Hi3: An Efficient and Secure Networking Architecture for Mobile Hosts

Andrei Gurtov, Dmitry Korzun, Andrey Lukyanenko, and Pekka Nikander
Helsinki Institute for Information Technology (HIIT), Finland
Department of Computer Science, Petrozavodsk State University, Russia

April 22, 2008

Abstract

The Host Identity Indirection Infrastructure (Hi3) is a networking architecture for mobile hosts, derived from the Internet Indirection Infrastructure (i3) and the Host Identity Protocol (HIP). Hi3 has efficient support for secure mobility and multihoming, which both are crucial for future Internet applications. In this paper, we describe and analyze Hi3 in detail. Compared to existing solutions, Hi3 achieves better resilience, scalability and security. Both our analysis and early measurements support the notion that Hi3 preserves the best of both approaches while improving performance compared to i3 and enhancing flexibility and security compared to HIP.

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., [4]), the current standards for adding mobility and security to the IP stack, at best, represent independent *point solutions* that do not integrate easily and sometimes interact badly.

In this paper, we describe the Host Identity Indirection Infrastructure ($Hi3$), analyze its operation and latency, and provide early measurement results. Compared to the Internet Indirection Infrastructure ($i3$) [28] and the Host Identity Protocol (HIP) [19], upon which $Hi3$ is based, $Hi3$ preserves the best of both approaches while greatly improving performance compared to $i3$ and enhancing flexibility and security compared to HIP.

In particular, we argue that an overlay infrastructure such as $i3$ is ideal for providing a secure, integrated *rendezvous infrastructure* for HIP [14], 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 $Hi3$ can be summarized as follows:

- Inheriting from HIP, $Hi3$ *integrates* mobility with end-host-based multi-address multihoming and basic security mechanisms. It also makes IPv4/IPv6 integration easy, including mobility and multi-homing across IPv4 and IPv6 [11, 22, 35].

[1] Mobility was considered as early as 1970 [26] but it was later decided to leave it out from the architecture.

- 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, $Hi3$ 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 plain IP, i.e., clearly better than the performance of systems based on pure overlay routing.

$Hi3$ 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, $i3$/Secure-$i3$, and IPsec-aware NAT is presented. In Section 3, we describe the $Hi3$ network architecture in detail. In Section 4, we analyze the latency of typical requests to $Hi3$. In Section 5, our implementation experience is described and measurement results are presented. Section 6 concludes the paper.

2 Background

$Hi3$ is based on ideas from $i3$, Secure-$i3$, and HIP. Furthermore, for protecting the data traffic, $Hi3$ 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 [19, 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 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 $i3$ identifiers used in triggers (see Section 2.2), but they are constructed entirely cryptographically.

[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 [25].

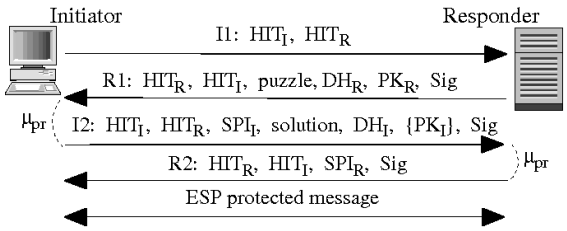


Figure 1: HIP base exchange.

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 [20] 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 μ_{pr} in Figure 1.

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 [22], and update this information as needed. For initial rendezvous, simultaneous movement, and location privacy, the HIP architecture includes the *rendezvous server* concept [14]. A HIP rendezvous server simply forwards HIP control packets to a registered HIP host. It can also provide a two-way forwarding function [25]. Functionally, a rendezvous server is similar to a single $i3$ server, as it forwards a packet to a registered IP address, based on the destination HIT in the packet.

2.2 Internet Indirection Infrastructure ($i3$)

To ease the deployment of services, Stoica *et al.* proposed an $i3$ overlay network that offers a rendezvous-based communication abstraction [28]. 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 $i3$ infrastructure to receive all packets that have the destination identifier id.

$i3$ 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-$i3$ 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-$i3$, 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-$i3$. In Secure-$i3$, 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 $i3$, a *proxy* located on the client and the server must be used. The proxy transparently intercepts DNS requests and forwards data packets to the $i3$ infrastructure. Recently, a capability to send data directly between the client and the server has been added to $i3$. Known as *shortcuts*, it allows efficient data transfer between hosts, but does not offer currently any cryptographic data protection.

Since $Hi3$ relies on features from the basic $i3$ architecture and the Secure-$i3$ extension, from here on we do not make a difference between them.

2.3 SPI multiplexed NAT

As argued by Walfish *et al.* [31,32] 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 $i3$ 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].

SPI multiplexed NAT (SPINAT), as proposed by Ylitalo *et al.* [30, 34], 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 [4], MAST [5], PeerNet [6], IPNL [7], and LIN6 [10].

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

[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 support in the form of STUN [27] or similar servers.

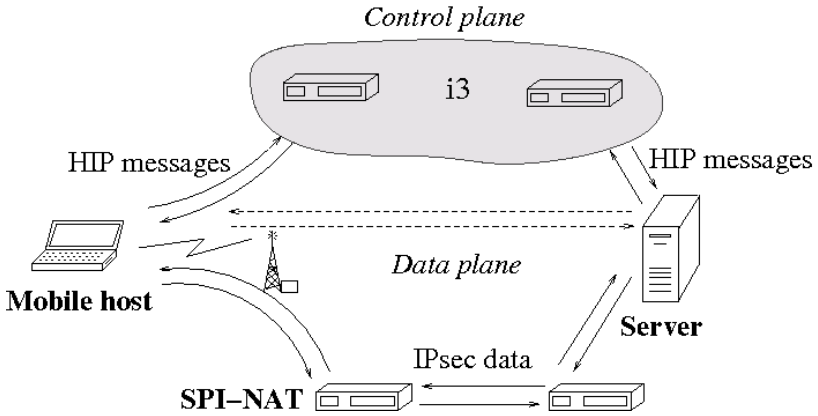


Figure 2: $i3$ as an instantiation of the HIP rendezvous server.

deployment and operational concerns into account.

3 $Hi3$ architecture

In this section, we describe the $Hi3$ architecture in detail. More specifically, we consider the particulars of separating session control, actual data delivery, and service naming. We analyze problems that induced by this separation and present the solutions. Additionally, we qualitatively analyze the key advantages of $Hi3$ and discuss perspective of the design.

The original concept of $Hi3$ was proposed by Nikander *et al.* in the sketch [8, 23], where they observed that a HIP rendezvous server and a single $i3$ server are functionally close. Therefore, the basic idea is to allow direct, IP-based end-to-end traffic while using an indirection infrastructure to route the HIP control packets.

