STRESS TESTING OF HOST IDENTITY PROTOCOL (HIP) IMPLEMENTATIONS

Oleg Ponomarev Andrei Gurtov

Helsinki Institute for Information Technology HIIT, Finland
Helsinki University of Technology TKK
oleg.ponomarev@hiit.fi, gurtov@hiit.fi

ABSTRACT

The Host Identity Protocol (HIP) was introduced almost a decade ago. There are three interoperating software implementations. HIP provides mobility and multi-homing in a secure way to the Internet hosts. Several studies evaluated the duration of HIP association establishment and mobility updates. In this paper, we perform stress testing of available HIP implementations on a multiprocessor server. When we started, the best result was only 12 connections per second, but after providing feedback to the developers, the stability of HIP implementations has improved and a new version is able to accept 112 connections per second. Moreover the number should increase up to 800 on an eight-core server by introducing more efficient multithreading support to the HIP implementations. This would make HIP performance comparable with current commercial SSL/TLS implementations.

KEYWORDS

HIPL, OpenHIP, hip4inter.net, performance evaluation, secure mobility

1. INTRODUCTION

Currently IP addresses serve both as end host identifiers and routing locators. This approach was suitable before the wide adoption of portable devices, but with increasing host mobility the existing Internet architecture needs a revision. There are various ongoing efforts to split host identifiers and locators [1]. One of the approaches, the Host Identity Protocol was introduced in 1999 [2]. Its architecture was published as a Request-for-Comment in 2005 [3]. At the time of writing, there are three active implementations running on several operating systems. Until now, only measurements of the duration of HIP base exchange [4] or mobility updates [5] and a study of HIP performance on lightweight hardware [6] were made.

HIP is used for several experimental as well as production environments. We suggested using HIP as one of the authentication methods for the DNS updates [7, 8]. There are also ideas on using HIP for Source Address Validation and Distributed Authentication [9]. Therefore, it is topical to investigate the possible peak performance of a server host serving multiple HIP clients.

The rest of the paper is organized as follows. We give a short overview of the protocol and existing implementations in Section 2, then present the results of our experiments in Section 3, and finish with conclusions in Section 4.

2. HOST IDENTITY PROTOCOL

2.1. Identifier/locator split architecture

The idea behind HIP is 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 the IP addresses. When HIP is used, upper layers do not any more rely on IP addresses as host names. Instead, Host Identifiers are used in the transport protocol headers for identifying hosts and establishing connections. IP addresses at the same time act purely as locators and are responsible for routing packets towards the destination. A Host Identifier is a public key of the host. For compatibility with IPv6 legacy applications, a Host Identifier is further represented by a 128-bit long cryptographic 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 multi-homing support.

