HI3: AN EFFICIENT AND SECURE NETWORKING ARCHITECTURE FOR MOBILE HOSTS

Andrei Gurtov, Dmitry Korzun, Pekka Nikander

June, 2005

HIIT
TECHNICAL
REPORT
2005-2

Hi3: An Efficient and Secure Networking Architecture for Mobile Hosts

Andrei Gurtov, Dmitry Korzun, Pekka Nikander

HIIT Technical Reports 2005-2
ISSN 1458-9478

Hi^3: An Efficient and Secure Networking Architecture for Mobile Hosts

Andrei Gurtov, Dmitry Korzun, Pekka Nikander
HIIT

Abstract

The Host Identity Indirection Infrastructure (Hi^3) is a networking architecture for mobile hosts, derived from the Internet Indirection Infrastructure (i^3) and the Host Identity Protocol (HIP). Hi^3 has efficient support for secure mobility and multihoming, which both are crucial for future Internet applications. In this paper, we describe and analyze Hi^3 in detail. Compared to Mobile IP, Hi^3 achieves better resilience, scalability and security. Considering all capabilities Hi^3 provides, its implementation is much simpler than a corresponding implementation based on existing or proposed IETF standards. Both our analysis and early measurements support the notion that Hi^3 performance is on the same level with Mobile IP.

1 Introduction

The original Internet Protocol (IP) stack was designed without explicit consideration for address agility[1] or IP-layer security. As argued elsewhere (e.g., [6]), the current standards for adding mobility and security to the IP stack, i.e., Mobile IP and IPsec, at best represent independent *point solutions* that do not integrate easily and sometimes interact badly [3].

In this paper, we enhance the Host Identity Indirection Infrastructure (Hi^3), introduced by Nikander *et al.* [24], analyze its performance and scalability, and provide early measurement results. Compared to Mobile IP, we are able to reach roughly equal performance with much improved resilience, scalability, and security properties. Compared to the Internet Indirection Infrastructure (i^3) [31] and the Host Identity Protocol (HIP) [20], upon which Hi^3 is based, Hi^3 preserves the best of both approaches while greatly improving performance compared to i^3 and enhancing flexibility and security compared to HIP.

In particular, we argue that an overlay infrastructure such as i^3 is ideal for providing a secure, integrated *rendezvous infrastructure* for HIP [20], basically forming a secure "control plane" for the Internet. For performance reasons, the actual "data plane" traffic should still be carried directly end-to-end, without involving the overlay.

The main benefits of Hi^3 can be summarized as follows:

- Inheriting from HIP, Hi^3 *integrates* mobility with end-host-based multi-address multi-homing and basic security mechanisms. It also makes IPv4/IPv6 integration easy, including mobility and multi-homing across IPv4 and IPv6 [16, 23, 39].
- To our knowledge, the system provides better protection against Denial-of-Service (DoS) attacks than any other comparable system.
- Due to its inherently decentralized nature, Hi^3 is very robust, with no single points of failure.
- The system is designed to facilitate separation of control and data packets into different "planes", thereby making it easier to build system architectures, where the control and data traffic flow different paths due to security, manageability, or other reasons.
- The overall system performance is comparable to Mobile IP, i.e., clearly better than the performance of systems based on pure overlay routing.

Hi^3 can be deployed in a piecewise manner without any flag days. Furthermore, all the perceived deployment steps give some benefits, providing motivation for people and organizations to perform the required upgrades.

The rest of the paper is organized as follows. In Section 2, background material on HIP, i^3/Secure-i^3, and IPsec-aware NAT is presented. In Section 3, we describe the Hi^3 network architecture in detail. In Section 4, we outline the use scenarios for Hi^3, and in Section 5 analyze them. In Section 6, our implementation experience is described and measurement results are presented. Section 7 concludes the paper.

2 Background

Hi^3 is based on ideas from i^3, Secure-i^3, and HIP. Furthermore, for protecting the data traffic, Hi^3 uses the IPsec-aware NAT, SPINAT. This section gives the necessary background on the technologies mentioned above.

2.1 Host Identity Protocol (HIP)

In HIP [21, 20], IP addresses are used to address and route packets just as today. Only in the upper parts of the stack the addresses are replaced with the host identifiers. These host identifiers form a new Internet-wide name space, the host identity

[1] Mobility was considered as early as 1970 [28] but it was later decided to leave it out from the architecture.

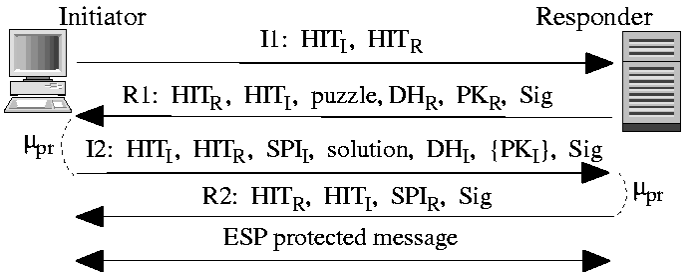


Figure 1: HIP base exchange.

name space. The identifiers in this name space are public cryptographic keys. With HIP, each host is directly identified with one or more public keys that each corresponds to a private key possessed by the host. Each host generates one or more public/private key pairs to provide identities for itself[2]. A host can prove that it corresponds to the identity by signing data with its private key. All other parties use the host identifier, i.e., the public key, to identify and authenticate the host.

Typically, a host identifier is represented by a 128-bit long identifier, the Host Identity Tag (HIT). A HIT is constructed by applying a cryptographic hash function over the public key. The purpose and function of HITs is similar to i^3 identifiers used in triggers (see Section 2.2), but they are constructed entirely cryptographically.

The introduction of new end-point identifiers changes the role of IP addresses. When HIP is used, IP addresses become pure topological labels, naming locations in the Internet. An end-point can change its IP address without breaking connections. Thus, the relationship between location names and identifiers becomes dynamic.

The actual HIP protocol [21] consists of a two-round-trip, end-to-end Diffie-Hellman key exchange protocol (called the *HIP base exchange*), a mobility exchange, and some additional messages. The purpose of the HIP base exchange is to create assurance that the peers indeed possess the private key corresponding their host identifiers (see Figure 1). Additionally, the exchange creates a pair of Encapsulated Security Payload (ESP) security associations (SAs), one in each direction. The base exchange requires cryptography processing for R1/I2 (solving the puzzle) and I2/R2 (checking the puzzle solution and authenticating the initiator) at initiator's and responder's sides, respectively. The delay is represented as the processing time $\mu_{\rm pr}$ in Figure 1; see also Section 5.

Once the HIP base exchange has been completed and the security associations are in place, the end-points can inform their peers about the additional IP addresses assigned to them [23], and update this information as needed. For initial rendezvous, simultaneous movement, and location privacy, the HIP architecture includes the *rendezvous server* concept [20]. A HIP rendezvous server simply forwards HIP control packets to a registered HIP host. It can also provide a two-way forwarding function [26]. Functionally, a rendezvous server is similar to a single i^3 server, as it forwards a packet to a registered IP address, based on the destination HIT in the packet.

[2]The problem of certifying the keys or otherwise creating trust relationships between them has explicitly been left out from the HIP architecture. It is expected that each system using HIP may want to take care of it in a different manner. For mere mobility and multi-homing, the systems can work without any explicit trust management, in an opportunistic manner [26].

2.2 Internet Indirection Infrastructure (i^3)

To ease the deployment of services, Stoica *et al.* proposed an i^3 overlay network that offers a rendezvous-based communication abstraction [31]. Instead of explicitly sending a packet to a destination, each packet is associated with a destination identifier; this identifier is then used by the infrastructure to deliver the packet. As an example, a host R may insert a trigger (id, R) in the i^3 infrastructure to receive all packets that have the destination identifier id.

i^3 provides natural support for mobility. When a host changes its address, the host needs only to update its trigger. When the host changes its address from R_1 to R_2, it updates its trigger from (id, R_1) to (id, R_2). As a result, all packets with the identifier id are correctly forwarded to the new address. Note that this change is completely transparent to the sender.

The primary aim of the Secure-i^3 proposal [1] was to provide a network architecture that is more robust against DoS attacks than today's networks. The basic idea is to protect against DoS attacks by hiding the IP addresses of the end-hosts from other users of the network. The indirection approach provides straightforward implementation for multicast, mobility, and multi-address multihoming. In Secure-i^3, there are two types of triggers, public and private. Public triggers are used to announce the existence of a service and are well known (announced on web pages, in the DNS, or on other public media). Private triggers are used for the actual communication between sender and the receiver(s), which are the only ones that know the private triggers.

Finally, we describe three advanced capabilities of Secure-i^3. In Secure-i^3, a public trigger cannot point to the end-host, but only to a private trigger to prevent cycles in the infrastructure and malicious misuse of triggers. Therefore, a *trigger chain* of two right-constrained triggers is used to insert a given identifier into the infrastructure. To run legacy applications over i^3, a *proxy* located on the client and the server must be used. The proxy transparently intercepts DNS requests and forwards data packets to the i^3 infrastructure. Recently, a capability to send data directly between the client and the server has been added to i^3. Known as *shortcuts*, it allows efficient data transfer between hosts, but does not offer currently any cryptographic data protection.

2.3 SPI multiplexed NAT

As argued by Walfish *et al.* [34, 35] and also elsewhere, by introducing IP-address-independent end-point identifiers, the connectivity problem created by NATs becomes easier to manage. Both the HITs in HIP and the trigger identifiers in i^3 are such address-independent identifiers. However, utilizing the identifiers for NAT traversal in an architecturally clean way requires that the NATs become aware of the identifiers[3].

[3]It is also possible to use new end-to-end identifiers with existing NATs, but this cannot be considered architecturally clean. It typically requires UDP encapsulation, constant state maintenance at the NAT, and external infrastructure

SPI multiplexed NAT (SPINAT), as proposed by Ylitalo *et al.* [38, 33], is an approach to establish a state for HITs during a HIP base or mobility exchange. The association at the SPINAT device consists of a HIT pair, IP address pair, and ESP SPI pair. The base or mobility exchange packets are routed based on the HITs in the HIP header. Once the state at the SPINAT device has been established, the device identifies connections using the SPI value and the destination IP address in the ESP-protected data packet headers. With a SPINAT-like approach it becomes possible to connect several IP realms into a single network where the upper layer identifiers are used to route packets between the realms.

2.4 Other related work

In addition to the work mentioned above, there has been a considerable number of other proposals to address the identifier / locator separation and consequently mobility, multi-homing, and security, both separately and in an integrated manner, both from the academic community and from the industry. For a partial list of proposals, consider FARA [6], MAST [7], PeerNet [10], IPNL [11], and LIN6 [14].

