Secure Multipath Transport For Legacy Internet Applications

Andrei Gurtov, Tatiana Polishchuk
Helsinki Institute for Information Technology HIIT
Helsinki University of Technology TKK

***Abstract*—Multi-interface mobile devices and multi-homed residential Internet connections are becoming commonplace. However, standard transport protocols TCP and SCTP are unable to take advantage of several available paths so that the application using a single transport connection would receive the aggregate bandwidth of all paths. Multihoming and advanced security features make the Host Identity Protocol a good candidate to provide multipath data delivery. In this paper, we design and implement a multipath scheduler that distributes the incoming traffic among multiple available paths. Using Fastest Path First scheduling, packets from a single TCP connection could be spread to multiple paths with no reordering. Our simulations confirm effectiveness and TCP-friendliness of multipath transfer for a range of path bandwidths and in the presence of cross-traffic.**[1]

Keywords: Internet, HIP, multipath routing, goodput, cross-traffic

I. Introduction

Multipath routing is an active area of research. Despite the fact that several techniques of utilizing path diversity have been proposed, multipath routing is not yet widely deployed in practice. Researchers study advantages of its implementation on different layers of the TCP/IP stack. In this paper we propose a design of a multipath scheduler on HIP layer. HIP multihoming [18] provides multiaddressing support in a functional layer between IP and transport. Taking this approach we combine the advantages of HIP advanced security with the obvious benefits of multipath routing such as better resource utilization, increased throughput and fault tolerance.

When data packets are sent over several paths inside one connection they can experience different propagation delays and arrive out of order. In case of TCP traffic, receiver sends duplicate acknowledgments to the sender, which will falsely indicate packet loss. It can lead to unnecessary retransmissions and a substantial reduction of the congestion window thereby reducing total throughput. We implement a variation of Fastest Path First scheduling algorithm which is known to be robust against reordering.

Load balancing among several paths requires some estimate of each path capacity. The HIP layer can attempt to estimate and control variations of such path parameters as delay, bandwidth and loss rate. We propose a simple $marking\ technique$, storing the estimated delivery time for a chosen packet, one per a TCP round-trip time cycle, to compare estimated delivery time of the marked packet to the actual time of its arrival. This enables our network system to react quickly to the change in the available bandwidth of the paths and redirect the useful traffic accordingly.

Network links can experience external cross-traffic which reduces available path bandwidth. To address the problem we propose a $multipath\ congestion\ avoidance\ scheme$. Novel feature of the scheme is that it takes control of congestion situation in the overall end-to-end multipath system unlike TCP which is restricted to control congestion in one path only.

The rest of the paper is organized as follows. Section II describes Host Identity Protocol and summarizes related work in the field of multipath scheduling. the HIP multipath transport architecture is introduced in Section III. Section IV presents the design and implementation of the multipath scheduling algorithm. Section V presents the experimental results. Conclusions and future work are given in Section VI.

II. Related Work

A. Host Identity Protocol

The Host Identity Protocol (HIP) [8], [16], [17] was proposed to overcome the problem of using IP addresses simultaneously for host identification and routing. The idea behind HIP is based on decoupling the network layer from the higher layers in the protocol stack architecture (see Figure 1). HIP defines a new global name space, the Host Identity name space, thereby splitting the double meaning of IP addresses. When HIP is used,

[1]Research supported by TEKES as part of the Future Internet program of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT).

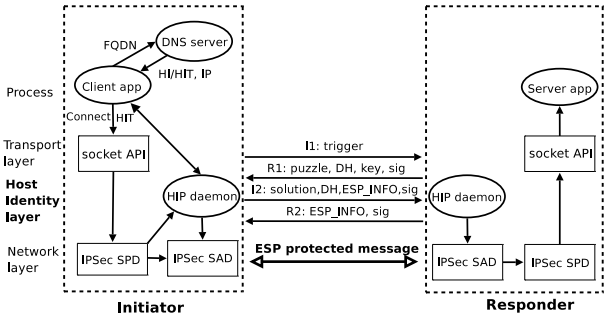

Fig. 1. Establishing HIP association with base exchange.

upper layers do not any more rely on IP addresses as host names. Instead, Host Identities (HI) are used in the transport protocol headers for establishing connections. IP addresses at the same time act purely as locators for routing packets towards the destination. For compatibility with IPv6 legacy applications, Host Identity is represented by a 128-bit long hash, the Host Identity Tag (HIT).

