Hybrid Delay-Based Congestion Control for Multipath TCP

Regel Gonzalez, Juan Pradilla, Manuel Esteve, Carlos E. Palau
Universitat Politècnica de València (UPV)
Valencia, Spain
regonus@doctor.upv.es, juaprace@teleco.upv.es, mesteve@dcom.upv.es, cpalau@dcom.upv.es

***Abstract*—Current algorithms for MPTCP (as LIA, OLIA, BALIA, or wVegas) present loss-based congestion control on the exception of wVegas. Delay-based congestion control allows a preventive action against congestion, capable to avoid loss up to some extent, unlike loss-based congestion control. Additionally delay-based congestion control induces lower delay and presents higher fairness, but poor performance interoperating with loss-based flows, as get a poor share of the available bandwidth. We propose DAIMD, a hybrid congestion control for Multipath TCP, based on the delay-based AIMD scheme, which benefits from better, preventive detection of congestion, a more responsive use of queues and consequently low induced delay, as well as the capability to coexist in fair conditions with loss-based flows in shared links. Our system presents its own analysis criteria for detecting incipient congestion that differs from other delay-based schemes on which it is based, such as CDG, delay-based AIMD and Vegas..**

***Keywords*—Delay-based congestion control, Multipath TCP, low delay, loss avoidance, hybrid congestion control, low queueing delay** .

I. Introduction

Multipath TCP [1][3][4][5] is a new transmission protocol that allows the use of multiple paths simultaneously on the network between two end hosts for a single data transmission. Unlike what is the case with conventional, single TCP, data transmission is not constrained to a single path. This MPTCP feature provides several considerable benefits in performance, reliability, resilience, congestion avoidance, among others aspects, compared with single-path TCP. MPTCP aims to be TCP-friendly by responsively taking a fair share instead of all the potentially available bandwidth on the whole set of paths [2][5][6][8]. A MPTCP connection is composed of several single TCP connections, which we called 'sub-connections' or 'subflows', across different paths, called ‘subpaths’, between the end-hosts. In upper layers, the whole set of MPTCP sub-connections behaves as a single TCP connection, with a unique data transmission end-to-end.

One main function of TCP is congestion control, which allows TCP connections to maintain an appropriate data transmission rate, not underutilizing the available capacity, without triggering collapse in the network. MPTCP congestion control [5][3] is more complex and requires an additional function: appropriate data load balancing among all sub-connections on the different paths. Typically TCP congestion control is loss-based: periodical data packet loss is necessary to regulate transmission rate; preventive action against congestion and loss is impossible [7]. On the other hand, TCP delay-based congestion control allows a more precise congestion detection which allows a quicker, preventive action against congestion before loss occurs, which is costly in performance, and a fine-grained rate regulation. Among current congestion control algorithms for MPTCP (LIA[5][6], OLIA, BALIA, wVegas[9]..) only wVegas [9], applies delay-based congestion control [9][10]. Not only spares the cost of loss in performance as far as possible, presents a more effective preventive action against congestion, and induces lower delay [9]; a finer transmission rate regulation can allow for a better, more accurate and fairer load balance among subflows. However classic delay-based congestion control [11] like Vegas [11] or wVegas [9] presents the downside of a very poor performance sharing links with loss-based background traffic, as loss-based congestion control is more aggressive hogging bandwidth.

Hybrid congestion control [12] combines aspects of both loss-based and delay-based congestion control, allowing the benefits of an early detection of incipient congestion, before loss occurs, and an appropriate performance interoperating with loss-based implementations.

We designed DAIMD, a novel, hybrid congestion control for Multipath TCP. DAIMD allows a more effective, preventive action against congestion than loss-based MPTCP implementations, and an appropriate performance with loss-based background traffic. DAIMD is based on delay-based AIMD [12] but presents a different criterion for triggering rate decrease. We implemented our congestion control approach on Linux kernel and evaluated its performance on simulations using real network stack instead of models. We investigated if DAIMD presents the expected benefits of hybrid congestion control, and performs as a functional MPTCP implementation, fulfilling the MPTCP congestion control goals [6][13].

This paper is structured as follows: Section II gives an overview of the congestion control goals of MPTCP, delay-based congestion control and the algorithms delay-based AIMD and LIA. Section III describes our proposed algorithm and implementation challenges. In Section IV we describe our simulation environment and setup, present simulation results, and evaluate our algorithm. Section V summarizes our results and gives an outlook on possible further approaches on congestion control for MPTCP and practical applications of MPTCP with our own hybrid congestion control approach in different utilization cases and environments.