So far, none of the proposals have gained major acceptance, partially because the time has not been ripe, and partially because many of the proposals have not properly taken deployment and operational concerns into account.

3 Hi^3 architecture

In this section, we describe the Hi^3 architecture in detail. More specifically, we discuss the particulars of separating service naming, session control, and actual data delivery. Additionally, we consider data protection, support for multiple IP realms, and mobility. We conclude the section with a qualitative comparison of HIP, i^3, and Mobile IPv6 versus Hi^3.

The Hi^3 sketch [24] by Nikander *et al.* was based on the observation that a HIP rendezvous server and a single i^3 server are functionally close to each other. Therefore, the basic idea in Hi^3 is to allow direct, IP-based end-to-end traffic while using an indirection infrastructure to route the HIP control packets. In the Hi^3 sketch, all end-to-end traffic was supposed to be protected with ESP; in this paper we also consider other encapsulation methods. We also add the details to the original proposal for DoS protection by not revealing the actual IP addresses of the hosts.

Since Hi^3 relies on features from the basic i^3 architecture and the Secure-i^3 extension, from here on we do not make a difference between them. Hence, whenever we write i^3, we refer to i^3/Secure-i^3.

3.1 Separating naming, control, and data

Figure 2 illustrates the use of i^3 as a decentralized instantiation of the HIP rendezvous server. The HITs act directly as public, 128-bit long i^3 trigger identifiers. Data traffic flows directly between the hosts, using plain IP routing, just as today (shown with dashed lines). Hence, the control messages and data traffic are conceptually separated into different "planes", as is customary in telecommunication networks. However, this initial design works well only within a single IP realm and additional mechanisms are needed for supporting multiple inter-connected realms; see Section 3.2.

In a security-aware i^3 instantiation, public trigger identifiers are only used for initial rendezvous and the server is supposed the create a private identifier for each connection. We make the observation that the i^3 public identifiers resemble the service identifiers in the layered naming architecture by Balakrishnan *et al.* [2], while the private trigger identifiers clearly form some "lower" naming layer. Utilizing HIP, we use fresh, newly generated host identifiers as private trigger identifiers. That is, upon a new HIP association the server host generates a new host identifier for the client. To secure the binding between the public and private triggers, i.e. between the service and host identifiers, we use cryptographic delegation [22]; see Section 3.5.

To establish a connection with a server, a client first sends a HIP I1 packet to i^3, with a service identifier as the destination[4]. The infrastructure passes the packet to the server that returns a delegated R1 packet with a suitable host identity[5]. The client verifies delegation, solves the puzzle, and sends I2 to the server through the infrastructure; the packet also lists one or more IP addresses that the server can use to reach the client[6]. The server verifies the puzzle solution, authenticates the client, and sends an R2 packet giving address(es) that can be used to reach it[7]. From there on, the end-hosts rely on normal HIP mechanisms and need not be aware of the additional protections offered to them; see Section 4 for more details.

3.2 Protecting end-to-end data traffic

For basic end-to-end data protection we use HIP. In its simplest form, HIP encapsulates all data traffic in ESP, protecting integrity, authenticity, and (optionally) confidentiality. However, HIP alone does not protect against distributed denial-of-service attacks. In plain HIP, the hosts always reveal their real IP ad-

support in the form of STUN [29] or similar servers.

[4]The client does not need to be aware of Hi^3 being involved. The server may simply list i^3 as its rendezvous server for the service, resulting in the I1 being sent to i^3.

[5]With a *delegated R1* we mean an R1 packet that has two keys, on corresponding to the original service identifier and another to a host identifier. The delegation could be based on the HIP DELEGATION parameter, as proposed by Koponen *et al.* [18]. See Section 3.5 for more details.

[6]The HIP LOCATOR parameter is used for listing the client's IP address(es).

[7]The HIP LOCATOR is used, just like in the case of client's address(es).

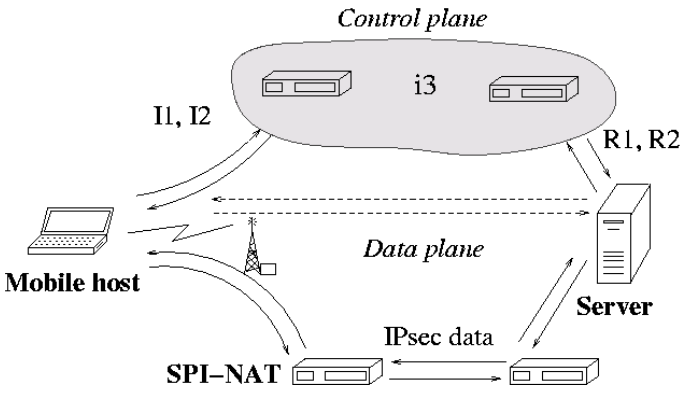


Figure 2: Basic Hi^3 architecture.

dress(es) to their potential peers. Therefore, a host could tell a large number of zombies to launch a coordinated bombing attack against the target host.

To protect against distributed denial-of-service, we extend the notion of using IPsec-aware middle boxes [24]. A number of SPINATs[8] (IPsec-aware middle boxes) are placed on or close to the possible data paths. These provide a fast-path barrier against bombing denial-of-service, simultaneously hiding the actual IP address of the servers. The method structurally resembles i^3 shortcuts [31] but is more secure than using shortcuts and works independently from the rendezvous infrastructure.

To employ SPINATs at the time the client and server inform each other about the IP addresses to be used for data traffic, they tell the addresses of SPINATs serving them instead of telling their real IP addresses. In other words, the use of SPINAT is completely controlled by the involved host, independent from the rendezvous infrastructure. In practical terms, in most cases the SPINAT can act by inspecting HIP base and mobility exchange packets flowing through it; see Section 3.3. Mobility performance and DoS resistance of SPINAT has been measured by Ylitalo *et al.* [37]. The results suggest that the efficiency of data plane is not significantly reduced by the presence of SPINATs.

Figure 2 illustrates the use of ESP envelopes and SPIs to implement the denial-of-service protection for the data traffic. As described in [33], it is easy to design such a middle box that forwards and filters traffic based on <dst,SPI> pairs. The filtering can be extended to include source addresses. In the typical case of the control packets passing through the middle box, the middle boxes can securely learn the appropriate mappings by listening to the signed control packets. If the control and data packets take completely different paths, there must be explicit signaling between policy points at the control and data path. For example, the hosts can use the HIP registration protocol [19] to create suitable initial state at some SPINAT.

As the SPINAT knows the allocated SPI mappings, including the source and destination IP addresses, for its basic functionality, it can easily filter out most unwanted traffic. A random attacker can't learn the real IP address of the server; it can only learn the IP address of the SPINAT. Getting packets through the SPINAT requires that the attacker knows a valid SPI, causing random packets to be effectively filtered. However, an attacker that establishes an (opportunistic) HIP association with the server learns a valid SPI, which it can communicate to a large number of zombies. Hence, source address spoofed traffic from zombies that have learned a valid SPI still form a potential problem. Applying heuristics based on ESP sequence numbers makes such coordinated attacks harder but not impossible; the zombies can increase the sequence number in rough synchrony, resulting in unwanted high-volume traffic where the sequence numbers mostly fall within the replay window.

[8]Note that even though the SPINATs in their basic form translate network addresses in order to hide the real IP address(es) of the server, that translation may still happen between IP addresses belonging to the same IP realm instead of distinct IP realms. Alternatively, if placed always on path (instead of close to the path), they can function as plain filters that do not perform address translation at all. The following discussion mostly applies to all cases, with just minor differences.

An obvious means to protect against zombie-based synchronized bombing attacks is to deploy source address filtering everywhere in the network. That would prevent zombies from sending valid-looking packets; the packet's source address would necessarily be different, resulting in the packets being dropped at the first SPINAT on the path.

In Hi^3 the IP source address field is no longer needed[9]. The control packets are explicitly routed by the identifiers; there the source HIT takes the function of the source IP address. The data packet destination is always based on the local by-HIP-created IP-layer state, and the source address is always ignored [21]. Hence, we surmise that the source address field could be used to record the actual path taken by the packet [5].

Utilizing the possibility of using HIP-based mobility, a server under an attack can move the legitimate traffic to other available SPINATs. Hence, a multi-homed site with multiple entry SPINATs or a host with suitably selected independent SPINATs can move legitimate traffic from the SPINAT under an attack to another one. The server can also use the HIP control packets to tell the attacked SPINAT to drop forwarding all traffic on the attacked SPI. This is structurally similar to a host dynamically changing its private trigger in i^3.

3.3 Supporting multiple IP realms

In the discussion above we have glossed over problems caused by multiple IP realms and the resulting partial connectivity. For the system to work properly in the current multi-realm IP reality, two requirements must be fulfilled. First, all hosts must be reachable through the i^3 infrastructure. Second, the hosts must know at least one public IP address of a SPINAT serving them so that they can tell that address to their peers at or behind the public Internet. There are multiple ways to fulfill the requirements.

We first consider the requirement of knowing a public IP address of a serving SPINAT. As the SPINATs are assumed to form a new piece of infrastructure, an anycast-based mechanism can be used to learn suitable nearby SPINATs. Alternatively, in a corporate environment SPINAT-related information could be naturally distributed along with other managed configuration data. Additionally, on-path, passive plain NATs could be detected directly, and the necessary state in them can be created with methods similar to STUN [29] or ICE [25].

To make hosts reachable by the i^3 infrastructure, the simplest way seems to be to locate the infrastructure in the public Internet, requiring the hosts in other IP realms to maintain active connectivity with that/those i^3 server(s) that hold their private trigger(s). In that way the packets sent to the private trigger can be always passed to the hosts over active connections. Alternatively, if a host is able to create semi-permanent state at some SPINAT with a public IP address, it can list the SPINAT's IP address at the private trigger, again resulting the packets coming to the right host. However, in this case the i^3 server does not use an existing connection for sending the packet but sends it to the SPINAT, which in turn forwards it according to the state associated with the HIT.

[9]The source addresses are often still useful, if for no other purpose for learning the identities of certain i^3 servers; see Section 4.2.

In any case, multiple realm support requires reachability state to be created at the SPINATs between the realms. This state can be either created explicitly, by hosts registering their identifiers at the cross-realm SPINATs. The resulting infrastructure resembles proactive hop-by-hop host routing, but takes place on a layer above the current IP routing layer. Alternatively, supposing the existence of a single most preferred realm (i.e., the public Internet), SPINATs at the realm boundaries can learn the identifiers of the hosts behind them. In order to remain reachable, the hosts must keep sending packets towards the preferred realm. In this case, the resulting infrastructure resembles link layer bridging.