HIP offers several benefits including end-to-end security, resistance to CPU and memory exhausting denial-of-service (DoS) attacks, NAT traversal, mobility and multihoming support.

To start communicating through HIP, two hosts must establish a HIP association. This process is known as the HIP Base Exchange (BEX) [17] and it consists of four messages transferred between the initiator and the responder. After BEX is successfully completed, both hosts are confident that private keys corresponding to Host Identifiers (public keys) are indeed possessed by their peers. Another purpose of the HIP base exchange is to create a pair of IPsec Encapsulated Security Payload (ESP) Security Associations (SAs), one for each direction. All subsequent traffic between communicating parts is protected by IPsec. A new IPsec ESP mode, Bound End-to-end Tunnel (BEET) [20] is used in HIP. The main advantage of BEET mode is low overhead in contrast to the regular tunnel mode.

Figure 1 illustrates the overall HIP architecture including the BEX. The initiator may retrieve the HI/HIT of the responder from a DNS directory [19] by sending a FQDN in a DNS query. Instead of resolving the FQDN to an IP address, the DNS server replies with a HI (FQDN→HI). Transport layer creates a packet with the HI as the destination identifier. During the next step HI is mapped to an IP address by the HIP daemon on the Host Identity layer. Finally, the packet is processed in the network layer and routed to the responder. As a result, the conventional 5-tuple socket becomes {protocol, source HI, source port, destination HI, destination port}.

Since neither transport layer connections nor security associations (SAs) created after the HIP base exchange are bound to IP addresses, a mobile client can change its IP address (i.e., upon moving, due to a DHCP lease or IPv6 router advertisement) and continue transmitting ESP-protected packets to its peer. HIP supports such mobility events by implementing an end-to-end three-way signaling mechanism [18] between communicating nodes. HIP multihoming uses the same mechanisms as mobility for updating the peer with the current set of IP addresses of the host.

B. Multipath transport architectures

Researchers have approached the problem of utilizing multiple network paths from various angles. Multipath transmission can be implemented on physical, link, network, transport or application layers [4], [9], [10], [11], [12], [13], [23]. Placing the function on a lower layer enables more efficient utilization of a particular link type and presents a generic solution for all upper-layer protocols and applications. On the other hand, solutions on upper layers are more tuned for the need of a specific application and could be implemented more easily (e.g., in the user space).

Transport layer applications can naturally obtain information on the quality of different paths. For example, SCTP [25] can perform measurements across several paths simultaneously, and then map flows on one or another path. TCP-MH [15] can detect when the current path has stopped working well, for instance, if the frequency of repetition becomes too high, and decide to try another path.

The advantages of network layer solutions, such as proposed in [4] for wireless network interfaces, are they are easy to deploy, totally transparent to applications and involve minimum modifications in the contrary to the application and transport layer solutions which involve many changes in the infrastructure.

Wedge-layer approaches implemented in HIP, LIN6 [6], MAST [5], MIM6 [14] conduct control exchange on a separate logical channel. This approach has an advantage of being able to maintain multiaddressing information across transport associations. Transport activity between two endpoints may well be able to use multiaddressing immediately and with no further administrative overhead. Moreover, wedge-based locator exchange protocols can be incorporated without necessitating modification to any host's IP or transport modules.

A number of applications or transport connections can be allocated independently to different paths [21]. As

an example, popular web browsers open several parallel TCP connections to download a page. Such approach avoids complications resulting from spreading packets from a single transport connection over multiple paths. However, it has an obvious drawback – if there are fewer active bulk transport connections than links, it is not possible to utilize all available paths. Simultaneous Multiaccess (SIMA) [22] implements such approach using flow binding extensions for HIP.

Several proposals in the related work assume the presence of a proxy in the network that can serve as a termination point of multipath TCP extensions [3]. Such approach works only for plain-text TCP communication and fails in the presence of IPsec encryption or authentication mechanisms. When TCP packets are protected with IPsec, the proxy is unable to observe or modify the packets. Therefore, if HIP is used end-to-end, proxy-based solutions are not applicable.

