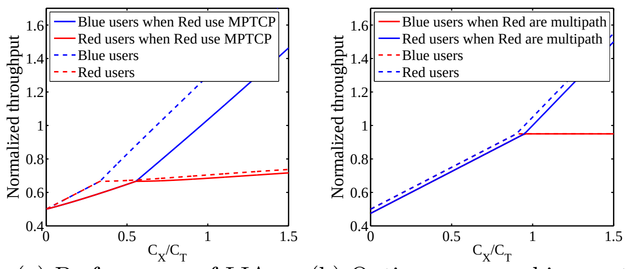
in many cases they are too small to be visible. The loss probability p_1 depends only on C_1 and is 0.02, 0.009, 0.004 for $C_1 = 0.75, 1, 1.5$ Mbps. We also show our analytical analysis of LIA, as well as the theoretical optimum with probing cost as defined above.

These figures have multiple implications. First, they show that MPTCP with LIA exhibits problem (P1) from the introduction: 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 C_1 of the streaming server. For N_1=N_2, type2 users see a decrease of about 30%. When N_1=3N_2, this decrease is between 50% to 60%. This is explained by the fact that LIA does not fully balance congestion, as shown in Figure 1(c). It excessively increases congestion on the shared AP (not in compliance with goal 3). We observe that LIA performs far from how an optimal algorithm with probing cost would perform. Furthermore, these figures show that the fixed point analysis predicts accurately the behavior of the algorithm: the two curves (theoretical and experimental) exhibit the same trend. As a summary for this section, we conclude that MPTCP with LIA is not Pareto-optimal and can penalize TCP users without any benefit for anybody.

3.3 Scenario B: MPTCP is not Pareto-optimal and can penalize other MPTCP users.

Consider the multi-homing scenario depicted in Figure 3. 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 C_X, C_Y, C_Z, and C_T.

Z and T host different video streaming servers. There are two types of users: N_B Blue users download contents from a server in Z, and N_R Red users download from a server in ISP T. Blue users use multi-homing and are connected to both ISPs X and Y to increase their reliability. Red users can connect either only to Y or to both X and Y. We assume that only ISPs X and T are bottlenecks and denote by p_X and p_T the loss probabilities. We assume that all paths have similar RTTs.



(a) Performance of LIA. (b) Optimum w. probing cost

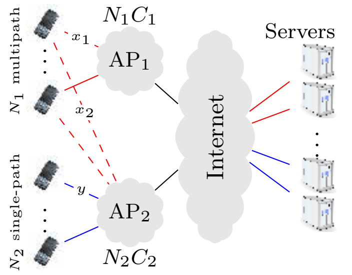
Figure 4: Scenario B: analytical results for 15 Blue, 15 Red users and C_T=36 Mbps where $C_Y = C_Z = 100$ Mbps. We show the normalized throughput $(15(x_1+x_2)/C_T$ and $15(y_1+y_2)/C_T)$ as a function of C_X/C_T. Dashed curves: normalized throughput when Red users connect only to ISP Y. Solid curves: the case when Red users upgrade to multipath. For all values of C_X/C_T, the throughput of all users decreases when Red users upgrade to MPTCP.

We first present a theoretical analysis of the rate that each user would achieve (to simplify the analysis, we assume similar numbers of Blue and Red users). There are two possible cases. When Red users connect only to Y, the analysis is the same as the one of scenario C, given in Section 3.4. Here, we analyze the case when Red users upgrade to MPTCP. The loss throughput formula (Equation (2)) shows that the throughput of the different connections are:

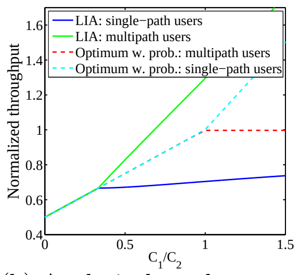
$$\begin{cases} y_1 = \dfrac{1/\text{rtt}}{2+\frac{p_X}{p_T}}\sqrt{\dfrac{2}{p_T}} \\ y_2 = \dfrac{p_X}{p_T} y_1 \end{cases}, \begin{cases} x_1 = \dfrac{1/\text{rtt}}{1+p_X/p_T}\sqrt{\max\dfrac{2}{p_X},\dfrac{2}{p_T}} \\ x_2 = \dfrac{1/\text{rtt}}{1+p_T/p_X}\sqrt{\max\dfrac{2}{p_X},\dfrac{2}{p_T}} \end{cases}$$

As shown in [18], Appendix B, this set of equations has a unique positive solution. A numerical evaluation of these formulas is depicted in Figure 4(a). Figure 4(b) depicts the performance of a theoretical optimum with probing cost (see [18], Appendix B). The results are presented for RTT=150 ms, $C_Y = C_Z = 100$ Mbps, and $C_T = 36$ Mbps. We consider 15 Blue users and 15 Red users in the network. We depict the normalized throughput $(15(x_1 + x_2)/C_T$ and $15(y_1 + y_2)/C_T)$ as a function of C_X/C_T. The results show that upgrading Red users to MPTCP with LIA decreases the performance for everyone. As an example, when $C_X/C_T \approx 0.75$, by upgrading the Red users we reduce the throughput of the Blue users by up to 21%. This decrease is about 3% when we use an optimal algorithm with probing cost (Fig. 4(b)).

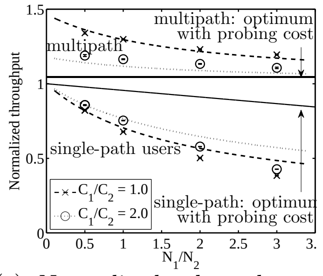
We emulate this scenario in our testbed in a similar manner as for Scenario A. The measurement results are reported in Table 1 for a similar setting with $C_X = 27$ Mbps. We observe that when Red users only connect to ISP Y, the aggregate throughput of users is close to the cut-set bound, 63 Mbps. However, Blue users get a higher share of the network bandwidth. Now consider that Red users upgrade to MPTCP by establishing a second connection through X (shown by dashed line in Figure 3). 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. This confirms our analytical observation and shows that MPTCP



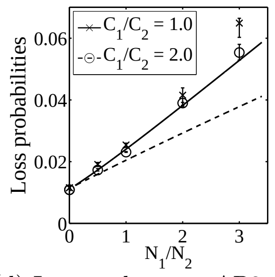
(a) Scenario C: N_1 multipath users and N_2 single-path users are connected to two APs with capacities N_1C_1 and N_2C_2 Mbps



(b) Analytical results: normalized throughput of all users using LIA (solid) or optimum with probing cost (dashed) for $N_1 = N_2$.



(c) Normalized throughputs using LIA, obtained by measurement (points) or analysis (lines).



(d) Loss prob. p_2 at AP2: LIA fails to balance the congestion.

Figure 5: Scenario C: MPTCP with LIA excessively penalizes TCP users (when $C_1/C_2 \geq 1$, for any fairness criterion, MPTCP users should not impact TCP users). We show the normalized throughputs ($(x_1+x_2)/C_1$ and y/C_2) received by the users, as well as p_2. The performance of LIA is far from the theoretical optimum with probing cost.