3.4 Handling alternative encapsulation formats

The HIP protocol is planned to be extended to support other encapsulation mechanisms in addition to ESP [21]. Independent of the encapsulation method, HIP requires that there is enough of information in the data packet so that it can be successfully demultiplexed and tagged with source and destination HITs. Furthermore, de-multiplexing must work independent of the source address in the packet. We surmise that as long as the information used for de-multiplexing is sufficiently hard to predict but easily verifiable with a state that can be formed from the public information in the HIP control packets, the above-outlined SPINAT-based protection can be easily generalized to future HIP encapsulation methods.

3.5 Delegating part of the processing to the infrastructure

When a service is registered to the i^3 infrastructure by creating a trigger for the service identifier, instead of pointing to a server host that directly implements the service, the trigger can point to an infrastructure node that handles the I1 packets directly. In HIP, the hosts handle I1s by replying with precomputed R1 messages. To implement service/host separation, we assumed above that the R1 messages use a host identity different from the service identity, explicitly denoting delegation from the service identifier to the host identifier.

We now propose a separate *service distribution function, SDF*, which can be integrated with the actual service nodes, if desired. As in HIP, the function handles I1 messages sent to the service by replying with suitable pre-computed R1 messages. In contrast to HIP, the R1 packets are not signed by a host key but by the service key. Instead of carrying just one public key, they carry two: the service key corresponding the HIT and used for signing the packet, and a host key. The host key is stored in a HIP parameter with a new parameter type, denoting delegation. The host key identifies a host that will provide the service to the client[10]. The signature in the packet allows the client to verify that the host is indeed authorized to provide the service[11].

The service distribution function allows I1 packets to be processed completely by the infrastructure. The servers need not see the I1 packets in the first place. Furthermore, the function

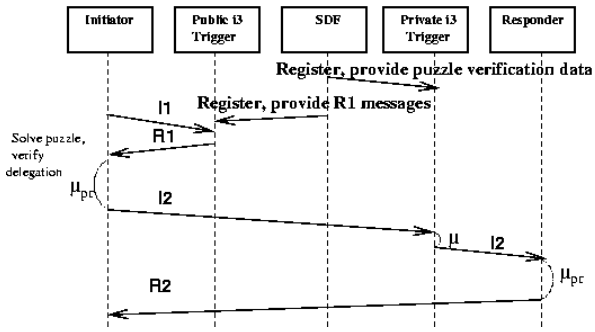


Figure 3: Hi^3 base exchange with delegation.

of forming the R1 packets and replying to I1 packets with R1 packets can be separated. Hence, the R1 packets can be created and signed by a well protected node that has authority over the service while the R1 distribution can take place directly by the i^3 server that stores the public trigger. All that is required is that the R1 generating node supplies the i^3 servers with pre-computed R1 packets.

Note that the host can no longer verify the puzzle alone, as it would become vulnerable to pre-computation attacks. Hence, the infrastructure *must* verify the puzzle origin; at the same time, it can easily verify that the puzzle solution is correct. For this, the node that generates the R1 messages can provide the i^3 server holding the private trigger with the necessary information for verifying the puzzle's origin and freshness.

The remaining problem of providing the server assurance of puzzle verification can be solved in several different ways. The simplest way is to let the server assume that any I2 packets coming from the i^3 server holding the private trigger have passed puzzle verification. If the server fully trusts the infrastructure and if there is a secure channel between the nodes, this can be deemed secure. Another solution is to let the service distribution function and the actual servers share data about puzzle creation, thereby allowing the servers to (re)verify the puzzle[12].

Figure 3 demonstrates the relaying of HIP handshake over i^3 with delegation of I1 processing to the infrastructure (see also Section 4.2).

Once the initiator has processed the R1 packet and produced the puzzle solution (in time $\mu_{\rm pr}$), it sends an I2. The I2 is now sent to the private trigger. The i^3 server that keeps the private trigger verifies that the puzzle solution is correct and created by the SDF before passing the I2 packet to the server. Effectively, this distributes the proposed i^3 DoS-filter function over all i^3 server nodes, allowing the puzzle to be formed and verified by different nodes. In sum the server and the responder spend time $\mu_{\rm pr}$ for verifying the puzzle and authenticating the client.

3.6 Mobility

In Hi^3, basic mobility between already communicating hosts can be provided directly at the HIP and SPINAT level, without involving the i^3 infrastructure. Only if the hosts lose direct reachability, they need to revert back to the infrastructure (see

[10] We surmise that the host would typically be a *virtual* host and that the host key would be used only for serving one client. However, the architecture does not impose these properties; there are clearly cases where a different pattern offers better utility.

[11] Compare this to the SPKI [9] use of signatures.

[12] For example, the service distribution function and the servers can use the same keyed PRNG, run in synchrony, to drive puzzle generation.

also Section 4.3). Even in that case hosts will use private triggers, being safe from attacks launched by third parties. For this to work, the hosts must keep the infrastructure updated with their current location information.

By combining the end-to-end mobility provided by HIP and the indirect mobility support provided by the infrastructure, the resulting mechanism is highly efficient (no triangle routing for regular data) and robust (a property inherited from i^3).

In addition to plain end-host mobility, Ylitalo has suggested how to apply HIP for network mobility [36]. Furthermore, the signaling delegation ideas by Nikander and Arkko [22] can be applied more generally to HIP, resulting in savings in the air interface. The support of these ideas in Hi^3 is left for future study.

3.7 Qualitative comparison

In this section, we compare the pros and cons of using HIP, i^3, and Mobile IPv6 versus Hi^3. We use the following evaluation criteria for compatibility with the previous studies [12].

- Mobility
  - simultaneous mobility and multihoming support,
  - fault-tolerance,
  - inverse mapping support.
- Security
  - Denial of Service (DoS) resistance,
  - end-to-end security and privacy,
  - accountability,
  - trust model.
- Efficiency
  - routing efficiency,
  - infrastructure cost.

The basic HIP protocol provides efficient and secure end-to-end connectivity. If the HITs and IP addresses of end points are known, it can work without additional infrastructure, thus having no issues with infrastructure cost, accountability, trust, or fault-tolerance. Basic HIP provides limited DoS protection by enabling the responder to make the initiator solve a computationally substantial puzzle before creating state in the responder. Mobility of one end point at a time is supported, but there is no way to perform the reverse mapping support [13]. HIP with a rendezvous server enables mobility of both end points, while preserving accountability and the trust model, since the rendezvous server is chosen by the responder.

The advantages of i^3 include better DoS protection, support for simultaneous mobility, and higher fault-tolerance when using a DHT with data replication. Disadvantages of i^3 include reliance on an extensive infrastructure, server scalability, use of UDP, and lack of traffic encryption. Since i^3 is an overlay network on top of the Chord DHT, it makes the infrastructure fairly complex. There is limited experience with widespread i^3 deployment, thus it is difficult to assess how scalable the servers are. The latency of relayed control traffic will mostly be affected by forwarding and network delays. However, relaying all control and data traffic through i^3 infrastructure would likely prove burdensome. By mutual agreement, the client and the server could use i^3 only for initial contact and afterward exchange the data directly using shortcuts.

The basic i^3 system does not provide data encryption, although it could be implemented as an add-on feature. i^3 also lacks encryption and privacy for control packets. When a public infrastructure is used, i^3's extensive infrastructure requirements bring other serious security issues including the possibility of malicious or misbehaving i^3 nodes that do not forward correctly and a lack of trust of arbitrary i^3 servers from end points. Note that Secure-i^3 introduced several constraints on the structure of triggers to prevent misuse of triggers by third parties and formation of loops in the topology. Finally, diagnosing problems in a distributed Internet system is always challenging, and the added indirection introduced by i^3 further complicates the situation.

A combination approach helps to address some of the separate shortcomings of HIP and i^3. The advantages of using i^3 as a control plane for HIP in Hi^3 include protection from DoS attacks, solving the double-jump problem, and providing an initial rendezvous service. By hiding parties' IP addresses until the HIP handshake partially authenticates them, Hi^3 provides additional protection against DoS attacks. Although some DoS protection could be provided by a HIP rendezvous server, the client's IP address is revealed to a server in the first control packet. Simultaneous mobility of both hosts in i^3 is supported by sending update control packets via i^3 when end-to-end connectivity is lost. Hi^3 inherits the challenges of the extensive i^3 infrastructure, including trust, accountability, and cost issues.

Comparing Mobile IP to HIP, basic Mobile IP does not provide any DoS protection mechanisms, end-to-end security, or support for multi-homing or co-existence of IPv4 and IPv6. Hi^3 inherits the benefits of HIP and provides better DoS protection than HIP does. End-to-end security can be added to Mobile IP with IKE [8]. In Mobile IP, the mobile node and home agent are assumed to have a business relationship, leading to a fairly clear trust and accountability model. While there are proposals for supporting mobility between IPv4 and IPv6 [30], we are not aware of any serious work to address end-host multi-homing with Mobile IP. From a fault tolerance point-of-view, the Mobile IP home agent forms a single point of failure. From an efficiency point of view, Mobile IP adds extra mobility-related headers to all packets while HIP/Hi^3 does not[14]. The capacity requirements for Mobile IP and Hi^3 appear to be at the same level; however, the administrative models differ considerably[15].

4 Scenarios

We describe three Hi^3 scenarios in detail. The first two scenarios are for establishment of a connection between a mobile client and a server: a pure setup and an optimized one, respectively. In the optimized setup the requirements for communication capacity are reduced. In the third scenario the case where two mobile hosts move at the same time is considered.

[13] A reverse lookup from an IP address to HIT (similar to reverse DNS) provides additional functionality, for example, for security purposes.

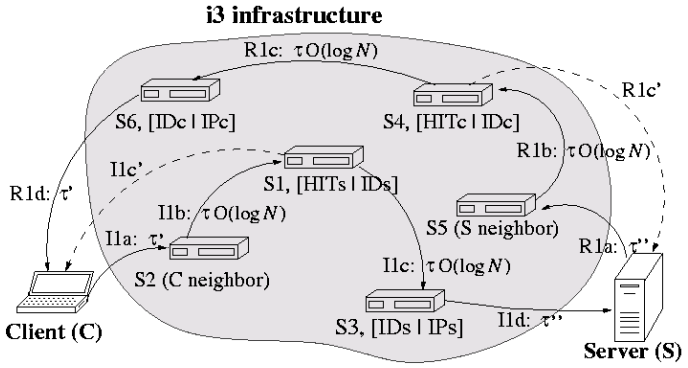
[14] While HIP currently requires ESP encapsulation, this may change in the future; see Section 3.4.

[15] In Hi^3, the infrastructure is distributed and may be controlled jointly by several organizations. Unless some structure is imposed on the trigger identifiers, the parties cannot easily control where an identifier is stored. In Mobile IP, on the other hand, the home agents are configured to the mobile hosts.

