Secure and Efficient IPv4/IPv6 Handovers Using Host-Based Identifier-Locator Split

Samu Varjonen Miika Komu Andrei Gurtov
Helsinki Institute for Information Technology
Helsinki University of Technology and University of Helsinki
firstname.lastname@hiit.fi

***Abstract*—Internet architecture is facing at least three major challenges. First, it is running out of IPv4 addresses. IPv6 offers a long-term solution to the problem by offering a vast amount of addresses but is neither supported widely by networking software nor has been deployed widely in different networks. Second, end-to-end connectivity is broken by the introduction of NATs, originally invented to circumvent the address depletion. Third, the Internet architecture lacks a mechanism that supports end-host mobility and multihoming in a coherent way between IPv4 and IPv6 networks.**

We argue that an identifier-locator split can solve these three problems based on our experimentation with the Host Identity Protocol. The split separates upper layer identifiers from lower network layer identifiers, thus enabling network-location and IP-version independent applications.

Our contribution consists of recommendations to the present HIP standards to utilize cross-family mobility more efficiently based on our implementation experiences. To the best of our knowledge we are also the first ones to show a performance evaluation of HIP-based cross-family handovers.

I. Introduction

The IPv6 address space is drastically larger than for IPv4, but IPv6 has not experienced a wide-scale deployment yet. Concurrent use of both addressing families cause problems for both network software and management due to non-uniform addressing. Existing legacy software is hard-coded to use IPv4 addresses and some of it can never be updated to support IPv6 due to its proprietary nature. The fact that IPv4 address space is almost exhausted does not make things any easier because a host might acquire only an IPv6 address in future networks. As a consequence, proprietary network software may have trouble to access the Internet in the future.

End-to-end communication between two hosts is not guaranteed anymore, even considering protocols for traversing NATs. To make things even more complicated, end-host mobility arises as a new requirement for the Internet. Users are used to staying continuously in contact with each other using cellular phones and may also want the same with other portable devices. Users may want to benefit from access technologies, such as WLAN and 3G, available on phones and other devices. Multiaccess is desirable for users, for example, to reduce monetary costs, to assess benefits from device proximity, or to obtain a faster connection. Even though cellular networks support mobility transparently, the same does not apply to WLAN mobility.

In the current Internet, an IP address both identifies and locates a host. However, this binding breaks when the address of the host changes. This is a problem both for relocating the mobile host and for maintaining long-term transport layer connections, which break upon such an event.

The identifier-locator split decouples the host identifier from its topological location. The new host identifier is present at the transport and upper layers to provide applications a fixed identifier independent of network location. The identifier-locator split introduces a layer between the transport and the network layers, and transforms identifiers dynamically into routable addresses and vice versa.

The concept of the Host Identity Protocol (HIP) [1], [2] is based on identity-locator split. It provides security, global end-host mobility, multihoming, NAT traversal, and Rendezvous/Relay services. The HIP stardard [3] describes end-host mobility and multihoming but handovers across IP families are left for further study. In this paper, we describe HIP-based cross-family handovers based on our implementation experimentation and performance evaluation. Compared to previous work [4], [5], [6], we focus on Linux rather than the BSD networking stack.

We proceed as follows in the rest of paper. In Section II, we describe HIP base exchange and mobility management as well as summarize the related work. In Section III, we outline the shortcomings in current HIP mobility specifications, propose a simple solution and share our experience in implementing cross-family handovers with Linux networking stack. We evaluate performance of intra-family and cross-family handovers for TCP flows in Section IV. Section V concludes the paper with a summary of our contributions.

II. Background

Host Identity Protocol (HIP) [1], [7] introduces a cryptographic namespace based on public-private key pairs. An identifier in the namespace is the public key of a public-private key that the end-host creates for itself. This identifier is called Host Identifier (HI).

