# Improving TCP-friendliness and Fairness for mHIP

Tatiana Polishchuk and Andrei Gurtov

*Abstract*—**Multihomed environments are getting increasingly common, especially for mobile users. mHIP was designed to provide secure multipath data transmission for the multihomed hosts and boost throughput of a single TCP connection by effectively distributing data over multiple available paths.**

**In this paper we develop a TCP-friendly congestion control scheme for mHIP secure multipath scheduling solution. We enable two-level control over aggressiveness of the multipath flows to prevent stealing bandwidth from the traditional transport connections in the shared bottleneck. We demonstrate how to achieve a desired level of friendliness at the expense of inessential performance degradation. A series of simulations verifies that mHIP meets the criteria of TCP-compatibility, TCP-equivalence and TCP-equal share, preserving friendliness to UDP and another mHIP traffic. Additionally we show that the proposed congestion control scheme improves TCP-fairness of mHIP.**

**Keywords:** Internet, HIP, multipath routing, TCP-friendliness, goodput

## I. Introduction

Multipath data transfer is a promising technique for enhancing reliability of Internet connections. New mobile devices and laptops are equipped with several network interfaces (e.g., WLAN, GPRS, 3G) and have multiple links to the Internet, which results in availability of multiple paths between a source and destination end host.

TCP [24] comprises a major share of the total Internet traffic. Among its other management tasks, TCP controls segment size, the rate at which data is exchanged, and network traffic congestion [25]. However, traditional TCP flow is constrained to use one path only per one connection between two communicating hosts. There are efforts within the networking community to overcome this limitation. Most of these efforts rely on the mechanisms which aggressively compete for network resources. Naive designs and implementations risk substantial unfairness to well-behaved TCP flows. Proper per-flow congestion control is required to limit aggressiveness of the proposed multipath solutions.

Other multipath communication methods, proposed to efficiently utilize multiple access links, unable to take advantage of all available multipath bandwidth because they do not properly consider end-to-end delay of packet transmission. Out-of-order data arrivals at a receiver cause unpredictable underutilization of spare network capacity. Packet reordering and non-congestion packet loss can significantly degrade TCP performance.

Manuscript received January 26, 2011

T. Polishchuk is with Helsinki Institute for Information Technology HIIT, Aalto University, Finland email:`firstname.lastname@hiit.fi`

A. Gurtov is with Helsinki Institute for Information Technology HIIT, Aalto University, and the Centre for Wireless Communications, University of Oulu, Finland email:`lastname@hiit.fi`

TCP-friendliness has emerged as a measure of correctness in Internet congestion control. The notion of TCP-friendliness was introduced to restrict non-TCP flows from exceeding the bandwidth of a conforming TCP running under comparable conditions. Protocols commonly meet this requirement by using some form of AIMD (Additive Increase Multiplicative Decrease) congestion window management, or by computing a transmission rate based on equations derived from AIMD model.

In the prior work [8] we proposed a multipath solution on HIP layer [7]. Multipath HIP (mHIP) combines the advantages of HIP advanced security with the benefits of multipath routing such as better resource utilization, increased throughput and fault tolerance. HIP multihoming extension [26] supports multiaddressing in a functional layer between IP and transport and provides HIP hosts with the ability to use multiple access networks simultaneously. Simultaneous Multiaccess (SIMA) [21] utilizes multihoming for assigning separate transport connections independently to different paths. We take alternative approach by using multiple parallel paths simultaneously inside one transport connection. mHIP multipath scheduler effectively distributes incoming data over available paths on the per-packet basis, taking into account their rapidly changing parameters. mHIP is a generic multipath solution. It was designed to schedule not only the most common TCP traffic, but also data from different transport protocols, which are not necessarily TCP-friendly, e.g. UDP, SCTP, DCCP. Simple congestion control measures were suggested to provide reliable multipath data delivery.

In this paper we study TCP-friendliness of multipath HIP design with respect to coexisting connections. The contributions of this work include the development of a two-level congestion control concept for a reliable multipath data transmission and methods of tuning aggressiveness of individual flows from the multipath bundle in order to provide a desirable level of TCP-friendliness while avoiding significant performance degradation. The proposed congestion control scheme also improves TCP-fairness for mHIP, which allows to relax the original assumption that the chosen paths should necessarily be bottleneck-disjoint.

The rest of the paper is organized as follows. Section II summarizes the related work. Preliminaries are presented in Section III and contain the review of multipath HIP simple congestion control and definitions of TCP-friendliness. Section IV presents the step-by-step work which was done to enable TCP-friendly congestion control for mHIP. We verify the correctness of the proposed congestion control scheme in Section V. Conclusions and future work are given in Section VI.

## II. Related Work

Despite the fact that multiple multipath solutions for multihomed hosts has recently emerged, multipath routing is not yet widely deployed in practice. Researchers study advantages of its implementation on different layers of the TCP/IP stack.

Transport layer solutions, such as SCTP [15], MPTCP [6], TCP-MH [18], can naturally obtain the most recent information on the quality of different paths and detect congestion situations in timely manner. For example, SCTP can perform measurements across several paths simultaneously, and then map flows on one or another path. However implementing multiaddressing and multipath functionality on transport layer involves significant re-factoring of the code, in particular separating connection and path-specific components so that functions such as congestion control could be implemented per each path. Transport layer multipath solutions are not easy to deploy also because they involve inevitable changes to the corresponding applications.

Network layer approaches [2], [5] are generally easy to deploy and involve only minimal changes, contrary to the application and transport layer solutions. They are transparent to the applications, but do not support proper per-flow congestion control, which is needed to provide the required level or TCP-friendliness to the external connections.