II. Congestion Control

Multipath TCP congestion control is more complex than classic, single-path TCP congestion control, as it requires additional functionality for appropriate traffic load balance among paths, which should guarantee fairness to TCP. Congestion control for Multipath TCP is based on three design goals [5][6]:

1) Improving connection throughput - A MPTCP connection should take up at least the same bandwidth as a single TCP connection would take up instead on the best path. Ideally, the fairest situation to TCP means to take up exactly the same amount of bandwidth. In any case, this share should not be excessively larger than what TCP would get. MPTCP aims to be fair to TCP, and therefore to take a fair share of the available capacity in the network. As an exception, on idle paths is appropriate and convenient to take up the whole available bandwidth.

2) Not harming other TCP connections – A MPTCP subflow should not take more bandwidth than a single TCP flow performing on the same path.

3) Load balance – MPTCP should utilize more the best, least congested paths and take traffic off from the worst, more congested paths.

Multipath TCP is based on the fairness goals of the Resource Pooling Principle [8] , which aims to increase fairness, efficiency and resource distribution, and to reduce congestion through the network, by rmaking a set of connections behave conjointly as a single one [8][3] in a responsable way in terms of traffic balance and fairness to other connections.

Multipath TCP has a global action against congestion by moving traffic off the worst, most congested paths to the best, least congested paths alleviates congestion in the network [6]. This way not only the MPTCP connection does achieve higher efficiency, a better distribution of resources and congestion avoidance. Besides the other flows sharing links with MPTCP subflows benefit from this action, as the overall congestion is reduced by redirecting traffic off the most congested links. MPTCP indirectly leads to a better network resource distribution among MPTCP and flows sharing links with MPTCP, and less overall congestion in the network [6].

Traditional loss-based congestion control, like Reno, follows the AIMD scheme [7]. The congestion window (CWND) which controls the number of packets sent in a round-trip time has an additive increase until packet loss is detected. Then the congestion window shrinks abruptly (multiplicative decrease) as loss is an indicative of an excessive transmission rate, to continue growing in further transmissions with additive increase.

Delay-based AIMD [12] is a hybrid congestion control algorithm. It follows the AIMD scheme, but the multiplicative decrease is triggered by a delay-based estimation of congestion, instead of loss. Due to its additive increase it is able to interoperate with loss-based flows, unlike classic delay-based congestion control.

III. DAIMD Design and Implementation

A. DAIMD Algorithm Design

We implemented the hybrid delay-based AIMD scheme on MPTCP, with some changes on the decision criterion that triggers the multiplicative decrease. Our DAIMD algorithm follows a delay-based criterion for the multiplicative decrease, whereas presents the additive increase, coupling and general behaviour of LIA [5], the first algorithm proposed by the IETF for Multipath TCP.

We developed our own criterion for triggering the multiplicative decrease, after being inspired on the criteria aspects of several delay-based algorithms. This is described in (1). In case of loss multiplicative decrease is also triggered. DAIMD follows the LIA rule for loss.

The backoff factor is chosen to make DAIMD performance similar to LIA, despite DAIMD earlier congestion detection. DAIMD detects the need of a multiplicative decrease before loss occurs, therefore gets a lower top value of the congestion window than LIA under the same conditions. DAIMD should have a less aggressive backoff, as the preventive action against congestion prevents as well the attaining of a higher congestion window size.

DAIMD scheme – Rules for each subflow r:

Increase on each ACK of subflow r

$$cwnd_r \leftarrow cwnd_r + min\left(1/\mathrm{cwnd}_r, \alpha / cwnd_{total}\right)$$

Decrease

$cwnd_r \leftarrow cwnd_r * 0.7$ on delay congestion detection

$cwnd_r \leftarrow cwnd_r * 0.5$ on every loss of subflow r

$$\alpha = \max\left(\frac{cwnd_j}{rtt_j^2}\right) / \left(\sum \frac{cwnd_i}{rtt_i}\right)^2$$

$$cwnd_{total} = \sum cwnd_i \text{ ,}$$

$rtt_r, cwnd_r$: round-trip time & CWND on path *r*

Delay Congestion Detection

If D>=D0 & cwnd>W0 & delay values that activate the first condition hold during T time (hysteresis)

D = packet delay; *D0* = delay that triggers delay-based backoff; *W0* = *CWND* threshold that activates delay-based action

(1)

B. *Implementation*

We implemented our new approach in Linux, a real operating system. We implemented our congestion control as a new Linux kernel module. For the sake of simplificity we did not implement the MPTCP operations nor the TCP protocol extension but only the coupled congestion control. This approach is sufficient to evaluate our congestion control approach, as the overhead signaling has no influence on congestion control behaviour.