We enhance this proposal in the following way. First, we generalize the $i3$ representation of the host identity namespace such that peers can use separate identifier layers for service and for hosts. Second, we concentrate on the $Hi3$ design for the control traffic and introduce all available control messages. Third, our discussion of $Hi3$ advantages and perspective gives the most comprehensive overview available at present.

3.1 Separating control, data, and naming

In HIP, a rendezvous server is used to fully support association setup between two end-hosts, simultaneous movement, location privacy, and third party referrals. The concept of the HIP rendezvous can be enhanced to an overlay rendezvous infrastructure, a distributed and decentralized instantiation of the HIP rendezvous server. In the $Hi3$ solution, an $i3$ network implements this infrastructure. Figure 2 illustrates this idea.

The infrastructure forms *the $Hi3$ control plane*, relaying HIP signaling messages. Data traffic flows directly between end-hosts, using plain IP routing and forming *the $Hi3$ data plane* (shown with dashed lines in Figure 2). HIP and $i3$ provide secure communications for the control plane; IPsec and IPsec-aware NAT (SPINAT) give basic protection to the data plane. This concept of $Hi3$ is shown in Figure 3.

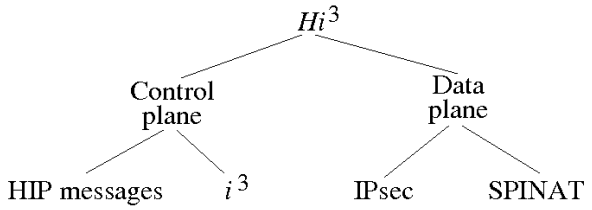


Figure 3: Separating data and control in $Hi3$, and related technologies

Inherited from HIP, $Hi3$ uses separate identifiers for location (IP addresses) and endpoint (HIT). HIT acts as an identifier for a public $i3$ trigger, reflecting in the infrastructure the HIP-based separation of naming. Note that if $i3$ identifiers are longer than 128-bits, then prefixes to HITs are applied.

For a host A, the pair of triggers $[\mathrm{HIT}_A \mid \mathrm{ID}_A]$ and $[\mathrm{ID}_A \mid \mathrm{IP}_A]$ can be stored in $i3$, where HIT_A is a public $i3$ identifier of A and ID_A is its private $i3$ identifier. The latter is constructed by A according to the constrained IDs technique [1, 16].

In this mere form, a set of all pairs of public and private identifiers is the $i3$ representation of the host namespace. Whenever a host needs to contact a peer, the host uses peer's public identifier; however, the messages flow obligatory via the private trigger too. Figure 4 shows a schematic distribution of the HIT-based triggers in $i3$ for communication between end-hosts C and S.

We enhance this mechanism to support a higher naming layer. Public trigger identifiers are used for association setup between a client and a server. When the association has been completed, the server creates a private identifier for the client (or uses an existed identifier for a group of clients). After that, only the private trigger is used to relay control messages from the client to the server.

Therefore, the $i3$ 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 can use fresh, newly generated host identifiers (or identifiers for a group of hosts) as private trigger identifiers. To secure the binding between the public and private triggers, i.e. between the service and host identifiers, the cryptographic delegation is used [21].

3.2 The data plane

In this section, we describe the $Hi3$ data plane that protects end-to-end data traffic and supports multiple IP realms.

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

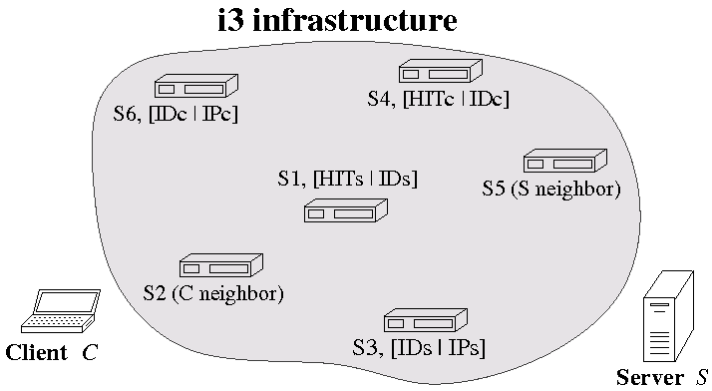


Figure 4: Schematic distribution of HIT-based triggers in $i3$ for $C \leftrightarrow S$ communication.

the target host.

To protect against distributed denial-of-service, we extend the notion of using IPsec-aware middle boxes [23]. A number of SPINATs[4] (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 $i3$ shortcuts [28] 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. Mobility performance and DoS resistance of SPINAT has been measured by Ylitalo *et al.* [33]. The results suggest that the efficiency of data plane is not significantly reduced by the presence of SPINATs.

As described in [30], 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 [15] 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

[4]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.

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 is 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.

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 $Hi3$ the IP source address field is no longer needed. 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 [20]. Hence, we surmise that the source address field could be used to record the actual path taken by the packet [3].

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 $i3$.

3.2.2 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 $i3$ 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 [27] or ICE [24].

To make hosts reachable by the $i3$ 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 $i3$ 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 $i3$ 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.

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.

3.3 The control plane

The control plane is used to relay HIP messages in two cases. First, when before direct end-to-end communication two end-hosts establish a HIP association. In this case the main benefit of using the control plane is protection against DoS; the IP addresses of both hosts are not revealed until mutual authentication is completed. Second, when having an established connection the hosts lose the direct end-to-end connectivity. Such a case is important for end-host mobility, e.g., the connectivity is lost after a simultaneous movement of both hosts. Therefore, the control plane is a trusted third-party that aims in establishing and keeping the data plane connectivity between communicating peers.

Let C and S be two communicating end-hosts. We assume that C is a HIP initiator (e.g., a mobile client) and S is a HIP responder (e.g., a stationary Internet server). Figure 4 illustrates a distribution of their HIT-based public and private triggers in $i3$; for first contacts the peers use neighbor $i3$ nodes that they happen to know ($S2$ and $S5$).

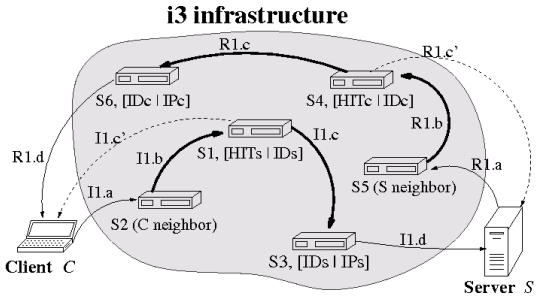
In the rest of this section, available request types to the control plane are described. Diagrams of packet flows of the requests are shown in Figure 5. Arrows represent paths, by which packets of requests follow. An arrow label is a packet name (e.g., I1 or R2) and a sequence mark (i.e., "a" is for the first part of the flow, "b" is for the next part, etc.). Thick arrows denote possible multi-hop Chord lookups [29], when the destination $i3$ node is not cached in the source $i3$ node.