Wedge-layer multipath solutions implemented in mHIP [8], LIN6 [11], MIP6 [16] have an advantage of being able to maintain multiaddressing information across transport associations. The transport activity between two endpoints may well be able to use multiaddressing immediately and with no further administrative overhead. Edge-based locator exchange protocols can be incorporated without necessitating modification to any host's IP or transport modules, which makes them the best choice to provide generic multipath functionality for legacy Internet applications and transport protocols.

mHIP naturally solves the tasks which are challenging for any multipath design. These include providing end-to-end security for each individual multipath flows, facilitating the ability to traverse NATs and middleboxes, as well as mobility support, which are inherited from the standard HIP protocol implementation. mHIP identifies individual multipath flows with SPI (Security Parameters Index) specific for each path, which guarantees the proper packet to path assignment, contributes to the flow congestion control and helps to prevent output data sequence from reordering. As a wedge-layer solution mHIP design does not require modifications to any transport modules or applications. Legacy IPv4 and IPv6 applications unaware of multiple paths can benefit from it transparently.

There is an effort in the community to create new methods which effectively and TCP-friendly utilize a spare network capacity. In [9] authors created a parallel multipath TCP solution, which controls data transmission over coordinated multiple TCP connections. They stressed the importance of TCP-friendliness for multipath schemes and suggested a way to find a balance between effectiveness and fairness. Their work provided a motivation to design a TCP-friendly congestion control over multipath flows inside one TCP connection.

When data packets are sent over several paths inside one connection they can experience different end-to-end delays and arrive out of order. In case of TCP traffic, packet reordering causes significant performance degradation. The authors of [19] surveyed and analyzed relevant techniques on coping with multipath TCP packet reordering. They conclude that there exists no one-fits-all solution to solve the problem of packet reordering for multipath TCP. Basing on the methods [3], [4], [20], [29] we suggest the improvement for multipath HIP which reduced the level of reordering on the receiver and significantly improved TCP-friendliness of our scheme.

According to the resource pooling principle [28] when several subflows of one connection share a bottleneck, their resource consumption adds up. Multipath connections with a large number of TCP-friendly subflows can compete unfairly against a smaller number of regular TCP connections. Each subflow is as aggressive as a single TCP, and a bundle of $n$ TCP-friendly subflows will hence use an approximately $n$ times greater share of the bottleneck resource than they should. *TCP-fair* multipath connection should displace no more TCP traffic than a traditional TCP stream would displace. A number of methods [9], [22], [10] were proposed to study and solve the TCP-fairness problem. The congestion control solution for mHIP, which we present further in this paper, is also designed to meet the TCP-fairness criterion.

## III. Preliminaries

### A. TCP-friendliness Definitions

*TCP-friendliness* is a generic term describing a scheme that aims to use no more bandwidth than TCP uses. In this paper we study mHIP congestion control in view of the criteria proposed in [27]:

A *TCP-compatible* flow, in the steady state, should use no more bandwidth than a TCP flow under comparable conditions, such as packet-loss rate and round-trip time (RTT). However, a TCP-compatible congestion control scheme is not preferred if it always offers far lower throughput than a TCP flow.

A *TCP-equivalent* scheme merely ensures the same throughput as TCP when they experience identical network conditions. Although a TCP-equivalent scheme consumes TCP-equivalent bandwidth when working by itself, it may not coexist well with TCP in the Internet.

*TCP-equal share* is a more realistic but more challenging criterion than TCP-equivalence and states that a flow should have the same throughput as TCP if competing with TCP for the same bottleneck. A TCP-equivalent flow may not be TCP-equal share, but the opposite is always true.

To be able to meet all three criteria a TCP-friendly scheme should use the same bandwidth as TCP in a steady-state region, while being aggressive enough to capture the available bandwidth and being responsive enough to protect itself from congestion, as the packet-loss condition changes in the paths in the transient state. *Aggressiveness* of a scheme describes how the scheme increases the throughput of a flow before encountering the next packet loss, while *responsiveness* describes how the scheme decreases the throughput of a flow when the packet-loss condition becomes severe.

In what follows we will examine the ability of our multi-path solution to adhere to the proposed definitions of TCP-friendliness. To evaluate its performance we intruduce the factor of friendliness metric:

$FF(flow) = \frac{T(flow)}{T(TCP)}$

Here $T(\cdot)$ denotes the average flow throughput in Mbps. $FF = 1$ indicates the solution satisfies the strongest TCP-equal share criterion, while solution resulting in $FF > 1$ is more aggressive than a typical TCP and the one with $FF < 1$ may be not TCP-compatible.

## B. Review of Multipath HIP with Simple Congestion Control

In the prior research [8] HIP multipath scheduling showed a potential to aggregate about 99% of the sum of individual paths bandwidth. Simple congestion detection and avoidance are able to prevent the sending rate of the multipath traffic from significant degradation caused by congestion in the paths. Before we start evaluating mHIP congestion control scheme in the view of TCP-friendliness criteria, we recall how it operates.

1) *Connection establishment*
During the base exchange HIP obtains information about the number of available interfaces on both communicating hosts and the number of available paths with the initial parameters such as available bandwidth and propagation delay.

2) *Updating parameters of the paths*
mHIP uses HIP signaling packets for path probing. The frequency of heartbeats can vary depending on the particular setup.

3) *Sending data*
HIP multipath scheduler optimally splits data among the paths according to their capacities. The details of scheduling algorithm are provided in [8].
mHIP stores packet-to-path assignments at the sender and also in the ESP packet headers, which are used according the HIP standard [13]. SPI number, specified in the packet header corresponds to the path which is assigned to deliver this particular packet.

4) *Congestion control*
Marking and multipath congestion avoidance techniques provide a simple congestion control for mHIP. One packet per round-trip time is marked on the departure to each path. The expected delivery time of the marked packet is stored at the sender and then compared to its actual arrival time value on the receipt of the corresponding ACK. If the estimated delivery time and the actual arrival time of the marked packet are noticeably different, the scheduler considers the path to be congested.
Multipath congestion avoidance technique specifies two indicators of the path congestion:
   - Case 1: standard TCP *dupack action*, when the sender is retransmitting the packet after the receipt of three duplicate acknowledgments from the receiver;
   - Case 2: observed delivery time of the marked packet exceeds its corresponding expected delivery time by more than some preset value.

   If any of the two indicators suggest congestion, the path is temporarily closed and the packets are redirected to the other available paths. mHIP sends regular probes to the congested path to detect when the path becomes again free for reliable data transmission.

