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

Samu Varjonen, Miika Komu, and Andrei Gurtov

***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 IPv4 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

While IPv6 address space is drastically larger than for IPv4, IPv6 has not experienced a wide-scale deployment yet. A “flag day” is not practically feasible and therefore the protocols have a co-exist for a long time. The concurrent use of both addressing families causes 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 may enforce new networks to employ only IPv6 adddresses. 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 with protocols specifically designed 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 network connection. Even though cellular networks support mobility transparently, the same does not apply globally to WLAN-based mobility.

Manuscript received December 8, 2009; revised February 6, 2010. The material in this paper was presented in part at the 17th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2009), Split-Hvar-Korčula, Croatia, Sept. 2009.

Authors are with the Helsinki Institute for Information Technology, Helsinki, Finland ({Samu.Varjonen, Miika.Komu, Andrei.Gurtov}@hiit.fi).

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 address changes.

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 translates the 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 specification [3] describes end-host mobility and multihoming but leaves handovers across IP families 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 Distributed Hash Table [11]. When the application uses a HIT or a LSI to establish new outgoing communications, IPsec layer intercepts the packet and ask HIP layer to trigger base exchange (BEX) to set up symmetric keys for the IPsec tunnel.

HIP can be deployed in two ways, non-opportunistic and opportunistic. In non-opportunistic way HIP needs infrastructure to map names to the identifiers and identifiers to routable IP addresses (locators). HIP specifies DNS extensions [10] for this purpose and a service using the OpenDHT [11] for home users who cannot publish their identifiers on DNS servers. In the opportunistic mode, the Initiator of the communications makes a leap of faith and tries to initiate communication solely using a locator without knowing Responder's identifier beforehand [12].

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 [13]. 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 (Figure 1 illustrates a base exchange). First, the Initiator starts the base exchange with an I1 packet (step 1 in Fig. 1). Upon receiving of the I1, the responder selects a precomputed R1 containing a computational puzzle (step 2 in Fig. 1). Second, the Responder replies with its public key and Diffie-Hellman key material in an R1 (step 3 in Fig. 1). After receiving of the R1, the Initiator checks the validity of the packet and solves the computation puzzle in the received R1 (steps 4 and 5 in Fig. 1). Third, the Initiator responds with an I2 packet that contains its public key and Diffie-Hellman key material (step 6 and 7 in Fig. 1). Fourth, the Responder concludes the BEX with an R2 packet if the solution in the received I2 control packet was correct (step 8, 9, and 10 in Fig. 1. 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. Based on the exchanged keying material, the end-hosts create IPsec security associations (steps 11 and 12 in Fig. 1).

After the BEX is completed successfully and both end-hosts have reached ESTABLISHED state, the two end-hosts can commence to send upper-layer traffic to each other over the encrypted ESP tunnel. From here on, state created during the BEX is called a HIP Association (HA).

After the BEX, the end-hosts lose their roles as Initiator and Responder because there is no need for such separation.

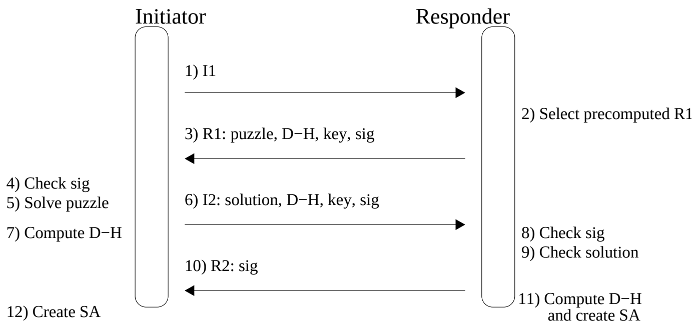

Fig. 1. HIP Base Exchange.

Now, the end-hosts can process mobility related packets which requires a 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 [14], [15] 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 a bit misleading because HIP architecture allows both hosts to move simultaneously [16]. 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 a random nonce (echo request) to the mobile node. The mobile node responds with a packet containing the same nonce (echo response). This procedure allows the correspondent node 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. 2 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.