Red users	Rate/user		Aggregate
	Blue users	Red users	
Single-path	2.5	1.5	59.8
Multipath	2.0	1.4	52.0

Table 1: Measurement results for scenario B, showing the effect of upgrading the Red users from regular TCP to MPTCP with LIA. The number of Red and Blue users is 15 and all values are recorded in Mbps. By upgrading Red users to MPTCP, the throughput drops for all users and the aggregate throughput falls by 13%.

with LIA is not Pareto-optimal and could penalize other MPTCP users without any benefit for anybody.

3.4 Scenario C: MPTCP users could be excessively aggressive towards TCP users.

We 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 (see Figure 5). Multipath users connect to both APs and they share AP2 with single-path users.

If the allocation of rates is proportionally fair, multipath users will use AP2 only if $C_1 < C_2$ and all users will receive $(N_1C_1+N_2C_2)/(N_1+N_2)$. When $C_1 > C_2$, a fair multipath user will not transmit over AP2. This fair allocation is represented by dashed lines in Figure 5(b) when we take into account the minimum probing cost. However, using MPTCP with LIA, multipath users get a larger share of bandwidth as soon as $C_1 \geq C_2/(2+N_1/N_2)$. We show this analytically. Let p_1 and p_2 be the loss probabilities at APs, x_1 and x_2 be rates that a multipath user receives over its paths, and y be the rate of a single-path user. Assume all RTTs are the same. When $C_1/C_2 < 1/(2+N_1/N_2)$, we have $p_1 > p_2$ and all users receive the same rate: $x_1+x_2 = y = (C_1+C_2)/2$. When $C_1/C_2 > 1/(2+N_1/N_2)$, we have $p_1 < p_2$ and the fixed point formula of LIA gives:

$$x_1 = \frac{p_2}{p_1+p_2}\frac{1}{\text{rtt}}\sqrt{\frac{2}{p_1}} \quad \text{and} \quad x_2 = \frac{p_1}{p_1+p_2}\frac{1}{\text{rtt}}\sqrt{\frac{2}{p_1}}.$$

Moreover, both the APs are bottlenecks and we have $x_1 = C_1$ and $x_2 + y = C_2$. As shown in [18], Appendix C, this set of equations has a unique positive solution that only depends on the ratio N_1/N_2 and C_1/C_2. Figure 5(b) reports a numerical evaluation of these fixed point equations for the case $N_1 = N_2$. We show the normalized throughputs ($(x_1+x_2)/C_1$ and y/C_2) received by the users, as well as p_2. We observe that LIA is fair with regular TCP users, as long as $C_1 < C_2/3$. However, as C_1 exceeds $C_2/3$, it takes most of the capacity of AP2 for itself.

We emulate the scenario in our testbed and measure the performance of MPTCP with LIA. The results are reported in Figures 5(c) and 5(d) for C_2=1 Mbps and C_1=1, 2Mbps, with N_2=10 and N_1=5, 10, 20, 30. As in scenario A, we also present the theoretical optimum with probing cost in Figure 5(c).

When $C_1/C_2 \geq 1$, multipath users should not use AP2 at all. However, our results show that, MPTCP users are disproportionately aggressive and exhibit problem (P2). Figure 5(d) shows the loss probability at AP2. We observe that LIA excessively increases congestion on AP2 and is unable to fully balance congestion in the network. Also, we have p_1=0.01 and 0.003 for C_1=1 and 2Mbps, respectively. These results confirm our analytical observation and show that LIA is overly aggressive towards TCP users.

4. OLIA: THE OPPORTUNISTIC LINKED INCREASES ALGORITHM

In this section, we introduce OLIA as an alternative for MPTCP's LIA. OLIA is a window-based congestion-control algorithm that couples the increase of congestion windows and uses unmodified TCP behavior in the case of a loss. The increase part of OLIA has two terms. The first term is an adaptation of Kelly and Voice's increase term and provides the Pareto-optimality (Kelly and Voice's algorithm is based on scalable TCP; the first term is a TCP compatible version of their algorithm that compensates also for different RTTs). The second term, with α, guarantees responsiveness and non-flappiness. We first present the algorithm and its Linux implementation. Then, we illustrate with an example its operation and its difference with LIA.

4.1 Detailed Description of OLIA

Let $\mathcal{R}_u$ be the set of paths available to user u and let $r \in \mathcal{R}_u$ be a path. We denote by $\ell_{1r}(t)$ the number of bits that were successfully transmitted by u over path r between the last two losses seen on r, and by $\ell_{2r}(t)$ the number of bits that are successfully transmitted over r after the last loss. If no losses have been observed on r up to time t, then $\ell_{1r}(t) = 0$ and $\ell_{2r}(t)$ is the total number of bits transmitted on r. Also, let $\ell_r(t) = \max\{\ell_{1r}(t), \ell_{2r}(t)\}$ and let $\text{rtt}_r(t)$ and $w_r(t)$ be respectively RTT and the window on r at time t. We define

$$\mathcal{M}(t) = \left\{ i(t) \,|\, i(t) = \arg\max_{p \in \mathcal{R}_u} w_p(t) \right\} \tag{3}$$

$$\mathcal{B}(t) = \left\{ j(t) \,|\, j(t) = \arg\max_{p \in \mathcal{R}_u} \frac{\ell_p(t)}{\text{rtt}_p(t)^2} \right\} \tag{4}$$

$\mathcal{M}(t)$ is the set of the paths of u with the largest window sizes at time t. $\mathcal{B}(t)$ is the set of the paths at time t that are presumably the best paths for u, as $1/\ell_r(t)$ can be considered as an estimate of packet loss probability on path r at time t, and the rate that path r can provide to a TCP user can be estimated by $\sqrt{2\ell_r(t)}/\text{rtt}_r$ [17].

Our algorithm is as follows (to simplify notation, we drop the time argument t; however, note that w_r, rtt_r, ℓ_r, $\mathcal{M}$, and $\mathcal{B}$ are all functions of time):

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

  $$\frac{w_r/\text{rtt}_r^2}{(\sum_{p \in \mathcal{R}_u} w_p/\text{rtt}_p)^2} + \frac{\alpha_r}{w_r}, \tag{5}$$

  where α_r is calculated as follows:

  $$\alpha_r = \begin{cases} \dfrac{1/|\mathcal{R}_u|}{|\mathcal{B} \setminus \mathcal{M}|} & \text{if}\, r \in \mathcal{B} \setminus \mathcal{M} \neq \emptyset \\ -\dfrac{1/|\mathcal{R}_u|}{|\mathcal{M}|} & \text{if}\, r \in \mathcal{M} \,\text{and}\, \mathcal{B} \setminus \mathcal{M} \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

  $\mathcal{B}\backslash\mathcal{M}$ is the set of elements in $\mathcal{B}$ but not in $\mathcal{M}$, $\emptyset$ is the empty set, and $|\mathcal{R}_u|$ is the number of paths available to u at the time. Note that $\sum_{r \in \mathcal{R}_u} \alpha_r{=}0$.

- For each loss on path r, **decrease** w_r by $\dfrac{w_r}{2}$.