5) *Assumptions and limitations*
Our approach corresponds to the class of disjoint multipath routing [23]. The paths are restricted to have independent bottlenecks. The scheduler resides at the sender side, no information from the receiver is available other than TCP acknowledgments (ACKs) received by the sender. At least one available path should not be congested at any given point of time.

## IV. Improving mHIP Step by Step

Next we examine mHIP congestion control in the view of TCP-friendliness criteria. We analyze the reasons why multi-path flows not always fairly share available bandwidth with TCP and propose the methods to improve TCP-friendliness of our multipath solution.

## A. Experimental Evaluation of mHIP with Simple Congestion Control

All simulations presented in this work were run using ns-2 network simulator [1]. A new protocol agent was implemented on the basis of TCP New Reno to deal with the multipath flow controlled by HIP. Existing TCP and UDP modules were also used to simulate external cross-traffic competing with HIP multipath flows for bottleneck bandwidth.

Consider a simulation model shown in Figure 1. A TCP traffic flow, controlled by multipath HIP, is sent from $n0$ to $n1$ over two available paths: $Path1 = n0 - n2 - n1$ and $Path2 = n0 - n3 - n1$ with the bandwidth of 8Mbps and 4Mbps respectively. Since multipath scheduler is distributing the traffic according to bandwidth-delay product of the paths, for simplicity the propagation delay is fixed to be the same for all the links and equals 30 ms. mHIP is calculating the end-to-end propagation delays in the paths, they can consist of any number of connected links and intermediate nodes. Node $n4$ is used for the path $n2 - n1 - n4$ construction, which accommodates a standard TCP New Reno flow, competing against one flow from the mHIP bundle for the bottleneck

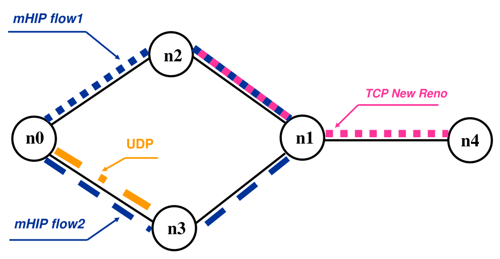

Fig. 1. 2-path simulation model.

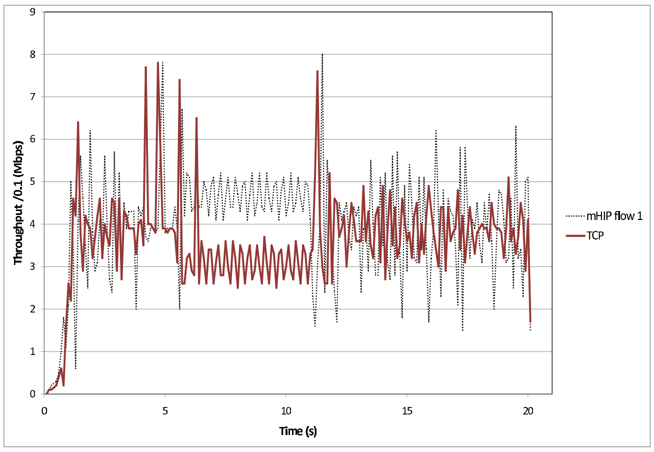

Fig. 2. mHIP flow competes with TCP New Reno flow in $Path1$.

link $n2 - n1$. Drop-Tail scenario was used to manage the bottleneck link, its size is 1.5 times the bandwidth-delay product of the link. The packet size in each flow is 1250 bytes. The simulation runs for 20 seconds, which we believe is sufficient to reflect the difference between the proposed congestion control solutions. Appropriate $Rwin$ values were used at the receivers to allow maximum throughput of the flows.

We begin our first experiment with an empty network and then allocate multipath HIP subflows to the two end-to-end paths. At the same time we start sending a TCP traffic from $n2$ to $n4$, which will compete with mHIP flow in the bottleneck link $n2-n1$. To simulate variable network conditions we also introduce cross-traffic to $Path2$. A 4Mbps UDP flow was scheduled between 5 and 11 seconds of the simulation run, triggering a congestion situation in $Path2$.

Figure 2 shows mHIP and TCP New Reno flow throughputs, averaged over 0.1 sec. As one can clearly conclude from the chart the flows do not share the bottleneck bandwidth fairly. mHIP (dotted curve) occupies more bandwidth, with the average of $T(mHIP1) = 3.98$Mbps and TCP takes just $T(TCP) = 3.56$Mbps resulting in the friendliness factor $FF = \frac{T(mHIP1)}{T(TCP)} = 1.11$.

Lets try to understand the reason why mHIP starts starving the TCP flow during the particular time period. In the beginning of the simulation run mHIP and TCP flows share the bandwidth mostly fair. At some point after 5 seconds the marking technique reports a congestion situation, resulting from the competition with UDP cross-traffic in $Path2$. Let $w$ be the number of packets at the sender, which corresponds to the $cwnd$ value of the global TCP flow controlled by mHIP. The multipath scheduler sends $w1$ packets to $Path1$ and $w2$ packets to $Path2$ in the share correspondent to path characteristics with the total $w1 + w2 = w$. According to the congestion avoidance scheme $Path2$ is closed and all the traffic from the congested $Path2$ is rerouted to $Path1$, meaning that at this same time $Path1$ receives not only its own share $w1$ but also extra $w2$ packets. In this region mHIP is dominating and stealing bandwidth from the competing TCP transport transmission in the bottleneck link $n2 - n1$.