When the mobile node moves (in step 1, Figure 2), its locators change 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 its local policies. For example, mobile node might not advertise expensive links for all correspondent nodes, or it might exclude some locators for privacy reasons. The 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 2). In step 2, the mobile node sends an initial UPDATE with the LOCATOR parameter that lists

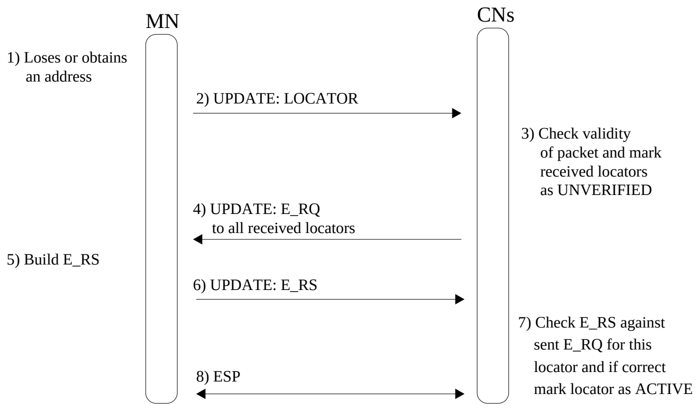
MN
CNs
1) Loses or obtains an address
2) UPDATE: LOCATOR
3) Check validity of packet and mark received locators as UNVERIFIED
4) UPDATE: E_RQ to all received locators
5) Build E_RS
6) UPDATE: E_RS
7) Check E_RS against sent E_RQ for this locator and if correct mark locator as ACTIVE
8) ESP

Fig. 2. Return routability tests and locator state.

the new locator set which the mobile node publishes to all of 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 2). 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 2). Next, the correspondent node builds an UPDATE packet containing an ECHO_REQUEST parameter (E_RQ in Figure 2) 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 the correspondent node in step 6. The correspondent node receives the UPDATE packet and validates its integrity and nonce in step 7. The correspondent node transitions now the state of the peer locator to ACTIVE. If the mobile node failed to respond within a certain time, the correspondent node would deprecate the locator and remove 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 a specific locator for HIP-related communications. For example, this can be used to employ load balancing for HIP.

C. Related Work

In Mobile IP [14], [17], 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 allows two end-hosts of different families to route MIPv6-related traffic directly between them without such triangular routing through the home agent. IPsec and MOBIKE [18] [19] can be used to protect Mobile IP traffic.

The MOBIKE protocol offers mobility functionality similar as in 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 [20].

A MIPv4 extension [21] 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 [22]. Teredo employs 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 [23] is a layer 3 multihoming protocol that offers locator agility for the transport protocols. SHIM6 has multiple similarities with HIP. For example, the protocol encodings are identical and the initial handshake are similar. 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 standardized for SHIM6.

Jokela et al. [24] 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 end-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. The focus of this paper is on end-to-end handovers even though HIP facilitates also end-to-middle operation using a HIP proxy [25].

In a Make-Before-Break (MBB) handover an end-host obtains a new locator before it loses its current address. In a Break-Before-Make (BBM) handover, the end-host loses its current address before it obtains a new address. The latter results in a gap in connectivity during which the end-host is not reachable, which further throttles 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 a host sets the preferred locator, its peer is forced to switch to it immediately. We argue that the preferred bit complicates handling of alternative locators and a host should prefer sending its locators in the base exchange with all preferred bits unset.

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 the purposes of fault tolerance or load balancing. This aids also cross-family handovers because then two communicating hosts know all the available addresses of each other.

At the Responder 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, this is beneficial for a mobile node employing precreated pools of R1 packets. As the R1 signature covers the Responder's IP address, it does not have to recreate its pools upon address changes.

Using the LOCATOR parameter in the base exchange benefits also HIP NAT traversal [26] which forbids preferred bits in NATted environments and assumes the LOCATOR to be placed in the R2 packet. De la Oliva et al. [27] also proposed a scheme for sending all the locators early in the communication in order to improve fault tolerance.