IV. Results and Evaluation

A. *Simulation Setup*

For the simulation we use the NS-3 simulator, the most widely used in the scientific community. We include real network stack from the kernel code into NS3. Unlike the use of models for simulation, the use of real network stack provides extremely precise results, which are considered equivalent to those obtained in real networks.

We used simple traffic scenarios to pursue a basic evaluation of our algorithm. As Fig. 1 shows there are two paths across the network between the two end-hosts that maintain an MPTCP connection. Each path is an ideal link with its queue sized by its bandwidth-delay-product. All MPTCP subconnections of the same host belong to the same MPTCP connection. Delay and capacity characteristics of each subpath, as well as the presence or not of additional loss-based background traffic (Reno) vary on each individual simulation. Specifications of the traffic scenario are indicated on the results. Sources are greedy generators which start at the same time.

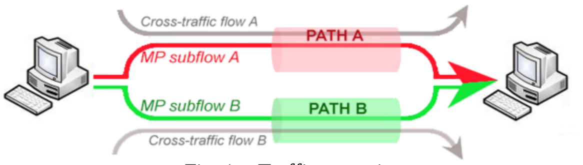

Fig. 1 - Traffic scenario

B. *Results*

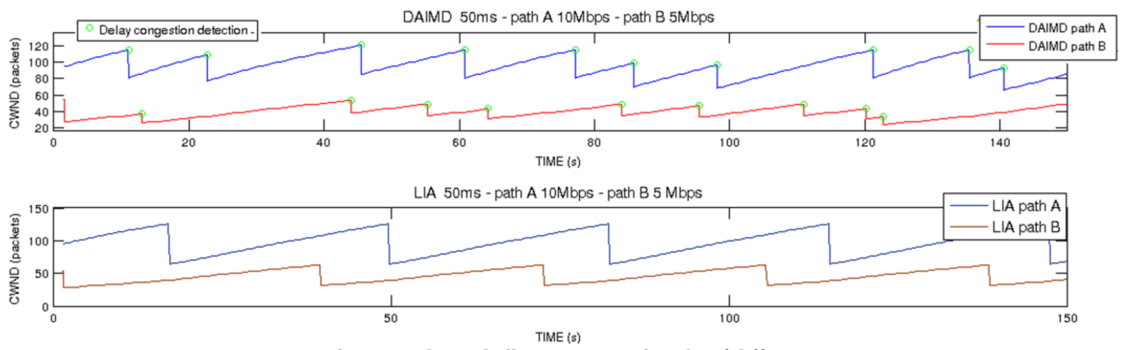

Fig. 2 – DAIMD and LIA on identical idle scenarios with paths of different capacities.

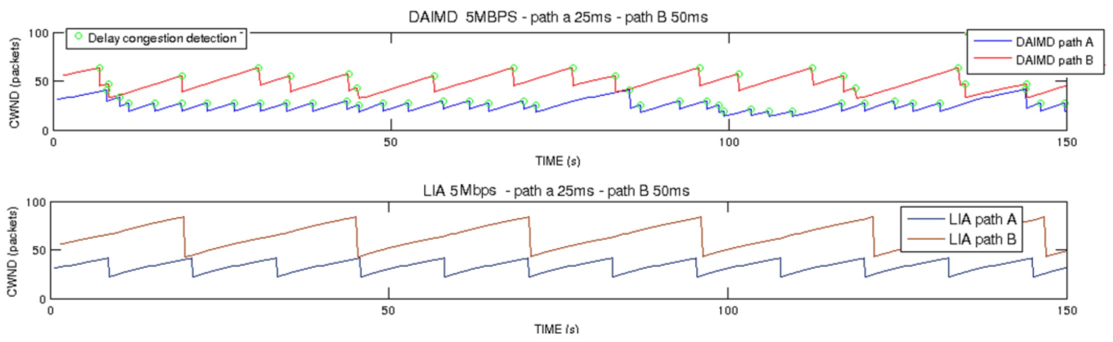

Fig. 3 – DAIMD and LIA on identical idle scenarios with paths of different delay.