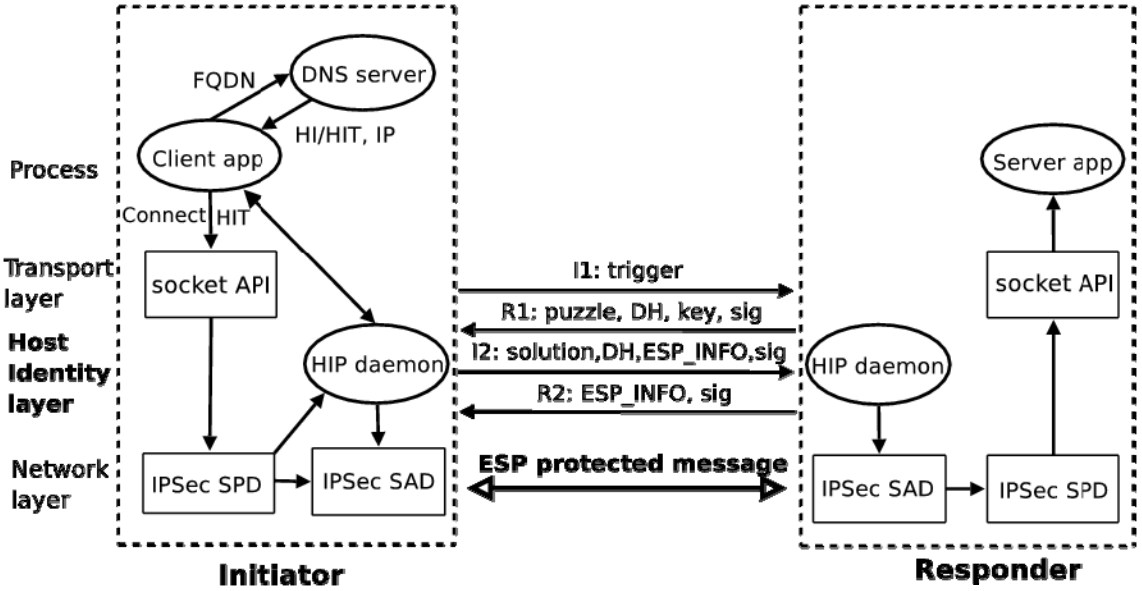

Figure 1. HIP architecture.

2.2. Base exchange

To start communicating through HIP, two hosts must establish a HIP association. This process is known as the HIP Base Exchange (BE) [10, 11, 12] and it consists of four messages transferred between the initiator and the responder. After BE 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) [13] 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 BE. The initiator may retrieve the HI/HIT of the responder from a DNS directory [14] by sending a FQDN in a DNS query. Instead of resolving the FQDN to an IP address, the DNS server replies with an HI (FQDN->HI). Transport layer creates a packet with the HI as the destination point 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 by the

network layer and delivered to the responder. As a result, the conventional 5-tuple socket becomes (protocol, source HI, source port, destination HI, destination port).

2.3. HIP implementations

HIP for Linux. HIPL [15] utilizes the BEET IPsec mode in official Linux kernel starting from the version 2.6.27. The HIP daemon can be executed right away on most modern Linux distributions. We tested release 1.0.4 that runs as a non-threaded single process.

OpenHIP supports two modes. It can run entirely in the user-space (on Linux, Windows XP and Mac OS X). This mode received most development attention and has more features [16]. The alternative mode on Linux is to patch the kernel and have data plane encryption in kernel-space. We studied the user-space mode by creating a virtual interface to pick packets to HITs, process packets and forward further. There was OpenHIP-0.6_RC1 version available at the moment of writing, it starts multiple threads.

HIP for inter.net This implementation provides the HIP daemon and system libraries for FreeBSD [17]. The encryption and decryption of the user data is done using BEET mode of IPsec in the kernel. Despite of FreeBSD 7.0 release in February 2008, one year later only FreeBSD 5 and 6 were supported. Therefore we downloaded 2008.08.17 snapshot of HIP for inter.net and installed it on FreeBSD 6.3-RELEASE-p2. The user is instructed to compile and install a custom kernel as well as a custom system library (libc) enhanced with HIP functions to use it.

PyHIP Although this implementation is mentioned on the IETF HIP working group page [18], its last release is from year 2003 and we did not use it for the experiments.

3. EXPERIMENTAL RESULTS

3.1. Experiment setup

It is nearly impossible to start multiple HIP sessions from the same host; therefore we used a kernel-based virtual machine [19] to run multiple independent clients on a few physical machines. Our virtualized clients had identical file-systems generated using logical volume manager [20] snapshots from one image. However the hosts get different IP addresses from the DHCP server and generate their own Host Identities on the first boot. All hosts were in the same local network with a typical packet round trip time of 1-2 milliseconds.

The clients ran a simple cycle: trigger a HIP base exchange, wait for one second, close the HIP association, and wait for one more second. One such iteration takes a little over 2 seconds. There were no transmissions on the data plane.

Additional clients could be started on PlanetLab [21], a distributed platform for research and development of global network services. It allows creating Linux virtual hosts (slivers) on more than 800 physical machines at over 200 sites. Users can install and run their own software with root privileges, but there are some security restrictions, for example the safe version of raw sockets supports only ICMP, TCP and UDP protocols, not HIP own protocol 139.

3.2. HIP for Linux

We used an HP ProLiant DL360 G4 server with two dual-core Intel Xeon 3GHz processors running Linux i686 kernel 2.6.27.5 as a test server. In *the first experiment*, we started 14 clients initiating about 6 connections per second to the server. Such activity caused approximately 50% utilization of one processor in the server (Figure 2). The number of connections and processor

utilization were averaged over 10-second periods. As an example, 6 connections per second at the 40th ten-second period means that 60 connections were established from second 400 to second 409 of the experiment. The utilization is shown as the sum for all the processors and could reach 400% on such a system.