3.3.1 Pure HIP association setup

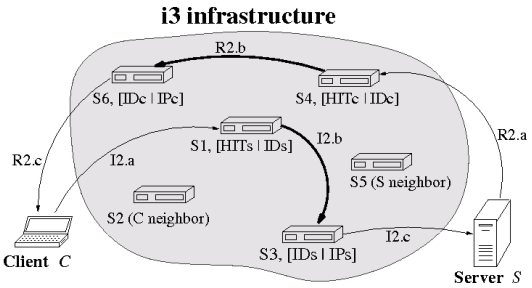
Figures 5(a) and 5(b) show the $Hi3$ establishment of a HIP association between C and S.

To establish a connection with a server, a client C sends an I1 packet to the IP address of a random $i3$ node it happens to have; in our case this node is $S2$ (path I1.a). The public trigger for S, HITs, is stored in $S1$, and $S2$ forwards the packet to $S1$ via $i3$ (path I1.b). The client obtains the correct $i3$ node for future contacts to S (path I1.c', in parallel with the primary branch I1.c–I1.d). The private trigger of S resides on $S3$, to which $S1$ forwards the packet (path I1.c), and finally $S3$ delivers it to S (path I1.d).

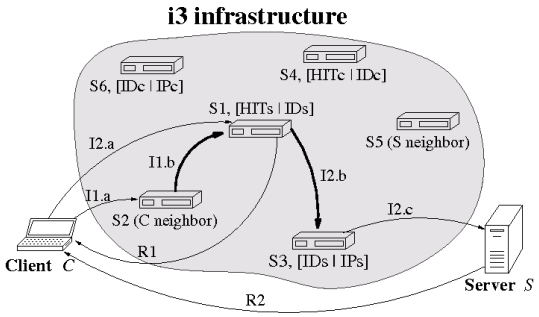
A similar procedure is followed by S to send an R1 reply packet to C. The neighbor $i3$ node $S5$ is contacted first (path R1.a). The public trigger for the client C, HITc, is stored in the node $S4$, to which $S5$ forwards the packet via $i3$. Then $S4$ notifies S about the correct $i3$ node for communicating with C (path R1.c', secondary branch) and forwards R1 to $S6$, which keeps the private trigger for the client (path R1.c). Finally, $S6$ delivers the packet to



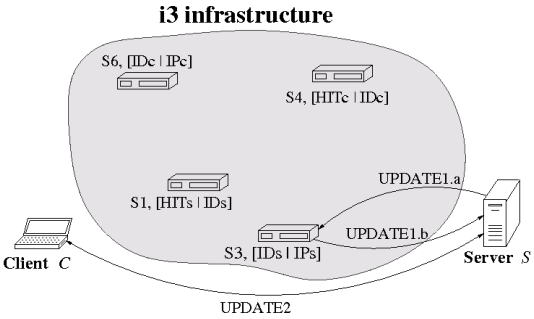
(a) Pure association setup: I1 and R1 packets



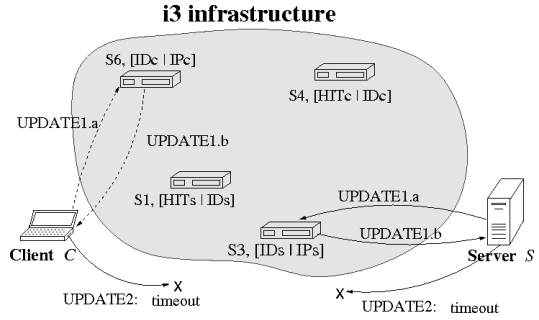
(b) Pure association setup: I2 and R2 packets



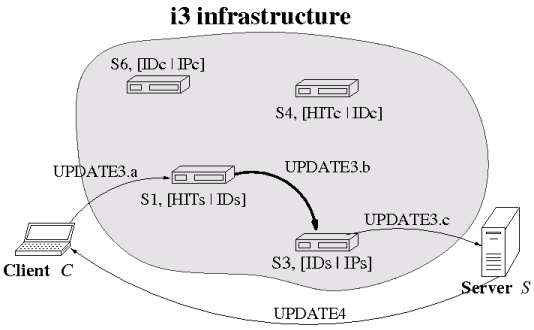
(c) Optimized association setup



(d) Location update



(e) Failure of the location update



(f) Double-jump resolving

Figure 5: Request types in $Hi3$.

the client (path R1.d).

The consequent I2–R2 exchange occurs in a similar manner, see Figure 5(b). The only difference is that the packets are sent straight to the $i3$ nodes keeping the public triggers, $S1$ and $S4$.

3.3.2 Optimized HIP association setup

In the pure form of association setup, the public and private triggers of both hosts must have been inserted to $i3$. It is, however, unreasonable to require a client to keep its triggers in $i3$ even temporarily. The solution is to delegate an initial part of the setup from S to $i3$. This way, the node $S1$, which keeps a public trigger of S, caches pre-computed R1 packets.

Figure 5(c) shows the optimized form of setup.

As in the pure association setup, C sends an I1 packet to the $S2$, its neighbor (path I1.a), and the packet is forwarded to $S1$ (path I1.b). Then, unlike in the pure setup, $S1$ replies directly to the client with an R1 packet since it has been cached (path R1). In this reply, $S1$ also notifies C about the correct node to contact S via $i3$.

The I2 packet is sent to $S1$ (path I2.a), then forwarded to $S3$ via $i3$ (path I2.b), and finally delivered to S (path I2.c). The packet is expected to contain the HIP LOCATOR parameter, listing the client's real IP address. The server replies with an R2 packet directly to the client (path R2). That is, the control plane is not involved in R2 delivering.

The optimization reduces the load of S, since it receives less packets and does not check the puzzle solution of C. Note that the cost of processing delegated is comparable with the cost of packet forwarding, and the main benefit of the optimization is due to the lower communication needs.

3.3.3 Location update

Both C and S can change their locations and, consequently, their IP addresses during the communication.

Typically, only one host changes its IP address and performs a *location update* at a time. If the change is due to the server, then S updates its private trigger in $i3$ (Figure 5(d)). The location update also causes the *HIP update exchange* [22] running over the data plane between C and S.

One update to $i3$ is sufficient independently on the number of hosts communicating with S via the private trigger. UPDATE1 and UPDATE2 packets can be sent in parallel. Therefore, no significant overhead is introduced, compared to HIP.

For client C, having a trigger pair in $i3$ is optional. Thus, if C changes its location, then the signaling packets, UPDATE2, run directly between hosts, and the control plane is not used.