TABLE I – Average Throughput of each Flow (in Mbps)

DAIMD	**Different capacity, same base delay** Path A – 10Mbps OWD 50ms Path B – 5Mbps OWD 50ms	**Different base delay, same capacity** Path A – 5Mbps OWD 25ms Path B - 5Mbps OWD 50ms
Path A	9.4	4.5
Path B	4.2	4.6
Total MPTCP throughput	13.6	9.1

1) Simulation on idle paths

In Fig. 2 and 3 is possible to appreciate the bare behaviour of the algorithm on idle paths, without the influence of other flows. The idle scenarios have different characteristics of delay and capacity, specified on the results. In order to compare DAIMD congestion control with its equivalent loss-based approach, also LIA performance on identical scenarios is displayed. It can be seen that DAIMD performance is very similar to LIA, but with a smaller peak-to-peak oscillation, more irregular and with higher frequency. Maximum congestion window size on the same path conditions is slightly lower for DAIMD than for LIA, as DAIMD stops the increase of the CWND before LIA does, due to an earlier detection of congestion. Each early congestion detection, before loss occurs, is marked on the images with a circle. Peaks without a circle represent loss events. No loss occurs in the DAIMD transmissions after the slow-start phase, proving a high effectiveness of the delay-based congestion control. The delay-based congestion control should act against congestion by reducing the rate early enough to avoid loss, but not too soon to end up underutilizing the available capacity. An inappropriate, imprecise trigger may lead to bandwidth underutilization and inability to compete with other flows for a fair share of the link. DAIMD is able to maintain an adequate throughput on each subpath (see Table I). Therefore DAIMD accomplishes the MPTCP congestion control design goal for idle paths and behaves as a functional congestion control for MPTCP with an effective, preventive action against loss.

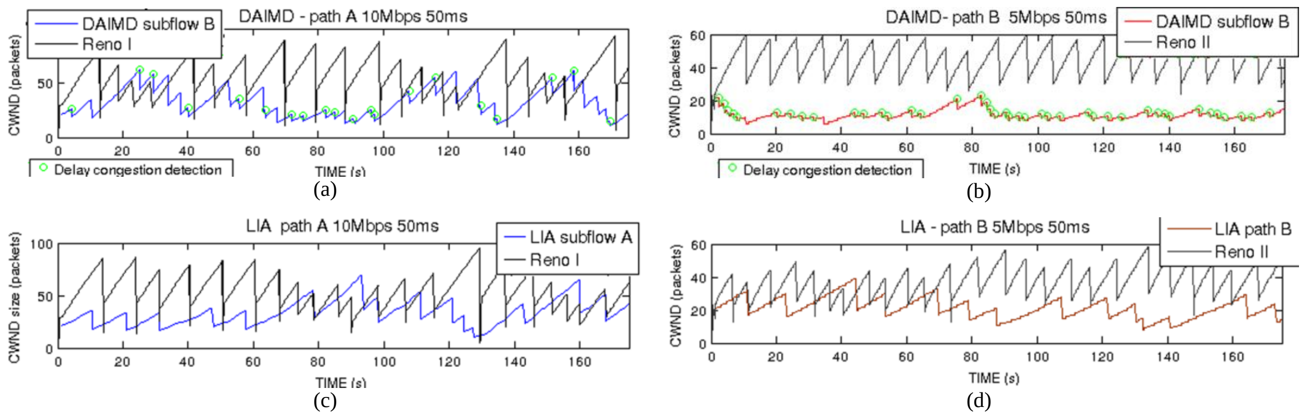

Fig. 4 - DAIMD and LIA on identical scenarios with paths with different capacity and Reno cross-traffic.