We can see from (3), (4), and (6) that if the best paths have the maximum window size, then $\alpha_r{=}0$ for any $r \in \mathcal{R}_u$. However, if there is any best path with a small window size, *i.e.* if $\mathcal{B}\backslash\mathcal{M}{\neq}\emptyset$, then α_r would be positive if $r \in \mathcal{B}\backslash\mathcal{M}$, negative if $r \in \mathcal{M}$, and zero otherwise. Hence, OLIA increases windows faster on the paths that are the best but that have small windows. The increase is slower on the paths with maximum windows.

4.2 Linux Implementation of OLIA

We implemented OLIA in the MPTCP release supported on the Linux kernel 3.0.0 [19]. Similarly to LIA, our 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 [1]. 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 similar minimum ssthresh as in TCP (2 MSS). The purpose of this modification is to avoid transmitting unnecessary traffic over congested paths when multiple paths are available to a user. The minimum congestion windows size is 1 MSS as in TCP. Our implementation is available online [18].

One important part of our implementation is the measurement of ℓ_r on a path r. This can be done easily by using information that is already available to a regular TCP user. Our algorithm for computing ℓ_r is as follows:

- For each ACK on r: $\ell_{2,r} \leftarrow \ell_{2,r}+$ (number of bits that are acknowledged by ACK)
- For each loss on r: $\ell_{1,r} \leftarrow \ell_{2,r}$ and $\ell_{2r} \leftarrow 0$

where $\ell_r = \max\{\ell_{1,r}, \ell_{2,r}\}$. $\ell_{1,r}$ and $\ell_{2,r}$ are initially set to zero when the connection is established. To compute a smoothed estimate of rtt_r, we use the algorithm, proposed in [20] and implemented in the Linux kernel.

4.3 Illustrative Example of OLIA's Behavior

To give more insight into how OLIA performs, we show the evolution of window sizes and α values for a two-path flow in Figure 6. The measurement results on our testbed are reported in Figures 7 and 8.

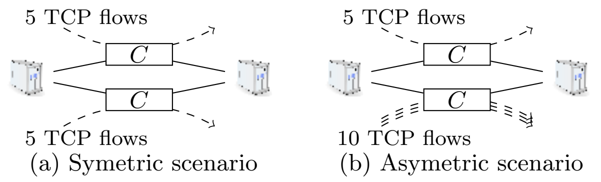


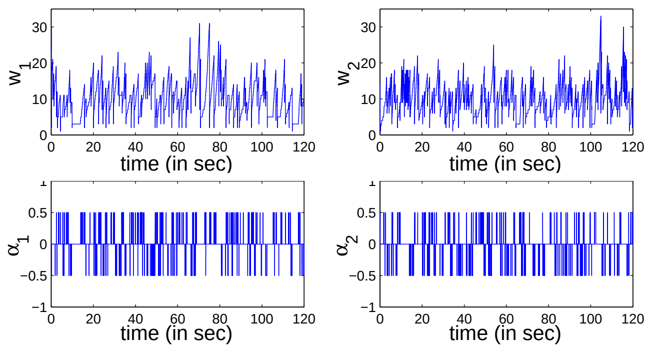
Figure 6: A multipath user sharing two bottlenecks of the same capacity C with single-path users.

We first consider a symmetric case, depicted in Figure 6(a). As both of the paths are equally good, a multipath user will benefit from using both of them. Figure 7(a) shows the evolution of w_r and α_r as a function of time. We observe that OLIA simultaneously uses both of the paths, similarly to LIA (Figure 7(b)), which is the desired behavior. There is no sign of flappiness as α_1 and α_2 react quickly and adjust w_1 and w_2 accordingly.

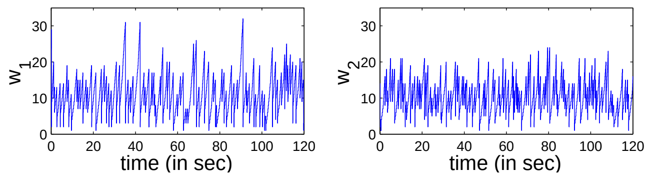
We now study the asymmetric scenario of Figure 6(b). In this case, the second path is shared with 10 TCP flows and it will be beneficial if multipath users use only the first path. This is what we observe in Figure 8(a). The window on the congested path is 1, most of the time (because of the first increase term). However, due to α, the window increases from time to time over the congested path whenever the path has the largest inter-loss distance ℓ_r. This increase is brief as losses occur more frequently on this path. LIA, however, transmits significant traffic over the congested paths and lower traffic, compared to OLIA, over the good path as depicted in Figure 8(b).

5. PARETO-OPTIMALITY OF OLIA

In this section, we build a fluid model of OLIA by using differential inclusions. We show that this model provides a Pareto-optimal allocation (Theorem 3) that satisfies the three design goals of MPTCP [10] (Corollary 2). Also, we prove that MPTCP with OLIA is fair with TCP: If all routes of a user have the same RTT, then OLIA maximizes the same fairness criteria as the regular TCP (Theorem 4).



(a) MPTCP - OLIA: window size and α_r as a function of time.



(b) MPTCP - LIA: window size.

Figure 7: Evolution of w and α values for a two-path flow. Each path is shared with 5 regular TCP users. OLIA uses both of the paths, similarly to LIA, and there is no sign of flappiness.

5.1 Fluid Model of OLIA

We consider a network model similar to [3]. The network is static and composed of a set $\mathcal{L}$ of links (or resources). We denote by $\mathcal{R}_u$ the set of paths available to a user u, each path being a set of links. If the route r is available to user u, we write $r \in R_u$. If a route r uses a resource ℓ, we write $\ell \in r$. Similarly, we refer to all routes that cross ℓ as $r \ni \ell$.

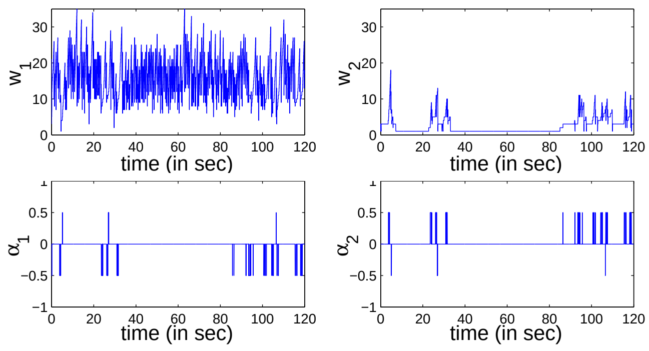
Let $x_r(t) \geq 0$ be the rate of traffic transmitted by the user u on a path $r \in \mathcal{R}_u$. We assume that the RTT of a route r is fixed in time and we denote it by rtt_r. In the fluid model, the rate x_r is an approximation of the window size divided by the RTT, i.e. $x_r = w_r/\text{rtt}_r$.

Let $p_\ell(\sum_{\ell \in r} x_r)$ be the loss rate at link ℓ. p_ℓ depends on the capacity of the link, C_ℓ, and the total amount of traffic sent through the link, $\sum_{\ell \in r} x_r$. We assume that p_ℓ is an increasing function of the total traffic. To simplify the notation, we omit the dependence on x and write only p_ℓ. However, note that if x varies with time, p_ℓ will also vary. We assume that the loss probabilities of links are independent and small; hence, the loss probability on a route r is $p_r = 1 - \prod_{\ell \in r}(1-p_\ell) \approx \sum_{\ell \in r} p_\ell$.