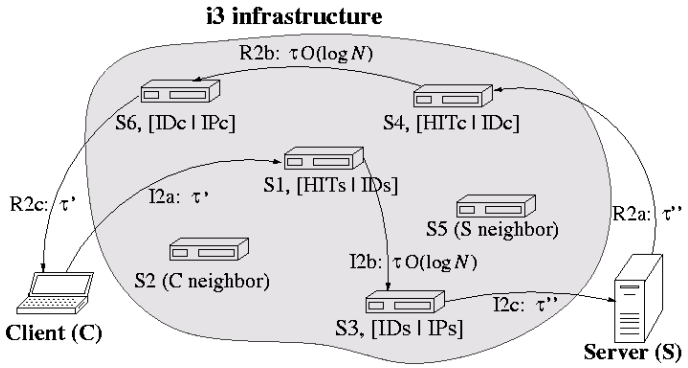
Describing the scenarios of association setup we assume the client is a mobile host and the server is a stationary one. This case is close to the initial contact scenario for Mobile IP (see the Appendix). However, Hi^3 easily allows more general scenarios, e.g., the server S can be a mobile node.

4.1 Pure HIP association setup

Figure 4 shows the setup of HIP connections in Hi^3. Arrows corresponds to packets' paths and are labeled with the packet name (e.g., I1 or R2), the sequence mark (i.e., "a" is for the fist part of a trip, "b" is for the next one, etc.). Average trip time follows in a label after column (e.g., τ' is time for a packet to travel from C to the infrastructure, τ'' is the same but for S, τ characterizes inter-domain connectivity of i^3 (see Section 5 for details).



(a) I1 and R1 packets



(b) I2 and R2 packets

Figure 4: HIP base exchange via Hi^3 infrastructure.

The client C sends an I1 packet to the IP address of a random i^3 server it happens to have, hopefully the closest neighbor server. In this case the server is $S2$, see the path I1a in Figure 4 (a). The public trigger for the server S, HITs, is stored in the i^3 server $S1$, and $S2$ forwards the packet to $S1$ via i^3 (path I1b). The client obtains the correct i^3 server for future contacts to the recipient server, $S1$ (path I1c', in parallel with the primary branch I1c–I1d). For security, the server S has also registered a private trigger that happens to reside on the server $S3$. Therefore, $S1$ forwards the packet to $S3$ (path I1c) that in turn delivers it to S (path I1d).

A similar procedure is followed by S to send an R1 reply packet to C. S first contacts its neighbor i^3 server $S5$ (path R1a). The public trigger for the client C, HITc, is stored in the server $S4$, and $S5$ forwards the packet to $S4$ via i^3. $S4$ notifies S about the correct i^3 server for communicating with C (path R1c', secondary branch) and forwards R1 to $S6$ that keeps the private trigger for the client (path R1c). $S6$ delivers the packet to the client (path R1d).

The consequent I2–R2 exchange occurs in a similar manner, see Figure 4 (b). The only difference is that the packets are sent straight to the i^3 servers keeping the public triggers, $S1$ and $S4$, respectively.

4.2 Optimized HIP association setup

Figure 5 shows the optimized setup. In this scenario, the i^3 server $S1$, which keeps the public trigger of S, also caches precomputed R1 packets of the server S. This slightly reduces the communication load of S by delegating a part of HIP cryptography processing to the infrastructure (see also Figure 3). This processing is comparable with a cost of packet forwarding by an i^3 server (μ). Since $\mu \ll \mu_{\text{pr}}$ the benefit is mainly in reducing communication needs.

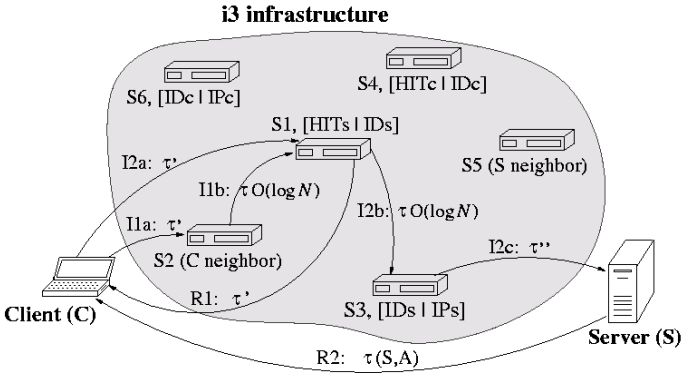


Figure 5: Optimized HIP base exchange via Hi^3 infrastructure.

As in the previous scenario, the client C sends an I1 packet to the i^3 server $S2$, its neighbor (path I1a). The packet is forwarded to $S1$ via i^3. However, $S1$ replies directly to the client with an R1 packet that it has cached (path R1). At the same time, it notifies C about the correct i^3 server for reaching S.

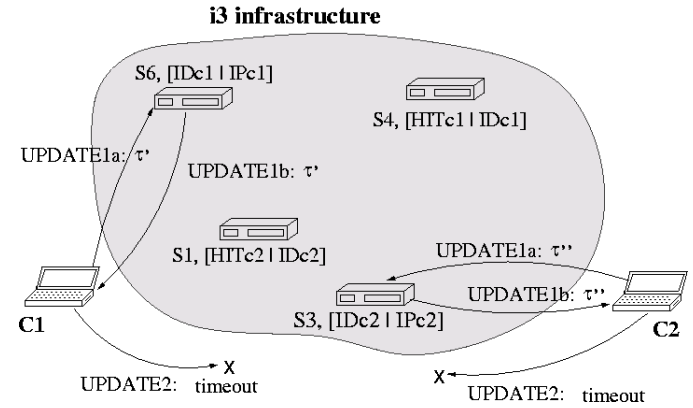
The I2 packet is sent to $S1$ (path I2a), forwarded to $S3$ via i^3 (path I2b), and then delivered to S (path I2c). The packet is expected to contain the HIP locator parameter, listing the client's real IP address. S replies with an R2 packet directly to the client (path R2).

4.3 Simultaneous host movement

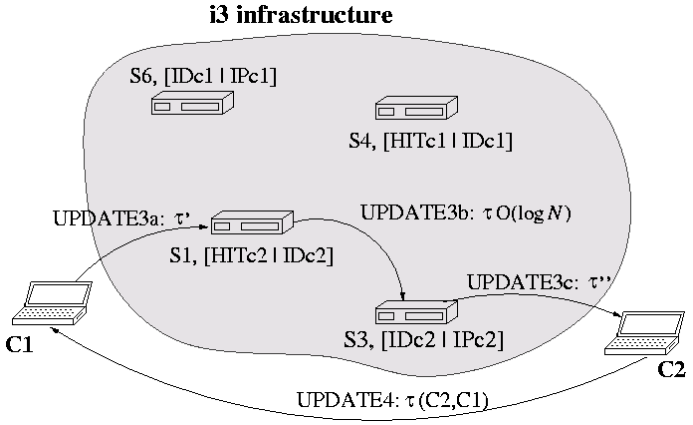
Consider the case when two mobile hosts $C1$ and $C2$ have setup a HIP connection to communicate each with another. This can be done with any of the scenarios mentioned above, but the division of roles to clients and servers is not important for this case.

Figure 6 (a) shows what happens if both $C1$ and $C2$ change their IP addresses simultaneously.

$C1$ sends its HIP UPDATE packet to the i^3 server that stores its private trigger, $S6$ in this case (path UPDATE1a). Also $C1$ sends the UPDATE packet directly to $C2$ (path UPDATE2). Approximately at the same time $C2$ is changing its IP address and



(a) UPDATE1 and UPDATE2 packets



(b) UPDATE3 and UPDATE4 packets

Figure 6: HIP simultaneous update via Hi^3 infrastructure.

performs the same UPDATE steps; it updates the private trigger in $S3$ and tries to notify $C1$ about the change.

Servers $S6$ and $S3$ reply with acknowledgments to $C1$ and $C2$, respectively (UPDATE1b paths). However, the UPDATE2 packets cannot be delivered due to the simultaneous change of IP addresses, and a timeout eventually occurs.

Assume that $C1$ always happens to detect the timeout first. It starts the Hi^3 simultaneous update scenario; see Figure 6 (b). $C1$ resends an UPDATE packet to $C2$ via $S1$, which stores the public trigger of $C2$ (path UPDATE3a). The i^3 server forwards the packet to $S3$ via i^3 (path UPDATE3b). Then, the update is delivered to $C2$ (path UPDATE3c). The mobile host $C2$ replies with its update directly to the new IP address of $C1$ (path UPDATE4), and from there on the communication proceeds normally.

5 Theoretical analysis

We analyze the following properties of the proposed Hi^3 solution: 1) latency of basic scenarios (Hi^3 performance) and 2) scalability (Hi^3 deployment). The analysis is based on typical simplified assumptions: Uniformity of all parties involved and inexhaustible communications needs (a host always has a need to communicate with other hosts). As the result, we obtain average estimates for the worst case.

5.1 Hi^3 parameters

We define three groups of important Hi^3 parameters. The first one is parameters that can be used to control Hi^3. Two other groups contain parameters that we can't manage directly: Outside parameters that are independent on the internal structure of i^3, and characteristics required from Hi^3 to be provided when the infrastructure is deployed.

5.1.1 Manageable and semi-manageable parameters

N : the number of servers in i^3.

τ : average transit time of a packet between two arbitrary i^3 servers, ≈ 100 ms.

μ : average time for an i^3 server to process a packet before forwarding it. This time depends on the infrastructure size[16]: $\mu = \mu(N)$, because any i^3 server maintains $O(\log N)$ state. This makes μ significant for analysis of large-size infrastructures. The reasonable range is 0.1–1.0 ms.

$\mu_{\rm pr}$: average processing time at a HIP host. This characterizes cryptography complexity: time of either I1/R2 or I2/R2 processing, see Figure 1. The range is close to 100–200 ms.

The key parameter for Hi^3 infrastructure deployment is N. It needs to be estimated depending on desired characteristics of Hi^3. Parameters τ, μ and $\mu_{\rm pr}$ should be considered as semi-manageable. Their management is possible, but limited. Processing times μ and $\mu_{\rm pr}$ can be reduced by the CPU power of i^3 servers and hosts. Transit time τ depends on i^3 topology and proximity properties; it can be shifted a bit, for instance, by introducing a large i^3 cluster.

5.1.2 Parameters outside of i^3

M : the number of mobile hosts, expected 10^6–10^9.

λ : the average rate of requests from an arbitrary mobile host to Hi^3, e.g., the number of setup initiations per 1 ms. We have used the estimate of one new connection establishment per 15 min[17], i.e., $\lambda \sim 10^{-6}$ ms^{-1}.

$\tau_A^{\rm Hi3}$: the average transit time between a HIP host A and a i^3 server. For wireline $\tau_A^{\rm Hi3} \sim 10$ ms, for wireless $\tau_A^{\rm Hi3} \sim 200$ ms.