III. HIP MULTIPATH TRANSPORT ARCHITECTURE

Next we compare two approaches, implementing multipath on transport and network layers, in detail. The first approach implements multipath capabilities on the transport layers, as an extension of TCP or SCTP protocols. Trilogy project [2] is taking this approach. We compare it with our approach, where multipath scheduling occurs at the wedge-layer, below HIP.

The SCTP protocol [25] supports a notion of multiple paths for fault-tolerance. Therefore, extending it to support simultaneous transfer over multiple paths is possible without significant changes to the protocol structure. On the contrary, TCP is built to run over a single path only. Connection-specific functions, such as flow control and connection establishment are tightly coupled with path-specific functions such as MTU discovery, congestion avoidance and retransmissions. Hence, implementing multipath transfer in TCP requires significant refactoring of the code, separating connection and path-specific components so that functions such as congestion control could be implemented per each path.

Figure 2 illustrates this approach. Essentially it requires turning TCP into SCTP with additional functionality. The only benefit compared to extending SCTP directly would be support for legacy applications. However, compatibility of TCP-only applications with SCTP could be provided at lower cost through a shim adaptation API.

The main task of the scheduler is to distribute packets from one connection over available links. According to the resource pooling principle specified in [26] ideally the overall throughput is the sum of all link bandwidths. Since TCP and SCTP cannot robustly differentiate between packet reordering and packet loss, the scheduler must minimize reordering at the receiver. However, given variable delays on the links, eventual out-of-order delivery is almost inevitable. Several improvements for TCP and SCTP do exist that make the reordering tolerable including the Eifel algorithm and DSACK [27]. Such mechanisms can dynamically adjust the DUPACK threshold to balance between packet loss recovery and avoiding spurious retransmissions.

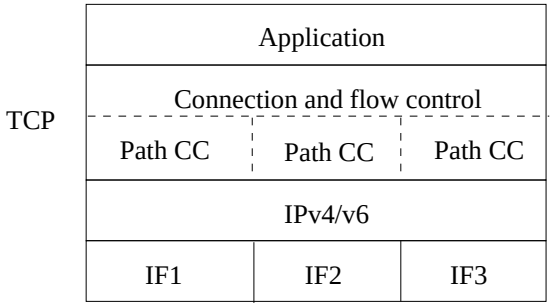

Fig. 2. Multipath TCP.

We take a different approach compared to extending transport protocols such as TCP and SCTP. Since multipath functionality requires proper security mechanisms to avoid accepting packets from spoofed IP addresses, it is logical to implement it on the HIP layer. HIP shields the presence of multiple paths from transport and application layers, presenting only the identity of the peer host, a Host Identity Tag (HIT). Therefore, all multipath functionality must be located below the HIP layer as the upper protocol layers only see a single path through HIT. In fact, when an application performs a DNS query from FQDN to an IP address, the HIP resolver removes all IP addresses from the result returned to the application, leaving only HIT (for IPv6 applications) or LSI (for IPv4 applications). The HIP layer is entirely responsible for mapping between HITs and current IP addresses of the peer host.

We propose a multipath transfer architecture as shown in Figure 3. The scheduler located below the HIP protocol maintains an estimate of each path parameters, including the congestion window, retransmission timer, and MTU. It spreads packets from TCP connections over available paths according to their capacity. The scheduler is TCP-friendly when allocating TCP traffic, competing fairly with other TCP connections. It can also schedule traffic from other protocols that are not necessarily TCP friendly, such as UDP, SCTP, DCCP over the links according to their capacity.

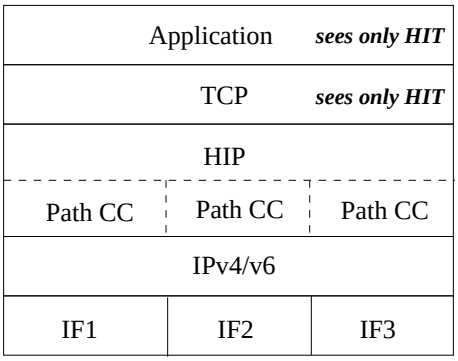

Fig. 3. Multipath HIP.