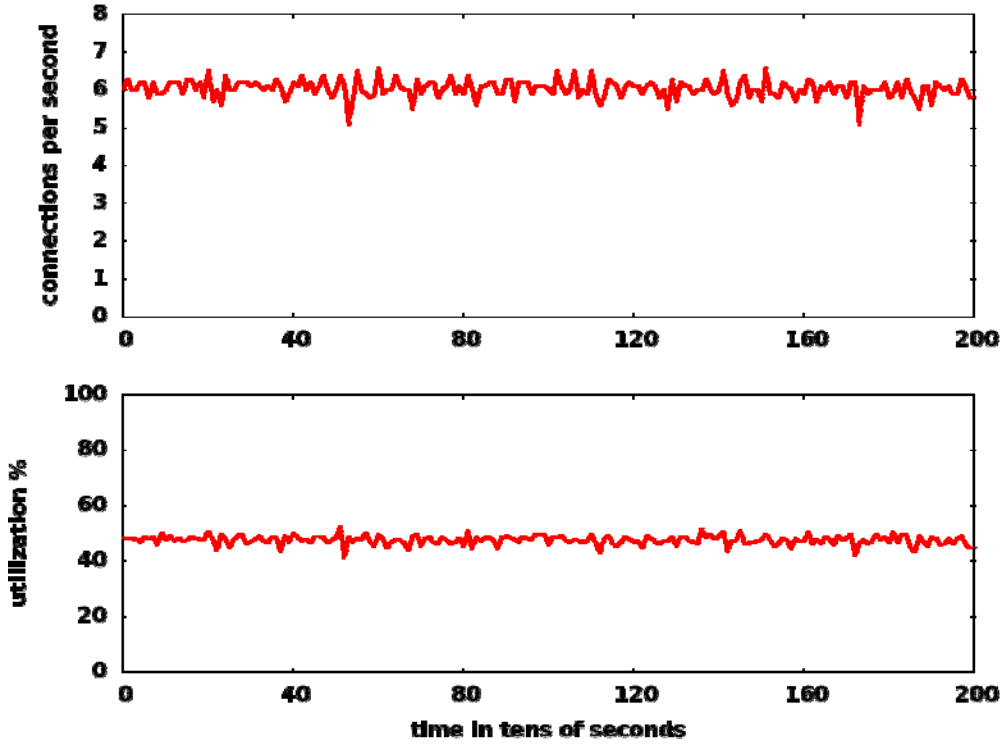

Figure 2. Number of base exchanges per second (top) and processor load (bottom) using HIPL implementation and i686 architecture at the server.

In *the second experiment*, we wanted to study the peak capacity. We increased the number of clients until the number of incoming connections stopped growing as the HIP daemon utilized one of the processors at 100%. The rate of incoming HIP connections and the corresponding total processor load are shown in Figure 3. The processor utilization is more than 100% due to Symmetric Multiprocessing (SMP) environment.

Since crypto-functions are executed much faster on x86_64 architecture, we performed *the third experiment*. We used an HP ProLiant DL380 G5 server with two dual-core Intel Xeon 5130 2GHz processors running Linux x86_64 kernel 2.6.27.19 as a test server. The rate of incoming HIP connections and the corresponding processor load are shown in Figure 4. In this case, we did not have a sufficient number of virtual machines to reach 100% CPU utilization, but with 18.05% summary load at 4.7 base exchanges per second we can estimate the peak capacity to be about 25-30 HIP connections per second.

We also tried to start clients on PlanetLab to reach the peak capacity, but only UDP encapsulation was available due to the security restrictions of PlanetLab. Unfortunately the implementation did not show good results in handling its UDP sockets, many packets were lost, and the new connection rate did not exceed five-six base exchanges per second in this test. We reported the issue to the developers.