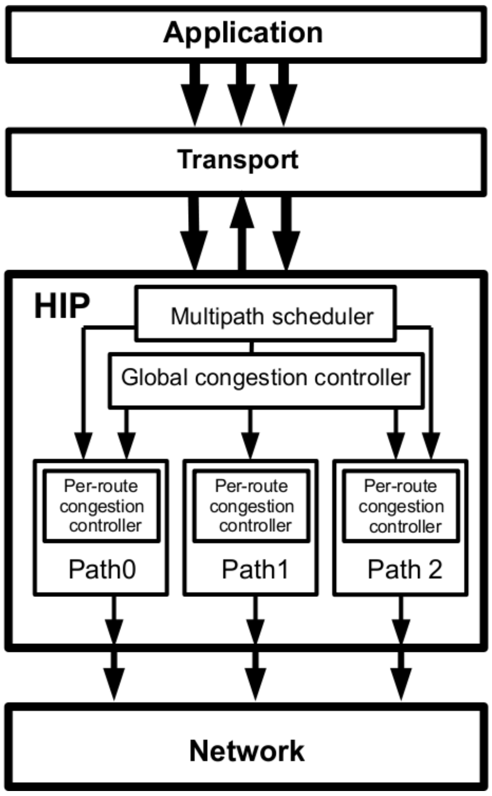
Application
Transport
HIP
Multipath scheduler
Global congestion controller
Per-route congestion controller
Path0
Per-route congestion controller
Path1
Per-route congestion controller
Path 2
Network

Fig. 3. Two-level multipath congestion control.

The proposed congestion control method is definitely more aggressive than AIMD policy of a typical TCP.

## B. Designing TCP-friendly Congestion Control for mHIP

We want our mHIP connections to coexist with other traffic providing opportunities for all to progress satisfactory. To limit aggressiveness of the flow growth we propose the following two-level congestion control scheme - per-path AIMD plus TCP global stream congestion control on top of it, and introduce a sender-side buffer to provide better control on the packet sequence in congestion situations.

The proposed twofold congestion control scheme is illustrated in Figure 3. Global congestion controller coordinates the work of the individual per-path controllers and balances traffic load between the paths according to their available capacity. If $cwnd$ capacity of the quickest path is exceeded, the path with the next minimum estimated arrival time is chosen.

An important property of the proposed scheme is that per-path controllers are connected so that the aggregated congestion window is a simple sum of per-flow congestion windows. Same rule applies to the threshold values. Connecting per-path congestion control parameters in such a way we guarantee the resulting multipath bundle behaves as a single TCP if all are sent to the same path.

Below we summarize the proposed updates to the mHIP multipath scheduling design presented in subsection III-B. Parts 1,2 and 5 (connection establishment, path parameters updates and assumptions) remain unchanged, while there are some additions to the rest:

3) *Sending data*
   After per-path congestion control limitations were introduced the scheduler takes in consideration the current sizes of per-path congestion windows. If $cwnd$ capacity of the best path is exceeded, the path with the next minimum estimated arrival time is chosen. If there is no available capacity in any of the paths, the packet is placed to the sender-side buffer until new ACK arrives.

4) *Congestion control*
   Marking is now removed from the congestion control scheme. Multipath congestion avoidance retains only one congestion indication, the standard TCP *dupack* event. Upon receipt of a preset number of $dupacks$ (3 for standard TCP) the scheduler determines from which path the packet is missing and halves $cwnd$ and $ssthresh$ values of the corresponding path. This action reduces data intake in the congested path and automatically redirects traffic to the other paths which have available capacity. If there is no capacity in the paths, extra data goes to the sender-side buffer. Maximum capacity of the buffer is set to TCP receiver window size $Rwin$, making it capable to occupy the maximum flight-size number of packets in case of severe congestion situations.

It should be noted that the congestion control parameters of the global TCP flow differs from the standard TCP New Reno only in the way how the congestion control window grows and decreases (AIMD parameters): the increase of the global $cwnd$ is now dictated by the cumulative increase of the per-flow congestion windows and the reaction on losses (*dupack action*) has changed so that the global $cwnd$ is not divided by half, but only the window corresponding to the path from which the packed was lost is decreasing.

## C. Experimental Evaluation of mHIP with the Updated Congestion Control

To validate correctness of the proposed congestion control scheme we repeat the experiments with the simulation scenario described in Section IV-A. Again, one of the multipath HIP flows sent to $Path1$ meets with the external TCP flow in the bottleneck link $n2 - n1$, while the other flow sent to $Path2$ is interrupted by UDP cross-traffic in the link $n0 - n3$.

The resulting throughputs of the two flows competing in $Path1$ are shown in Figure 4. mHIP average flow throughput is $T(mHIP) = 3.56$Mbps and TCP takes about $T(TCP) = 3.98$Mbps resulting in the fairness factor $FF = \frac{T(mHIP)}{T(TCP)} = 0.89$.

Here we observe the opposite extreme: mHIP flow behaves too leniently and is not able to occupy available bandwidth effectively. In the following section we analyze the problem and propose a method to solve it.

## D. Balancing between Aggressiveness and Responsiveness

Competition with the external traffic naturally influences effectiveness of multipath scheduling. Mistakes in the expected delivery time estimations result in the output sequence reordering at the receiver. TCP sender receives multiple $dupacks$ in

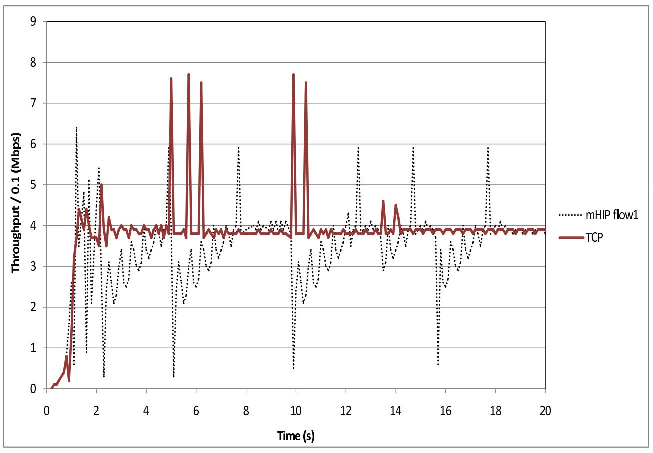

Fig. 4. mHIP flow controlled by the proposed twofold multipath congestion control is suppressed by TCP.

response to reordering, which mHIP scheduler treats as an indication of congestion. In response to the congestion mHIP scheduler halves congestion window of the corresponding path, reducing aggressiveness of the traffic flow. This precaution could be too strict in case when the missing sequence numbers are not lost but just slightly delayed in competition with the external flows.