τ_{AB} : the average transit time between a HIP host A and a HIP host B via IP. Approximately one can assume $\tau_{AB} = \tau_A^{\rm Hi3} + \tau + \tau_B^{\rm Hi3}$.

5.1.3 Required Hi^3 characteristics

$\tau^{\rm Hi3}$: the average internal transit time of a HIP packet to cross the infrastructure, i.e. the time the packet travels inside i^3.

[16]The Mobile IP case is similar: a home agent plays the role of a forwarding server and μ depends on the number of mobile clients the home agent serves.

[17]This measures the rate of new IP-level contacts between hosts, not the rate of new transport or application level connections. Given that many mobile hosts are idle for long times or tend to communicate with a fairly small set of hosts, this figure seems to be conservative.

τ^{out} : the average outside time of a HIP packet, i.e., the time the packet travels outside of i^3.

L^{Hi3} : the latency of a HIP message exchange, e.g., base exchange or mobility simultaneous update, assuming the infrastructure is involved.

W : workload of Hi^3. Measured as the average number of requests to an i^3 server either to serve HIP packets, or for fixed time processing, e.g., μ-processing of forwarding.

5.2 Latency of an association setup

5.2.1 Pure association setup

According to Figure 4 (Section 4.1), the latency of HIP base exchange in the pure association setup consists of four trips over i^3 ($4\tau^{\text{Hi3}}$), four trips to/from the infrastructure, both for C and S ($4\tau_C^{\text{Hi3}} + 4\tau_S^{\text{Hi3}}$), and time $2\mu_{\text{pr}}$ spent for R1/I2 (by client) and I2/R2 (by server) processing.

$$L_{\text{s}}^{\text{Hi3}} = 4(\tau^{\text{Hi3}} + \tau^{\text{out}}) + 2\mu_{\text{pr}} = \\ = 4(\tau_C^{\text{Hi3}} + \tau^{\text{Hi3}} + \tau_S^{\text{Hi3}}) + 2\mu_{\text{pr}}$$

i^3 uses the Chord DHT [32] to route a packet and requires a sequence of $O(\log N)$ hops toward the destination. Therefore, each lookup needs time $t \approx (\tau+\mu)O(\log N)$ in average. Hence, $t \le \alpha(\tau+\mu)\log N$ for a constant $\alpha > 0$. According to [32], $\alpha \approx 1/2$. Below we assume the worst case for the average lookup time: $t = \frac{1}{2}(\tau+\mu)\log N$

The setup spends time $4\tau^{\text{Hi3}} = 6t$ inside the infrastructure (see Figure 4). Thus, the latency is

$$L_{\text{s}}^{\text{Hi3}} = 4\tau_C^{\text{Hi3}} + 3(\tau+\mu)\log N + 4\tau_S^{\text{Hi3}} + 2\mu_{\text{pr}} \quad (1)$$

and

$$\begin{aligned} 4\tau^{\text{Hi3}} &= 3(\tau+\mu)\log N \\ 4\tau^{\text{out}} &= 4\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + 4\tau_S^{\text{Hi3}} \end{aligned} \quad (1a)$$

5.2.2 Optimized association setup

There are only two lookups (time $2t$). Extra processing is delegated to i^3 in the optimized association setup (see Figure 5 in Section 4.2 and Figure 3 in Section 3.5). The cost of this processing is comparable with μ; since $\mu \ll \mu_{\text{pr}}$ and $\mu \ll (\tau+\mu)\log N$ we can neglect this.

The latency for this case is

$$L_{\text{so}}^{\text{Hi3}} = 4(\tau^{\text{Hi3}} + \tau^{\text{out}}) = \\ = 3\tau_C^{\text{Hi3}} + (\tau+\mu)\log N + \tau_S^{\text{Hi3}} + \tau_{SC} + 2\mu_{\text{pr}} \quad (2)$$

and

$$\begin{aligned} 4\tau^{\text{Hi3}} &= (\tau+\mu)\log N \\ 4\tau^{\text{out}} &= 3\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + \tau_S^{\text{Hi3}} + \tau_{SC} \end{aligned} \quad (2a)$$

5.2.3 Benefit of the setup optimization

Based on (1) and (2), the absolute benefit of the setup optimization on average is

$$L_{\text{s}}^{\text{Hi3}} - L_{\text{so}}^{\text{Hi3}} = \tau_C^{\text{Hi3}} + 2(\tau+\mu)\log N + 3\tau_S^{\text{Hi3}} - \tau_{SC}$$

Assuming that for $N > 3$

$$\tau_{SC} \approx \tau_C^{\text{Hi3}} + \tau + \tau_S^{\text{Hi3}} < \tau_C^{\text{Hi3}} + \frac{\tau+\mu}{2}\log N + \tau_S^{\text{Hi3}}$$

we obtain that the benefit is more than

$$\frac{3}{2}(\tau+\mu)\log N + \text{RTT}(S, \text{Hi3}) \quad (3)$$

For typical values given in Section 5.1 and a conservative value $N = 500$ (see Section 5.5), the benefit is in the order of more than one second or even two seconds for scenarios with mobile S. This can be considered significant.

5.3 Latency of simultaneous update

The update scenario was introduced in Section 4.3 (see Figure 6). Ignoring the initial timeout, two HIP packets are involved, one lookup is performed (time t), and only the UPDATE3 packet crosses the infrastructure—the UPDATE4 packet moves directly from $C2$ to $C1$. The latency of simultaneous update is

$$L_{\text{u}}^{\text{Hi3}} = 2(\tau^{\text{Hi3}} + \tau^{\text{out}}) = \\ = \tau_{C1}^{\text{Hi3}} + \frac{\tau+\mu}{2}\log N + \tau_{C2}^{\text{Hi3}} + \tau_{C2,C1} \quad (4)$$

and

$$\begin{aligned} 2\tau^{\text{Hi3}} &= \frac{\tau+\mu}{2}\log N \\ 2\tau^{\text{out}} &= \tau_{C1}^{\text{Hi3}} + \tau_{C2}^{\text{Hi3}} + \tau_{C2,C1} \end{aligned} \quad (4a)$$

5.4 Comparison with Mobile IP

5.4.1 Setup latency

The Mobile IP messages, corresponding to the HIP association setup, are given in Appendix A.1. The latency of an initial contact is

$$L_{\text{s}}^{\text{MIP}} = \tau_{\text{MN}}^{\text{HA}} + \tau_{\text{CN}}^{\text{HA}} + \mu + \tau_{\text{MN,CN}} \quad (5)$$

where $\tau_{\text{MN}}^{\text{HA}}$ and $\tau_{\text{CN}}^{\text{HA}}$ are times for a packet to reach HA from MN and CN, respectively; μ is the time to process a packet by HA for forwarding; $\tau_{\text{MN,CN}}$ is the one-way trip time between MN and CN.

Let us compare the optimized association setup in Hi^3 and initial contact in Mobile IP, where C and S correspond to MN and CN, respectively. Obviously, $\tau_{SC} \sim \tau_{\text{MN,CN}}$, $\tau_C^{\text{Hi3}} \sim \tau_{\text{MN}}^{\text{HA}}$ and $\tau_S^{\text{Hi3}} \sim \tau_{\text{CN}}^{\text{HA}}$. Hence,

$$L_{\text{so}}^{\text{Hi3}} - L_{\text{s}}^{\text{MIP}} = 2\tau_C^{\text{Hi3}} + (\tau+\mu)\log N + 2\mu_{\text{pr}} - \mu$$

Considering the cost μ of forwarding a packet by HA as negligible, the setup scenario in Mobile IP is faster approximately for time

$$\text{RTT}(C, \text{Hi3}) + (\tau+\mu)\log N + 2\mu_{\text{pr}} \quad (6)$$

In reality, in the typical case the connection setup time in Mobile IP is dominated by the TCP connection establishment and not by the route optimization.

5.4.2 TCP connection setup

Let $L_{\text{TCP}}^{\text{Hi3}}$ be the latency of a TCP connection setup in Hi^3. As HIP in its current state does not support TCP piggybacking,

$$L_{\text{TCP}}^{\text{Hi3}} = L_{\text{s*}}^{\text{Hi3}} + L_{\text{TCP}}$$

where L_{s*}^{Hi3} is either L_{s}^{Hi3} or L_{so}^{Hi3}, and $L_{\mathrm{TCP}} = 3\tau_{CS}$ is the latency of a standard TCP tree-way handshake that runs directly between hosts C and S.

The corresponding latency of TCP connection establishment for the case of Mobile IP can be estimated roughly as

$$L_{\mathrm{TCP}}^{\mathrm{MIP}} = 3\left(\tau_{\mathrm{MN}}^{\mathrm{HA}} + \mu + \tau_{\mathrm{CN}}^{\mathrm{HA}}\right)$$

due to the fact that TCP connection delay in Mobile IP is typically dominated by the TCP three-way handshake flowing through HA (see Appendix A.2).

Assuming that $\mu \approx 0$ (comparing with other delays) and $\tau_{CS} \approx \tau_{C}^{\mathrm{Hi3}} + \tau + \tau_{S}^{\mathrm{Hi3}}$, even with optimized setup Hi^3 requires more time than Mobile IP:

$$\begin{aligned} L_{\mathrm{TCP}}^{\mathrm{Hi3}} - L_{\mathrm{TCP}}^{\mathrm{MIP}} \approx 2\mathrm{RTT}(C, \mathrm{Hi3}) + \\ + \tau\log(2^4 N) + 2\mu_{\mathrm{pr}} + \mathrm{RTT}(S, \mathrm{Hi3}) \end{aligned} \quad (7)$$

In real terms, using the values from Section 5.1 for a mobile client and stationary server and a conservative value of $N = 500$, Mobile IP appears to be roughly 2.5 second faster; for a smaller N the value is accordingly smaller. Almost 1.5 seconds of the additional 2.5s delay is caused by the initial HIP packets crossing[18] the i^3; the rest consist of cryptographic processing and additional delay at the client side air interface caused by transporting HIP packets. On the other hand, if HIP would support TCP piggybacking in the I2 and R2 packets, this time could be cut by $2\tau_{CS}$ or 400–500 ms. Furthermore, the delay is caused only at the first TCP connection between any two hosts, and again only after the HIP association has expired. At any subsequent TCP connection between two already communicating hosts, Hi^3 and MIP are have roughly equal performance, as in both cases all packets are sent directly between the hosts.