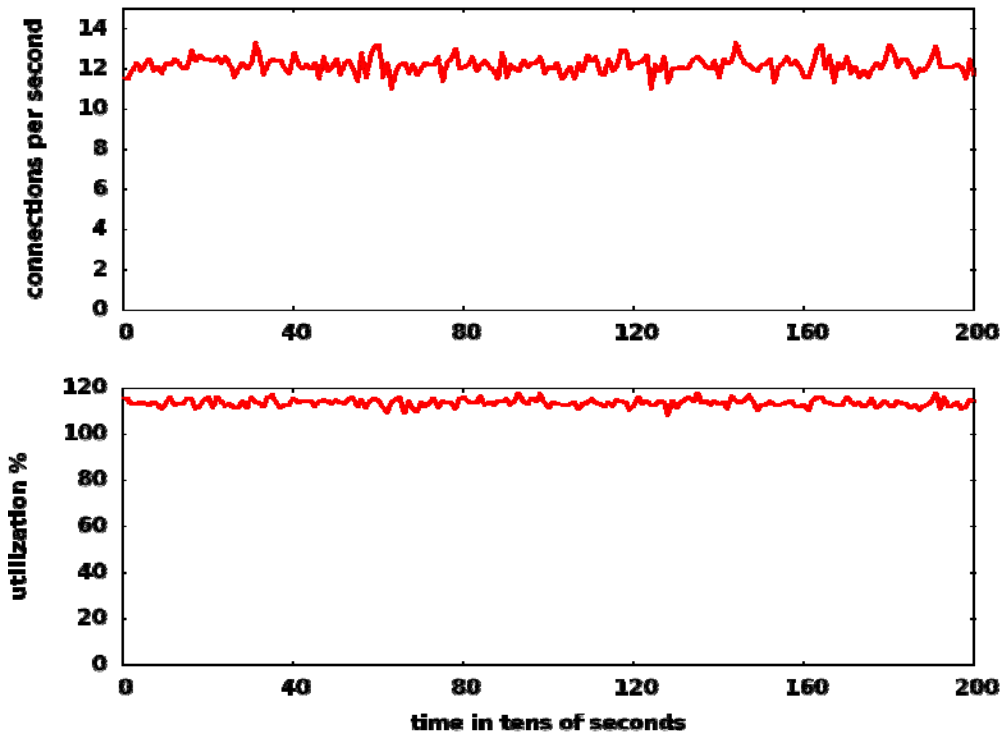

Figure 3. Number of base exchanges per second (top) and processor load (bottom) using HIPL implementation and i686 architecture at the server.