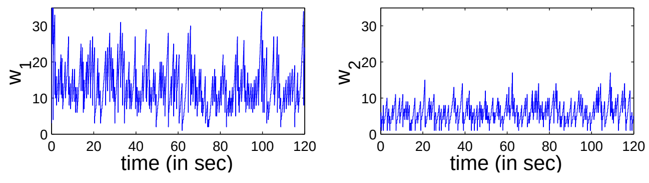
When p_r is small, a user u receives acknowledgments on a route $r \in \mathcal{R}_u$ at rate x_r and increases the window w_r as Equation (5). Losses occur at rate $p_r x_r$ on r, and the user decreases w_r by half whenever it detects a loss. We consider a fluid approximation of OLIA in which we replace the stochastic variations of rates by their expectation. This leads to the differential equation:

$$\frac{dx_r}{dt} = x_r^2 \left(\frac{1/\text{rtt}_r^2}{(\sum_{p \in \mathcal{R}_u} x_p)^2} - \frac{p_r}{2} \right) + \frac{\alpha_r}{\text{rtt}_r^2}, \qquad (7)$$

α_r depends on the values p_p and w_p for all paths $p \in \mathcal{R}_u$ of users u. It is defined by Equation (6). To compute α_r, we approximate ℓ_r by its average: $l_r = 1/p_r$.



(a) MPTCP - OLIA: window size and α_r as a function of time.



(b) MPTCP - LIA: window size.

Figure 8: Evolution of w and α for a two paths flow. The first path is shared with 5 TCP flows and the second with 10. OLIA uses only the good path. LIA transmits significant traffic over the congested path and less than OLIA over the good path.

For a user u, the set of best paths $\mathcal{B}_u$ and the set of paths with maximum window size $\mathcal{M}_u$ depend non-continuously on the probability of loss on each route, as well as on the various window sizes of the routes of this user. This implies that the right-hand side of Equation (7) is not a continuous function of x_r. Therefore, this differential equation is not well defined and can have no solutions. A natural way to deal with a differential equation with a discontinuous right-hand size is to replace the differential equation (7) by a differential inclusion $dx/dt \in F(x)$ where the discontinuous α_r of (7) is replaced by the convex closure of the possible values of α in a small neighborhood of x [21, 22].

We show in [18], Appendix D, that the differential inclusion corresponding to the Equation (7) is

$$\frac{dx_r}{dt} = x_r^2 \left(\frac{1/\text{rtt}_r^2}{(\sum_{p \in \mathcal{R}_u} x_p)^2} - \frac{p_r}{2} \right) + \frac{\bar{\alpha}_r}{\text{rtt}_r^2}, \qquad (8)$$

where $\bar{\alpha} = (\bar{\alpha}_1 \dots \bar{\alpha}_{|\mathcal{R}_u|})$ is such that

$$(\bar{\alpha}_r \cdot |\mathcal{R}_u|) \in \begin{cases} [\mathbf{1}_{|\mathcal{B}_u|=1}, 1] & \text{if } r \in \mathcal{B}_u \setminus \mathcal{M}_u \\ [-1, -\mathbf{1}_{|\mathcal{M}_u|=1}] & \text{if } r \in \mathcal{M}_u \setminus \mathcal{B}_u \\ [-\mathbf{1}_{|\mathcal{B}_u| \geq 2}, \mathbf{1}_{|\mathcal{M}_u| \geq 2}] & \text{if } r \in \mathcal{M}_u \cap \mathcal{B}_u \\ \{0\} & \text{if } r \notin \mathcal{M}_u \cup \mathcal{B}_u \end{cases} \qquad (9)$$

with $\sum_{r \in \mathcal{R}_u} \bar{\alpha}_r = 0$ and $\sum_{r \in \mathcal{B}_u} \bar{\alpha}_r = 1/|\mathcal{R}_u|$ if $\mathcal{B}_u \cap \mathcal{M}_u = \emptyset$. The notation $\mathbf{1}_{|\mathcal{B}_u|=1}$ means that this term is equal to 1 if $|\mathcal{B}_u| = 1$ and 0 otherwise. For example, when there is only one best path (*i.e.* $|\mathcal{B}| = 1$), $\alpha_r = 1/|\mathcal{R}_u|$ for $r \in \mathcal{B}_u \setminus \mathcal{M}_u$. If there are two or more best paths (*i.e.* $|\mathcal{B}| \neq 1$), then $\alpha_r \in [0, 1/|\mathcal{R}_u|]$ for $r \in \mathcal{B}_u \setminus \mathcal{M}_u$.

Note that there are multiple $\bar{\alpha}$ that correspond to definition (9). The differential inclusion might have multiple solutions, but this does not affect our analysis [18].

5.2 Pareto Optimality of OLIA

A fixed point of the congestion control algorithm (8) is a vector of rates $x = (x_1 \dots x_{|\mathcal{R}|})$ such that there exists $\bar{\alpha}$ satisfying (9) and such that, Equation (8) is equal to zero for any route r. We say that x is a non-degenerate allocation of rates if each user transmits with a non-zero rate on at least one of its paths. In practice, due to re-establishment routines in traditional TCP, the allocation of rates will not be degenerate. Hence, in our analysis, we consider only the non-degenerate fixed points and analyze their properties.

THEOREM 1. *Any non-degenerate fixed point x of OLIA congestion control algorithm, given by Equation* (8)*, has the following properties:*

(i) Only the best paths will be used, i.e. *paths r with maximum $\sqrt{2/p_r}/\mathrm{rtt}_r$.*

(ii) The total rate obtained by a user u is equal to the rate that a regular TCP user would receive on the best path available to u:

$$\sum_{r\in\mathcal{R}_u} x_r = \max_{r\in\mathcal{R}_u} \frac{1}{\mathrm{rtt}_r}\sqrt{\frac{2}{p_r}}.$$

PROOF. The proof is given in Appendix A. □

This theorem implies the following corollary:

COROLLARY 2. *OLIA satisfies the three design goals suggested by the RFC [10].*

PROOF. The proof is given in Appendix B. □

The following theorem gives a global optimality property of OLIA. For a rate allocation x, we define the total congestion cost by $C(x) = \sum_\ell \int_0^{\sum_{r\ni\ell} x_r} p_\ell(y)dy$.

THEOREM 3. *Any non-degenerate fixed point x of our congestion control algorithm* (8) *is Pareto optimal, i.e.:*

- *It is impossible to increase the quantity $\sum_{r\in\mathcal{R}_u} x_r/\mathrm{rtt}_r^2$ for some users without decreasing it for others or increasing the congestion cost $C(x)$.*

PROOF. The proof is given in Appendix C. □

Remark 1. If the probability p_ℓ is sharp around C_ℓ, i.e. if $p_\ell(y) \approx 0$ when $y < C_\ell$ and p_ℓ grows rapidly when y exceeds C_ℓ, then the cost C is a binary function: it is very small if the capacity constraints $\sum_{r\in\ell} x_r \leq C_\ell$ are respected and grows rapidly otherwise. In this case, Theorem 3 shows that if x is a fixed point of our algorithm, it is impossible to increase the quantity $\sum_{r\in\mathcal{R}_u} x_r/\mathrm{rtt}_r^2$ for some users without decreasing it for others while respecting the capacity constraints.