The fact that the first connection takes significantly more time in Hi^3 versus Mobile IP is natural from the very different signaling patterns. Hi^3 requires that the hosts first communicate through the infrastructure, while Mobile IP allows them to directly exchange packets with the home agent. On the other hand, Hi^3 also protects both the infrastructure and the server much better than Mobile IP does. To get more comparable results between Hi^3 and Mobile IP, the analysis should include an IKE exchange to secure the end-to-end traffic for Mobile IP. However, given the difficulties in estimating the likely packet paths (see the Appendix), such an analysis is left for further study.

5.4.3 Simultaneous update latency

The Mobile IP analog for a simultaneous update is presented in Appendix A.3. The latency is

$$L_{\mathrm{u}}^{\mathrm{MIP}} = 5(\tau_1^{\mathrm{HA}} + \mu + \tau_2^{\mathrm{HA}}) + 2(\tau + \mu) + \tau_{21} \quad (8)$$

where τ_1^{HA} and τ_2^{HA} are time for a packet to travel to HA (either HA1, or HA2) from MN1 and MN2, respectively; τ and τ_{21} are cross trip time HA1–HA2 and MN1–MN2, respectively.

For comparison of Hi^3 and Mobile IP let us estimate the difference $L_{\mathrm{u}}^{\mathrm{MIP}} - L_{\mathrm{u}}^{\mathrm{Hi3}}$. Obvious assumption is $\tau_{C2,C1} \sim \tau_{21}$, $\tau_{C_i}^{\mathrm{Hi3}} \sim \tau_{\mathrm{MN}_i}^{\mathrm{HA}}$ for $i = 1, 2$, and τ and μ are the same for both solutions.

$$L_{\mathrm{u}}^{\mathrm{MIP}} - L_{\mathrm{u}}^{\mathrm{Hi3}} = 4(\tau_1 + \mu + \tau_2) + \mu - \frac{\tau + \mu}{2}\log\frac{N}{2^4} \quad (9)$$

For all N, except extremely large, the Hi^3 solution is faster. The corresponding bound for N can be easily computed by equating (9) to zero:

$$N_{\mathrm{bound}} = 4^{(4(\tau_1+\mu+\tau_2)+\mu)/(\tau+\mu)} \quad (10)$$

Only for $N > N_{\mathrm{bound}}$ the Mobile IP solution for simultaneous update has smaller latency than Hi^3.

Example 1. Assume μ is negligible comparing with τ. Let $\tau_1 = \tau_2 = 200$ ms and $\tau = 150$ ms. Then

$$N_{\mathrm{bound}} = 4^{4\cdot 400/150} \approx 2.5 \cdot 10^6 \sim 10^6$$

As it will be shown in Section 5.5, typical values for N are in order of several hundreds. Therefore, the threshold N_{bound} seems not to be reachable in practical scenarios.

5.4.4 Discussion

The comparative analysis above shows that the performance of Hi^3 and Mobile IP is comparable.

On one hand, the setup in Hi^3 is slower with extra latency (6) or (7). However, the difference is close to several $\mathrm{RTT}(C, S)$ for a reasonable infrastructure size N; see estimates for N in Section 5.5, and appears to be reasonable in reality.

On the other hand, recovery from a simultaneous mobility situation is faster in Hi^3, see Eq. (9). In this case, the difference is also of order of a few RTTs between the mobile nodes.

5.5 Scalability

The important problem for Hi^3 deployment is estimation of N, the number of required i^3 servers to provide the certain average latency. In this paper, we make estimates based on the latency of association setup only; we expect simultaneous movements to be relatively rare, and estimates based on them are left for further study.

Consider the following scenario of mobile peers interaction shown in Figure 7. Let M be the total number of mobile hosts that use the infrastructure. The rate λ of Hi^3 requests are the same for each host. We limit the analysis only to requests for an association setup, due to their expensiveness for the infrastructure. Any host with rate λ initiates a HIP-based connection with a random peer.

5.5.1 Workload

Each request affects, on average, $3\log N$ of i^3 servers in the pure association setup case or $\log N$ in the optimized setup case. Hence, a HIP host loads $3\lambda\log N$ or $\lambda\log N$ servers per 1 ms. Taking this into account, the average workload of an i^3 server in the pure association setup case is

$$W_{\mathrm{s}} = \frac{3\lambda M\log N}{N} \quad (11)$$

In case of the optimized setup, the workload is

$$W_{\mathrm{so}} = \frac{\lambda M\log N}{N} \quad (12)$$

[18] In our current analysis we have assumed that the i^3 servers are distributed uniformly over the Internet with the average transfer delay of 100ms. If the servers were closer to each other, the delay would decrease accordingly.

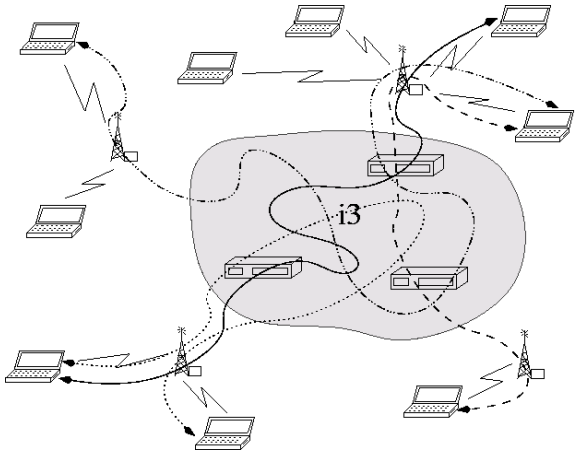


Figure 7: Mobile peers communications in Hi^3.

The workload can be considered as the average number of requests to an i^3 server, as well as the average number of packet relays (μ-processing) performed by the infrastructure per time unit, e.g., 1 ms.

5.5.2 Estimation of the infrastructure size

The major constituent of the association setup latency, that Hi^3 can directly influence on, is τ^{Hi3}. The location of a host A is an essentially more important factor to τ_A^{Hi3}, than the properties of the infrastructure.

Consider the case of the pure association setup to estimate N. According to Eq. (11), $3\log N = \dfrac{WN}{\lambda M}$. Applying the representation of τ^{Hi3} from Eq. (1a) one obtains $\tau^{\mathrm{Hi3}} = \dfrac{NW}{4\lambda M}(\tau+\mu)$. This allows to estimate the number of i^3 servers as

$$N = 4\frac{\lambda M \tau^{\mathrm{Hi3}}}{W(\tau+\mu)} \tag{13}$$

The same equation can be derived based on Eqs. (12) and (2a) in case of the optimized base exchange.

Estimation based on Eq. (13) must be used carefully. One should keep in mind that for a correct result the equalities

$$\frac{4\tau^{\mathrm{Hi3}}}{\tau+\mu} = C\log N \quad \text{and} \quad \frac{W}{\lambda M} = C\frac{\log N}{N}$$

have to be hold, where the constant $C = 3$ for case of the pure association setup and $C = 1$ for the optimized one. This is due to (1a), (2a), (11), and (12).

Example 2. Let us study the properties for a particular instance of Hi^3 with $N = 2^9 = 512$ i^3 servers and $M = 10^7$ mobile hosts. Assume the following typical values for the other parameters:

$$\lambda = 10^{-6}\ \mathrm{ms}^{-1},\ \tau = 100\ \mathrm{ms},\ \mu = 1.0\ \mathrm{ms},$$
$$\mu_{\mathrm{pr}} = 150\ \mathrm{ms},\ \tau_A^{\mathrm{Hi3}} = 200\ \mathrm{ms}\ \ \forall\ \text{mobile host } A.$$

In case of the pure association setup, $4\tau^{\mathrm{Hi3}} \approx 2700$ ms and $W_{\mathrm{s}} \approx 53\%$. The average latency in the worst case is

$$L_{\mathrm{s}}^{\mathrm{Hi3}} \approx 800 + 2700 + 800 + 300 = 4600\ \ (\mathrm{ms})$$

This is very conservative scenario and the infrastructure workload is medium.

For an optimized setup, $4\tau^{\mathrm{Hi3}} \approx 900$ ms and $W_{\mathrm{so}} \approx 18\%$. The average latency in the worst case is

$$L^{\mathrm{Hi3}} \approx 600 + 900 + 200 + 500 + 300 = 2500\ \ (\mathrm{ms})$$

Therefore, the infrastructure of the same size provides for the optimized setup less workload and internal transit time by 3 times comparing with the pure setup. The over overall latency is less by almost half.

Example 3. Let us use the same assumptions as in Example 2 while the number of i^3 servers is smaller, $N = 256 = 2^8$ (half reduction comparing with the previous example). In this case of the pure setup $4\tau^{\mathrm{Hi3}} \approx 2400$ ms and $W_{\mathrm{so}} \approx 94\%$. The workload is very high while the internal transit time is slightly less due to the half reduction of the infrastructure size. The latency is

$$L^{\mathrm{Hi3}} \approx 800 + 2400 + 800 + 300 = 4300\ \ (\mathrm{ms})$$

Indeed, this infrastructure is not able in general to provide the satisfactory service for $M = 10^7$ mobile nodes.

The case of optimized setup results in $4\tau^{\mathrm{Hi3}} \approx 800$ ms, $W_{\mathrm{so}} \approx 31\%$, and the latency is

$$L^{\mathrm{Hi3}} \approx 600 + 800 + 200 + 500 + 300 = 2400\ \ (\mathrm{ms})$$

The infrastructure is reasonably loaded and the latency can be considered satisfactory. The latency can be decreased even more by further reducing the number of i^3 servers, however this also increases the workload.

5.5.3 Discussion

Estimation (13) is valid for the case of inexhaustible communications of a homogeneous set of mobile peers via a large homogeneous infrastructure. Inexhaustible communications are the worst case of network behavior but possible in overloaded environments. The homogeneity assumption is too strong for detailed modeling of modern networks, but it captures average properties and can be useful to describe some parts of the infrastructure, e.g., large homogeneous i^3 clusters.

The infrastructure size has to be reduced to decrease the latency. A large infrastructure reduces the performance; in the worst case the size N is proportional to the crossing time τ^{Hi3}.

The workload is also a primary parameter in the estimation. One can control the average workload of an i^3 server by varying $0 < W\mu < 100\%$ (computational utilization) and $0 < W\tau < B$ (throughput utilization). Utilization can be reduced by increasing the infrastructure size.

Although the size N is proportional to the number of hosts M, this does not result in a large infrastructure comparable with the size of the hosts set. A HIP association setup is relatively rare event and growth of N is reduced by small λ (see Examples 2 and 3). Furthermore, this proportionality of N and M is due to the inexhaustible communication and uniformity assumptions for Eqs. (11) and (12), i.e., the worst case. This is easy to understand, if one considers a case when all hosts simultaneously initiate HIP setups. Due to the uniformity assumption, a part of the infrastructure proportional in size to M is involved in serving these requests.

In practical terms, our analysis shows that even for millions of mobile hosts an infrastructure consisting of just a few hundred nodes is sufficient.