The protocol employs two fixed-length representations of HIs because varying length identifiers are inconvenient in networking APIs for existing legacy stacks and protocol header encodings [8]. The first representation type is Host Identity Tag (HIT). It has the same size as an IPv6 address. The HIT is generated by hashing the HI and concatenating it with an orchid prefix (2001:10::/28) [9]. The second representation type is Local Scope Identifier (LSI) that is the size of an IPv4 address to support legacy applications. LSIs are valid only within the local host due to high collision probability of two hosts choosing the same LSI.

To use HITs and LSIs, an application uses the existing system resolver to resolve names to HITs and IP addresses using host files, DNS [10] or OpenDHT [11]. When the application uses a HIT or a LSI to establish new outgoing communications, networking stack intercepts the packet and triggers a base exchange (BEX) in the networking stack to set up symmetric keys for the IPsec tunnel.

A. Base Exchange

HIP Base EXchange (BEX) [1] is a secure Diffie-Hellman exchange that authenticates the end-hosts to each other using their public keys, and negotiates algorithms and symmetric keys for IPsec ESP [12]. The BEX is protected against replay attacks and authenticated with public-key signatures.

In HIP terminology, the client-side is referred as *Initiator* and the server-side as *Responder*. The BEX consists of four messages. First, the Initiator starts the base exchange with an I1 packet. Second, The Responder replies with its public key and Diffie-Hellman key material in an R1. Third, the Initiator responds with an I2 packet that contains its public key and Diffie-Hellman key material. Fourth, the Responder concludes the BEX with an R2 packet. After this, the HIP state (HIP association) can transition to ESTABLISHED state on both sides. The end-hosts have agreed on SPI numbers and symmetric keys for IPsec ESP. Finally, the applications can commence communication over the created IPsec tunnel.

After the BEX, the hosts lose their role as Initiator and Responder because there is no need for such separation. Now, the end-hosts can process mobility related packets which requires different kind of state handling as discussed in the next section.

B. Mobility Management

This section summarizes HIP-based mobility as described in RFC5206 [3]. We use Mobile IP terminology [13], [14] for denoting two communicating end-hosts, i.e., Mobile Node is a moving node and Correspondent Node is an immobile node. It should be noticed that the terminology can be a bit misleading because HIP architecture allows both hosts to move simultaneously. We use the HIP state machine terminology [1], [3] extensively here. We also refer to routable IP addresses as *locators*.

The core idea in HIP-based mobility is that when a mobile node detects a change in its locators, it sends its complete new set of locators to all of its correspondent nodes. A correspondent node receives the new locator set and verifies each address in the set for reachability by sending an UPDATE packet with random nonce (echo request) to the mobile node. The mobile node responds with a packet containing the same nonce (echo response). These packets containing echo request and response packets protect the correspondent node from replay attacks because they allow it to securely verify that the mobile node is in the location it claims to be. This procedure is also referred as the *return routability test*. It should be noticed that there are no separate return routability tests for addresses used in the BEX because the BEX itself acts as an implicit return routability test.

In HIP-based mobility, a locator pair has ACTIVE, DEPRECATED and UNVERIFIED states. Fig. 1 illustrates HIP-based mobility from the view point of locator pair state. For simplicity, retransmissions and optional negotiation of new D-H key material are excluded from the figure.

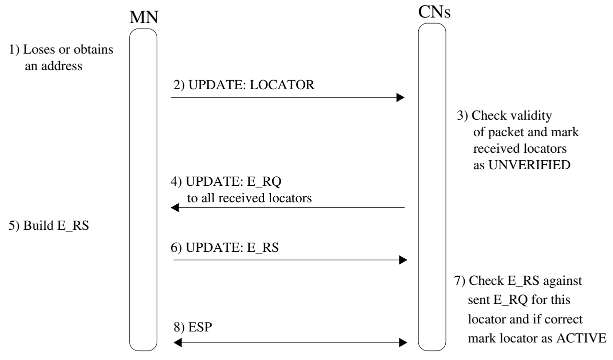
MN
CNs
1) Loses or obtains an address
2) UPDATE: LOCATOR
3) Check validity of packet and mark received locators as UNVERIFIED
4) UPDATE: E_RQ to all received locators
5) Build E_RS
6) UPDATE: E_RS
7) Check E_RS against sent E_RQ for this locator and if correct mark locator as ACTIVE
8) ESP

Fig. 1. Return routability tests and locator state.

