                                                 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

Table of Contents

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: $l_{2r} \leftarrow l_{2r}$ + (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

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

   multiplied by MSS_r * bytes_acked.

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

   alpha_r is calculated as follows:

   - If r is in collected_paths, then

$$alpha_r = \frac{1/number\_of\_paths}{|collected\_paths|}$$

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

$$alpha_r = -\ \frac{1/number\_of\_paths}{|max\_w\_paths|}$$

   - Otherwise, alpha_r=0.

|collected_paths| and |max_w_paths| are the number of paths in
collected_paths and in max_w_paths. Note that the sum of all alpha_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 alpha_r, guarantees
responsiveness and non-flappiness of our algorithm.

By definition of alpha_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 alpha_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

$$\frac{w\_r/rtt\_r^2}{(SUM\ (w\_p/rtt\_p))^2} + \frac{1}{w\_r * number\_of\_paths * |collected\_paths|} \qquad (2)$$

  multiplied by MSS_r * bytes_acked.

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

$$\frac{w\_r/rtt\_r^2}{(SUM\ (w\_r/rtt\_r))^2} - \frac{1}{w\_r * number\_of\_paths * |max\_w\_paths|} \qquad (3)$$

  multiplied by MSS_r * bytes_acked.

  - Otherwise, increase w_r by

$$\frac{(w\_r/rtt\_r^2)}{(SUM\ (w\_r/rtt\_r))^2} \qquad (4)$$

  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 >= 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. 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>.

[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