Remark 2. As pointed out by Kelly [2], as $C(x)$ is an increasing function of rates, single-path congestion control mechanisms are always Pareto optimal and the choice of an allocation of rates is only a matter of fairness. However, if we have multiple paths, it is likely that an algorithm will lead to a non-Pareto optimal allocation [2]. Theorem 3 guarantees that this cannot happen with OLIA. As a consequence, our algorithm will not exhibit either problem P1 nor P2.

Remark 3. Although the utility function of each user $\sum_{r\in\mathcal{R}_u} x_r/\mathrm{rtt}_r^2$ could appear to be an ad-hoc utility function, it reflects the fact that like TCP, OLIA favors paths with low rtt. When all paths belonging to a user have the same RTT, this theorem implies that the rate allocation of OLIA is such that one user cannot increase its rate without decreasing the rate of some other users. Hence, OLIA can successfully avoid problems P1 and P2. When RTTs over paths available to a user are different, satisfying goals 1 and 2 of the RFC [10] can lead to sending traffic on paths that are not the least congested but have a small round trip times. Therefore, using a TCP-compatible algorithm, it is not possible to avoid problems P1 and P2 in all possible settings. However, we can see from Theorem 1 that by using OLIA, only the best paths available to a user would be used. This indicates that OLIA provides an allocation as close as or closer to the optimal than any TCP-compatible algorithm. To completely avoid problems P1 and P2, it is necessary to depart from the compatibility with regular TCP by using congestion mechanisms that are less sensitive to round trip times, such as CUBIC [23] or STCP [24].

5.3 TCP Compatibility

As we show in Appendix C, OLIA maximizes the utility function $V^*(x)$ given by Equation (11). We now show that our algorithm is fair with the regular TCP under the assumption (A): all the paths belonging to a user u have the same RTT rtt_u. Under this assumption, $V^*(x)$ simplifies as follows:

$$V(x) = \sum_{u\in\mathcal{U}} -\frac{1}{\mathrm{rtt}_u^2 \sum_{r\in\mathcal{R}_u} x_r} - \frac{1}{2}\sum_{l\in\mathcal{L}} \int_0^{\sum_{r\ni l} x_r} p_\ell(x)dx,$$

where x is the set of all the rates of the users.

THEOREM 4. *Under the assumption (A), the congestion control algorithm defined by Equation* (8) *converges to a maximum of the utility function V:*

$$\lim_{t\to\infty} V(x(t)) = \max_{x\geq 0} V(x).$$

PROOF. The proof is given in Appendix D. □

This implies that OLIA maximizes the same utility function as the regular TCP of [25] where we replace the rate of a connection by the total rate that a user achieves on all its paths. If the probabilities of loss p_ℓ are sharp around C_ℓ, then our algorithm converges to an optimum of the following global maximization problem:

$$\max \sum_{u\in\mathcal{U}} -\frac{1}{\mathrm{rtt}_u^2 \sum_{r\in\mathcal{R}_u} x_r} \quad \text{subject to} \quad \begin{cases} \sum_{r\ni\ell} x_r \leq C_\ell \\ x_r \geq 0. \end{cases}$$

This is analog to the TCP maximization problem.

6. OLIA EVALUATION: MEASUREMENTS AND SIMULATIONS

In this section, we study the performance of MPTCP with OLIA, through measurements and by simulations. We first perform measurements on our testbed to show that OLIA outperforms LIA in all the scenarios from Section 3, as evidence that OLIA solves problems P1 and P2. Results from this section are in line with our theoretical analysis from Section 5. We then study the performance of OLIA in a data center by using htsim simulator [7].

6.1 Performance of OLIA in Scenarios A,B, C

In this section, we study the performance of MPTCP with OLIA, in the scenarios A,B and C described in Sections 3.2

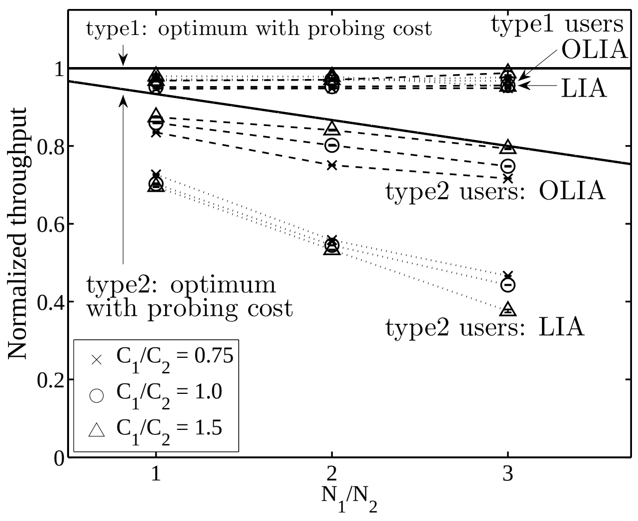


Figure 9: Scenario A - Normalized throughput of type1 and type2 users: we compare performance of LIA and OLIA. By using OLIA, type2 users achieve up to 2 times higher rates. OLIA performs close to the theoretical optimum with probing cost.

to 3.4. We show that in practice, OLIA is very close to the theoretical optimum with probing cost. These results are obtained through measurements over our testbed, by using our Linux implementation of OLIA.

6.1.1 Scenario A

We have shown in Section 3.2 that when the addition of an extra link does not help (like in Scenario A), using MPTCP with LIA can reduce the throughput of competing TCP users. In this section, we show by measurements that MPTCP with OLIA significantly outperforms MPTCP with LIA and comes close to the theoretical optimum with probing cost.

Figures 9 and 10 report measurements obtained on the testbed shown in Figure 2. Figure 9 depicts the normalized throughput of type1 and type2 users that use LIA or OLIA. The results show that OLIA performs close to an optimal multipath algorithm that transmits the minimum traffic over congested paths (theoretical optimum with probing cost). OLIA significantly outperforms LIA: by using OLIA, type2 users achieve rates up to two times higher than with LIA, with no reduction for type1 users.

Figure 10 depicts the measured loss probability p_2 on the shared access point. We observe that OLIA balances congestion much better than LIA. When we use OLIA, p_2 increases only by a factor of 1.3 in the worst case, whereas with LIA, p_2 increases by a factor of 5. p_1 is almost the same when using LIA or OLIA.

6.1.2 Scenario B

We now show the performance of OLIA in the scenario B described in Section 3.3. As we have shown, OLIA is Pareto optimal. Hence, taking into account the minimum probing cost, we expect only 3% reduction in the Blue users' rates and in the aggregate throughput when we upgrade Red users to OLIA (see Figure 4(b)).

Table 2 presents the measurements for the scenario described in Section 3.3 using OLIA. We set C_X=27, C_T=36, C_Z=100, all in Mbps. We have 15 Red and 15 Blue users. We set RTTs to 150 ms over all paths. Our results show that there is a 3.5% decrement in aggregate throughput when we update Red users to OLIA, which is much smaller than the 13% reduction we observed when we used LIA (see Table 1). This 3.5% reduction in the aggregate throughput is due to the minimum traffic transmitted by users over congested paths and cannot be reduced as it is bounded below by 1/rtt packets/sec.