When the mobile node moves (in step 1, Figure 1), its set of locators changes and it builds a LOCATOR parameter that contains the new locator set. The mobile node can exclude some locators from the LOCATOR parameter according to local policies. As an example, mobile node might not advertise expensive links for all correspondent nodes, or it might exclude some locators for privacy reasons. Corresponding node transitions the state of locator to DEPRECATED, when the mobile node excludes the particular locator from its locator set (not shown in Figure 1). In step 2, the mobile node sends an initial UPDATE, containing the LOCATOR parameter listing the locator set which the mobile node publishes to its correspondent nodes.

Now, the correspondent node receives the UPDATE packet and validates the packet by verifying packet checksum, correctness of the signature, sequence number and comparison of SPI number with existing SAs (step 3 in Figure 1). Then, the correspondent node processes the LOCATOR parameter from the UPDATE packet.

The correspondent node marks all received locators as UNVERIFIED and deprecates existing locators excluded from the new locator set (not shown in Figure 1). Next, the correspondent node builds an UPDATE packet containing an ECHO_REQUEST parameter (E_RQ in Figure 1) containing a random nonce value and sends it to mobile node's locator to be tested for reachability (step 4). The correspondent node repeats this for all of the locators contained in the locator set of the mobile node.

In step 5, the mobile node receives the packet and echoes the same nonce in an ECHO_RESPONSE parameter to correspondent node in step 6. The correspondent node receives the UPDATE packet and validates it. In step 7, the correspondent node finds the response sent by mobile node and verifies the nonce. The correspondent node transitions now the state of the peer locator to ACTIVE. If the mobile node does respond within a certain time, the correspondent node deprecates the locator and removes the locator from its peer locator list.

It should be noticed that locators can be present already in the base exchange. When a locator has a so called *preferred bit* set, the sender of the locator enforces the recipient to use the specific locator for HIP-related communications for e.g. load-balancing purposes.

C. Related work

In Mobile IP [13], [15], each node has a home address that identifies the node independently of its location. When the mobile is not located in its home address, the mobile node informs its Home Agent (HA) on its current address (Care-of-Address). Datagrams destined to the mobile node are tunneled to its current address through its home agent. MIPv6 includes an optimization that two allows end-hosts to route MIPv6-related traffic directly between them without triangular routing through the home agent. IPsec and MOBIKE [16] [17] can be used to protect Mobile IP traffic.

The MOBIKE protocol offers similar behavior in mobility as HIP. For example, the LOCATOR is similar to ADDITIONAL_*_ADDRESS (where * is IPV4 or IPV6) and the return routability test is similar as in HIP. The MOBIKE standards allow the mobile node to send additional addresses of different family than those currently in use [18].

A MIPv4 extension [19] introduces dual stack mobility by tunneling IPv6 over IPv4. This approach needs dual stack HA and triangular routing to offer movement between IPv4 and dual stack networks. Cross-family handovers, where nodes move from IPv4 network to IPv6 networks or vice versa, is left somewhat unclear in the specification.

Teredo is an IPv6-over-IPv4 tunneling protocol that includes a mechanism to avoid triangular routing [20]. Teredo uses UDP encapsulation and encodes additional information into the IPv6 addresses. Teredo defines a dedicated IPv6 prefix (2001:0::/32) for the tunnel which can be used by any IPv6-capable networking software.

SHIM6 [21] is a layer 3 multihoming protocol that offers locator agility for the transport protocols. SHIM6 has multiple similarities when compared with HIP, for example the initiating handshake. At the time of writing SHIM6 did not have specification for the usage of IPv4. In our opinion, our work with cross-family handovers is beneficial also for the SHIM6, when the usage of IPv4 is specified in SHIM6.

Jokela et al. [22] first discussed about cross-family handovers in HIP but showed no performance or implementation evaluation. Their primary environment was FreeBSD, while we have implemented cross-family handovers for the Linux networking stack. Furthermore, we specifically focus on the fault tolerance aspects of handovers rather than load balancing.