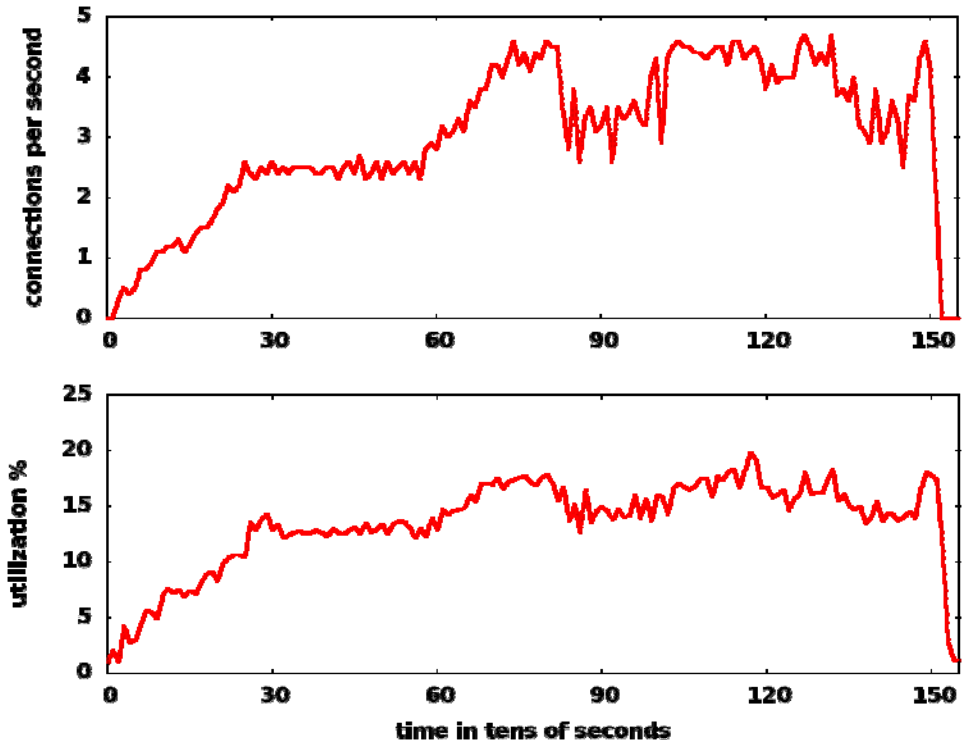

Figure 4. Number of base exchanges per second (top) and processor load (bottom) using HIPL implementation and x86_64 architecture at the server.

3.3. OpenHIP

We used Intel Pentium IV 3GHz desktop running Linux i686 kernel 2.6.24-23 as the server *in the first experiment*. The clients ran on PlanetLab slivers after a few changes to the configuration (UDP encapsulation) and minor alterations to the source code (to avoid using netlink sockets). Figure 5 shows the average number of HIP base exchanges per second and the corresponding processor utilization as the number of clients is increasing. The implementation served 33 base exchanges per second using the basic hardware.

An HP ProLiant DL360 G5 server with two quad-core Intel Xeon E5440 2.83 GHz processors running Linux x86_64 kernel 2.6.27.12 was the 64-bit architecture test server *in the second experiment*. The rate of incoming HIP connections and the corresponding processor load are shown in Figure 6. The implementation served 112 base exchanges per second when one of the CPU cores was utilized at 100% and other processors were almost idle.

3.4. HIP for inter.net

Unfortunately we could not start the experiments with this implementation at the moment of writing. After 10-30 iterations the HIP daemon at the server fails due to an internal error and thus we were not able to collect any performance data. We reported results to the developers.

3. CONCLUSIONS

The Host Identity Protocol is suggested as one of the methods to authenticate hosts for DNS Updates [7, 8], for use in mobile networks [22], and for validation of source IP addresses [9, 23]. Unfortunately, the existing implementations did not allow providing services to any significant number of clients using HIP when we started writing the paper. By providing feedback to developers, we were able to improve the performance and stability of implementations, when memory leaks in HIPL and dynamic allocation failures in OpenHIP were fixed. However both implementations still do not benefit from multiple processors and multiple cores common in the modern hardware. We suggest starting multiple worker threads to improve the situation.

After profiling the HIPL implementation, we noticed that the RSA_sign function takes much longer time than it should. It requires providing intermediate values for efficient calculations, but those were not stored after generating the keys. We suggested a patch to developers and hope the next HIPL version to serve about twice more connections per second. Another reason for the performance difference is the approaches to handling the data packets. HIPL has to update Linux kernel tables after every base exchange, while OpenHIP keeps the associations only in the process internal data structures.

With a multi-threaded implementation using all the processors, we believe at least 800 HIP connections per second should be served by an eight-core server running x86_64 Linux kernel. It would be comparable to other similar technologies such as TLS [24] with 517 connections per second on dual Pentium III at 933Mhz using 1024bit RSA keys [25].

ACKNOWLEDGEMENTS

The authors would like to thank all developers for their contributions to HIP community, and especially Miika Komu, Jeff Ahrenholz and Tom Henderson for their comments. We also appreciate valuable feedback given by Andrei Khurri.