6.1.3 Scenario C

Finally, we study the performance of MPTCP with OLIA in scenario C described in Section 3.4. Theorems 1 and 4 imply that by using our algorithm, multipath users do not send any traffic on their path crossing AP2. Hence, in theory, OLIA provides a fair allocation among users and performs as an optimal algorithm (Figure 5(b), dashed lines). Next, we show by measurements that OLIA is also fair in practice.

Figure 11 depicts the normalized throughput of single-path and multipath users, as a function of N_1/N_2 and for C_1/C_2=1, 2. We show the results for LIA and OLIA, as well as for an optimal algorithm with minimum probing cost. This figure shows that with OLIA multipath users transmit only one packet per RTT over AP2. Compared to LIA, type2 users receive up to 2 times higher throughput. Hence, OLIA is less aggressive than LIA towards regular TCP users.

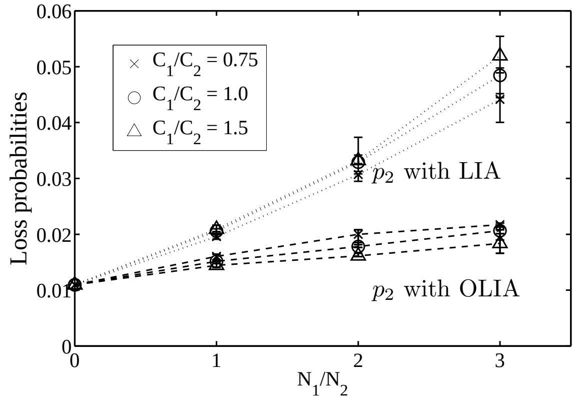


Figure 10: Scenario A - Loss probability p_2 at shared AP: we observe that OLIA significantly reduces the congestion level at this bottleneck and improves the congestion balancing.

Red users	Rate/user		Aggregate
	Blue users	Red users	
Single-path	2.2	1.8	59.3
Multipath	2.2	1.7	57.8

Table 2: Measurement results for scenario B showing the effect of upgrading the Red users from regular TCP to MPTCP with OLIA. We observe a small drop of 3.5% in the aggregate throughput, which is due to the overhead of minimum traffic (1/rtt) over the congested path. Compared to LIA (see Table 1), we see significant improvement.

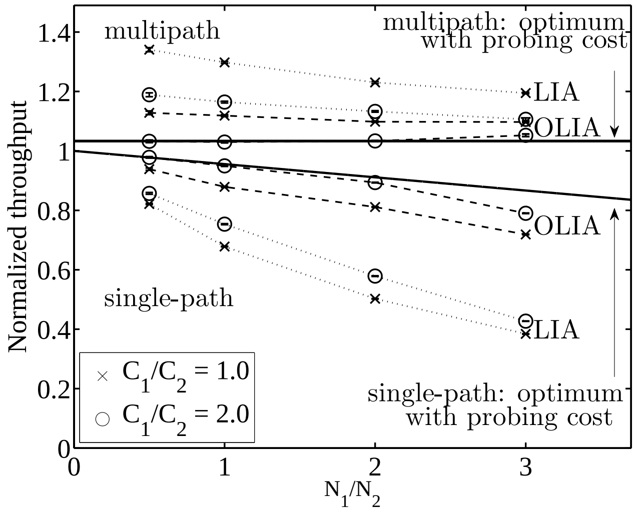


Figure 11: Scenario C - Normalized throughput of single-path and multipath users: we compare the performance of LIA and OLIA. We observe that by using OLIA, type2 users achieve up to 2 times higher rates. OLIA performs close to the theoretical optimum with probing cost.

Figure 12, shows the measured loss probability p_2. The results show again that OLIA balances congestion in the network and reduces the loss probability in bottlenecks much better than LIA. In particular, we observe that by increasing N_1 from 0 to $3N_2$, p_2 increases by a factor of 2 using OLIA, whereas the increase is in the order of 4 to 6 times when using LIA. p_1 is almost the same when using OLIA or LIA.

6.2 Performance of OLIA in Data Center and Dynamic Scenarios

The three preceding examples show that by providing a better congestion balance, MPTCP with OLIA outperforms MPTCP with LIA in Scenarios A, B, and C. In this section, we show that, by being non-flappy and as responsive as LIA, OLIA can fully use the multiple paths available in a data center. Our study is based on a series of scenarios in which MPTCP with LIA is studied in [7]. Because of space constraints, we present the results for only two of the cases where LIA was shown to be very efficient. We observe that OLIA performs as well or better than LIA in these two scenarios. This indicates that it is not flappy and has a very good responsiveness. These results are obtained using `htsim` simulator used in [7], provided by Raiciu et al. We implemented OLIA in the simulator and use the same scenarios as [7].

6.2.1 *Static FatTree Topology*

We first study exactly the same scenario as in [7], Section 4.2-Throughput: the network is a FatTree with 128 hosts, 80 eight-port switches, 100Mb/s links. Each host sends a long-lived flow to another host chosen at random. Figure 13(a) shows the aggregate throughput achieved by long-lived TCP and MPTCP (LIA and OLIA) flows. We show the results for different numbers of subflows used. Our results show that OLIA can successfully exploit the multiple paths that exist in the network and can use the available capacity.

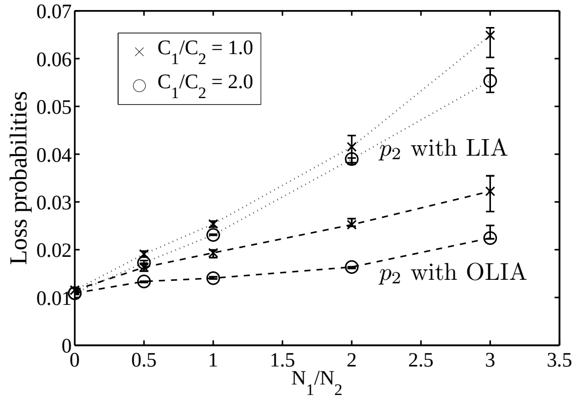
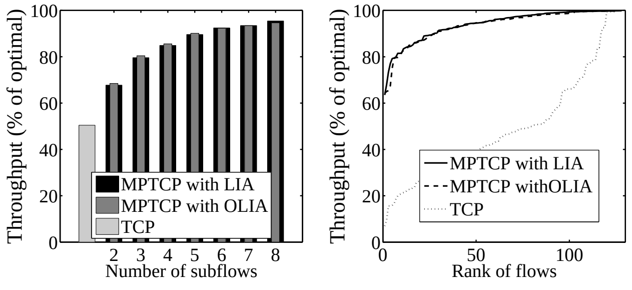


Figure 12: Scenario C - Loss probability p_2 at shared AP: we observe that OLIA significantly reduces the congestion level at this bottleneck (4 to 6 times lower compared tp LIA).



(a) Aggregated throughput. (b) Throughput of users.