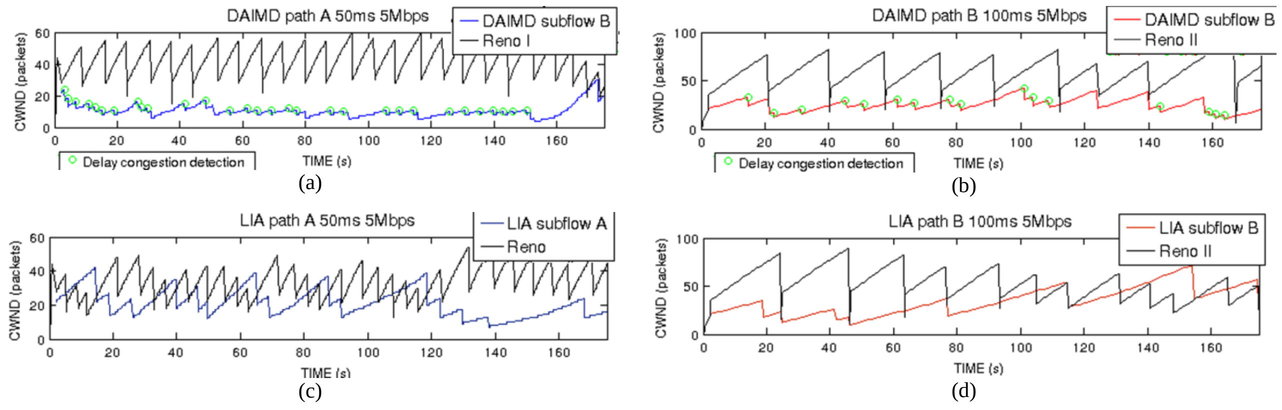

Fig. 5 - DAIMD and LIA on identical scenarios with paths with different delay and Reno cross-traffic.

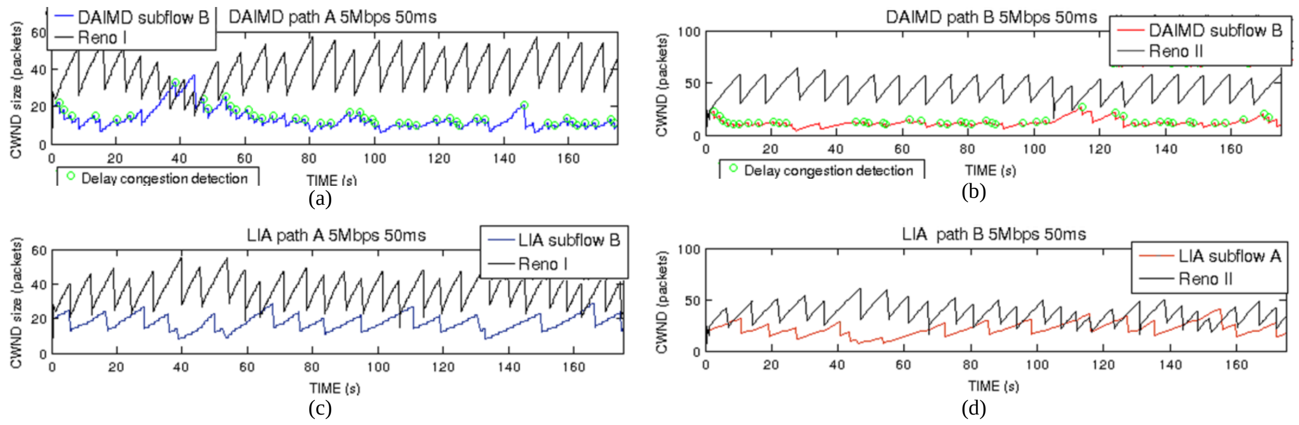

Fig. 6 - DAIMD AND LIA on identical scenarios with identical paths and Reno cross-traffic.

TABLE II - AVERAGE THROUGHPUT OF FLOWS (in Mbps)

	Paths with different capacity			Paths with different delay			Equal paths		
	DAIMD	LIA	Reno (DAIMD)	DAIMD	LIA	Reno (DAMID)	DAIMD	LIA	Reno (DAIMD)
Path A	3.4	3.6	5.8	1.4	1.8	2.9	1.3	1.5	3.5
Path B	1	1.7	3.7	1.1	1.6	2.8	1.2	1.7	3.7
Total MPTCP throughput	4.4	5.3	mptcp ideal share: 5	2.5	3.4	mptcp minimum ideal share: 2.5	2.5	3.2	mptcp minimum ideal share: 2.5

2) Performance on shared paths

Fig. 4 to 6 show the performance of both DAIMD and LIA on scenarios with background traffic. We chose Reno, the standard TCP implementation, as the background traffic type. A single Reno flow is performing on each subpath.

