On Scalability Properties of the Hi3 Control Plane

Dmitry Korzun[*]
Helsinki Institute for Information Technology (HIIT)
P.O.Box 9800, FIN-02015 HUT, Finland
`dkorzun@hiit.fi`

Andrei Gurtov[†]
University of California, Berkeley
School of Information Management and Systems (SIMS)
102 South Hall, Berkeley, CA 94720-4600
`gurtov@cs.berkeley.edu`

Abstract

The Host Identity Indirection Infrastructure ($Hi3$) is a general-purpose networking architecture, derived from the Internet Indirection Infrastructure ($i3$) and the Host Identity Protocol (HIP). $Hi3$ combines efficient and secure end-to-end data plane transmission of HIP with robustness and resilience of $i3$. The architecture is well-suited for mobile hosts given the support for simultaneous host mobility, rendezvous and multi-homing. Although an $Hi3$ prototype is implemented and tested on PlanetLab, scalability properties of $Hi3$ for a large number of hosts are unknown.

In this paper, we propose a simple model for bounds of size and latency of the $Hi3$ control plane for a large number of clients and in the presence of DoS attacks. The model can be used for a first approximation study of a large-scale Internet control plane before its deployment. We apply the model to quantify the performance of the $Hi3$ control plane. Our results show that the $Hi3$ control plane can support a large number of mobile hosts with acceptable latency.

1 Introduction

The original Internet Protocol stack was designed without explicit consideration for address agility or IP-layer security. Recently, much effort has been applied to develop an architecturally sound solution to address these shortcomings (refer, e.g., to [2, 3, 5, 4, 9]).

The Host Identity Indirection Infrastructure ($Hi3$) [15, 7] is a novel general-purpose networking architecture, derived from the Internet Indirection Infrastructure ($i3$) [17] and the Host Identity Protocol (HIP) [12, 8, 10, 16]. $Hi3$ benefits from HIP by efficient end-to-end

[*]The author is a visiting research scientist at HIIT. Permanent position is at Department of Computer Science, Petrozavodsk State University, Lenin St., 33, Petrozavodsk, Republic of Karelia, 185910, Russia. Email: `dkorzun@cs.karelia.ru`

[†]The author is a visiting research scientist at SIMS. Permanent position is at Helsinki Institute for Information Technology (HIIT), P.O.Box 9800, FIN-02015 HUT, Finland. Email: `gurtov@hiit.fi`

data transfer over IPsec, IPv4/v6 interoperability, basic mobility, multi-homing, Denial-of-Service (DoS) protection, as well as IETF standardization and strong support of the industry. Integration with $i3$ allows for simultaneous mobility of end points, additional DoS protection, and the initial rendezvous service for $Hi3$ hosts. $Hi3$ can be gradually deployed, has been included into public HIPL and OpenHIP implementations, and tested on PlanetLab [6].

$Hi3$ consists of separate control and data planes. The control plane running over $i3$ is used only for HIP messages; the user data flows via the end-to-end data plane over IPsec. Main performance issues for a control plane infrastructure are scalability, DoS resilience, and robustness to node failures. Despite initial promising measurement results, the $Hi3$ performance at larger scales remains unknown. In this paper, we develop an analytical model of the first two issues and apply it to quantify the $Hi3$ performance. The third property, robustness, is inherited by $Hi3$ from Chord and has been analyzed elsewhere [18].

Our model can be used as a first approximation of a large-scale Internet control plane before its deployment. In practical terms, the model provides us with a conservative estimate of the $Hi3$ capacity. Despite conservative estimating, our results show that the $Hi3$ control plane can support Internet applications for a large number of mobile hosts.

Our analysis shows that the $Hi3$ control plane scales well; a few hundred infrastructure servers are sufficient to support millions of clients. The internal latency of the infrastructure is satisfactory and does not exceed a few hundred milliseconds in most cases. Additionally, we describe $Hi3$ behavior under a load of Distributed DoS (DDoS) attacks generated by zombies, a set of compromised and exploited PCs.