III. Cross-family IPv4/IPv6 Handovers

A. Scope of HIP Handovers

In this paper, a handover refers to a change in the locator set of an end-host. When the locator set changes, the host can perform a handover procedure to sustain HIP and upper layer connectivity. A vertical handover describes end-host movement between different link-layer access technologies, such as WLAN and UMTS, and a horizontal handover refers to movement within the same type of access technology devices. HIP can support both vertical and horizontal handovers because it operates above link layer.

In a make-before-break handover a host obtains a new locator before it loses its current address. In a break-before-make handover, the host loses its current address before it obtains a new address. The latter results in a gap in connectivity during which host is not reachable which causes disruption to existing connections at the transport layer.

B. Cross-Family Handovers

HIP specifications [1], [3] offer a possibility to include LOCATOR parameters in the R1 and I2 packets. However, these two documents explain only the load balancing case with the preferred bit set. When the a host sets the preferred locator, its peer is forced to switch to it immediately and this complicates handling of alternative locators. We argue that a host should send its locators in the base exchange with all preferred bits unset by default.

When a host receives locators with all preferred bits unset, they should be considered as alternative addresses for the peer. The host does not have to use these locators immediately, but can use them for fault tolerance or load balancing purposes. This aids also cross-family handovers because then two communicating hosts know all the available addresses of each other.

On the Responders side the LOCATOR parameter could be placed into the R2 packet instead of the R1. The LOCATOR in the R2 packet facilitates mobile devices to serve as Responders better. For instance, a mobile node could disable an expensive link until the base exchange completes. Also, the mobile node employing precreate spools of R1 packets does not have to recreate its pools upon mobility events when the locator is absent from the R1.

Using the LOCATOR parameter in the base exchange benefits also HIP NAT traversal [23], which forbids preferred bits in NATted environments. De la Oliva et al. [24] also proposed a shceme for sending all the locators early in the communication in order to maximize the fault tolerance.

C. Peer Locator Learning

It is possible to delay the exposure of additional locators to the peer. This can occur e.g. for privacy reasons to avoid exposing of the topology of the corporation of the end-host. Alternatively, the end-host can even be unaware of some its locators in NATted environments [23] where the peers of the end-host observe the address of a NAT middlebox and not actual end-host address. In either case, a correspondent node should be able to inform about its additional locators after the base exchange without sending addional locators.

As an example, let us consider that two hosts have established the base exchange over IPv6 without additional locators. Then, one of the hosts becomes mobile and moves to an IPv4-only network. The mobile node informs correspondent node on its new location with an UPDATE. Now, the correspondent node can choose to break connectivity for privacy reasons or send an echo request from its previously unadvertised IPv4 address.

This case is not defined in the HIP mobility standard [3]. To achieve better flexibility, we propose that correspondent node should be able to send echo requests from previously unadvertised addresses and the mobile node should reply to them with echo responses.

As an example of scenario, NAT middleboxes alter source addresses of UDP encapsulated HIP packets and the end-host sending the packets may be unaware of this. As a consequence, the packet receiver learns a new address of the originating host that was not advertised in the included LOCATOR parameter.

D. Teredo Experiments

In general, basic Teredo-based connectivity was successful in our experimentation. We discovered some problems as well, for example, when the mobile node moved into an IPv6-only network and could not derive a Teredo address in the absence of an IPv4 address. The mobile node sent an UPDATE packet to the Teredo address of the correspondent node, but the local router did not know what to do with the non-routable Teredo address. In order to work, this case would have required a Teredo relay in the network of the mobile node or a global IPv6 address for the corresponding node.

Miredo, the Teredo implementation for Linux, decreased the troughput due to the tunneling overhead and unoptimized implementation. Especially in make-before-break handovers, it took 30 seconds at the maximum for the Miredo software to notice a mobility event that required changing the topology-dependent Teredo address. HIP daemon reacted instantly by sending an UPDATE packet advertising the old but unfortunately invalid Teredo address. As a summary, there is room for performance improvements in the Miredo implementation.