Results show that DAIMD behaves similarly to LIA. We consider the design congestion control goals of Multipath TCP achieved on the simulations, but not strictly regarding the first goal. As can be seen in the plots and in Table II, DAIMD takes nearly the same bandwidth as a single TCP flow would get instead on the best path. The ideal fair share, according to the Resource Pooling Principle, and the design fairness aims of Multipath TCP is exactly what a single TCP flow would get. On the scenario with paths with different capacities DAIMD gets a lower throughput than the expected fair share (5Mbps). However, DAIMD gets a high, acceptable utilization of this corresponding ideal share. We consider that even not fulfilling the first design goal strictly the result is sufficiently good to consider the MPTCP throughput goal acceptably achieved. In the other simulations, DAIMD throughput achieves exactly this ideal fair share unlike LIA under the same path conditions. LIA accomplishes the first goal, but exceeding the ideal fair share.

The second design goal, not to get more capacity than a single TCP flow on a shared path is also fulfilled, as can be seen on Table II. In plots 4.a, 4.c, 4.d, 5.c, 5.d, and 6.a it can be seen that both DAIMD and LIA get more throughput than the Reno flow sharing the same subpath for an instant in few, exceptional occasions, which affects little to the average throughput of the flows. The average throughput results prove that this MPTCP goal is achieved, despite these momentary situations. The third design goal is the appropriate load balance among subflows. As well as LIA, DAIMD uses more the best paths, and less the worst, most congested paths. But DAIMD in these simulations takes off comparatively more traffic from the most congested paths than LIA. The utilization of the worst path, regarding the overall use of paths, is proportionally lower in DAIMD than in LIA. This fact suggests that due to its delay-based sensitiveness to congestion DAIMD may be able to move off more traffic than LIA from congested links. In that case DAIMD would accomplish this goal more effectively than LIA, and would get closer to the Resource Pooling Principle aims. MPTCP contributes to a fairer and more efficient distribution of resources among flows in the network (including background traffic flows), as an effect of the alleviation of congestion on the most congested links by moving off traffic, and a more efficient distribution of its flows, which is not only beneficial for the own MPTCP connection performance, also indirectly for the flows sharing links with MPTCP subflows. As a secondary effect flows sharing links with multipath TCP tend to equalize, up to some extent, their share. The use of our hybrid congestion control on MPTCP may strengthen this effect, as it improves the load balance.

DAIMD presents some loss episodes on these simulations. The less responsible use of the queues of loss-based flows forces periodical queue overload, packet loss, and network congestion. In that type of congestion situation loss is more difficult to avoid for the delay-based congestion control. DAIMD is able to prevent loss in most occasions by anticipating the queue overload and triggering multiplicative decrease. This spares the cost of loss and further retransmissions to the connection performance. Compared to loss-based congestion control DAIMD presents a more responsible use of the queues, preventing overload and loss, and consequently inducing less queuing delay.

V. CONCLUSIONS AND OUTLOOK

In this paper, we proposed a novel, hybrid congestion control approach for Multipath TCP, based on delay-based AIMD with a different mechanism for detecting incipient congestion using delay analysis.

Following an AIMD scheme, the increase scheme is based on LIA, the first loss-based congestion algorithm for Multipath TCP proposed by the IETF, as well as the coupling, load balance and general functionality. Multiplicative decrease follows the delay-based AIMD general idea, but with a different criterion for triggering the decrease. The backoff factor is adjusted to compensate the effect of early congestion detection, compared with the late loss-based action.

We developed this new algorithm, implemented it on a real system, Linux, and tested it with NS-3 simulator using real network stack code to evaluate its performance on different network scenarios.

In the standard case studies tested with simulations DAIMD proves to behave appropriately in terms of loss avoidance, incipient congestion detection, fairness, adequate performance on idle links as well as interoperating with loss-based flows in shared links, Multipath TCP congestion control goals including resource pooling, and responsive use of queues.

DAIMD is able to achieve the expected benefits of a delay-based scheme like loss avoidance, intra-protocol fairness, proactive action against congestion and low delay induction. Additionally DAIMD possesses the capability of interoperating in fair conditions with loss-based flows, which is impossible for classic delay-based congestion control. Only hybrid congestion control such as Compound and delay-based AIMD are able to achieve this desirable characteristic while applying a responsive, preventive action against congestion by using congestion delay-based detection. DAIMD is the first proposed hybrid congestion control for multipath TCP. Its proactive action against congestion prevents up to some extent queue overload, and therefore presents a more responsive use of queues, inducing less delay, and spares the important cost in performance of loss to the extent that it is possible. Simulations have shown an important loss reduction, and even complete loss avoidance in the absence of loss-based background traffic.