C. Peer Locator Learning

A host can delay or decline altogether to advertise its additional locators to its peers to e.g. avoid exposing of sensitive topology information. Alternatively, the end-host can even be unaware of some its locators in NATted environments [26] where the peers of the end-host observe the address of a NAT middlebox and not the 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 additional locators. As an example, let us consider that two hosts have established a base exchange over IPv6 without exchanging additional locators. Then, one of the hosts becomes mobile and moves to an IPv4-only network. The mobile node informs its correspondent node about 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.

The use of unadvertized addresses is not defined in the HIP mobility specification [3]. To achieve better flexibility, we propose that correspondent node should be able to send echo requests from previously unadvertised addresses. Also, the mobile node should reply to them with echo responses. We refer this as *peer locator learning*.

As a second 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 recipient learns a new address of the originating host that was not advertised in the included LOCATOR parameter.

Peer locator learning is further depicted in Figure 3. In step 1, the MN changes its attachment point to the network and obtains one IPv4 address and one IPv6 address. In step 2 MN sends the new locator set to its CN. Upon receiving the locator set, the CN starts the return routability tests and sends one echo request to the IPv4 address and one echo request to the IPv6 address. When the MN receives the echo request from its IPv4 address, it checks the locator lists it has for the active CNs. The MN notices that the locator is already known and sends an echo response to the CN (see Fig. 3 step 3). Upon receiving the echo request from its IPv6 locator, the MN checks its locator lists for the active CNs and does not find the used locator (see Fig. 3 step 4). Finally, MN adds this locator to the list and starts connectivity tests for the locator, and sends an echo response to the CN (step 5 in Fig. 3).

Peer locator learning is also beneficial in cases such as simultaneous end-host mobility. In this case both end-hosts move simultaneously and lose connectivity due to the fact that the end-hosts do not know where to send the UPDATE control packet. This is generally solved with third party rendezvous service as described by Hobaya et al. [16].

In their paper, Hobaya et al. also describe a problem where the simultaneous UPDATE procedures of the end-hosts are interleaved and renders the security associations in an asymmetric state. Hobaya et al. provide a solution for this problem by enforcing UPDATE retransmissions. In their use scenarios, only one of the end-hosts is able to send ESP traffic to the other but not vice versa because the other end-host mistakenly cleared its retransmission buffers. In contrast, with peer locator learning, the end-host unable to the send ESP data can find a new locator from the IP header of the received UPDATE control packet. After the end-host has discovered the new address, it could trigger the return routability tests and, as a result, both of the nodes could continue communicating. We believe that our peer locator learning technique would result in faster handovers than UPDATE retransmission as proposed by Hobaya et al.

D. Teredo Experiments

We wanted to validate that our implementation works in the presence of NATs with Teredo. 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. Another problem was that the mobile node sent an UPDATE packet to the Teredo address of the correspondent node, but the local router could not route 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 throughput 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

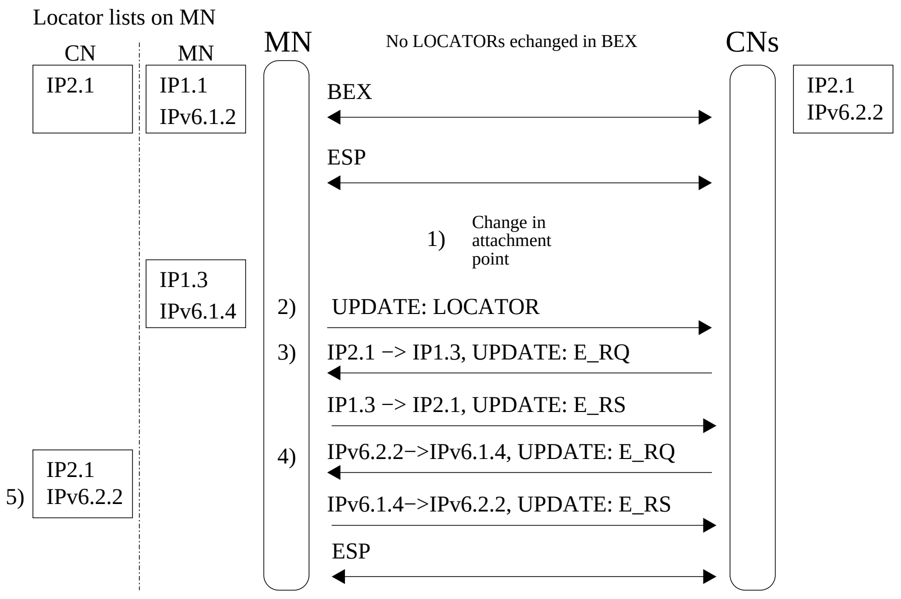
Locator lists on MN
CN
MN
IP2.1
IP1.1
IPv6.1.2
MN
No LOCATORs echanged in BEX
CNs
IP2.1
IPv6.2.2
BEX
ESP
1) Change in attachment point
IP1.3
IPv6.1.4
2) UPDATE: LOCATOR
3) IP2.1 -> IP1.3, UPDATE: E_RQ
IP1.3 -> IP2.1, UPDATE: E_RS
4) IPv6.2.2->IPv6.1.4, UPDATE: E_RQ
IP2.1
5) IPv6.2.2
IPv6.1.4->IPv6.2.2, UPDATE: E_RS
ESP