3.3.4 Simultaneous host movement

It may happen that both hosts change their locations at once, an event known as the *double-jump problem*. Note that simultaneous mobility of C and S is rare compared with the usual location update.

Figure 5(e) and 5(f) show what happens if both C and S change their addresses simultaneously. The hosts update their private triggers in $i3$ (Figure 5(e), UPDATE1 packets). For C this update is optional (dashed arrows in the diagram). In parallel, the hosts start a HIP location update over the data plane, UPDATE2 packets. The exchange fails since each host uses the out-of-date IP address to contact the peer. This failure can be discovered by a timeout.

At this point the hosts need the control plane to recover from the double-jump (Figure 5(f)). The double-jump can be discovered by both hosts, but the client is responsible for starting the recovery. It sends the first packet of the HIP update exchange (UPDATE3, addressed to HIT_S) to S via $i3$. After receiving this packet, S continues the update talking directly to C.

Obviously, in this scenario the most delay is due to timeout for UPDATE2 packets. To improve this, C can send UPDATE2 and UPDATE3 packets in parallel, especially when the double-jump is likely. Unnecessary UPDATE packets are ignored by S.

3.3.5 HIT insertion

Let HIT_A be a HIT of a host A and IP_A be a recent IP address of A. There are two reasons for A to insert HIT_A into $i3$. First, A is a server, thus HIT_A has to be in $i3$ permanently; second, A is an initiator of a pure association setup, thus HIT_A is used by the responder.

To insert HIT_A, A constructs a private $i3$ identifier ID_A. Then A sends two requests to $i3$, namely inserting the public trigger $[\mathrm{HIT}_A \mid \mathrm{ID}_A]$ and private trigger $[\mathrm{ID}_A \mid \mathrm{IP}_A]$. Each request requires a Chord lookup to assign an $i3$ node, say $S0$, for storing the trigger. In successful case, $S0$ sends directly to A the acknowledgment, which contains a recent node's IP address.

3.3.6 HIT refreshment and HIT re-insertion

Trigger's time-to-live is limited. For instance, the recent implementation[5] applies the value of 30 sec. Keeping HIT_A alive in $i3$ requires the host A to refresh regularly both public and private triggers, sending the refreshment requests directly to the corresponding $i3$ nodes.

If the $i3$ node crashes, the trigger is lost. The end-host can re-insert it by sending the insertion or refreshment request to any $i3$ node. Similarly, if the trigger has been removed before the refreshment was received, then the trigger is re-inserted. However, the trigger might be located to another node, since $i3$ undergoes node joining and leaving.

In successful case, the acknowledgment is received. It always contains a recent IP address of the $i3$ node that keeps the trigger.

3.3.7 HIT removal

HIT removal happens either automatically after the public and private triggers expire, or explicitly by sending two requests. Although the latter case seems redundant, it can be important for a host to remove HIT as soon as possible when an attack via the triggers has been discovered.

3.3.8 A case with separate naming for hosts and service

Although the requests were designed without taking separating naming into account, the corresponding modification is clear. A public identifier (HIT) is used only for a first contact with the server. Whenever a client is trusted, it uses a given server's private trigger, e.g., for resolving from a double-jump or for establishing new connections.

3.4 Discussion of the design

Let us summarize the key design ideas of $Hi3$ architecture and their consequences. We also compare the pros and cons of using HIP and $i3$ versus their combination, $Hi3$.

[5] `http://i3.cs.berkeley.edu/impl/`

The integration of the $i3$ triggers mechanism allows using two name layers; private triggers identifiers form a host namespace like in HIP, and public triggers are considered as a service namespace. The latter is close to endpoint descriptors for applications in the namespace model of Komu *et al.* [12]. Hence, this $Hi3$ feature can be used to fill the gap between HIP and application layers.

When this separating naming is used, the control plane performance increases since a client talks not via the public trigger's node but straight via the node with the private trigger. In this case, the security can suffer since the rule of compulsory usage of public/private trigger pair is violated. This rule is due to Secure-$i3$, and its violation substantially reduces the infrastructure to the case of basic $i3$. Hence, extra security mechanisms should be applied like the cryptographic delegation [21] or the HIP registration extension [15].

Any successful request ends with the acknowledgment to the end-host. Some requests can fail owing to packet losses in a network or $i3$ inconsistency and unavailability. This problem is not very significant since $i3$ is based on top of Chord DHT, which is known as resilient to massive failure of nodes [29], to dynamic node joining and leaving [17], and to pathological states [18]. Therefore, the problem happens rarely, and a simple resubmission of a request after a timeout is adequate.

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 protection against DoS 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 [6]. 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 $i3$ include better protection against DoS, support for simultaneous mobility, and higher fault-tolerance when using a DHT with data replication. Disadvantages of $i3$ include reliance on an extensive infrastructure, server scalability, use of UDP, lack of traffic encryption, and complexity of $i3$ as an overlay network. There is limited experience with widespread $i3$ 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 $i3$ infrastructure would likely prove burdensome, and by mutual agreement, the client and the server could use $i3$ only for initial contact and afterward exchange the data directly using shortcuts.

The basic $i3$ system does not provide data encryption, although it could be implemented as an add-on feature. There is no encryption and privacy for control packets. When a public infrastructure is used, $i3$'s extensive infrastructure requirements bring other serious security issues including the possibility of malicious or misbehaving $i3$ nodes that do not forward correctly and a lack of trust of arbitrary $i3$ nodes from end points. Note that Secure-$i3$ 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

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

$i3$ further complicates the situation.

The combination approach of $Hi3$ helps to address some of the separate shortcomings of HIP and $i3$. The advantages of using $i3$ as a control plane for HIP 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, $Hi3$ provides additional protection against DoS attacks. Although some protection against DoS 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 $i3$ is supported by sending update control packets via $i3$ when end-to-end connectivity is lost. $Hi3$ inherits the challenges of the extensive $i3$ infrastructure, including trust, accountability, and cost issues.

4 Latency estimation

In this section, we analyze the latency of an $Hi3$ system, introducing a few key parameters, relations among them, and assumptions on their reasonable values. Although some parameters depend on a lot of factors such as instant of time, type of a request considered, or CPU power of $i3$ nodes, here we focus on the most essential details. It allows analyzing upper bounds for the parameters and defines a base for evaluation of various $Hi3$ properties. In particular, an analysis of the basic latency components for the control plane is presented. Another application of these results can be found in our companion paper [13] that analytically evaluates $Hi3$ scalability.

4.1 Primary parameters of $Hi3$

N

The number of nodes in $i3$. The values can be from several dozens, as in the current implementation on PlanetLab, to several hundred or thousand for a real large-scale overlay network.