DAIMD behaves satisfactorily as a functional MPTCP congestion algorithm, as it fulfills the MPTCP performance and fairness goals. The first design goal demands that a whole MPTCP connection should take at least the same bandwidth as a TCP connection would take performing over the best path, an aim that DAIMD achieves in general terms. In some simulations DAIMD is closer than LIA to the ideal fair share, ***exactly*** the same capacity a TCP flow would get on the best path. The second goal refers to fairness to TCP on each individual path: a MPTCP subflow must not take more available bandwidth than any TCP flow sharing the same path, which is a condition that DAIMD fulfills in all simulations. On idle links it is instead convenient to fulfill the whole available capacity, which is something DAIMD has been able to do in these special cases. The third design goal states that the load should be appropriately balanced among the available paths. Results suggest that comparatively DAIMD balances traffic even better as LIA, as it moves more traffic off from the most congested path and utilizes proportionally more the best path. This can be seen in Table II.

Results suggest that DAIMD may improve on LIA fairness to TCP in some respects, as in some cases DAIMD throughput is closer to the ideal fair share than LIA rate and DAIMD makes a fairer load distribution than LIA in the case studies.

In future work, in order to more thoroughly evaluate our own algorithm, we will extend the simulation scenarios, and make a more thorough study of its behaviour regarding queues. The DAIMD mechanism for preventing congestion can be also applied in other MPTCP algorithms, and we will implement this mechanism in OLIA and BALIA. Also, we will use DAIMD in several potential applications of MPTCP not yet fully explored and test the benefits of the use of MPTCP in combination with this delay-based hybrid approach that minimizes loss and reduces delay. We will investigate its application in special case studies of Internet of Things, wireless mobile phone networks, and military communications.

Acknowledgments

This research has received funding from the European Union's Horizon 2020 research and innovation programme as part of the ‘Interoperability of Heterogeneous IoT Platforms’ (INTER-IoT) project under Grant Agreement n˚ 687283.

References

[1] A. Ford, C. Raiciu, M. Handley and O. Bonaventure, “TCP extensions for multipath operation with multiple addresses” *RFC 6824, IETF,* 2013

[2] F. Kelly and T. Voice, “Stability of end-to-end algorithms for joint routing and rate control” in *Computer Communication Review* 35, pp. 5–12, 2005

[3] P. Key, L. Massouli ́L . and Towsley, D., “Combining multipath routing and congestion control for robustness” in *40th IEEE Conference on Information Sciences and Systems*, CISS, 2006

[4] H. Han, S. Shakkottai, C.V. Hollot, R. Srikant and D. Towsley, “Multi-path TCP: a joint congestion control and routing scheme to exploit path diversity in the internet” in *IEEE/ACM Trans. Netw.* 14(6), pp. 1260–1271, 2006

[5] C. Raiciu, M. Handly and D. Wischik, “Coupled congestion control for multipath transport protocols” *RFC 6356, IETF,* 2011

[6] D. Wischik, C. Raiciu, A. Greenhalgh and M. Handley, “Design, implementation and evaluation of congestion control for multipath TCP” in *Proceedings of USENIX Conference on Networked Systems Design and Implementation*, NSDI, 2011

[7] M. Allman, V. Paxson and E. Blanton, “TCP congestion control. RFC 5681”, *IETF,* 2009

[8] D. Wischik, M. Handley, and M. B. Braun, "The resource pooling principle" in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 47-52, 2008

[9] M. Xu, Y. Cao and E. Dong, “Delay-based congestion control for multipath TCP”, draft-xu-mptcp-congestion-control-02, Internet Draft *IETF,* July 2015

[10] Y. Cao, M. Xu and X. Fu, “Delay-based congestion control for multipath TCP” in *20th IEEE International Conference on Network Protocols (ICNP),* pp. 1-10, October 2012

[11] S. H. Low, L. Peterson, and L. Wang, "Understanding TCP Vegas: a duality model," in *Proc. of ACM SIGMETRICS,* pp. 226-235, 2001

[12] D. Leith, R. Shorten, G. McCullagh, J. HeFfner, L. Dunn, and F. Baker, "Delay-based AIMD congestion control," in *Proc. of PFLDNeT Workshop,* 2007

[13] R. Gonzalez and M. Kuehlewind, “Implementation and evaluation of coupled congestion control for multipath TCP”, *18th EUNICE Conference in Information and Communication Technologies,* August 2012