Fig. 3. Peer locator learning example case, where the end-hosts exchange no LOCATORs in the BEX and the MN learns the IPv6 address of its CN from the return routability tests

sending an UPDATE packet advertising the old but, unfortunately, still invalid Teredo address. As a summary, there is room for performance improvements in the Miredo implementation.

E. Implementation of Cross-Family Handovers

This section discusses some of the issues we faced when implementing cross-family handovers with HIP. We chose the *HIP for Linux* (HIPL) implementation [28] as our experimentation tool. Most of the changes involved only the UPDATE packet parameters. Only minor changes were required in the processing of R1 and I2 packets.

To make cross-family handovers possible, we implemented a new function to uniformly build LOCATOR parameters containing all the locators of the local host. Modifications also included introducing of LOCATOR parameter to R1 and I2 control packets.

The challenges we faced ranged from trivial to more complicated. A trivial problem was that base exchanges with locators triggered return routability tests before the state was ESTABLISHED on both sides. As a solution, we had to delay the triggering of address verifications (ECHO_REQUESTS) to avoid the state machine to reject the early return routability tests. A trickier problem originated from the sockets API that has separate raw sockets for IPv4 and IPv6. The HIP software needs the raw sockets are needed for sending and receiving of HIP control packets. We experienced a problem where one of the sockets included a base exchange packet and the other an UPDATE packet. We had to change the implementation to handle the packets in the correct order. This was just an optimization to the handovers because the problem could also be solved by just dropping the UPDATE packet and relying on retransmissions of the mobile node.

Performance measurements described in the Section IV revealed features in HIPL that were too aggressive in their behavior and did not conform to the specifications.

First, the MN triggered the first UPDATE control packet immediately after obtaining a new address. This resulted in an ICMP message informing the MN that the CN is unreachable because the network interface was not fully initialized. As a result from the destination unreachable error, the implementation queued the UPDATE control packet to the retransmission queue for ten seconds. It would have been beneficial to start with a low interval for the retransmission and increase it exponentially as described in the specification [1]. This is also discussed by Shütz et al. [29].

Second, solely relying to the address notifications from the kernel, i.e. netlink events, as the only indication for a handover, and, reacting too aggressively to the events, resulted in excess UPDATE control packets. This situation is depicted in Figure 4. For example, in a case where the MN has two addresses, reacting instantly on netlink messages would unecessarily result in multiple UPDATE control packets. When the interface goes down, the addresses are deleted one by one and this results in two UPDATE control packets. One contains a LOCATOR parameter with one locator (see Fig. 4 step 1) and the second contains zero addresses (see Fig. 4 step 2), for which there is no guarantees that it can be event sent. As a response to this, the CN tests at least the origin of the UPDATE packet for return routability. When the CN obtains the new attachment point and its interfaces are brought up, the kernel informs about the new addresses (two in the example), the kernel informs about the addresses one at a time that will result in two more UPDATE control packets (see Fig. 4 steps 3 and 4). The CN will unoptimally trigger at least three more return routability tests despite that the address used to send the previous control packet (UPDATE) would not have to be tested again.