Figure 13: Performance of OLIA in a FatTree with many possible parallel paths between users. It successfully explores the path diversity that exists in the network and uses the available capacity (a sign of non-flappiness). LIA performs similarly as, in this scenario, it can successfully balance the congestion.

This is a sign that it is not flappy. Regular TCP shows a poor performance. Figure 13(b) shows the throughput of individual users ranked in order of achieved throughputs, for LIA and OLIA with 8 subflows per user and with TCP; LIA and OLIA provide similar fairness among users and are more fair than TCP. We observe that, in this scenario, LIA performs close to an optimal algorithm and exhibits a similar performance to OLIA. The reason is that the users have multiple equally good paths. Hence, LIA also successfully balances the congestions in the network, similarly to OLIA, and performs optimally. We measured the loss probabilities of links available to users and results confirm our reasoning.

6.2.2 *Dynamic Setting with Short Flows*

We study the same scenario as the one described in Section 4.3.4-ShowFlows of [7]. The scenario is a 4:1 oversubscribed FatTree where each host sends to one other host. One-third of the hosts send a continuous flow by using either TCP, MPTCP with LIA (8 subflows) or with OLIA (8 subflows). The remaining hosts send short flows of size 70Kbyte every 200ms on average (they generate these flows according to a Poisson process). They use regular TCP. This is a highly

algorithm	Short flow finish time (mean/stdev)	Network core utilization
MPTCP - LIA	98 ± 57 ms	63.2%
MPTCP - OLIA	90 ± 42 ms	63%
Regular TCP	73 ± 57 ms	39.3%

Table 3: Performance of OLIA in a highly dynamic setting. It uses the available capacity as efficient as LIA, but decreases the average completion time of short flows by 10%**.**

dynamic setting in which changes occur in the order of milliseconds. Table 3 shows the average completion time for short flows and the network core usage. Figure 14 shows the distribution of completion times of short flows. Our results show that although OLIA uses the available capacity as efficiently as LIA, the average completion time of short flows decreases by 10% using OLIA. Moreover, we observe in Figure 14 that OLIA decreases the completion time of both fast and slow short flows. For slow flows, the decrease is more than 25%. This shows that OLIA has a better responsiveness than LIA, is more fair to TCP users, and uses capacity quickly when it is available. With TCP, we have a lower average completion time for short flows, but very low network utilization.

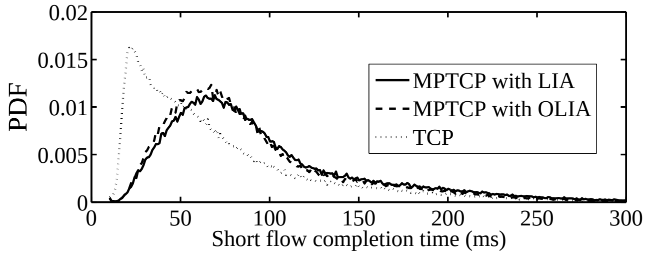


Figure 14: Completion time of short flows competing with long-lived TCP, MPTCP with LIA, or MPTCP with OLIA flows in a highly dynamic setting. OLIA reacts faster to the changes in the network and is fairer toward short flows.

7. CONCLUSION

We have shown that MPTCP with LIA suffers from important performance issues. Moreover, it is possible to build an alternative to LIA, which performs close to an optimal algorithm with probing cost while being as responsive and non-flappy as LIA. Our theoretical results show that our proposed algorithm, OLIA, is Pareto-optimal and satisfies the three design goals of MPTCP [10]. Moreover, we have shown through measurements and by simulation that OLIA is as responsive and non-flappy as LIA, and that it solves identified problems with LIA.

Multiple directions could be explored to go further. A first one would be to act on the minimum probing traffic rate by an adjustment of the retransmit timer – in our current implementation, the minimum window size is 1 and the minimum probing rate is $1/\text{rtt}_r$. Another direction comes from the fixed point analysis of Theorem 3. The stability and convergence of OLIA is another important question that will be studied in future work.

8. REFERENCES

[1] M. Allman, V. Paxon, and E. Blanton. Tcp congestion control. In *RFC 5681*, September 2009.

[2] F.P. Kelly. Mathematical modelling of the internet. *Mathematics unlimited-2001 and beyond.*

[3] 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.

[4] F. Kelly and T. Voice. Stability of end-to-end algorithms for joint routing and rate control. *ACM SIGCOMM CCR*, 35, 2005.

[5] 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. *ToN*, 14, 2006.

[6] W.H. Wang, M. Palaniswami, and S.H. Low. Optimal flow control and routing in multi-path networks. *Performance Evaluation*, 52, 2003.

[7] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handly. Improving datacenter performance and robustness with multipath tcp. *ACM Sigcomm*, 2011.

[8] D. Wischik, M. Handly, and C. Raiciu. Control of multipath tcp and optimization of multipath routing in the internet. *NetCOOP*, 2009.

[9] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handly. Design, implementation and evaluation of congestion control for multipath tcp. *Usenix NSDI*, 2011.

[10] C. Raiciu, M. Handly, and D. Wischik. Coupled congestion control for multipath transport protocols. *RFC 6356 (Experimental)*, 2011.

[11] A. Ford, C. Raiciu, M. Handley, S. Barre, and J.Iyengar. Architectural guidelines for multipath tcp development. *RFC 6182 (informational)*, 2011.

[12] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. Tcp extensions for multipath operation with multiple addresses. *IETF Internet Draft*, 2011.

[13] C. Cetinkaya and E.W. Knightly. Opportunistic traffic scheduling over multiple network paths. In *INFOCOM*, 2004.

[14] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *USENIX*, 2004.

[15] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda. Multipath congestion control for shared bottleneck. In *PFLDNeT workshop*, 2009.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18, 2000.

[17] V. Misra, W.-B. Gong, and D. Towsley. Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. In *SIGCOMM*, 2000.

[18] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec. Non pareto-optimality of mptcp: Performance issues and a possible solution. *EPFL Technical report. Available at* *http://infoscience.epfl.ch/record/177901*, 2012.

[19] http://mptcp.info.ucl.ac.be/.

[20] V. Jacobson. Congestion avoidance and control. In *ACM SIGCOMM CCR*, volume 18, 1988.

[21] N. Gast and B. Gaujal. Mean field limit of non-smooth systems and differential inclusions. *ACM SIGMETRICS Performance Evaluation Review*, 38(2):30–32, 2010.

[22] N. Gast and B. Gaujal. Markov chains with discontinuous drifts have differential inclusion limits. *Performance Evaluation*, 2012.

[23] S. Ha, I. Rhee, and L. Xu. Cubic: A new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review*, 42, 2008.

[24] T. Kelly. Scalable tcp: Improving performance in highspeed wide area networks. *ACM SIGCOMM CCR*, 33, 2003.

[25] S. Kunniyur and R. Srikant. End-to-end congestion control schemes: Utility functions, random losses and ecn marks. *ToN*, 11, 2003.

APPENDIX

We provide sketches of proofs for the results in Section 5. Detailed proofs are available in [18], Appendices E to H.