IV. Performance Measurements

In this section, we describe the measured impact of cross-family handovers. To avoid issues with TCP timeouts documented in detail elsewhere [25], we just measured UDP throughput.

We conducted the measurements on two identical laptops (Intel Core 2, 2 GHz CPU). We concentrated on the processing cost by minimizing the network latency (RTT 0.484 ± 0.143 ms), and therefore the laptops were connected via single Gigabit router to each other. Both machines were running Ubuntu Jaunty Jackalope Linux with 2.6.28 kernel and HIPL release 1.0.3.

We triggered handovers using *ip* from ip-tools package that allows manipulation of the network interfaces. In the test cases the CN sent UDP packets continuously to the MN. We used the Wireshark to capture the traffic and to analyze the gathered data. The handover is measured to start from the sending of the first UPDATE control packet with LOCATOR parameter (step 2 in Figure 1) and to stop when the first ESP is received using the new address (step 8 in Figure 1).

Tables I and II show that cross-family Make Before Break (MBB) and Break Before Make (BBM) handovers tend to last 8 milliseconds longer than handovers where the family does not change.

TABLE I
Durations of intra-family handovers.

Direction	Duration, ms
MBB IPv4 to IPv4	53 ± 12
MBB IPv6 to IPv6	56 ± 6
BBM IPv4 to IPv4	41 ± 12
BBM IPv6 to IPv6	40 ± 6
Total average	47 ± 10

TABLE II
Durations of cross-family handovers.

Direction	Duration, ms
MBB IPv4 to IPv6	56 ± 6
MBB IPv6 to IPv4	53 ± 16
BBM IPv4 to IPv6	56 ± 8
BBM IPv6 to IPv4	54 ± 11
Total average	55 ± 11

We observed that it took 10 ms (± 1) from sending the UPDATE control packet with LOCATOR parameter and receiving of the UPDATE control packet with ECHO_REQUEST (steps 2 - 4 in Figure 1). Handling of the ECHO_REQUEST and creation of needed SAs took 19 ms (± 5) in intra-family handovers and 40 ms (± 8) in cross-family handovers (step 5 in Figure 1). It took 6 ms (± 2) from sending of the ECHO_RESPONSE to the receiving of the first ESP packet (steps 6-8 in Figure 1). Most of the processing time was spent in processing of the UPDATE control packet with ECHO_REQUEST parameter, as Pääkkönen et al. [26] have also observed. We suspect that the processing time is doubled in cross-family handovers due to unoptimized code.

V. Conclusions

Cross-family handovers can be used as a transition mechanism towards IPv6 now that IPv4 address space is almost depleted. In this paper, we have shown three key contributions. 1) We described a shortcoming in current HIP mobility specifications preventing cross-family handovers and suggested a simple solution to it. 2) Our performance evaluation on our implementation indicates that HIP-based cross-family handovers perform as well as intra-family handovers. 3) Our approach is compatible with NATted networks because it can make use of Teredo-based end-to-end tunnels.

Acknowledgments

The authors thank Jan Melen, Ari Keränen, Philip Matthews, Joakim Koskela, Toni Ruottu, Tobias Heer, Stefan Götz, and Barış Boyvat for helpful discussions.

References

[1] R. Moskowitz, P. Nikander, P. Jokela, and T. R. Henderson, "RFC 5201: Host Identity Protocol," Apr. 2008.

[2] A. Khurri, E. Vorobyeva, and A. Gurtov, "Performance of Host Identity Protocol on lightweight hardware," in *MobiArch '07: Proceedings of the 2nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*. New York, NY, USA: ACM, Aug. 2007, pp. 1–8.

[3] P. Nikander, T. Henderson, C. Vogt, and J. Arkko, "RFC 5206: End-Host Mobility and Multihoming with the Host Identity Protocol," Apr. 2008.