The rest of the paper is organized as follows. In Section 2, background material on HIP and $i3$ is provided. In Section 3, we describe the $Hi3$ control plane, introduce our modeling approach and derive cost estimates of basic requests to the control plane. Workload scenarios are given in Section 4. The scenarios illustrate the use of $Hi3$ capacity by mobile, stationary and zombie hosts. In Section 5, scalability properties of $Hi3$ are analyzed. Based on the workload scenarios, we derive trends for the size and latency of the control plane. Section 6 concludes the paper.

2 Background

We begin by giving the necessary background on $i3$, HIP, and the $Hi3$ architecture.

2.1 Host Identity Protocol (HIP)

In HIP [13, 12], IP addresses are used to locate and route the 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 a public key that corresponds to a private key, possessed by the host. Each host generates one or more public/private key pairs to provide identities for itself. 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 HITs are constructed entirely cryptographically.

The actual HIP protocol [13] 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 to their host identifiers. Additionally, the exchange creates a pair of Encapsulated Security Payload (ESP) security associations (SAs), one in each direction.

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 [17]. 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 A may insert a trigger $[\mathrm{ID} \mid \mathrm{IP}_A]$ in the *i*3 infrastructure to receive all packets that have the destination identifier ID.

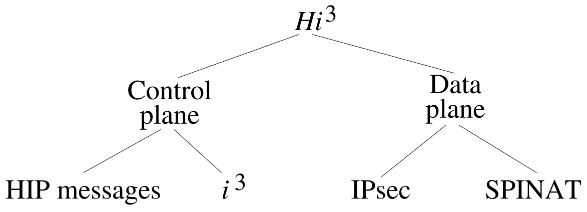
*i*3 provides natural support for mobility. When a host changes its address, the host needs only to update its trigger. When the host changes its address from IP_1 to IP_2, it updates its trigger from $[\mathrm{ID} \mid \mathrm{IP}_1]$ to $[\mathrm{ID} \mid \mathrm{IP}_2]$. As a result, all packets with the identifier ID are correctly forwarded to the new address.

The next step of *i*3 evolution is the Secure-*i*3 proposal [1]. Its main goal is to provide enhancements against DoS attacks and misuse of the infrastructure. The basic idea is hiding the IP addresses of the end-hosts from other users of the network. In Secure-*i*3, there are two types of triggers, public $[\mathrm{ID} \mid \mathrm{ID}']$ and private $[\mathrm{ID}' \mid \mathrm{IP}]$. Public triggers are used to announce the existence of a service with identifier ID. Private triggers are used for the actual communication between sender and receiver(s).

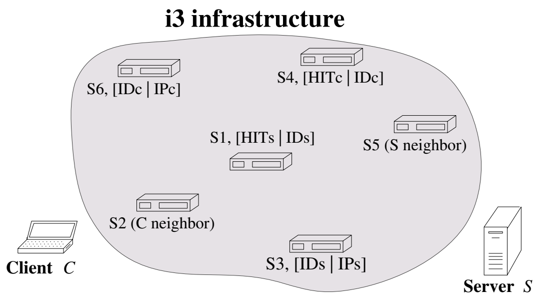
2.3 Host Identity Indirection Infrastructure (*Hi*3)

The *Hi*3 sketch [15] 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 end-to-end traffic (the *Hi*3 data plane) while using an indirection infrastructure to route the HIP control packets (the *Hi*3 control plane). The concept of this separation is shown in Figure 1(a). *i*3 has an important capability to map HITs from a flat namespace to IP addresses using the underlying Chord DHT, a key property missing from a stand-alone rendezvous server.

The control plane is used to relay HIP messages during the association establishment and when the direct end-to-end connectivity is lost, e.g, after a simultaneous movement of both hosts. The main benefit of using the control plane during association establishment is protection against DoS attacks. This way, the IP addresses of communicating hosts are not revealed until mutual authentication is completed. The data plane provides further protection



(a) Separating data and control in $Hi3$



(b) Schematic distribution of HIT-based triggers in $i3$ for $C \leftrightarrow S$ communication

Figure 1: The $Hi3$ architecture.

against DoS if IP addresses become revealed to a group of DDoS attackers after one of them establishes a HIP connection over $Hi3$.