τ

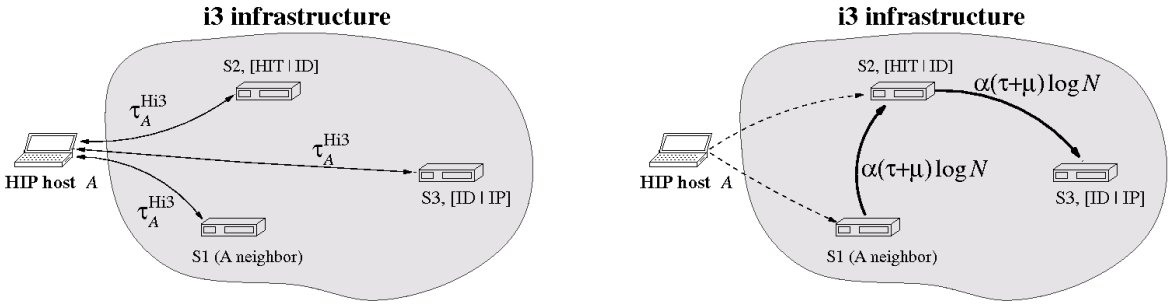
Transmission cost for sending a packet directly between two $i3$ nodes. The conservative estimate for the upper bound is 50–200 ms.

μ

Processing cost for an $i3$ server to serve a packet received before forwarding it. The cost depends on the infrastructure size, $\mu = \mu(N)$, since an $i3$ node maintains $O(\log N)$ state. Typically, the cost does not exceed 0.1–1.0 ms.

μ_{pr}

Processing cost for a host to execute HIP-related cryptography, i.e., cost of either I1/R2 or I2/R2 processing, see Figure 1. Typically, the cost is less than 100–200 ms.



(a) Direct communications with $i3$ nodes

(b) Lookups in $i3$ due to $Hi3$

Figure 6: Transmission costs of a packet in $Hi3$.

τ_A^{Hi3}

Transmission cost for sending packets directly between a HIP host A and an $i3$ node, or the one-way trip time. For wireline $\tau_A^{\text{Hi3}} \sim 10$ ms, for wireless $\tau_A^{\text{Hi3}} \sim 200$ ms.

There are three types of $i3$ nodes that may communicate directly with a HIP host A, see Figure 6(a). The node $S1$ is an arbitrary $i3$ node that A contacts first. The node $S2$ keeps the public trigger [HIT | ID] inserted by A, or a trigger of an A's peer. The node $S3$ stores the A's private trigger.

τ_{AB}

Transmission cost for sending a packet directly between HIP hosts A and B. A reasonable approximation is $\tau_{AB} = \tau_A^{\text{Hi3}} + \tau + \tau_B^{\text{Hi3}}$.

k

The number of packets in a request, including acknowledgment packets. Some packets can flow in parallel, and the latency analysis either considers only the most expensive of them or assumes conservatively that the packets are sent sequentially. For instance, a request for the pure association setup consists of $k = 4$ packets, and the notification packet (path I1.c', Fig. 5(a), p. 10) is not taken into account. For simultaneous host movement, conservative analysis assumes UPDATE2 and UPDATE3 packets are sent one after another.

4.2 Latency parameters

Comparing to HIP, performance overhead in $Hi3$ appears because of involving $i3$ as a third-party. That is, let us define basic latency components caused by this triangle routing.

τ^{Hi3}

Internal latency of a packet inside $i3$. A packet *enters* $i3$, when the first-hop $i3$ node receives the packet, and *exits* $i3$, when the last-hop $i3$ node forwards it to the end-host.

Although this entering/exiting behavior is typical for almost all packets, some of them, like an initial packet of the HIT insertion or an I1 packet of the optimized setup, do not leave

$i3$ running to another host. Such a packet reaches a target $i3$ node and does not go further; the node processes the packet received and then sends another packet backward. Similarly, a packet can be created inside $i3$; for instance, when the packet is an acknowledgment for a trigger inserted successfully or an R1 packet in the optimized setup.

Note also that some packets can never cross $i3$, if direct communication between end-hosts is possible. For instance, R2 packets always flow over the data plane in the optimized setup since initiator's IP address is already known to the responder.

Chord lookups

They are conducted by some packets of a request and can essentially affect the internal latency, especially for large N. A request to the $Hi3$ control plane uses $i3$ lookups in the following cases, see Figure 6(b). (1) An arbitrary $i3$ node ($S1$ in our case) is requested by A to forward a packet to the public trigger location. (2) The node $S2$, which stores the public trigger, looks up the private trigger.

During a lookup the packet visits usually at most $O(\log N)$ nodes [29] as a sequence of $i3$ hops towards the destination. Each hop takes the time $\tau + \mu$, where μ is for processing the packet at a node of the sequence, and then τ is for traveling to the next hop. Although the number of hops is fewer by one than the number of transmissions, the difference is negligible, and we assume a lookup to take time $t = (\tau+\mu)O(\log N)$. In other words, $t \le \alpha(\tau+\mu)\log N$ for some positive constant α and for all large N. Taking $t = \alpha(\tau+\mu)\log N$, we obtain an upper bound for the cost of a lookup, where α controls how many lookups are covered within this bound. Note that experiments in [29] showed α to be equal to $1/2$ on average.

τ^{out}

External latency of a packet, i.e., the time for a packet to be outside $i3$, both before entering and after exiting. If a packet runs directly between end-hosts, then τ^{out} is the only latency of the packet ($\tau^{\text{Hi3}} = 0$).

T^{Hi3} **and** T^{out}

The internal and external latencies of a request. Obviously, T^{Hi3} is a sum of internal latencies and T^{out} is a sum of external latencies for all packets of the request. We shall also use the estimates $T^{\text{Hi3}} = k\tau^{\text{Hi3}}$ and $T^{\text{out}} = k\tau^{\text{out}}$, where τ^{Hi3} and τ^{out} take averages over all k packets of the request.

L^{Hi3}

The total latency of a given request to the control plane.

$$L^{\text{Hi3}} = T^{\text{Hi3}} + T^{\text{out}} = k(\tau^{\text{Hi3}} + \tau^{\text{out}}) \tag{1}$$

where τ^{Hi3} and τ^{out} are averaged over all k packets.

Table 1: Latency estimates of requests to the $Hi3$ control plane. ($C \leftrightarrow S$ communication.)