IV. Multipath Scheduler

We formulate the multipath scheduling problem as an online optimization problem. A network traffic scheduler assigns a path for each packet arriving from the TCP sender to minimize its delivery time and packet reordering at the receiver. Paths characteristics such as capacity and delay are analyzed and the path which provides the earliest delivery for the given packet is chosen.

The goal is to maximize the throughput of our multipath network, the number of packets successfully delivered to the receiver. To achieve the goal we schedule packets to arrive with minimal reordering, which in case of TCP traffic provides increasing throughput by reducing delays due to unnecessary retransmissions.

A. Assumptions and limitations

Our approach will correspond to the class of disjoint multipath routing (DMPR) algorithms[24]. The paths are restricted to have independent bottlenecks. We make the following assumptions:

- Only TCP data traffic source is considered.
- The scheduler resides only on the sender, 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.

B. Problem statement

Given an ordered sequence of TCP data packets each of size S arriving from the sender, n available paths connecting the sender and the receiver, each of which could consist of a number of consecutively connected links, with the following end-to-end path characteristics: D_i - delay in the path i; B_i - bottleneck bandwidth of the path i.

The scheduler is not allowed to change the order in which packets are sent even if multiple packets are available at the same time. But several packets could be scheduled to depart at the same time if assigned to different parallel paths.

The scheduling problem is to assign the packets to the paths to minimize reordering at the receiver subject to keeping the throughput maximum.

C. Multipath scheduling algorithm

We use a variation of the the Fastest Path First scheduling suggested in the paper [7], which is also referred as the Earliest Delivery Path First in [4] and has the property of eliminating reordering fully in case if all the packets are of the same size.

For each arriving packet p the expected delivery time t_{pi} if sent to route i is to be estimated. Then the packet is sent to the path with minimum value of t_{pi}.

We calculate the expected delivery time for each packet according to the following formula:

If $t^{\text{now}} < t_i^{\text{free}}$ - route i is currently busy delivering other packets, then

$t_{pi} = t_i^{\text{free}} + D_i + S/B_i$; set $t_i^{\text{free}} = t_i^{\text{free}} + S/B_i$;

otherwise (the route is free):

$t_{pi} = t^{\text{now}} + D_i + S/B_i$; set $t_i^{\text{free}} = t^{\text{now}} + S/B_i$,

where

t_i^{free} keeps the information about the availability of the path i,

t^{now} - current wall-clock time.

If several paths share the value of the estimated delivery time for a packet, we choose a path with the earliest expected arrival time of the last packet sent on this path. If the tie still exists, the path with the smallest sequence number of the last packet sent on the path is chosen.

D. Complexity considerations

The algorithm calculates estimated delivery times on each of the paths for each packet and compares the values to find the minimum. The number of operations per packet depends only on the number n of disjoint paths available and is constant when n is fixed.

Spacial complexity is linear in the flight size - the number or packets which have been sent but not yet acknowledged. One integer space per packet is used to store packet-to-path assignment in the one-dimensional array and is released after successful arrival of the packet.

E. Multipath congestion avoidance

FPF algorithm is robust against packet reordering when applied to a multipath system with stable bandwidths and delays of the links. But in practice network links often experience external cross-traffic and packet

losses, which result in the variations of the available end-to-end path bandwidths. Without additional modifications, our multipath system would have performance problems. The scheduler would continue calculations basing on the initial values of paths characteristics unaware of the recent changes. As a result, packets would arrive out of order. Packet reordering leads to needless retransmissions and confuses the TCP congestion control. The application would experience delays and reduced throughput. An example of this problem is analyzed in Section V-C and is illustrated in Figure 9.

Obviously, we need to enable the scheduler to react to path changes in timely manner. For that purpose we design a simple $marking\ technique$ as a part of the multipath congestion avoidance scheme. To detect a congestion situation quickly, we are marking packets on their departure to each path. The expected delivery time of the marked packet is stored and then compared to its actual arrival time value on the receipt of the corresponding ACK. After the path parameters are updated, another departing packet is marked. Hence, change of path characteristics should be detected in the period of one TCP round-trip time. Then, the scheduler would again fairly distribute packets to the paths according to their capacity and delay with minimal reordering, maximizing the total end-to-end throughput.

Next we propose a $multipath\ congestion\ avoidance\ technique$ which works effectively not only for a single path, but also gives a performance enhancement to the end-to-end multipath systems.