To differentiate between the reordering signals and actual losses we propose the following modifications to mHIP congestion control scheme. First, we increase $dupthresh$ value defining the number or $dupacks$ which serve as an indication of congestion. This method is proposed in the related work [4], [29] as a cure from the mild packet reordering. Compared with the default $dupthresh$ of three, the proposed techniques improves connection throughput by reducing the number of unnecessary retransmissions. But one should adjust $dupthresh$ value carefully since making it too large slows down the reaction of the system to the actual losses and can significantly degrade the overall performance in the networks with high loss rates.

Additionally we introduce a new time variable ADDR (allowable delay due to reordering), which keeps how much time has elapsed since the congestion situation in some path was reported. If the missing sequence number has arrived successfully during this allowable time period and the corresponding ACK arrives to the sender, $cwnd$ and $ssthresh$ of the path should be returned to the values prior to the congestion notification. ADDR is chosen to be less than the shortest RTT among the paths used to deliver multipath flow. It will assure accurate differentiation between the packets delayed due to reordering and their duplicates retransmitted after the loss was reported.

## E. Controlling Friendliness with a Receiving Buffer

Another way to control aggressiveness of the multipath flow is to locate a sufficiently large buffer at the receiver and use SACK [12] together with SMART option [17]. Standard TCP with cumulative acknowledgement scheme often does not provide the sender with sufficient information to recover

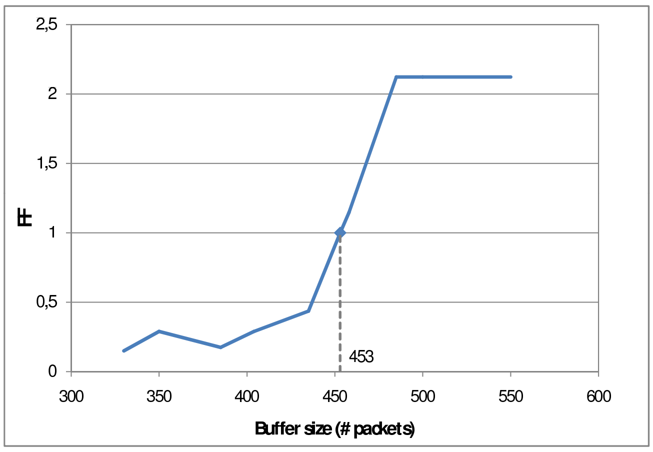

Fig. 5. Factor or friendliness depends on the size of receiving buffer.

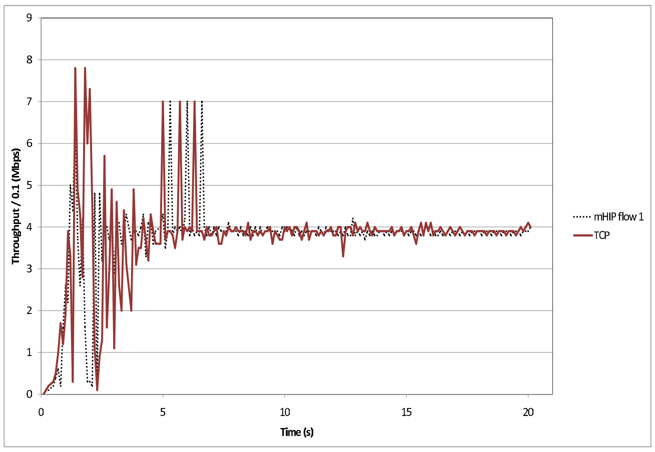

Fig. 6. mHIP flow 1 friendly coexists with TCP New Reno flow.

quickly from packet losses. When TCP with SACK option is used, each acknowledgement contains the information about up to three non-contiguous blocks of data that have been received successfully. SMART variation of the SACK provides each acknowledgement with the sequence number of the last received packet that caused the receiver to generate the acknowledgement.

We include SACK with SMART option into our multipath congestion control scheme as follows. When out-of-order packet arrives at the receiver, it is stored in the buffer and the receiver sends a cumulative acknowledgement with SACK containing the information on the sequence number of that packet. On the receipt of a duplicate acknowledgement the sender determines from which particular path the next expected packet is missing and counts the number of such events per path. After receipt of the three consecutive acknowledgements about the missing packet from the same path, the $dupack\ action$ is invoked and the missing packet is retransmitted in the standard New Reno way. The difference from the previous multipath HIP proposal is that the sender now stores the information about the packets, retrieved from the SACK, and does not retransmit the reordered packets, which are buffered at the receiver.

The buffer stores all the packets received out of order until the holes in the sequence are filled. It will hide multipath reordering from the global TCP and improve performance of the multipath connection. When the buffer capacity is limited, the packets arriving after the buffer is full, are dropped and considered lost. It will cause regular TCP retransmissions and reductions of the global congestion window. As a result, aggressiveness of the multipath flow is limited by the choice of the buffer size.

Figure 5 illustrates how the choice of the receiver buffer size influences the resulting factor of friendliness for some fixed network settings. The developer, seeking to achieve the optimal factor of friendliness $FF = 1$, should set the buffer size to about 453 packets in this particular setup.

The dependency between the buffer size and the friendliness factor could be non-monotonic in some small intervals. The reason is that the factor of friendliness is influenced by the number of reordered packets in the multipath HIP flows. If the capacity of the receiving buffer is limited and some packets are dropped, the recovery time can vary depending on the current congestion conditions. But in the global scope by increasing the size of the buffer, one can provide multipath flow with better conditions in comparison to the regular TCP against which it competes, making it more aggressive. The buffer of maximum capacity will accommodate all the packets reordered in competition against TCP. Hence, further increase of the buffer size will not influence the factor or friendliness any more.

## V. Final Validation

Below we provide the final experimental validation of the effectiveness of our proposed modifications to mHIP congestion control. Again, we repeat the experiment described in Section IV-A with the last version of mHIP with two-level congestion control scheme and all the proposed modifications applied.