Request type	k	$T^{\text{Hi3}} = k\tau^{\text{Hi3}}$	$T^{\text{out}} = k\tau^{\text{out}}$
Pure association setup	4	$6\alpha(\tau+\mu)\log N$	$4\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + 4\tau_S^{\text{Hi3}}$
Optimized association setup	4	$2\alpha(\tau+\mu)\log N$	$3\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + \tau_S^{\text{Hi3}} + \tau_{SC}$
Location update, $A \in \{C, S\}$	2	$\tau+\mu$	$2\tau_A^{\text{Hi3}}$
Double-jump	2	$\alpha(\tau+\mu)\log N$	$\tau_C^{\text{Hi3}} + \tau_S^{\text{Hi3}} + \tau_{SC}$
HIT insertion, $A \in \{C, S\}$	4	$2\alpha(\tau+\mu)\log N$	$4\tau_A^{\text{Hi3}}$
HIT refreshment/removal, $A \in \{C, S\}$	4	$2(\tau+\mu)$	$4\tau_A^{\text{Hi3}}$

4.3 Latency of available requests

The latency estimates are summarized in Table 1. Note that we consider the conservative case when $i3$ does not use caching of nodes. The optimistic case is discussed later in Section 4.4.

4.3.1 Pure association setup

There are $k = 4$ packets, namely I1, R1, I2, and R2, each crossing $i3$. HIP-related cryptographic processing with the cost μ_{pr} is needed at both end-hosts: R1/I2 (initiator) and I2/R2 (responder). Therefore, $T^{\text{out}} = 4\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + 4\tau_S^{\text{Hi3}}$. Two first packets (I1 and R1) use two $i3$ lookup each. For two last packets (I2 and R2) one lookup is sufficient. In total, there are six lookups and $T^{\text{Hi3}} = 6\alpha(\tau+\mu)\log N$.

4.3.2 Optimized association setup

The same packets are used as in the pure association setup, but the I1/R1 processing is delegated to the $i3$ node where $[\text{HIT}_S \mid \text{IP}_S]$ is stored. One lookup for the I1 packet is used to find this node. Both I1 and R1 packets are outside $i3$ for time τ_C^{Hi3} each. The I2 packet leaves $i3$ after one lookup to route between the nodes with a public and a private trigger, the external time is $\tau_C^{\text{Hi3}} + \tau_S^{\text{Hi3}}$. The R2 packet does not use $i3$ and travels directly from S to C in time τ_{SC}. The cost of cryptographic processing is the same as for the pure association setup. In total, there are two lookups, $T^{\text{Hi3}} = 2\alpha(\tau+\mu)\log N$, and $T^{\text{out}} = 3\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + \tau_S^{\text{Hi3}} + \tau_{SC}$.

4.3.3 Location update (IP update)

Assuming a host A, either C or S, keeps HIT in $i3$, an address change $\text{IP}_A \rightarrow \text{IP}'_A$ requires A to update its private trigger $[\text{ID}_A \mid \text{IP}_A] \rightarrow [\text{ID}_A \mid \text{IP}'_A]$. Two packets are involved ($k = 2$), an update request by A and the response. The request packet travels directly from A to the $i3$ node storing the private trigger and the node replies backs. The external latency is $T^{\text{out}} = 2\tau_A^{\text{Hi3}}$. The internal latency $T^{\text{Hi3}} = \tau + \mu$ estimates the cost of $i3$ update and forward operations.

4.3.4 Simultaneous host movement

When C has discovered the double-jump by a timeout it sends the first packet of the HIP update exchange to $i3$. As in the location update, $k = 2$. We assume that C remembers the IP address of an $i3$ node storing HIT_S. The first packet crosses $i3$ with one lookup and with external latency latency $\tau_C^{\text{Hi3}} + \tau_S^{\text{Hi3}}$. Then, S sends the response packet that travels directly to C in time τ_{SC}.

4.3.5 HIT insertion

Host A sends a packet to any $i3$ node (cost τ_A^{Hi3}); the packet is routed to the target $i3$ node to store the public trigger, and in the worst case the Chord lookup is used (cost $\alpha(\tau+\mu)\log N$); then the acknowledgment runs backward (cost τ_A^{Hi3}). Insertion of the private trigger is similar. Considering sequential order of the insertion for public and private triggers, such a request takes $T^{\text{Hi3}} = 2\alpha(\tau+\mu)\log N$ and $T^{\text{out}} = 4\tau_A^{\text{Hi3}}$.

4.3.6 HIT refreshment and HIT removal

As both requests have the same packet flow, we assume that there is no difference in latency. Having recent IP addresses of both $i3$ nodes, host A sends directly a packet to each and then receives an acknowledgment. Internal latency of such a packet together with its acknowledgment is estimated as $\tau + \mu$, accounting thereby the consumption of both transmission and processing resources of $i3$. The external latency is $2\tau_A^{\text{Hi3}}$. Considering the sequential scenario of sending the packets, the total latency is doubled.

Note that when HIT refreshment becomes HIT re-insertion, the latency is the same as in HIT insertion.

4.4 Discussion

The analysis shows that the most expensive latency components are due to 1) Chord lookups and 2) triangle routing via $i3$ nodes. Nevertheless, this overhead is not very high.

Even if a request involves Chord lookups, the performance is not affected significantly. First, such a request is rare. Association setup and HIT insertion appear at most on per connection basis. The double-jump can happen only when both end-hosts are mobile and the probability that they move simultaneously is usually low.

Second, any $i3$ node may cache IP addresses of another $i3$ node that happens to be a target of a Chord lookup. The next lookup with this target will take one hop instead of $O(\log N)$. More precise, $i3$ node first tries to use the cache and then, if there is no appropriate address in the cache or the address is not valid, the node calls the Chord lookup. Thus, the internal latency is reduced to $2(\tau+\mu)$ since only two $i3$ nodes are involved.

The following conditions make the $i3$ cache usage more efficient: 1) end-hosts are active and communications are repeatable (a lot of requests and many connections between the same end-points); 2) $i3$ is stable (node joining and leaving make the cache inconsistent). The former is typical for many applications, while the latter depends on the overlay network.

The external latency can be reduced by appropriate trigger allocation. First, end-hosts should first contacts $i3$ nodes that are close to them. Second, since an end-host constructs

its private trigger, the trigger can be stored at a nearby $i3$ server; hence, the overhead due to communication with the end-host via this node is low. Third, some mechanisms, like prefixes to HITs, are going to appear in future implementations of $i3$ to control allocation of public triggers.

5 Experimental evaluation

In this section, we present results comparing the performance of $Hi3$, $i3$, direct HIP, and direct plain IP. It is obvious that a direct connection between C and S has better throughput and latency than solutions based on $i3$ or $Hi3$. Our goal is a high-level estimation of the overhead of additional functionality offered by $Hi3$. We consider a scenario where a mobile host C (e.g. laptop) as an initiator contacts a stationary server S.