To sum up, the previous case leads to MN unoptimally sending three update packets to the CN which triggers four

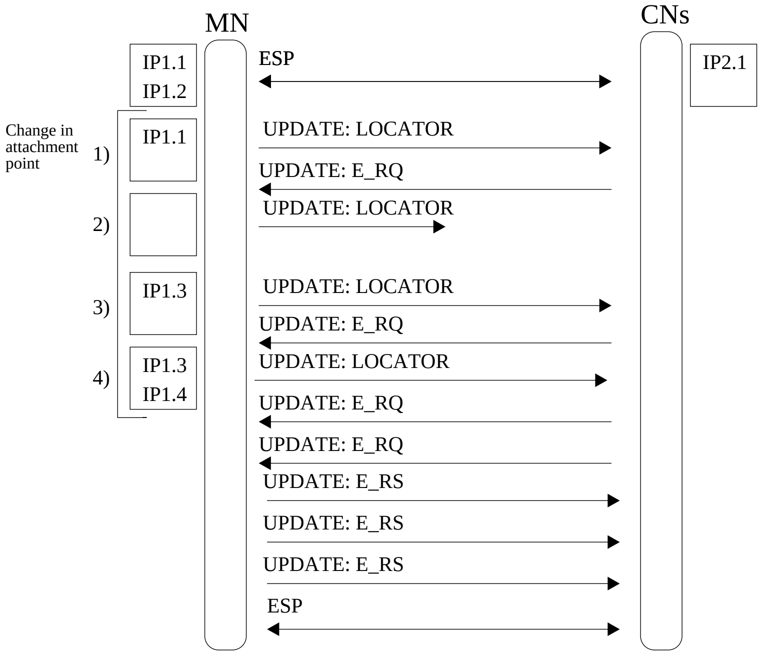

Fig. 4. Results of triggering the handover too fast after a change in the addresses on an interface.

return routability tests. However, it would suffice for the MN to send a single UPDATE control packet and the CN to trigger return routability tests for the addresses found in the locator set contained in the UPDATE packet. As a straw-man solution for this, we delayed the handover so that all the conseqtutive netlink events could be handled as a single event. However, even this is not the most optimal solution because it increases the latency of the handover in overall, while in some cases, such as the second case described in this section reacting to the netlink messages decreases the handover latency. Further, triggering of handovers should be triggered also monitoring of the end-to-end connectivity as we describe later in this section.

Figure 5 depicts the growth of the TCP sequence number during a BBM handover. The base exchange is concluded at time T1. MN loses network connectivity at T2 and regains it at T3. At point of time T4 the interfaces are fully operational and MN triggers the HIP update procedure. We observed that, after the update procedure at T4, there is some extra latency before the MN sent the first IPSec ESP packet at T5.

This latency varied so that it may even out the difference in the overall latency presented in Section IV in Tables I and II. This conforms to the findings of Shütz et al. [29] where they found a similar period of inactivity after a period of disconnectivity. According to them, TCP waits for the current retransmission timeout to expire while the new address is obtained or the connectivity is otherwise restored before TCP tries to retransmit. Shütz et al. [29] suggest an improvement to this situation. Their solution also tries to minimize the period of inactivity by introducing a more aggressive way to enforce the retransmissions after an end-host receives or sends the last echo response in the update procedure. In our opinion, this feature is a welcome improvement to decrease the handover latency.

To improve the chances that transport layer survives connectivity loss automatically, we implemented a heartbeat probe. The heartbeat is used to monitor the connectivity between the hosts. The heartbeat is essentially an ICMPv6 messages inside the ESP tunnel between two end-hosts. As a naive approximation, the implementation triggers the update procedure after *n* consecutive heartbeats are lost. Care has to be taken to avoid choosing a too long interval for the heartbeat to avoid TCP aborting the connection. Also, intermediary hosts, such as NAT boxes, may time out an idle ESP tunnel when the heartbeat interval is too long.