We consider two indicators of path congestion:

1. the standard TCP DUPACK action, when the sender is retransmitting the packet after receipt of three duplicate acknowledgments from the receiver;

2. the observed delivery time of the marked packet exceeds its corresponding expected delivery time value.

If any of the two indicators suggest congestion, the path is temporarily closed and the packets are redirected to other available paths.

F. Path probing

To determine whether a path has already recovered from congestion, we occasionally send probing packets to the closed path. Currently we open the path for one TCP packet and marking it to compare its expected and observed delays values on the receipt of the corresponding ACK. In the future we plan to use HIP heartbeat packets for this purpose.

If the path is still congested, the TCP probing packet would cause some reordering on the receiver. There is a trade-off between the unnecessary reordering and the idleness of the unused path. A reasonable time between probes needs to be chosen. For the simulations described in Section V, we set the time between probing packets to 1 second. Further research is needed to determine the best way of choosing such a value.

There is a constraint on which packet could be chosen for probing purposes. It should not be a duplicate packet to ensure it would come in regular order when path is not congested anymore.

G. ACK processing

On receipt of an ACK, the scheduler determines whether it comes in the regular order (sequence number surpasses its predecessor by one), is a duplicate (is equal to the sequence number of its predecessor) or cumulative (surpasses the previous ACK sequence number by more than one).

If ACK for a marked packet is received in the regular order, we compare the actual delivery time and expected delivery time values for this packet. Based on the result of this comparison we determine congestion situation in the corresponding path as described earlier. The receipt of a cumulative ACK covering marked packet sequence number, means congestion in one of the paths. However, this information is not sufficient to define the congested path number. The marking is reset and the following departing packet is to be examined.

V. Experimental evaluation

We choose goodput of the multipath system to be a metric for evaluation of our first simulation results. $Goodput$ is the application level throughput, the amount of data per unit of time (in Mbps) delivered by the network from source to destination, excluding protocol overhead and retransmitted data packets.

The simulations were performed using publicly available ns-2 simulator [1]. We evaluate performance of our algorithm implemented on two simple network topologies with two and three available paths between one source-destination pair.

A. Ideal system with two paths

Systems with stable path characteristics we call $ideal$. The paths are not experiencing any packet losses or cross-traffic interruptions. Taking these unrealistic assumptions we aim to place a limit on the best performance which is possible to achieve by using multiple paths simultaneously.

First we construct a network topology with two available paths. Two nodes $n0$ and $n1$ are connected by two parallel paths through two additional nodes $n2$ and $n3$ as shown in Figure 4.

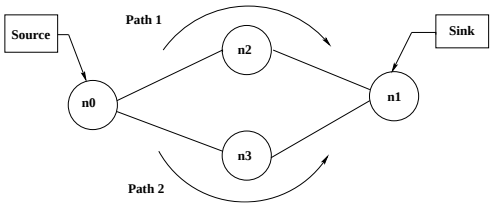

Fig. 4. Simple simulation topology with 2 paths.

The topology could present several disjoint paths and several nodes on each path. For the sake of clarity we consider in this paper only two- and three-path scenarios. And since we are measuring end-to-end propagation delays in the paths, they can consist of any number of connected links.

A TCP sender (source) is attached to the node $n0$ and TCP sink to the node $n1$. The traffic is produced by a FTP source. The sender and the sink are connected by four bi-directional duplex links having variable bandwidth and delay characteristics. To analyze how bandwidth variations can influence the resulting goodput of the multipath system, we fixed the delays in all four links to 10 ms and tried different combinations of two paths bandwidths.

TCP packets are of the same size equal to 1250 bytes. Maximum receiving window size is 100 packets to allow the system achieve its maximum goodput value. The simulated run time was set to 10 seconds, which is sufficient to ensure an adequate sample.

The resulting goodput of the two-path system with variable link bandwidths is illustrated in Figure 5. The increase of the goodput is almost linear provided by the increase of the total bandwidth of two paths. The maximum is achieved at 19.32 Mbps when bandwidths of both paths are at their maximal values equal to 10 Mbps.