5.1 The $Hi3$ implementations

We used the Linux OpenHIP implementation [9] to perform the experiments. The $Hi3$ prototype currently supports only the basic HIP exchange over $i3$. The implementation, in addition to HIP and $i3$, is less than 500 lines of code. In the implementation, HIP control packets including the IP header are tunneled through $i3$ servers running on PlanetLab. Our $Hi3$ implementation was included into the public OpenHIP release (available from http://www.openhip.org).

We have also implemented and tested $Hi3$ for HIP on Linux (HIPL) implementation (available from http://hipl.hiit.fi). It has following differences from the OpenHIP implementation. First of all, HIPL always answers using the same method as it has received the I1 (i.e., if it receives an I1 packet through $i3$ it will answer through $i3$ otherwise it will answer directly). Secondly, HIPL inserts into $i3$ four pairs of triggers, as HIPL has two RSA HITs and two DSA HITs. Every pair of DSA or RSA HITs consists of one public and one anonymous HIT. The public HIT should be available to everyone, while anonymous is available for friendly hosts only and is being changed from time to time. Thirdly, the I1 and R1 packets do not contain any IP addresses. We do not want to reveal IP addresses too early to avoid direct DoS attacks. Only after the sender solves the puzzle in the R1 packet (solution goes inside the I2 packet together with IP addresses of the sender) we send IP addresses in the R2 packet. The field for IP addresses is called LOCATOR and defined in the HIP specification. Generally, HIP uses this field only if the number of IP addresses is greater than one. In the $Hi3$ implementation for HIPL we always send this field even if there is only one IP address, as there is no standard IP header in $Hi3$ messages.

5.2 Data throughput and latency in LAN

We first consider a case where both a client and a server reside in a local network. Both hosts register their HITs on $i3$ 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 $i3$ and $Hi3$. We measured TCP connections setup, TCP throughput and round-trip time. The results given in Table 2 are averaged over a small number of repetitions, as the variance of measurements was negligible.

Table 2: Performance results of IP, $i3$, and $Hi3$ for throughput and latency.

Solution	TCP setup latency, ms	TCP throughput, Mbyte/s	RTT, ms
IP	0.4	10	0.2
$i3$	620	0.08	280
$Hi3$	900	5	1

The results above are heavily influenced by lack of $i3$ shortcuts, and by the fact that all $i3$ 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 $i3$, all packets flow through $i3$ servers with additional time required for packet routing inside $i3$ infrastructure. In $Hi3$, the HIP base exchange packets flow through the $i3$ infrastructure, while the TCP three-way handshake occurs directly between the sender and the receiver with additional overhead of IPsec encryption.

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 $i3$, all data flow through proxies and the infrastructure that results in a high overhead. In $Hi3$, 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 $i3$, the RTT is taken after the client and the server have cached the location of the triggers. For $Hi3$, 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 $Hi3$ remain at a reasonable level with plain IP.

5.3 Data throughput and latency in WAN

After evaluating the case where the client and the server were placed in the same LAN, we measured $Hi3$ performance when the client and the server are located far from each other. In particular, the client was located in Helsinki, Finland and the server in Berkeley, USA. We measured the time to establish a HIP security association between the client and the server, throughput of data transmission, and RTT. The one-way latency between the client and the server was approximately 100 ms.

Figure 7 shows the duration of establishing a HIP security association through $Hi3$ and directly over IP. The time was measured with `ping` and includes one RTT in addition to the delay of HIP base exchange. For plain IP, the operation took on the average 900 ms. In this scenario, the IP address of the server is known to the client immediately and the client could

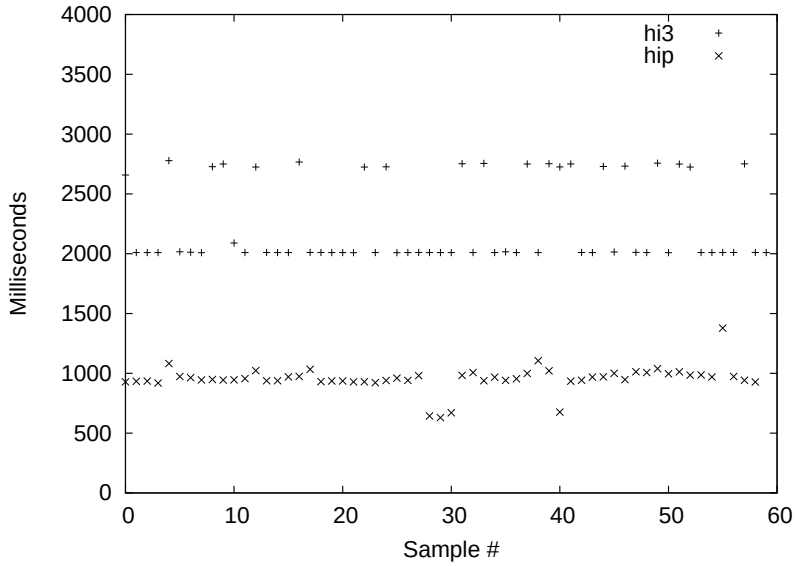


Figure 7: Duration of the association establishment.

start the flooding attack. We also measured the operation over $i3$ where IP addresses of communicating parties remain hidden until both are authenticated. When the base exchange is run over $i3$ it takes about two seconds with several samples of 2.7 seconds. We conjecture that the spikes might appear from the task scheduling delays on busy PlanetLab servers.

Throughput of a TCP data transmission is shown in Figure 8. In this experiment, we compared throughput of unprotected connection over IP, throughput over direct HIP, over $Hi3$, and over $i3$. The throughput of direct IP, HIP and $Hi3$ is on the level of 2.5 Mbps. This fact suggests that at this transmission speed, the overhead of HIP encryption is insignificant. Furthermore, it illustrates the efficiency of data transmission in $Hi3$. In contrary, throughput of $i3$ is twice less, about 1.3 Mbps. It is because the data packets have to be sent through $i3$ servers located on PlanetLab that slows down the transmission.

The RTT measured with ping between the client and the server is shown in Figure 9. For plain IP, HIP and $Hi3$, the RTT is the same (210 ms) with little variation. For $i3$, the RTT is higher (380 ms) and more variable as the ping packets go through $i3$ servers.

The presented results cannot be used as a definite bound of $Hi3$ and $i3$ performance, since $i3$ servers in the experiments were located on public PlanetLab servers. Some servers might be under high CPU or traffic load that affected the throughput and RTT results. However, the measurements do suggest the general trend that the data plane of $Hi3$, based on direct connectivity between hosts, is more efficient than tunneling all data through infrastructure as in $i3$.