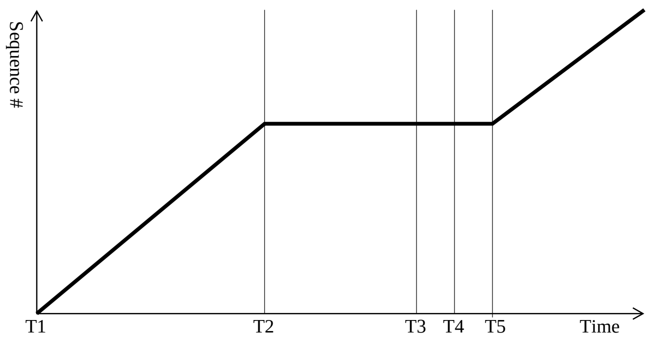

Fig. 5. Sequence number generation during a BBM handover.

For the heartbeats to work, the end-host need to discover more than one address of its peer. An advantage in using ICMPv6 packets to implement the heartbeat is the fact that the heartbeat mechanism must be supported only on the end-host using the heartbeat. The end-host on the other side of the tunnel does not have to trigger heartbeats at all. It merely has to support replying to ICMPv6 messages inside the tunnel.

The implementation currently sends the heartbeat on regular intervals. It could be optimized to send only when the IPsec tunnel is idle. However, sending heartbeats all the time and gathering the monitoring results from them is a better choice for multihoming cases as explained by Gurtov et al. [30].

IV. Performance Measurements

In this section, we show the performance of cross-family handovers based on our implementation. To avoid issues with TCP timeouts documented in detail elsewhere [29], we primarily 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* command 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 Wireshark to capture the traffic at the MN and to analyze the gathered data. The handover was measured to begin from MN sending of the first UPDATE control packet with LOCATOR parameter (step 2 in Figure 2) and to cease when the MN received the first ESP packet on the new address (step 8 in Figure 2).

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 remains unchanged.

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 a delay of 10 ms (± 1) from MN sending the UPDATE control packet with LOCATOR parameter and receiving the UPDATE control packet with ECHO_REQUEST (steps 2 - 4 in Figure 2). Handling of the ECHO_REQUEST and creation of the related SAs took 19 ms (± 5) in intra-family handovers and 40 ms (± 8) in cross-family handovers (step 5 in Figure 2) at the MN. The delay between MN sending the ECHO_RESPONSE and receiving the first ESP packet (steps 6-8 in Figure 2) was 6 ms (± 2). Most of the processing time was spent in processing of the UPDATE control packet with ECHO_REQUEST parameter as Pääkkönen et al. [31] have also observed. We suspect that the processing time was doubled in cross-family handovers due to unoptimized code.

In Figures 6 and 7, we depict the difference of moving from IPv4 to IPv4 and from IPv4 to IPv6 in the BBM handover case. The mobile node changes its network point of attachment and obtains a new locator (see Fig. 6 step 1 and Fig. 7 step 1). The major difference is the use of ARP messaging in IPv4 (see Fig. 6 steps 2 and 4) versus ICMPv6 neighbor discovery (see Fig. 7 step 2) in IPv6.

Operationally, ARP and ICMPv6 neighbor discovery do not differ much. In ARP, a end-host broadcasts a “Who has” message to the network. The message contains the target IP address. The end-host possessing the address responds informing that the queried IP address is located at the link-local address of the end-host. In IPv6 neighbor discovery, the end-host first announces to the nearest router that it listens to IPv6 multicasts and excludes its own address from the multicasts it wants to receive. This way the IPv6 router discovers the presence of multicast listeners on its directly attached links [32]. Then, the end-host broadcasts a neighbor solicitation message asking who has the target IPv6 address. The end-host possessing the IPv6 address answers with a neighbor advertisement containing the link-local address of the end-host.

In intra-family case with IPv4, the peer locator was discovered after the UPDATE procedure (see Fig. 6 step 3). Intra-family handover with IPv6 or cross-family handover towards IPv6 did not incur neighbor discovery after the UPDATE procedure (see Fig. 7 step 3). We observed that in cross-family handovers from IPv4 to IPv6, the interface kept broadcasting ARP queries on the previously used IPv4 addresses while MN did not have a working IPv4 address anymore. In our tests, this behavior resulted in circa five messages (see Fig. 7 step 4), after which the ESP traffic started flowing again (see Fig. 7 step 5).