We compare resulting goodput values to the sum of the goodput of two corresponding unipath systems. For these unipath systems bandwidths of their single paths are set to the values of our $Path1$ and $Path2$ respectively. The result of the comparison shows that the two-path system produces about 99% of the sum of two path bandwidths.

The resulting goodput depends not only on the bandwidth of the paths in the system but also on the packet sizes and the link delays. As an example, the observed goodput of the two-path system with the path bandwidths of 10 Mbps and 0 Mbps is not exactly 10 Mbps as one could have expected, but equals to 9.6 Mbps because of the delay in the path.

In the ideal system no reordering occurs at the receiver during the simulation period. This observation confirms applicability of FPF algorithm.

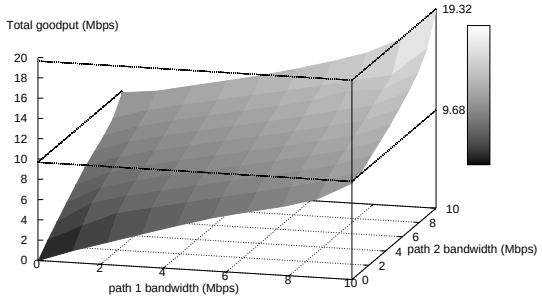

Fig. 5. Total goodput of ideal two-path system.

B. Ideal system with three paths

In the next experiment, we add $Path3$ connecting the same TCP sender and the sink through an additional node $n4$ as shown in Figure 6. This time we fix $Path1$ and $Path2$ bandwidth characteristics to 4 Mbps and 5 Mbps respectively and change the bandwidth of $Path3$, keeping the rest of the system parameters unchanged.

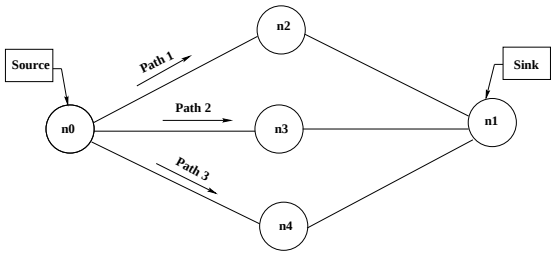

Fig. 6. Simple simulation topology with 3 paths.

Figure 7 shows the observed linear increase of the resulting goodput of the three-path system compared against the initial two-path system. Obviously adding one more path to the system noticeably improves its performance. Adding a path with the bandwidth of 10 Mbps more than doubles the goodput of the two-path system with 4 and 5 Mbps links.

Next, we set bandwidths of the paths to 2, 4 and 5 Mbps. Again, we compare our multipath system to three corresponding unipath systems. The results are summarized in Table I. The comparison confirms the result obtained earlier for the ideal two-path system. The multipath system produces goodput which exceeds goodput of any of the unipath systems and is about 99% of their sum.

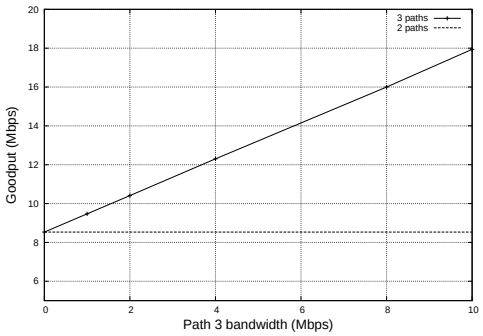

Fig. 7. Adding $Path3$ with variable bandwidth to the ideal two-path system.

TABLE I
IDEAL THREE-PATH SYSTEM SIMULATION RESULTS.

Topology	Bandwidth (Mbps)	Goodput(Mbps)
single path - path 0	2	1.91
single path - path 1	4	3.75
single path - path 2	5	4.68
Sum	2, 4 and 5	10.34
multipath (3 paths)	2, 4 and 5	10.27

C. Experiments with cross-traffic and losses

For experiments with cross-traffic we attach three UDP source agents to the node n_0 and three sinks (used for simulation purposes only) to the nodes n_2, n_3 and n_4 respectively (Figure 8). As in the previous example, bandwidths of three paths are set to 2, 4 and 5 Mbps. Cross-traffic patterns are chosen to start and end at different time points in different links and have variable intensity as shown in Table II. Each cross-traffic flow occupies at least 80 percent of the bandwidth in the corresponding link.