### A. TCP-friendliness

Figure 6 illustrates significant improvement in TCP-friendliness of the mHIP flow when it competes against TCP for the bottleneck link bandwidth. Finally both mHIP and TCP flows are able to achieve comparable average throughputs of $T(mHIP1) = 3.80$Mbps and $T(TCP) = 3.71$Mbps with the friendliness factor $FF = \frac{T(mHIP1)}{T(TCP)} = 1.02$. The competition demonstrated high variation about the average during a short stabilization phase. This unfairness is rather moderate and can be tolerated as far as the flows quickly achieve stability and later coexist friendly.

### B. UDP-friendliness

An interesting observation is that the second mHIP flow in $Path2$ behaves also about friendly competing against the UDP cross-traffic which we used to simulate variable network conditions between 5 and 11 seconds. On this interval mHIP achieves the throughput of $T(mHIP2) = 4.20$Mbps. The solid curve in Figure 7 corresponds to the UDP cross-traffic

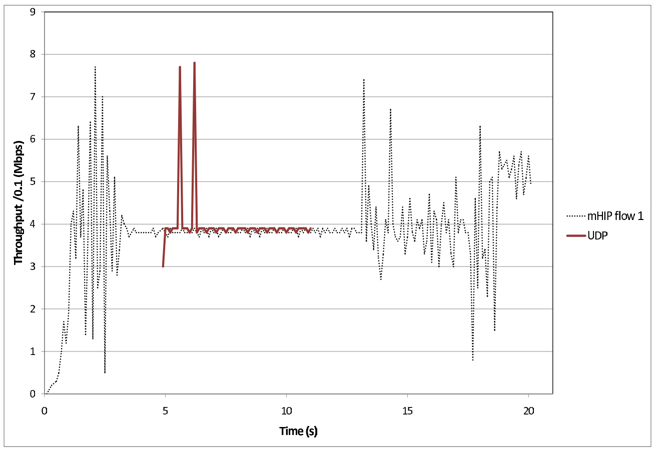

Fig. 7. mHIP flow 2 competes almost friendly with UDP cross-traffic.

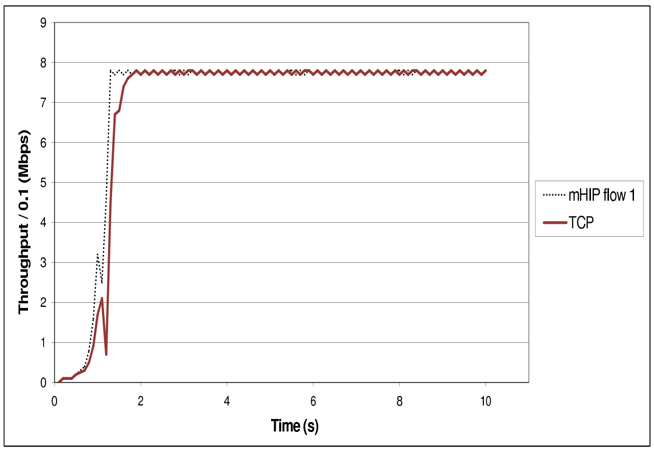

Fig. 8. Testing TCP-compatibility and equivalence of mHIP.

flow with the average flow throughput $T(UDP) = 3.98$Mbps. The flows fight during negligible time period and then find stability to share the bottleneck about fairly with a moderate unfairness of $FF = \frac{T(mHIP2)}{T(UDP)} = 1.05$.

## C. TCP-compatibility and TCP-equivalence

According to the definitions TCP-compatible flow, in the steady state, should use no more bandwidth than a TCP flow under comparable conditions, while TCP-equivalent scheme ensures the same throughput as TCP when they experience identical network conditions. We send mHIP to the empty 2-path network with no cross-traffic to determine how effectively the protocol is able to use a spare network capacity in the steady state.

Figure 8 shows mHIP flow occupies no more available bandwidth than a TCP flow sent to the same path making it TCP-compatible. Moreover, mHIP achieves the same average flow throughput of 7.8Mbps as TCP in the steady state and thus meets the criteria of TCP-equivalence.

## D. TCP-fairness in the Shared Bottlenecks

A flow is *TCP-fair* if its arrival rate does not exceed the rate of a conformant TCP connection in the same circumstances.

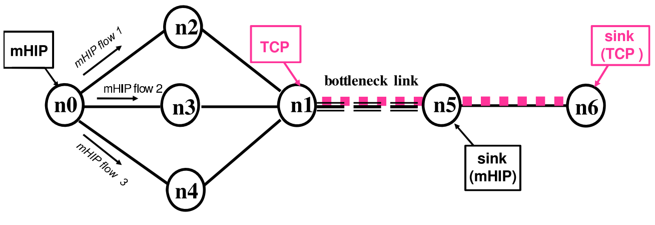

Fig. 9. Simulation model for testing TCP-fairness of mHIP.

Put another way, a TCP-fair flow sharing a bottleneck link with $N$ other flows should receive less than or equal to $1/(N+1)$ of bandwidth available.

Previously mHIP multipath scheduling assumed the paths are bottleneck-disjoint. This automatically liberated us from the necessity to prove TCP-fairness of our solution since multiple flows of a single multipath HIP connection never shared the same bottleneck link. Now we would like to relax the original assumption about the bottleneck-disjointness of the paths. The two-level congestion control scheme introduced in this paper provides TCP-fairness of mHIP. We support the last statement with the following experimental results.

We modified the simulation model by adding one more parallel path between the nodes $n0$ and $n1$ to accommodate an additional mHIP flow and added link $n5-n6$ as shown in Fig. 9. TCP traffic flow, controlled by multipath HIP, is now sent from $n0$ to $n5$ over three available paths: $Path1 = n0-n2-n1-n5$, $Path2 = n0-n3-n1-n5$ and $Path3 = n0-n4-n1-n5$. $Path4 = n1-n5-n6$ now accommodates a standard TCP New Reno connection, which meets all the mHIP flows in the bottleneck link $n1-n5$. To provide compatible starting conditions for their competition all four paths are set to have similar end-to-end characteristics (RTT, queue lengths and types).