For each 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 acts as a public $i3$ identifier of the host A and ID_A is a private $i3$ identifier constructed by $i3$. Let C and S be two communicating hosts. Figure 1(b) shows how the corresponding HIT-based public and private triggers are distributed in $i3$ for $C \leftrightarrow S$ communication.

In the data plane, end-to-end traffic is encrypted with ESP, but other encapsulation methods are possible too [7]. The IPsec-aware NAT (SPINAT) provides privacy and protection against DoS of the data plane [19] by dynamic mapping between the actual IP addresses and the NAT addresses using the IPsec Secure Parameter Index (SPI). For the rest of this paper, we concentrate on the $Hi3$ control plane only.

3 Control plane

In this section, we describe available request types to the $Hi3$ control plane and estimate their computational cost and latency.

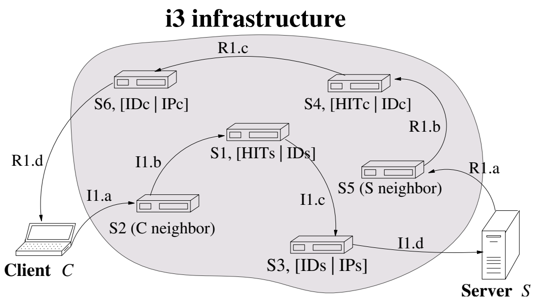
3.1 Request types

Consider two hosts that use $Hi3$ for communication. Let one of them be a HIP initiator C (e.g., a mobile client) and the other one be a HIP responder S (e.g., a stationary Internet server). Figure 2 shows main scenarios when $C \leftrightarrow S$ communication involves the $Hi3$ control plane. For details refer to a complete $Hi3$ description [7].

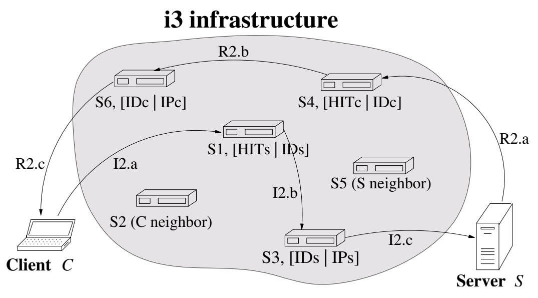
3.1.1 Association setup

The client C starts an $Hi3$ *association setup* with the server S for initiating the communication. This requires $i3$ to contain a server's HIT as a valid public trigger $[\mathrm{HIT}_S \mid \mathrm{ID}_S]$ (at node $S1$ in Figures 1(b) and 2(a)) and a valid private trigger $[\mathrm{ID}_S \mid \mathrm{IP}_S]$ (at node $S3$).

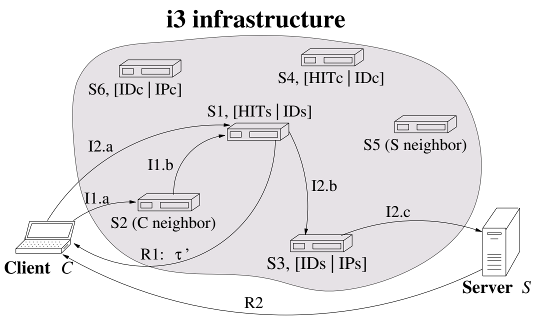
The association setup is implemented via the HIP base exchange [13]. The client C uses an IP address of either the $i3$ node that keeps the S's public trigger (node $S1$), or a random $i3$



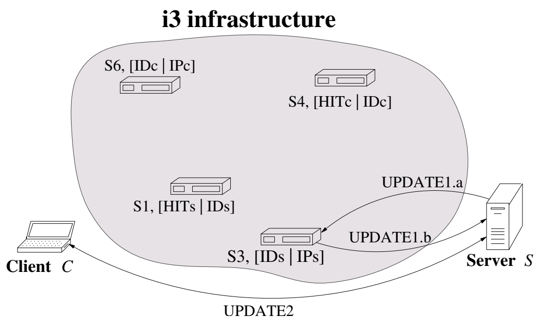
(a) Pure association setup: I1 and R1 packets