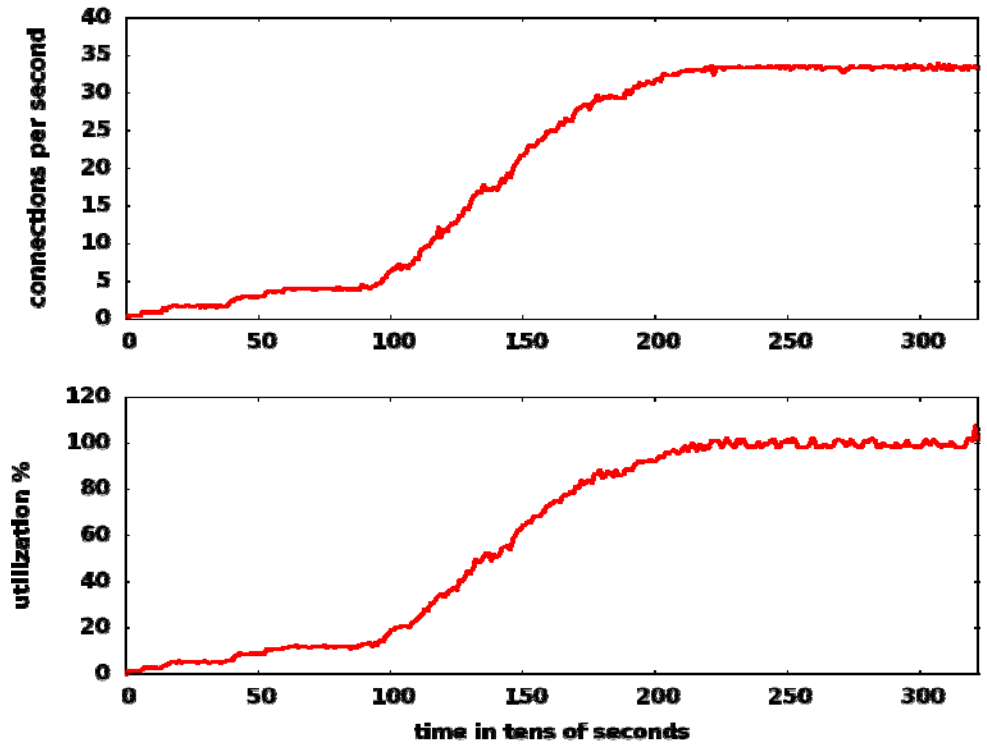

Figure 5. Number of base exchanges per second (top) and processor load (bottom) using OpenHIP implementation and i686 architecture at the server.

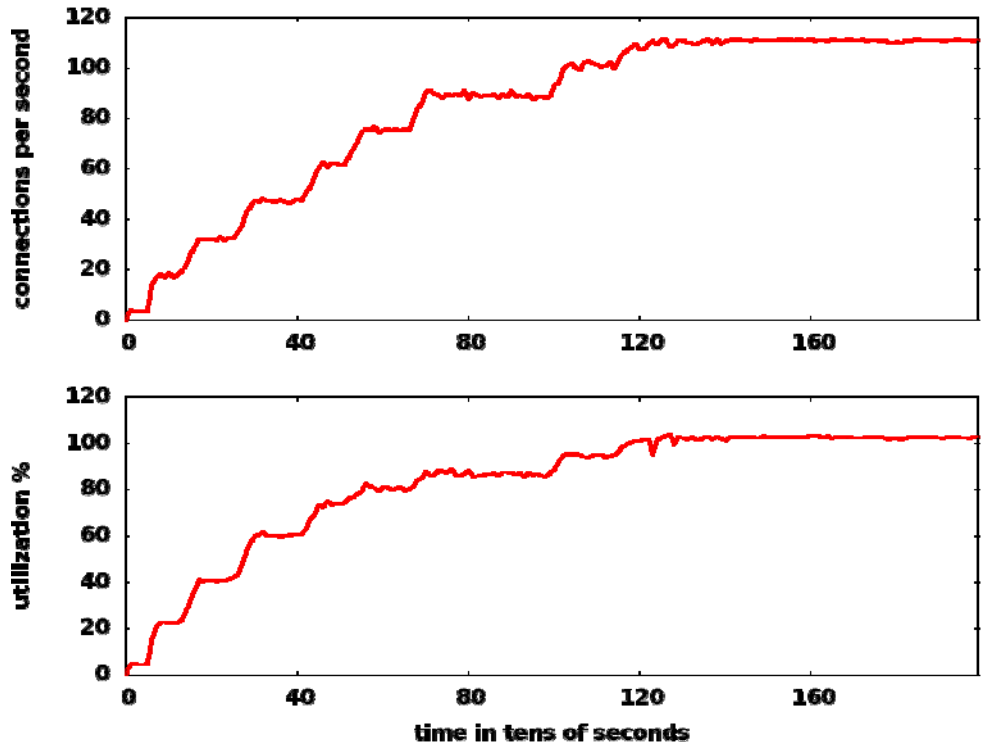

Figure 6. Number of base exchanges per second (top) and processor load (bottom) using OpenHIP implementation and x86_64 architecture at the server.