Multiple experiments with various path characteristics confirmed that mHIP flows inside one TCP connection share available bandwidth mostly fairly and still friendly to the external TCP flow. The observed friendliness factor lies within the interval $[0.95, 1.03]$. A typical example of such a bandwidth distribution is shown in Figure 10. mHIP bundle behaves almost as a standard TCP when all of its flows occasionally meet in one link. This result confirms that after we improved the congestion control scheme and limited the increase of the global TCP congestion window, our mHIP solution also meets the TCP-fairness criterion.

## E. The Cost of Friendliness

We achieved the desired level of TCP-friendliness for our multipath HIP solution and would like to evaluate the cost in terms of performance degradation paid for this improvement.

We calculate the total throughput $TT$ of the traffic flow controlled by multipath HIP. In the experiment where mHIP with simple congestion control policy demonstrated an excessive unfriendliness competing against TCP NewReno, $TT(mHIP) = 6.45$Mbps. After we applied a series of modifications to mHIP congestion control, similar experiment with

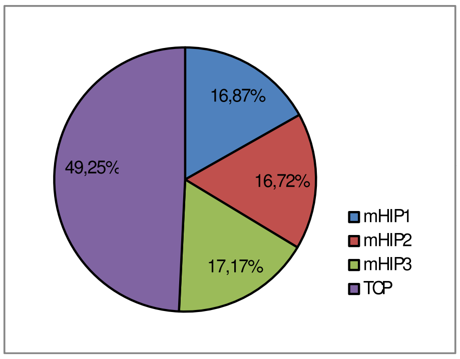

Fig. 10. Three mHIP flows from one connection compete against one TCP NewReno for the bottleneck bandwidth.

the TCP-friendly mHIP resulted in $TT(mHIP) = 5.30$Mbps, which corresponds to $\sim$18% performance reduction. A number of experiments with different network conditions confirmed the desired TCP-friendliness can be achieved at the cost of about 15-20% performance degradation.

## VI. Conclusions and Future Work

We showed a way how to tune aggressiveness of the multipath data transmission controlled by mHIP without loosing its responsiveness in competition with cross-traffic. We designed a twofold congestion control scheme, and adjusted it to meet the TCP-friendliness definitions. Simulation results verify the improved congestion control algorithm meets TCP-compatibility, TCP-equivalence and TCP-equal share criteria under the proposed testing conditions, and allows mHIP to coexist friendly with TCP, UDP and mHIP connections.

The proposed congestion control scheme also assures TCP-fairness of mHIP. Having achieved the fairness of mHIP subflows in sharing the common bottleneck links we now can relax the original assumption that the paths should necessarily be bottleneck-disjoint.

The work could be extended to find a method to dynamically adjust mHIP congestion control variables and enable adaptivity to random congestion scenarios including extreme cases. We will continue examining mHIP friendliness in competition against different transport protocols and compare the results against the alternative multipath proposals.

## Acknowledgments

This work was supported in part by TEKES as part of the Future Internet program of the ICT cluster of the Finnish Strategic Centers for Science, Technology and Innovation.

## References

[1] Network simulator ns-2. http://www.isi.edu/nsnam/ns/, last checked 15/10/2010.
[2] S. Barre and O. Bonaventure. Shim6 implementation report : Linshim6. Internet draft, draft-barre-shim6-impl-03.txt, September 2009.
[3] S. Bhandarkar and A. L. N. Reddy. TCP-DCR: Making TCP robust to non-congestion events. In N. Mitrou, K. P. Kontovasilis, G. N. Rouskas, I. Iliadis, and L. F. Merakos, editors, *NETWORKING*, volume 3042 of *Lecture Notes in Computer Science*, pages 712–724. Springer, 2004.
[4] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *ACM Computer Communication Review*, 32:2002, 2002.
[5] K. Chebrolu, B. Raman, and R. R. Rao. A network layer approach to enable TCP over multiple interfaces. *Wirel. Netw.*, 11(5):637–650, 2005.
[6] A. Ford, C. Raiciu, S. Barre, and J. Iyengar. Architectural guidelines for multipath TCP development. Technical report, June 2010. Internet draft, draft-ietf-mptcp-architecture-01, work in progress.
[7] A. Gurtov. *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley and Sons, 2008.
[8] A. Gurtov and T. Polishchuk. Secure multipath transport for legacy Internet applications. In *Proc. of BROADNETS'09*, Madrid, Spain, Sept. 2009.
[9] T. J. Hacker, B. D. Noble, and B. D. Athey. Improving throughput and maintaining fairness using parallel TCP. In *IEEE InfoCom*, 2004.
[10] T. Ishida, K. Ueda, and T. Yakoh. Fairness and utilization in multipath network flow optimization. In *Proc. of 2006 IEEE International Conference on Industrial Informatics*, pages 1096–1101, 2006.
[11] M. Ishiyama, M. Kunishi, and F. Teraoka. An analysis of mobility handling in LIN6. In *Proc. of International Symposium on Wireless Personal Multimedia Communications (WPMC'01)*, Aug. 2001.
[12] V. Jacobson, and R. T. Braden. TCP extensions for long-delay paths. RFC 1072, IETF, Oct. 1988.
[13] P. Jokela, R. Moskowitz, and P. Nikander. Using the Encapsulating Security Payload (ESP) transport format with the Host Identity Protocol (HIP). RFC 5202, IETF, Mar. 2008.
[14] P. Nikander, T. Henderson, C. Vogt, and J. Arkko. End-host mobility and multihoming with the Host Identity Protocol (HIP). RFC 5206, IETF, Apr. 2008.
[15] A. Jungmaier, E. Rescorla, and M. Tuexen. Transport layer security over Stream Control Transmission Protocol. RFC 3436, IETF, Dec. 2002.
[16] J. Kempf, J. Arkko, and P. Nikander. Mobile IPv6 security. *Wirel. Pers. Commun.*, 29(3-4):389–414, 2004.
[17] S. Keshav, and S. P. Morgan. SMART Retransmission: Performance with Overload and Random Losses. In *Proc. of INFOCOM'97, Annual Joint Conference of the IEEE Computer and Communications Societies*, page 1131, 1997.
[18] K.-H. Kim and K. G. Shin. Improving TCP performance over wireless networks with collaborative multi-homed mobile hosts. In *Proc. of the 3rd Int. conf. on Mobile systems, applications, and services (MobiSys'05)*, pages 107–120, June 2005.
[19] K.-C. Leung, V. O. Li, and D. Yang. An overview of packet reordering in Transmission Control Protocol (tcp): Problems, solutions, and challenges. *IEEE Transactions on Parallel and Distributed Systems*, 18:522–535, 2007.
[20] R. Ludwig and R. H. Katz. The Eifel algorithm: making TCP robust against spurious retransmissions. *SIGCOMM Comput. Commun. Rev.*, 30(1):30–36, 2000.
[21] S. Pierrel, P. Jokela, and J. M. Melen. Simultaneous Multi-Access extension to the Host Identity Protocol. IETF Internet-draft: draft-pierrel-hip-sima-00, June 2006.
[22] C. Raiciu. Coupled multipath-aware congestion control, March 2010. Work in progress.
[23] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. Disjoint multipath routing using colored trees. *Comput. Netw.*, 51(8):2163–2180, 2007.
[24] W. R. Stevens. *TCP/IP illustrated (vol. 3): TCP for transactions, HTTP, NNTP, and the Unix domain protocols*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1996.
[25] W. R. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001, IETF, Jan. 1997.
[26] P. Nikander, T. Henderson, C. Vogt, and J. Arkko. End-host mobility and multihoming with the Host Identity Protocol (HIP). RFC 5206, IETF, Apr. 2008.
[27] S.-C. Tsao and N. Chiao. Taxonomy and evaluation of TCP-friendly congestion-control schemes on fairness, aggressiveness, and responsiveness. *Network, IEEE*, 21(6):6–15, Nov. 2007.
[28] D. Wischik, M. Handley, and M. B. Braun. The resource pooling principle. *SIGCOMM Comput. Commun. Rev.*, 38(5):47–52, 2008.
[29] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: A reordering-robust TCP with DSACK. In *Proc. of IEEE ICNP*, pages 95–106, 2003.