TABLE II
CROSS-TRAFFIC FLOWS PARAMETERS.

Link	Start time (s)	End time (s)	Rate (Mbps)
$n_0 - n_2$	1.0	2.0	4.0
$n_0 - n_3$	3.0	5.0	3.0
$n_0 - n_4$	3.5	6.0	2.0

Now we study the effect of applying our multipath congestion avoidance scheme described in Section IV-E to the three-path system with the cross-traffic scheduled as shown in Table II. We compare total goodput produced by the multipath system without any special congestion control actions taken with goodput of the same system with our multipath congestion avoidance scheme. To get more complete picture of the performance, we plotted the sending rates of the system without congestion control measures against the system with our multipath congestion control on.

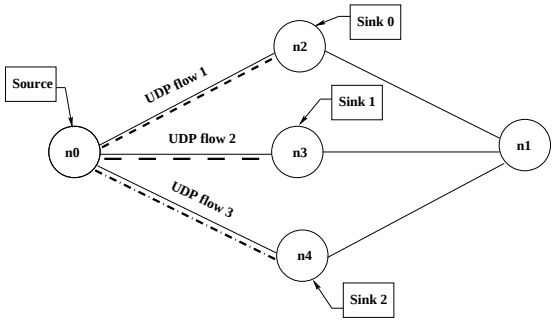

Fig. 8. Cross-traffic flows simulation.

Figure 9 illustrates noticeable degradation of the sending rate due to the existence of cross-traffic flows. Such a decrease in the sending rate is naturally accompanied by the goodput reduction. In particular, we observed about 43% reduction from the goodput of the ideal multipath system. Because of the cross-traffic, we obtain about 5.9 Mbps of goodput instead of 10.4 Mbps from the ideal system without cross-traffic. Most of the sending rate reduction is caused by unnecessary retransmissions and packet losses in the useful TCP traffic.

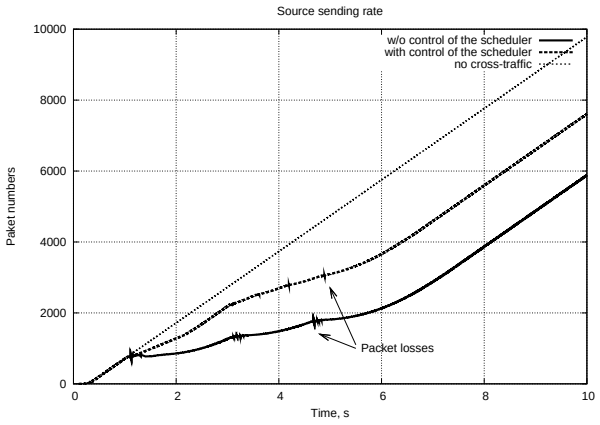

Fig. 9. Sending rate of the three-path system with cross-traffic.

We improve the situation by applying our multipath congestion avoidance technique in combination with the congested path probing. As a result, the goodput of our system achieves the value of 7.6 Mbps which is by 28% better than without the technique . As we can clearly see from the picture, less retransmissions and losses occur conforming an increase of the sending rate.

VI. Conclusions and future work

We have proposed a design of an online multipath scheduling algorithm for HIP which effectively distributes packets from a TCP connection over available links. It requires modifications only in the HIP daemon at the sender. Legacy IPv4 and IPv6 applications unaware of multiple paths can benefit from it transparently.

The initial experimental investigation has demonstrated robustness of the scheduling algorithm applied. In the ideal system with no cross-traffic, overall goodput of the simple multipath system is nearly the sum of link bandwidths.

When cross-traffic was introduced to the system, we were able to effectively decrease the number or retransmissions and packet losses. The result was achieved by applying a multipath congestion avoidance scheme, which includes redirection of the traffic to the less congested paths and consequent path probing.

Since HIP is using the IPSec encapsulation, we need to study the influence of its anti-replay sliding window [28] on the reordering of the resulting flow at the receiver.

The proposed traffic splitting algorithm does not explicitly change neither the TCP congestion window growing rate nor its recovery speed. We do not expect our multipath scheduler to behave more aggressively than any of the TCP variants; on the contrary, we believe it should demonstrate fairness and friendliness to the TCP flows. We follow up with careful experimental study on the subject to support our assumptions.