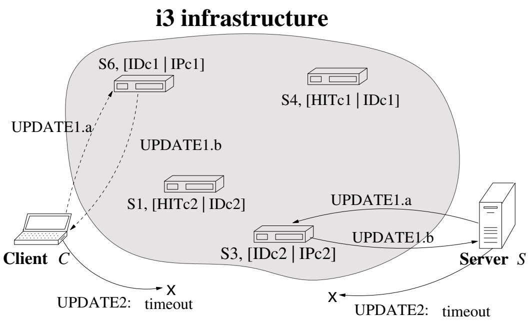
(b) Pure association setup: I2 and R2 packets



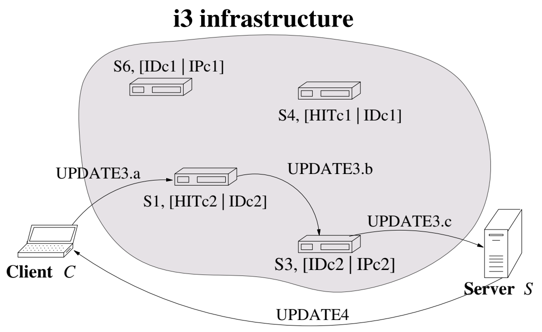
(c) Optimized association setup



(d) Location update



(e) Failure of the location update



(f) Double-jump resolving

Figure 2: Request types in $Hi3$.

node ($S2$). The latter case is typical, and the nearest $i3$ server would be typically contacted.

There are two variants of the association setup in $Hi3$, pure and optimized. In pure setup, the client C inserts, perhaps temporarily, its HIT into $i3$, the public trigger [HIT_C | ID_C] (at node $S4$) and the private trigger [ID_C | IP_C] (at node $S6$). These triggers are needed for replying to an I1 packet with an R1 packet during the HIP base exchange. Figures 2(a) and 2(b) show the packet flow.

In optimized setup the insertion of a client's HIT into $i3$ is not necessary; replying to I1 with R1 is delegated to $i3$ and the R2 reply packet is sent directly to the client. The packet flow is presented in Figure 2(c).

3.1.2 Mobility update

Let us assume that 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 by the server S, then S updates its private trigger in $i3$ (Figure 2(d)). The location update also causes the *HIP update exchange* [14] between C and S running over the data plane. One update to $i3$ is sufficient independently of the number of hosts communicating with S. For C having a permanent trigger pair in $i3$ is optional. Thus, if C changes its location, the HIP update exchange runs directly between hosts (only UPDATE1 messages use $i3$ as shown in Figure 2(d)).

It may happen that both hosts change their locations at the same time, an event known as the *double-jump problem*. We assume that simultaneous mobility of C and S is rare compared to the usual location update. This scenario proceeds as follows. The hosts update their triggers in $i3$ (for C it is optional) as shown in Figure 2(e). In parallel, the hosts start a HIP location update on the $Hi3$ data plane. The exchange fails since each host uses the out-of-date IP address to contact the peer. This failure is discovered by a timeout.

At this point the hosts need to use the control plane to recover from the double-jump (Figure 2(f)). The double jump can be discovered by both hosts, but the client C is responsible for starting the recovery. C sends the first packet of the HIP update exchange (addressed to HIT_S) to S via $i3$. After receiving this packet, S continues the update talking directly to C.

3.1.3 Auxiliary requests

For $Hi3$ association setup and mobility update, the $Hi3$ control plane has to support three auxiliary requests, namely *HIT insertion*, *HIT refreshment*, and *HIT removal*.

Let HIT_A be a HIT of a host A. HIT_A is inserted as a trigger to $i3$ when A is an initiator of a pure association setup. If A is a server, HIT_A needs to stay in $i3$ permanently. In the current $i3$ implementation[1] a trigger needs refreshing every 30 seconds. If the $i3$ node crashes, the trigger is lost until the host refreshes the trigger (re-insertion). Therefore, HIT removal can be done by sending a message to $i3$ or automatically after the trigger expires.