REFERENCES

[1] Nikander, P. (2007) “Identifier / locator separation: Exploration of the design space (ILSE)”, draft-nikander-ram-ilse-00, IETF

[2] Moskowitz, R. “Host Identity payload architecture”, draft-moskowitz-hip-arch-00, IETF, Dec 1999.

[3] Moskowitz, R. and Nikander, P. (2006) “Host Identity Protocol architecture”, RFC 4423, IETF

[4] Henderson, T.R., Ahrenholz, J.M. and Kim, J.H. (2003) “Experience with the host identity protocol for secure host mobility and multihoming”, in *Proc of Wireless Communications and Networking, 2003*, Vol. 3, pp. 2120-2125

[5] Jokela, P. et al. (2004) “Handover Performance with HIP and MIPv6”, in *Proc. of Wireless Communication Systems*, 2004, pp324-328.

[6] Khurri, A., Vorobyeva E. and Gurtov, A. (2007) “Performance of Host Identity Protocol on Lightweight Hardware”, in *Proc. of ACM MobiArch*’07.

[7] Ponomarev, O. and Gurtov, A. (2009) “Using DNS as an Access Protocol for Mapping Host Identity Tags to IP Addresses”, draft-ponomarev-hip-hit2ip-03, IETF

[8] Ponomarev, O. and Gurtov, A. (2007) “Using DNS as an Access Protocol for Mapping Host Identifiers to Locators”, Routing in Next Generation Workshop, Madrid, Spain

[9] Kuptsov, D. and Gurtov, A. (2009) “SAVAH: Source address validation with Host Identity Protocol”, Presentation at The Second AsiaFI School on Mobile and Wireless Networks, Beijing , China

[10] Moskowitz, R., Nikander, P., Jokela, P. and Henderson T. (2008) “Host Identity Protocol”, RFC 5201, IETF

[11] Gurtov, A. (2008) Host Identity Protocol (HIP): Towards the Secure Mobile Internet, ISBN 978-0-470-99790-1, Wiley and Sons. (Hardcover, 332 p)

[12] Gurtov, A. Komu, M., Moskowitz, R. (2009) Host Identity Protocol (HIP): Identifier/Locator Split for Host Mobility and Multihoming, Internet Protocol Journal, 12(1):27-32

[13] Jokela, P., Moskowitz, R., Nikander, P. and Jokela, P. (2008) “Using the Encapsulating Security Payload (ESP) transport format with the Host Identity Protocol (HIP)”, RFC 5202, IETF

[14] Nikander, P. and Laganier, J. (2008) “Host Identity Protocol (HIP) Domain Name System (DNS) Extension”, RFC 5205, IETF

[15] Infrastructure for HIP project, retrieved February 27, 2009, from http://infrahip.net/

[16] OpenHIP, retrieved February 27, 2009, from http://openhip.org/

[17] HIP for inter.net Project, retrieved February 27, 2009, from http://hip4inter.net/

[18] HIP Working Group, retrieved February 27, 2009, from http://tools.ietf.org/wg/hip/

[19] Kernel Based Virtual Machine, retrieved February 27, 2009, from http://kvm.qumranet.com/

[20] LVM2 Resource Page, retrieved February 27, 2009, from http://sourceware.org/lvm2

[21] PlanetLAB, an open platform for developing, deploying, and accessing planetary-scale services, retrieved May 20, 2009, from http://planet-lab.org

[22] Novaczki, S., Bokor, L. and Imre, S. (2007) “A HIP Based Network Mobility Protocol”, in *Proc. of International Symposium on Applications and the Internet Workshops*

[23] Kuptsov, D. and Gurtov, A. (2009) Source Address Validation with Host Identity Protocol, to appear in Proc. of MobiSec'09

[24] Dierks, T. and Rescorla, E. (2008) “The Transport Layer Security (TLS) Protocol Version 1.2”, RFC 5246, IETF

[25] Coarfa, C., Druschel P. and Wallach, D.S. (2006) “Performance analysis of TLS Web servers” in *ACM Transactions on Computer Systems,* Vol. 24, Issue 1, pp 39-69