We are currently implementing HIP multipath scheduler on Linux. When it is completed, we will be able to measure the benefits of multipath routing in real networks, including WLAN and 3G links.

References

[1] The network simulator ns-2. http://www.isi.edu/nsnam/ns/ns-documentation, last checked 23/02/2009.

[2] Trilogy project. http://www.trilogy-project.org/, last checked 23/02/2009.

[3] R. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance enhancing proxies intended to mitigate link-related degradations. RFC 3135, 2001.

[4] 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.

[5] D. Cocker. Multiple address service for transport (MAST). In *Proc. of Symposium on Applications and the Internet (SAINT'04)*, Tokyo, Japan, January 2004.

[6] C. de Launois, B. Quoitin, and O. Bonaventure. Leveraging network performance with IPv6 multihoming and multiple provider-dependent aggregatable prefixes. *Comput. Netw.*, 50(8):1145–1157, 2006.

[7] R. Greco and G. Galante. Load balancing over multipaths using bandwidth-aware source scheduling. In *Proc. of International Symposium on Wireless Personal Multimedia Communications (WPMC'04)*, December 2005.

[8] A. Gurtov. *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley and Sons, 2008.

[9] 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. *IEEE/ACM Trans. Netw.*, 14(6):1260–1271, 2006.

[10] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Deployable multipath communication scheme with sufficient performance data distribution method. *Comput. Commun.*, 30(17):3285–3292, 2007.

[11] X. Hesselbach, R. Fabregat, B. Baran, Y. Donoso, F. Solano, and M. Huerta. Hashing based traffic partitioning in a multicast-multipath MPLS network model. In *LANC '05: Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking*, pages 65–71, 2005.

[12] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: aggregating AP backhaul capacity to maximize throughput. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 89–104, 2008.

[13] F. Kelly and T. Voice. Stability of end-to-end algorithms for joint routing and rate control. *SIGCOMM Comput. Commun. Rev.*, 35(2):5–12, 2005.

[14] J. Kempf, J. Arkko, and P. Nikander. Mobile IPv6 security. *Wirel. Pers. Commun.*, 29(3-4):389–414, 2004.

[15] 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.

[16] R. Moskowitz and P. Nikander. Host Identity Protocol architecture. IETF RFC 4423, May 2006.

[17] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Experimental Host Identity Protocol (HIP). IETF RFC 5201, Apr. 2008.

[18] P. Nikander, T. Henderson, C. Vogt, and J. Arkko. End-host mobility and multihoming with the Host Identity Protocol (HIP). IETF RFC 5206, Apr. 2008.

[19] P. Nikander and J. Laganier. Host Identity Protocol (HIP) domain name system (DNS) extension. IETF RFC 5205, Mar. 2008.

[20] P. Nikander and J. Melen. A bound end-to-end tunnel (BEET) mode for ESP: draft-nikander-esp-beet-mode-09, Aug. 2008. Work in progress.

[21] R. Penno, S. Raghunath, and J. Iyengar. LEDBAT practices and recommendations. RFC, IETF, 2009.

[22] S. Pierrel, P. Jokela, and J. M. Melen. Simultaneous Multi-Access extension to the Host Identity Protocol: draft-pierrel-hip-sima-00, June 2006.

[23] S. Ramabhadran and J. Pasquale. Stratified Round Robin: a low complexity packet scheduler with bandwidth fairness and bounded delay. In *Proc. of ACM SIGCOMM'03*, pages 239–249, Aug. 2003.

[24] S. Ramasubramanian, H. Krishnamoorthy, and M. Krunz. Disjoint multipath routing using colored trees. *Comput. Netw.*, 51(8):2163–2180, 2007.

[25] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960, IETF, Oct. 2000.

[26] D. Wischik, M. Handley, and M. B. Braun. The resource pooling principle. *SIGCOMM Comput. Commun. Rev.*, 38(5):47–52, 2008.

[27] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: A Reordering-Robust TCP with DSACK. In *IEEE ICNP*, pages 95–106, 2003.

[28] F. Zhao and S. Wu. Analysis and improvement on IPSec anti-replay window protocol. In *ICCCN 2003: Proceedings of the 12th International Conference on Computer Communications and Networks*, pages 553–558, 2003.