6 Experimentation experience

It is obvious that a direct connection between C and S has better throughput and latency than solutions based on i^3, Hi^3 or Mobile IP. Our goal is a high-level estimation of the overhead. The subsequent analysis of the latter allows us to compare the efficiency of alternative solutions.

6.1 The Hi^3 implementation

We used the Boeing Linux HIP implementation [13] to create an Hi^3 prototype. The other available HIP implementation for the Linux kernel [4] supports only IPv6 at the moment, although an IPv4 port is forthcoming. The Hi^3 prototype currently supports only the basic HIP exchange over i^3. The implementation, in addition to HIP and i^3, is less than 500 lines of code. In the implementation, HIP control packets including the IP header are tunneled through i^3 servers running on PlanetLab. IP addresses are not yet hidden in the prototype; some effort is required to fix difficulties with accepting HIP packets with changing IP addresses in the HIP implementation.

Our code is publicly available for download. The implementation has been recently updated to the Boeing HIP version 3.1 and i^3 version 0.2.

6.2 Data throughput and latency

We consider the following scenario of communication. A mobile host C (e.g. laptop) as an initiator contacts a stationary server S. This corresponds to Hi^3 scenarios introduced in Sections 4.1 and 4.2, and to Mobile IP scenarios, listed in Appendices A.1 and A.2.

For experiments with i^3 and Hi^3, we used both a client and a server that reside in a local network. Both hosts register their identities on i^3 servers on PlanetLab. As a base line, we use a direct TCP connection between the hosts via IP. This is a conservative scenario, as it presents the worse case for i^3 and Hi^3. We measured TCP connections setup, TCP throughput and round-trip time. The results given in Table 1 are averaged over a small number of repetitions, as the variance of measurements was negligible.

The results above are heavily influenced by lack of i^3 shortcuts, and by the fact that all i^3 servers were located in the US while the client and the server were located in Europe. Furthermore, some PlanetLab servers are limited to use only a certain bandwidth by the system administrators and there is a smaller internal limit per an executing task.

The TCP connection setup is the time from sending the SYN segment to the receiver, till the time when an acknowledgment of SYN-ACK is sent. In case of direct communication, all packets are exchanged in a local network only. For i^3, all packets flow through i^3 servers with additional time required for packet routing inside i^3 infrastructure. In Hi^3, the HIP base exchange packets flow through the i^3 infrastructure, while the TCP three-way handshake occurs directly between the sender and the receiver with additional overhead of IPsec encryption.

Table 1: Performance results of IP, i^3, and Hi^3 for throughput and latency.

Solution	TCP setup latency, ms	TCP throughput, Mbyte/s	RTT, ms
IP	0.4	10	0.2
i^3	620	0.08	280
Hi^3	900	5	1

The TCP connection throughput is measured with `ttcp` omitting the connection setup. For direct communication, packets flow in a local network and throughput is the highest. In i^3, all data flow through proxies and the infrastructure that results in a high overhead. in Hi^3, data flow directly but encrypted with IPsec that decreases the throughput over plain IP approximately by half.

The RTT is measured with `ping` after the initial connection setup. For plain IP, the definition of RTT is standard. For i^3, the RTT is taken after the client and the server have cached the location of the triggers. For Hi^3, the RTT is taken within an IPsec tunnel directly between the client and the server.

Despite the conservativeness of the scenario, the throughput and latency of Hi^3 remain at a reasonable level with plain IP. Even for such demanding applications as multimedia conferencing, the throughput and latency of Hi^3 appear acceptable.

For comparison with Mobile IP, we consider two scenarios where the home agent is located 1) in the same network with the client and the server, 2) remote network. We did not yet carry out measurements comparing $Hi^3$with MIP. The comparison is based on empirical analysis and the Mobile IP performance results from the literature.

In the first case, the setup latency is increased compared with the plain IP. According to Table 1, TCP connection setup is close to $2 * \mathrm{RTT}(C, S)$ where 1.5*RTT is the communication latency and 0.1 ms is the end point processing. In case 1) above, the latency grows by two times due to an additional point of indirection. Therefore, we expect the TCP connection setup for Mobile IP to be 0.8 ms.

In case 2), assuming that the home agent is located as far as the i^3 server in our testbed, RTT(C,i^3)=RTT(C,HA)=200 ms. As in Section 5.4.2, we estimate the TCP connection establishment as 3*RTT=600 ms. This is comparable with i^3, but the latter requires more time for routing packets inside the infrastructure.

Considering Mobile IP with route optimization, TCP throughput and RTT in cases 1) and 2) are similar to plain IP or Hi^3(if Mobile IP with IPsec is used). As the result, we can conclude that Mobile IP has overhead comparable with Hi^3.

6.3 Handover performance

Our theoretical analysis shows that the HIP handover delay is less than in Mobile IP, see Section 5.4. This is confirmed with experimental study of Jokela *et al.* [17], where they show that the average delay from the beginning of the handover until the recovery of the TCP stream was 8.05 seconds for Mobile IPv6 and 2.46 seconds for HIP. However, for Hi^3 this difference should be smaller due to additional delays in the infrastructure.

Table 2: Code size for HIP, i^3, Mobile IP, IKE and SIIT.

Component	Number of lines
HIP (Boeing hipd v 3.1)	11714
i^3 (v 0.2)	29664
Hi^3 additions	378
MIPv6 (MIPL mipv6-2.0-rc1)	20174
MIPv4 (Dynamics 0.8.1)	26908
IKE (Racoon 2004-08-18)	41374
SIIT (Linux SIIT 1.0.5)	2092

6.4 Code size

To get a rough figure for comparing the complexity of the solutions, we ran `wc -l` over the source code of the various software components. The results are given in Table 2. The total size of Hi^3 is about 42000 LoC, while the total size of Mobile IPv4, v6, IKE, and SIIT, without integration, is about 64000 LoC. Furthermore, Hi^3 supports multi-homing (for both IPv4 and v6) and mobility between IPv4 and IPv6 while a solution based on the listed Mobile IP, IKE, and SIIT[19] [27] modules would not. Hence, the Hi^3 implementation is clearly simpler and provides more functionality than a corresponding implementation based on the current IETF standards.

7 Conclusions

In this paper, we addressed in an integrated manner the problems of mobility, multi-homing, and IP-layer security, including denial-of-service and distributed denial-of-service protection. Relying on the Host Identity Protocol (HIP) proposal, we separated the end-point identifier and locator functions of IP addresses and introduced a new sublayer to the TCP/IP stack. Based on the observation that a HIP rendezvous server and a single Internet Indirection Infrastructure (i^3) server node provide functionally the same service, we integrated i^3 with HIP, resulting in a clear separation of HIP control packets and user data packets. To provide protection against distributed denial of service attacks, we added an optional layer of IP-address-hiding middle boxes at the data path. These middle boxes are controlled by the end-hosts, making deployment and accountability easy. Finally, we separated service and host identifiers from each other. The resulting service identifiers are secure (but not human friendly) and act as first class citizens. Mere possession of a single service identifier is sufficient for creating a secure connection with an instance of the service.

We compared the resulting system to its origins, HIP and i^3, and to Mobile IP. Our qualitative analysis shows that the system is more robust and secure than either of the base systems. Compared to Mobile IP, the difference in robustness and security seems even larger. The system also provides more functionality that Mobile IP, by supporting end-host multi-homing and integrating IPv4 and IPv6 in an elegant manner.

By using conservative mathematical models, we estimated performance and scalability of the system. Our initial results show that even with relatively large numbers of mobile nodes, an infrastructure of a few hundred nodes is sufficient. The analysis also shows that the signaling latency in the system is dominated by inter-node transmission times within the infrastructure. Compared to Mobile IP, this causes a second of additional delay at the first contact. On the other hand, in the case of connectivity loss due to simultaneous movements, our system appears to perform better than Mobile IP.

Our initial performance measurements show that in terms of throughput and data path latency, our system exceeds i^3 and is comparable to Mobile IP. In terms of complexity (as measured by the implementation line count), our system is clearly simpler than a system based on current IETF standards providing the same functionality.

[19] SIIT is an IETF standard for making IPv4 and IPv6 interoperate.

Acknowledgments

The authors want to thank Jari Arkko, Börje Ohlman, Jukka Ylitalo, Heikki Mahkonen and Jan Melen for fruitful discussions on this problem space. The authors are also grateful to Tom Henderson, Ion Stoica, Karthik Lakshminarayanan, Dilip Joseph, Miika Komu, Teemu Koponen, and Anthony Joseph.

References

[1] D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica. Towards a more functional and secure network infrastructure. Technical Report UCB/CSD-03-1242, University of California at Berkeley, 2003.

[2] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the Internet. In *Proc. of ACM SIGCOMM'04*, pages 343–352, Aug. 2004.

[3] F. Bao, R. Deng, Y. Qiu, and J. Zhou. A scheme of mobile firewall in Mobile IPv6. Work in progress, draft-qiu-mip6-mobile-firewall-00.txt, Feb. 2005.

[4] C. Candolin, M. Komu, M. Kousa, and J. Lundberg. An implementation of HIP for Linux. In *Proc. of the Linux Symposium*, July 2003.

[5] C. Candolin and P. Nikander. IPv6 source addresses considered harmful. In *Proc. of Sixth Nordic Workshop on Secure IT Systems*, Nov. 2001.

[6] D. Clark, R. Braden, A. Falk, and V. Pingali. FARA: Reorganizing the addressing architecture. *ACM Computer Communication Review*, 33(4):313–321, 2003.

[7] D. Crocker. Multiple address service for transport (MAST): An extended proposal: draft-crocker-mast-01.txt, Sept. 2003. Work in progress. Expired in February, 2004.

[8] V. Devaparalli. Mobile IPv6 operation with IKEv2 and the revised IPsec architecture, Feb. 2005. Work in progress.

[9] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylönen. SPKI certificate theory. RFC 2693, IETF, Sept. 1999.

[10] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. PeerNet: Pushing peer-to-peer down the stack. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Berkeley, CA, USA, Feb. 2003. Springer-Verlag.

[11] P. Francis. IPNL: A NAT-extended Internet architecture. In

Proc. of ACM SIGCOMM'01, San Diego, CA, USA, Aug. 2001.

[12] T. R. Henderson. Host mobility for IP networks: A comparison. *IEEE Network*, 17(6):18–26, Nov. 2003.

[13] T. R. Henderson, J. M. Ahrenholz, and J. H. Kim. Experience with the host identity protocol for secure host mobility and multihoming. In *Proc. of the IEEE Wireless Communications and Networking Conference (WCNC'03)*, Mar. 2003.