**Tatiana Polishchuk** received her M.Sc in Computer Science from Moscow University of Electronic Technologies (1998) and M.Sc in Applied Mathematics and Statistics from State University of New York at Stony Brook(2007). Tatiana is a Ph.D. candidate in Computer Science at Helsinki University and works as a researcher within the Networking Research Group at the Helsinki Institute for Information Technology since August 2008. Her research area includes developing multipath and multicast solutions for Internet protocols.

**Andrei Gurtov** received his M.Sc (2000) and Ph.D. (2004) degrees in Computer Science from the University of Helsinki, Finland. He was appointed a Professor at University of Oulu in the area of Wireless Internet in December 2009. He is also a Principal Scientist (on leave currently) leading the Networking Research group at the Helsinki Institute for Information Technology focusing on the Host Identity Protocol and next generation Internet architecture. He is co-chairing the IRTF research group on HIP and teaches as an adjunct professor at the Aalto University and University of Helsinki. Previously, his research focused on the performance of transport protocols in heterogeneous wireless networks. In 2000-2004, he served as a senior researcher at Sonera Finland contributing to performance optimization of GPRS/UMTS networks, intersystem mobility, and IETF standardization. In 2003, he spent six months as a visiting researcher in the International Computer Science Institute at Berkeley working with Dr.Sally Floyd on simulation models of transport protocols in wireless networks. In 2004, he was a consultant at the Ericsson NomadicLab. Prof. Gurtov is a co-author of over 100 publications including a book, research papers, patents, and IETF RFCs. His articles received more than 1000 citations according to Google Scholar. He is a senior member of IEEE.

CALLS FOR PAPERS

## Call for Papers

Prospective authors are invited to submit original research papers for publication in the upcoming issues of our Infocommunications Journal.

Topics of interests include the following areas:

- Data and network security
- Digital broadcasting
- Infocommunication services
- Internet technologies and applications
- Media informatics
- Multimedia systems
- Optical communications
- Society-related issues
- Space communications
- Telecommunication software
- Telecom. economy and regulation
- Testbeds and research infrastructures
- Wireless and mobile communications

Theoretical and experimentation research results achieved within the framework of European ICT projects are particularly welcome. From time to time we publish special issues and feature topics so please follow the announcements. Propo-sals for new special issues and feature topics are welcome.

Our journal is currently published quarterly and the editors try to keep the review and decision process as short as possible to ensure a timely publication of the paper, if accepted. As for manuscript preparation and submission, please follow the guidelines published on our website
http://www.infocommunications.hu/for_our_authors.

Authors are requested to send their manuscripts via electronic mail (preferably) or on a CD by regular mail to the Editor-in-Chief:

*Csaba A. Szabo*
*Dept. of Telecommunications,*
*Budapest University of Technology and Economics*
*2 Magyar Tudosok krt. Budapest 1117 Hungary*
*E-mail: szabo@hit.bme.hu*

## Call for Proposals of Special Issues

Infocommunications Journal welcomes proposals for Special Issues – collections of papers dedicated to a particular topic of interest for the readers of the journal.

A Special Issue can be based on a recent high quality workshop or conference or papers can be collected from open call. Invited papers can be part of the special issue as well.

A Special Issue can fill in a whole issue, in which case the number of papers is expected to be 8 to 10, or it can be a Mini – Special Issue. In the latter case, at least 3, preferably 4 papers are required.

Proposals for special issues should include:

– contact information
(name, e-mail, title, affiliation and address);
– resume(s) of the proposer(s), with a representative list of recent publications and related experience (Editorial Board memberships, Guest Editorships, or roles in relevant conferences' program committees);
– the proposed title for the special issue;
– the way the special issue will be compiled (contributions solicited from a technical event, or to be collected from call for this special issue;
– intent to include invited papers should be also indicated, if possible with the names of professionals who are planned to be invited;
– scope and motivation and description of the special issue;
– guest editors (if different from the proposers) with detailed contact information and resumes.

Proposals should be sent to the Editor-in-Chief:

*Csaba A. Szabo*
*Dept. of Telecommunications,*
*Budapest University of Technology and Economics*
*2 Magyar Tudosok krt. Budapest 1117 Hungary*
*E-mail: szabo@hit.bme.hu*