A. SKETCH OF PROOF (THEOREM 1)

Proof of *(i)*. Let x be a non-degenerate fixed point of OLIA. For any path $p \in \mathcal{R}_u$, the equation dx_p/dt contains two terms, denoted term A and term B in the next equation:

$$0 = \frac{dx_p}{dt} = x_p^2 \underbrace{\left(\frac{1/\text{rtt}_p^2}{(\sum_{s\in\mathcal{R}_u} x_s)^2} - \frac{p_p}{2} \right)}_{\text{term } A} + \underbrace{\bar{\alpha}_p}_{\text{term } B} . \qquad (10)$$

Assume that there exists a non-best path $r \notin \mathcal{B}_u$ such that $x_r > 0$. We show that this leads to a contradiction.

Equation (10) shows that the term A is positive for r and hence is strictly positive for any best paths (by definition of best path). If $\mathcal{B}_u \cap \mathcal{M}_u \neq \emptyset$, there exists a best path p with maximum window size. Thus, we have $x_p \neq 0$, which implies that $dx_p/dt > 0$ as α_p is non-negative. If $\mathcal{B}_u \cap \mathcal{M}_u = \emptyset$, then there exists $p \in \mathcal{B}_u$ such that $\alpha_p > 0$, which implies that $\alpha_p > 0$ and thus $dx_p/dt > 0$. In both cases, we have $dx_p/dt > 0$ which contradicts that $dx_p/dt = 0$.

This shows that for any non-best path $r \notin \mathcal{B}_u$, we must have $x_r = 0$.

Proof of *(ii)*. Because of *(i)*, we have $\bar{\alpha}_r = 0$ for all routes $r \notin \mathcal{B}_u$. Thus, we can show that $\bar{\alpha}_r = 0$ for all paths $r \in \mathcal{R}_u$. Therefore, any fixed point x satisfies $x_r = 0$ or $\sum_{p\in\mathcal{R}_u} x_p = 1/\text{rtt}_r\sqrt{2/p_r}$. By assumption, x is non-degenerate, which means that there exists a route $r \in \mathcal{R}_u$ such that $x_r \neq 0$. This concludes the proof of *(ii)*. □

B. SKETCH OF PROOF (COROLLARY 2)

Point *(ii)* of Theorem 1 implies that OLIA satisfies goal 1: the total rate that OLIA gets ($\sum_{r\in\mathcal{R}_u} x_r$) is the same as the rate that a regular TCP would get on its best link ($\max_{r\in\mathcal{R}_u} \sqrt{2/p_r}/\text{rtt}_r$). Moreover, as OLIA only uses its best paths, it does not transmit more than a regular TCP does on any of its paths and satisfies goal 2. Finally, as OLIA only uses its best path, it perfectly balances congestion and satisfies goal 3. □

C. SKETCH OF PROOF (THEOREM 3)

Let x^* be a fixed point of the algorithm and define the utility function $V^*(x)$ as

$$\sum_{u\in\text{users}} -\frac{1}{\tau_u^2 \sum_{r\in\mathcal{R}_u} \frac{x_r}{\text{rtt}_r^2}} - \frac{1}{2} \sum_{\ell\in\text{links}} \int_0^{\sum_{r\in\ell} x_r} p_\ell(x)dx, \quad (11)$$

where τ_u is defined by: $\tau_u = (\sum_{r\in\mathcal{R}_u} x_r^*)/(\sum_{r\in\mathcal{R}_u} x_r^*/\text{rtt}_r^2)$.

The function V^* is a non-positive function that goes to $-\infty$ when $x \to \infty$. Therefore, it has a maximum, attained for a finite x. By concavity of V^*, a necessary and sufficient condition for a point x to be a maximizer of $\mathcal{U}$ is that for every route r, we have

$$\frac{\partial V^*}{\partial x_r}(x) \leq 0 \quad \text{and} \quad \frac{\partial V^*}{\partial x_r}(x) = 0 \text{ or } x_r = 0.$$

By the definition of V^* and Theorem 1, the fixed point x^* is a maximum of V^*. Since V^* is an increasing function of $\sum_{r\in\mathcal{R}_u} x_r/\text{rtt}_r^2$ and a decreasing function of the congestion cost, it is impossible to increase $\sum_{r\in\mathcal{R}_u} x_r/\text{rtt}_r^2$ for some users without decreasing it for others or increasing the congestion cost. □

D. SKETCH OF PROOF (THEOREM 4)

Theorem 4 assumes that all the paths belonging to user u have the same round trip time rtt_u. In that case, the function V is the same as V^* of Equation 11 with $\tau_u = \text{rtt}^2$.

Let x be one of the solutions of the differential inclusion given by Equation (8). Then, there exists a function $\bar{\alpha}(t)$ satisfying Equation (9) for all t and such that dx_r/dt satisfies Equation (8).

When running the algorithm, the derivative of $V(x(t))$ w.r.t. time satisfies $dV/dt = \sum_{u,r}(\partial V/\partial x_r)(dx_r/dt)$. Thus:

$$\begin{aligned}
\frac{d}{dt}V(x(t)) =& \sum_{u\in\mathcal{U}} \sum_{r\in\mathcal{R}_u} \frac{\partial V}{\partial x_r}\frac{dx_r}{dt} \\
=& \sum_r \left(\frac{1}{\text{rtt}_u^2(\sum_{r\in\mathcal{R}_u} \frac{w_r}{\text{rtt}_u})^2} - \frac{p_r}{2} \right) \\
& \cdot \left(\frac{w_r^2}{\text{rtt}_u} \left(\frac{1}{(\sum_{p\in\mathcal{R}_u} w_p)^2} - \frac{p_r}{2} \right) + \frac{\alpha_r}{\text{rtt}_u} \right) \\
=& \sum_{u\in\mathcal{U}} \sum_{r\in\mathcal{R}_u} x_r^2 \left(\frac{1}{\text{rtt}_u^2(\sum_{p\in\mathcal{R}_u} x_p)^2} - \frac{p_r}{2} \right)^2 \quad (12) \\
&+ \sum_{u\in\mathcal{U}} \sum_{r\in\mathcal{R}_u} \left(\frac{1}{\text{rtt}_u^2(\sum_{p\in u} x_p)^2} - \frac{p_r}{2} \right) \frac{\bar{\alpha}_r}{\text{rtt}_u^2} \quad (13)
\end{aligned}$$

By definition of $\bar{\alpha}$, we have $\sum_{r\in\mathcal{R}_u} \alpha_r = 0$. Moreover, when all rtt are equal, the best paths are the paths with minimal probability loss and $\alpha_r \leq 0$ for such paths. Thus:

$$\sum_{r\in\mathcal{R}_u} \alpha_r p_r = \sum_{r\in\mathcal{B}_u} \alpha_r p_r + \sum_{r\notin\mathcal{B}_u} \alpha_r p_r \leq \sum_r \alpha_r p_{\min} = 0.$$

These two properties together show that the term (13) is non-negative. Since (12) is also non-negative, this shows that $dV(x(t))/dt \geq 0$ for all t. Thus, the function V is non decreasing. Since V is non-positive, this shows that $\lim_{t\to\infty} dV(x(t))/dt = 0$.

Let x^* be a limit point of $x(t)$. We show in [18], Appendix H, that x^* is a fixed point of the algorithm. Thus, by Theorem 3, it is a maximizer of V. □