[14] M. Ishiyama, M. Kunishi, and F. Teraoka. An analysis of mobility handling in LIN6. In *Proc. of International Symposium on Wireless Personal Multimedia Communications (WPMC'01)*, Aug. 2001.

[15] D. B. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. RFC 3775, IETF, June 2004.

[16] P. Jokela, P. Nikander, J. Melen, J. Ylitalo, and J. Wall. Host identity protocol: Achieving IPv4 - IPv6 handovers without tunneling. In *Proc. of Evolute workshop 2003: "Beyond 3G Evolution of Systems and Services"*, Nov. 2003.

[17] P. Jokela, T. Rinta-Aho, T. Jokikyyny, J. Wall, M. Kuparinen, H. Mahkonen, J. Melen, T. Kauppinen, and J. Korhonen. Handover performance with HIP and MIPv6. In *Proc. 1st International Symposium on Wireless Communication Systems, ISWCS'04*, Sept. 2004.

[18] T. Koponen, A. Gurtov, and P. Nikander. Application mobility with Host Identity Protocol. In *Proc. of NDSS Wireless and Security Workshop*, San Diego, CA, USA, Feb. 2005. Internet Society.

[19] J. Laganier, T. Koponen, and L. Eggert. Host identity protocol (HIP) registration extension: draft-koponen-hip-registration-00, Feb. 2005. Work in progress.

[20] R. Moskowitz and P. Nikander. Host identity protocol architecture: draft-ietf-hip-arch-02.txt, Jan. 2005. Work in progress. Expires in August, 2005.

[21] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host identity protocol: draft-ietf-hip-base-02, Feb. 2005. Work in progress. Expires in August, 2005.

[22] P. Nikander and J. Arkko. Delegation of signalling rights. In *Proc. of Security Protocols, 10th International Workshop*, pages 203–212, Cambridge, UK, Apr. 2002. Springer.

[23] P. Nikander, J. Arkko, and T. Henderson. End-host mobility and multi-homing with host identity protocol: draft-ietf-hip-mm-01, Feb. 2005. Work in progress.

[24] P. Nikander, J. Arkko, and B. Ohlman. Host Identity Indirection Infrastructure (Hi3): draft-nikander-hiprg-hi3-00, June 2004. Work in progress (Expired).

[25] P. Nikander, H. Tschofenig, T. Henderson, L. Eggert, and J. Laganier. Preferred alternatives for tunnelling HIP (PATH): draft-nikander-hip-path-00.txt, Feb. 2005. Work in progress. Expires in August, 2005.

[26] P. Nikander, J. Ylitalo, and J. Wall. Integrating security, mobility, and multi-homing in a HIP way. In *Proc. of Network and Distributed Systems Security Symposium (NDSS'03)*, San Diego, CA, USA, Feb. 2003. Internet Society.

[27] E. Nordmark. Stateless IP/ICMP translation algorithm (SIIT). RFC 2765, IETF, Feb. 2000. `http://www.ietf.org/rfc/rfc2765.txt`.

[28] J. Postel and S. Crocker. A possible protocol plateau. RFC 48, Apr. 1970.

[29] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN: Simple traversal of user datagram protocol (UDP) through network address translators (NATs). RFC 3489, IETF, Mar. 2003.

[30] H. Soliman and G. Tsitsis. Dual stack mobile IPv64: draft-soliman-v4v6-mipv4-01.txt. Internet draft, IETF, Oct. 2004. Work in progress. Expires in April 2005.

[31] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. of ACM SIGCOMM'02*, Pittsburgh, PA, USA, Aug. 2002.

[32] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. of ACM SIGCOMM'01*, San Diego, CA, USA, Aug. 2001.

[33] H. Tschofenig, A. Nagarajan, V. Torvinen, J. Ylitalo, and J. Grimminger. NAT and firewall traversal for HIP: draft-tschofenig-hiprg-hip-natfw-traversal-00, Oct. 2004. Work in progress. Expires in April 18, 2005.

[34] M. Walfish and H. Balakrishnan. The location/identity split is useful for middleboxes, too. In *Proc. of Workshop on HIP and Related Architectures*, Nov. 2004. `http://hiprg.piuha.net/workshop/`.

[35] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. In *Proc. of the 7th USENIX Symposium on Operating System Design and Implementation (OSDI 2004)*, San Fransisco, CA, USA, Dec. 2004. ACM Press.

[36] J. Ylitalo. Re-thinking security in network mobility. In *Proc. of NDSS Wireless and Security Workshop*, San Diego, CA, USA, Feb. 2005. Internet Society.

[37] J. Ylitalo, J. Melen, P. Nikander, and V. Torvinen. Re-thinking security in IP based micro-mobility. In *Proc. of 7th Information Security Conference (ISC04)*, Sept. 2004.

[38] J. Ylitalo and P. Nikander. BLIND: A complete identity protection framework for end-points. In *Proc. of the Twelfth International Workshop on Security Protocols*, Apr. 2004.

[39] J. Ylitalo and P. Nikander. A new name space for end-points: Implementing secure mobility and multi-homing across the two versions of IP. In *Proc. of the 5th European Wireless Conference, Mobile and Wireless Systems beyond 3G (EW2004)*, pages 435–441, Feb. 2004.

A Mobile IP scenarios

We describe two Mobile IP scenarios in detail. In the first scenario, a Mobile Node (MN) establishes a connection with a Corresponding Node (CN) using Mobile IPv6 Route optimization [15]. In the second scenario, we consider the case where two Mobile Nodes move at the same time. These two scenarios form a baseline that we compare Hi^3 against. They were selected since in the first scenario Mobile IP performs optimally while in the second one its asymmetry and consequent problems are highlighted.

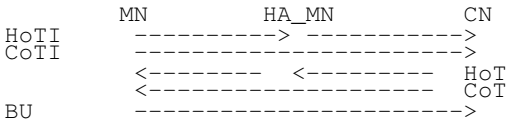


Figure 8: Route optimization in MIPv6.

A.1 Initial contact

Figure 8 illustrates the message exchange, where a Mobile Node (MN) establishes a connection with a Corresponding Node (CN) using route optimization. For brevity, we only consider Mobile IPv6.

When initiating the connection, the MN sends two messages in parallel to the CN: HoTI and CoTI. The HoTI (Home address Test Initiation) message is sent through the Home Agent (HA) using reverse tunneling. The CoTI (Care-of address Test Initiation) message is sent directly to the CN. As the CN receives these messages, it replies with a HoT and CoT, respectively. The HoT (Home address Test) and CoT (Care-of address Test) messages are sent individually as the corresponding Initiation messages are received, without any state being created at the CN.

The purpose of the HoT and CoT messages it to make sure that the MN is reachable, at the same time, both at its current Care-of address and via its Home address. Consequently, the HoT and CoT messages carry a nonce, i.e., a fresh random number. By showing to have received the nonces, the MN is able to prove the CN that it can receive messages sent to both addresses.

Once the MN has received both a HoT and a CoT message from the CN, it combines the nonces in the messages and sends a Binding Update (BU). When the CN receives a BU, it verifies that the BU correctly reflects the nonces sent in recent HoT and CoT messages. If this test passes, the CN creates a piece of binding state, starting to send messages directly to the MN's current care-of address.

Before the binding state is created, the MN and CN can communicate, but only through the Home Agent. Consequently, the nodes can establish any transport layer connection in parallel with the binding procedure. Hence, with mobile IP the only cost at an initial contact is the additional delay caused by packets flowing through the Home Agent.

A.2 Establishing a TCP connection

While in HIP a TCP connection can be opened only after the HIP base exchange, in Mobile IP the TCP connection can be opened in parallel with the binding process. As a consequence, the TCP connection delay in Mobile IP is dominated by the TCP handshake flowing through the home agent, not by the binding procedure. However, HIP also provides IPsec-comparable end-to-end security while Mobile IP does not. Hence, while Mobile IP allows TCP connections to be opened much faster than HIP or Hi^3 does, the difference is not that large if we assume that in addition to Mobile IP signaling also IKE must be run. Unfortunately, such an analysis would be fairly complex due to the potential different timing scenarios caused by the Mobile IP binding process and the IKE exchange running in parallel, resulting in some IKE messages flowing through the home agent while others flowing directly between the hosts. Consequently, for brevity, such an analysis falls beyond the scope of this paper.

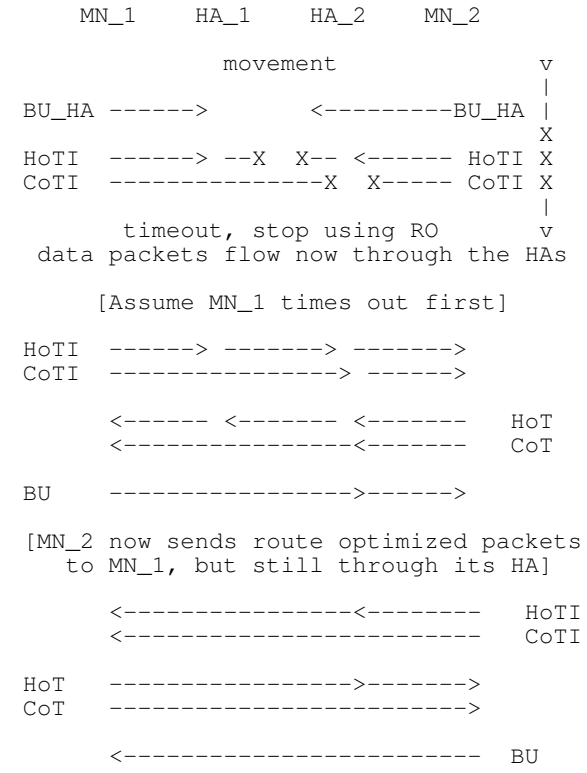


Figure 9: Binding update in MIPv6.

A.3 Simultaneous movement

Consider a case where two Mobile Nodes, MN_1 and MN_2, communicate with each other using route optimization, and then move at the same time, as illustrated in Figure 9. As a result of the simultaneous movements, the nodes initiate the binding update procedure towards their home agents, HA_1 and HA_2, and each other. Due to the protocol asymmetry, they assume that their peers have not moved, causing initial packet loss and some complex protocol dynamics. Note that the nodes can send the binding updates to their home agents without the need of return routability testing, as it is assumed that there is always a trust relationship between the mobile node and its home agent.

Initially, once the nodes have moved, they send Binding Updates to their Home Agents. This allows other nodes to reach them, without route optimization, as soon as the Binding Updates are processed by the Home Agents. However, any nodes that use route optimization continue to send to the previous care-of address of the mobile nodes.

Already in parallel with Binding Updates towards the Home Agents, the nodes may start to update the route optimization cache towards their peers. Now, in our case where both nodes are mobile and use route optimization, they will send the HoTI and CoTI messages towards the previous care-of address of their peer, as they assume that the peer has not moved.