[4] J. Ylitalo, J. Melén, P. Nikander, and V. Torvinen, "Re-thinking Security in IP-Based Micro Mobility," in *Lecture Notes in Computer Science*, 2004, pp. 318 – 329, iSBN 978-3-540-23208-7.

[5] S. Novaczki, L. Bokor, and S. Imre, "Micromobility support in HIP: survey and extension of host identity protocol," in *Electrotechnical Conference. MELECON 2006. IEEE Mediterranean*, May 2006, pp. 651 – 654.

[6] P. Jokela, T. Rinta-Aho, T. Jokikyyny, J. Wall, M. Kuparinen, J. Melén, T. Kauppinen, and J. Korhonen, "Handover performance with HIP and MIPv6," in *1st International Symposium on Wireless Communication Systems*, 2004, pp. 324 – 328.

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

[8] R. Moskowitz and P. Nikander, "RFC 4423: Host Identity Protocol (HIP) Architecture," Apr. 2006.

[9] P. Nikander, J. Laganier, and F. Dupont, "RFC 4843: An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)," Apr. 2007.

[10] P. Nikander and J. Laganier, "RFC 5205: Host Identity Protocol (HIP) Domain Name System (DNS) Extension," Apr. 2008.

[11] J. Ahrenholz, "HIP DHT Interface: draft-ahrenholz-hiprg-dht-04," Mar. 2009, work in progress. Expires in Sep, 2009.

[12] P. Jokela, R. Moskowitz, and P. Nikander, "RFC 5202: Using ESP Transport format with HIP," Apr. 2008.

[13] C. Perkins and et al, "RFC 3344: IP Mobility Support for IPv4," Aug. 2002.

[14] J. Manner and M. Kojo, "RFC 3753: Mobility Related Terminology," Jun. 2004.

[15] D. Johnson, C. Perkins, and J. Arkko, "RFC 3775: Mobility Support in IPv6," Jun. 2004.

[16] V. Devarapalli and P. Eronen, "RFC 5266: Secure Connectivity and Mobility Using Mobile IPv4 and IKEv2 Mobility and Multihoming (MOBIKE)," Jun. 2008.

[17] P. Eronen, "RFC 4555: IKEv2 Mobility and Multihoming Protocol (MOBIKE)," Jun. 2006.

[18] G. Tsirtsis and H. Soliman, "RFC 4977: Problem Statement: Dual Stack Mobility," Aug. 2007.

[19] G. Tsirtsis, V. Park, and H. Soliman, "RFC 5454: Dual Stack Mobile IPv4," Mar. 2009.

[20] C. Huitema, "RFC 4380: Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," Feb. 2006.

[21] E. Nordmark and M. Bagnulo, "RFC 5533: Shim6: Level 3 Multihoming Shim Protocol for IPv6," Jun. 2009.

[22] P. Jokela, P. Nikander, J. Melen, J. Ylitalo, and J. Wall, "Host Identity Protocol: Achieving IPv4 - IPv6 handovers without tunneling," in *Proceedings of Evolute workshop 2003: Beyond 3G Evolution of Systems and Services*, University of Surrey, Guildford, UK, Nov 2003.

[23] M. Komu, T. Henderson, P. Matthews, H. Tschofenig, and A. Keränen, "Basic HIP Extensions for Traversal of Network Address Translators: draft-ietf-hip-nat-traversal-09.txt," Jun. 2009, work in progress. Expires in Dec, 2009.

[24] A. de la Oliva and M. Bagnulo, "Fault tolerance configurations for HIP multihoming: draft-oliva-hiprg-reap4hip-00," Jul. 2007, work in progress. Expires in Jan, 2008.

[25] S. Shütz, L. Eggert, S. Schmid, and M. Brunner, "Protocol enhancements for intermittenly connected hosts," in *ACM SIGCOMM Computer Communication Review*, Jul. 2005, pp. 5 – 18.

[26] P. Paakkonen, P. Salmela, R. Aguero, and J. Choque, "Performance analysis of HIP-based mobility and triggering," in *Proceedings of International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2008. WoWMoM 2008*, Newport Beach, CA, Jun 2008.