After the handover, TCP had to retransmit some of the data because some of it was received out-of-order. In overall, the amount of TCP retransmits did not differ in intra-family or cross-family handovers. The similar amounts of retransmitted TCP segments, in the handovers, can be also explained. TCP handles the retransmissions inside the ESP tunnel. TCP is not affected by the change of the address family, i.e., TCP is connected to a HIT and the handover is transparent to TCP.

We observed the Credit Based Authorization (CBA, see [3] Section 3.3.1) operational in some intra-family cases. CBA allows ESP traffic to commence before the completion of return routability tests. We noticed that the CBA was too aggressive and resulted in ICMP reroute messages. We also performed the same tests with UDP which did not show any significant difference in the amount of lost packets.

V. Conclusions and Future Work

Cross-family handovers can be used as an IPv6 transition mechanism now that the 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

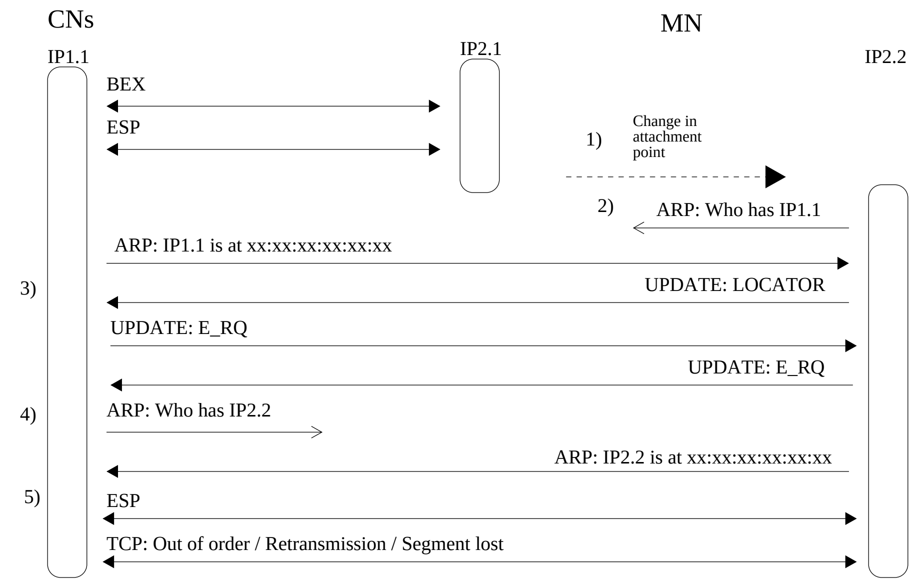
CNs
MN
IP1.1
IP2.1
IP2.2
BEX
ESP
1) Change in attachment point
2) ARP: Who has IP1.1
ARP: IP1.1 is at xx:xx:xx:xx:xx:xx
3)
UPDATE: LOCATOR
UPDATE: E_RQ
UPDATE: E_RQ
4) ARP: Who has IP2.2
ARP: IP2.2 is at xx:xx:xx:xx:xx:xx
5) ESP
TCP: Out of order / Retransmission / Segment lost

Fig. 6. An intra-family Break Before Make handover from IPv4 to IPv4 with ARP traffic before and after the handover.