6 Conclusions

In this paper, we addressed in an integrated manner the problems of mobility, multi-homing, and IP-layer security, including protection against distributed denial-of-service attacks. Re-

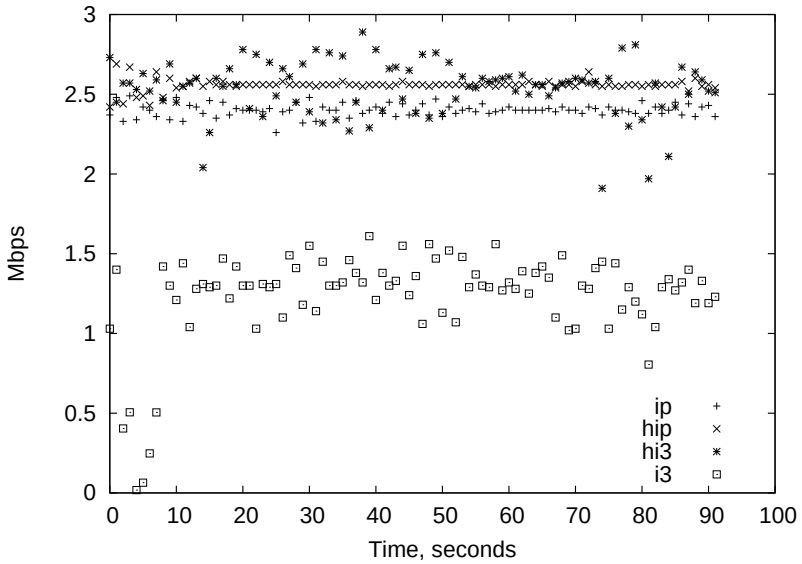


Figure 8: Data throughput.

lying on the Host Identity Protocol (HIP), 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 ($i3$) server node provide functionally the same service, we integrated $i3$ 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). 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 HIP and $i3$. Our qualitative analysis shows that the system is more robust and secure than either of the base systems. The analysis shows that the signaling latency in the system is dominated by inter-node transmission times within the infrastructure. Our performance measurements show that in terms of throughput and data path latency, $Hi3$ improves over $i3$.

Acknowledgments

The authors want to thank Jari Arkko, Börje Ohlman, Jukka Ylitalo, 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 for useful comments that helped to improve the paper.

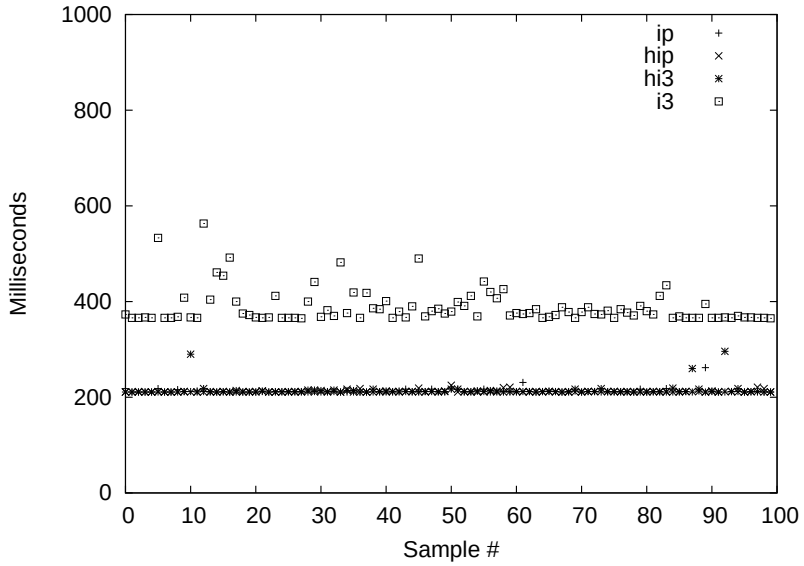


Figure 9: Data plane RTT.

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

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

[5] 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.

[6] 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.

[7] P. Francis. IPNL: A NAT-extended Internet architecture. In *Proc. of ACM SIGCOMM'01*, San Diego, CA, USA, Aug. 2001.

[8] A. Gurtov, D. Korzun, and P. Nikander. Hi3: An efficient and secure networking architecture for mobile hosts. Technical Report TR-2005-2, HIIT, June 2005.

[9] 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.

[10] 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.

[11] 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.

[12] M. Komu, S. Tarkoma, J. Kangasharju, and A. Gurtov. Applying a Cryptographic Namespace to Applications. In *Proc. of the first ACM workshop on Dynamic Interconnection of Networks (DIN 2005)*, Cologne, Germany, Sept. 2005. ACM Press.

[13] D. Korzun and A. Gurtov. On scalability properties of the Hi3 control plane. *Computer Communications*, 29(17):3591–3601, Nov. 2006.

[14] J. Laganier and L. Eggert. Host identity protocol (HIP) rendezvous extension: draft-ietf-hip-rvs-05, June 2006. Work in progress. Expires in December, 2006.

[15] J. Laganier, T. Koponen, and L. Eggert. Host Identity Protocol (HIP) registration extension: draft-koponen-hip-registration-02, June 2006. Work in progress.

[16] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica. On securing forwarding infrastructures: Protecting the data plane from an untrusted control plane, 2005. Unpublished manuscript.

[17] J. Li, J. Stribling, T. M. Gil, R. Morris, and M. F. Kaashoek. Comparing the performance of distributed hash tables under churn. In *Proc. of the 3rd International Workshop on Peer-to-peer systems*, pages 87–99, 2004.

[18] D. Liben-Nowell, H. Balakrishnan, and D. Karger. Observations on the Dynamic Evolution of Peer-to-Peer Networks. In *1st Workshop on P2P Systems and Technologies*, Cambridge, MA, March 2002.

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

[20] R. Moskowitz, P. Nikander, P. Jokela, and T. R. Henderson. Host Identity Protocol: draft-ietf-hip-base-10, Oct. 2007. Work in progress. Expires in May, 2008.

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

[22] P. Nikander, J. Arkko, and T. Henderson. End-host mobility and multi-homing with host identity protocol: draft-ietf-hip-mm-05, Mar. 2007. Work in progress, Expires September, 2007.

[23] P. Nikander, J. Arkko, and B. Ohlman. Host identity indirection infrastructure (hi3). In *The Second Swedish National Computer Networking Workshop*, November 2004.

[24] P. Nikander, H. Tschofenig, X. Fu, T. Henderson, and J. Laganier. Preferred alternatives for tunnelling HIP (PATH): draft-nikander-hip-path-01.txt, Mar. 2006. Work in progress.

[25] 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.

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

[27] 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.

[28] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proc. of ACM SIGCOMM'02*, pages 73–88, Pittsburgh, PA, USA, Aug. 2002.

[29] 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.

[30] H. Tschofenig and M. Shanmugam. Traversing HIP-aware NATs and Firewalls: Problem Statement and Requirements: draft-tschofenig-hiprg-hip-natfw-traversal-04, Mar. 2006.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.