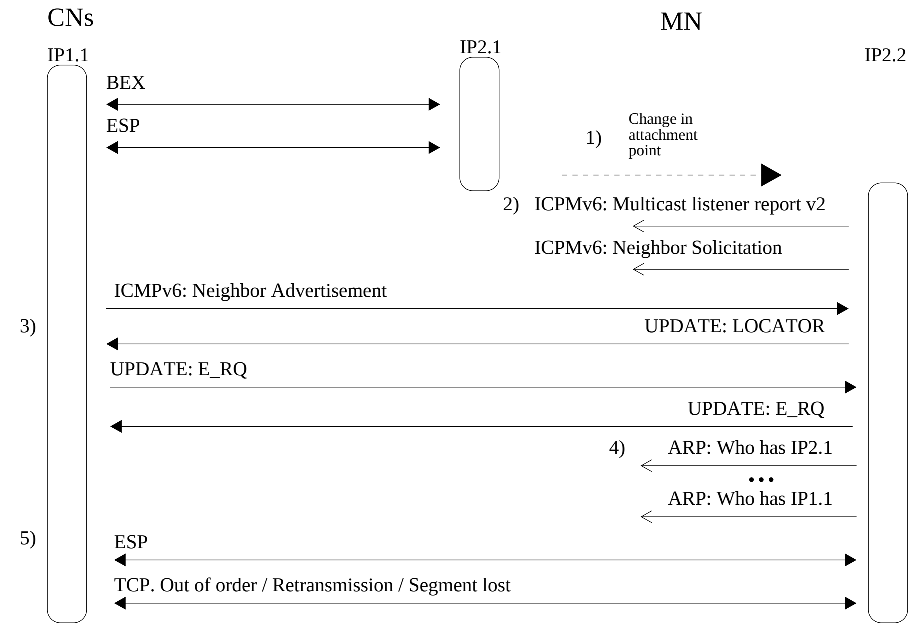
CNs
MN
IP1.1
IP2.1
IP2.2
BEX
ESP
1) Change in attachment point
2) ICPMv6: Multicast listener report v2
ICPMv6: Neighbor Solicitation
ICMPv6: Neighbor Advertisement
3)
UPDATE: LOCATOR
UPDATE: E_RQ
UPDATE: E_RQ
4) ARP: Who has IP2.1
...
ARP: Who has IP1.1
5) ESP
TCP. Out of order / Retransmission / Segment lost

Fig. 7. Cross-family Break Before Make handover from IPv4 to IPv6 with the ICMPv6 neighbor discovery traffic and with ARP messages for the lost IPv4 connectivity.

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.

As future work we intend to research solutions for triggering the handovers and find an optimal solution for the triggers. As a result of our findings, the HIPL implementation has a new mobility architecture which we plan to measure and evaluate further.

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] M. Komu and J. Lindqvist, "Leap-of-Faith Security is Enough for IP Mobility," in *Proceedings of the 6th Annual IEEE Consumer Communications and Networking Conference IEEE CCNC 2009*, Las Vegas, NV, Jan 2009.

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

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

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

[16] F. Hobaya, V. Gay, and E. Robert, "Host Identity Protocol extension supporting end-host simultaneous mobility," in *Proceedings of 2009 Fifth International Conference on Wireless and Mobile Communications*, 2009, pp. 261–266.

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

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

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

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

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

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

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

[24] 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.

[25] P. Salmela and J. Melen, "Host Identity Protocol Proxy," in *Communications in Computer and Information Science*, nov 2007, pp. 126 – 138.

[26] M. Komu, T. Henderson, P. Matthews, H. Tschofenig, and A. Keränen, "RFC5770: Basic HIP Extensions for Traversal of Network Address Translators," Feb. 2010.

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

[28] A. Pathak, M. Komu, and A. Gurtov, "HIPL: Give a name to your linux box," Nov. 2009.

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

[30] A. Gurtov and T. Polishchuk, "Secure Multipath Transport for Legacy Internet Applications," in *Proc. of BROADNETS'09*, sep 2009.

[31] 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.

[32] S. Deering, W. Fenner, and B.Haberman, "RFC 2710: Multicast Listener Report (mld) for IPv6," Oct. 1999.

Samu Varjonen received his M.Sc (2006) degree in Computer Science from the University of Helsinki, Finland. He is a PhD student at University of Helsinki. At the present he is working as a researcher in the Networking Research group at the Helsinki Institute for Information Technology focusing on the Host Identity Protocol and trustworthy communication in distributed systems. He has also taken part in the IETF standardization work of Host Identity Protocol.

Miika Komu is a postgraduate student at the Helsinki University of Technology (renamed as Aalto University during 2010). He works as a researcher for the same university at the Data Communications Software research group in the Department of Computer Science and Engineering of the Faculty of Information and Natural Sciences, Finland and for Helsinki Institute for Information Technology. He is involved in IETF standardization and is a developer in the HIP for Linux (HIPL) project.

Andrei Gurtov Andrei Gurtov received his M.Sc (2000) and Ph.D. (2004) degrees in Computer Science from the University of Helsinki, Finland. At the present, he is Principal Scientist 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 Helsinki University of Technology 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. Dr. Gurtov is a co-author of over 80 publications including a book, research papers, patents, and IETF RFCs. URL: http://www.hiit.fi/ gurtov.