3.2 Transmission and processing costs

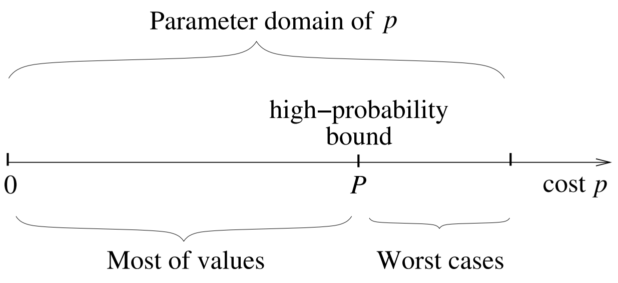
In this section, we make a preliminary analysis of transmission and processing costs in $Hi3$ according to the ideas given in previous work [7].

3.2.1 Upper bounds for costs

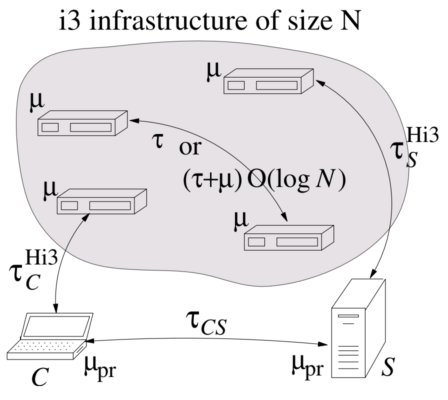
Let p be a cost parameter, for instance the round-trip time, latency, or the number of hops. The classical worst-case analysis aims in finding an upper bound $\overline{P}$ such that $p \leq \overline{P}$ for all possible p. A more robust metric is a high-probability bound P such that $p \leq P$ with high

[1]Available at `http://i3.cs.berkeley.edu`

probability (w.h.p.)[2]. Figure 3(a) shows the parameter domain division based on this bound. Most p values concentrate w.h.p. in the interval $[0, P]$; other values appear rarely and form the worst-case domain of the cost.



(a) High-probability bound



(b) Costs

Figure 3: Assumptions for the analysis.

Given its clear limitations, this approach stills allows to discover some basic trends for conservative cases. It is often applied for cost analysis of concrete DHT systems [18, 11]. The accuracy of trends can be improved by using typical values such as the average or median for some parameters instead of the high-probability bound.

Finding a theoretically approved high-probability bound is quite challenging and meets a lot of difficulties. Formally, one needs to define exactly what a high probability is and give assumptions on the probability distribution. Alternatively, one can apply statistical methods and calculate, based on measurements, a confidence interval for the parameter, e.g., that the bound estimated is valid for 95%. In many cases, however, various practical reasons are enough to provide a reasonable empirical value for the bound. We shall follow this simpler but more practical approach. It is useful as a first approximation of the parameters.

3.2.2 Costs inside and outside of *i*3

Let N be the size of *i*3 (the number of nodes). A request to the *Hi*3 control plane affects two network parts, inside and outside of *i*3 (see Figure 3(b)).

The outside part is characterized by the one-way trip time τ_A^{Hi3} from a HIP host A to an *i*3 node (or backwards). The parameter τ_A^{Hi3} depends on the host's location, but not on the *i*3 size N. Some requests to the control plane can continue as direct host-to-host communication, e.g., in the optimized association setup an R2 packet is sent by S to C directly (see Figure 2(c)). The cost of direct IP-based transmission between C and S is estimated as the one-way trip time $\tau_{CS} = \tau_{SC}$.

Communication inside *i*3, i.e. between *i*3 nodes, can be divided into two types. (1) If two nodes use direct IP-based communication, then the one-way trip time is τ. (2) If a node

[2]Throughout this paper we use the term w.h.p. to mean with probability at least $1 - c/N$ for some constant $0 < c \ll N$, where N is the size of network.

makes a lookup to route a packet to the target node, then the packet visits w.h.p. $O(\log N)$ nodes [18] as a sequence of $i3$ hops towards the destination. Each hop takes time $\tau + \mu$, where μ is the forwarding cost of a packet at an $i3$ node: matching against the $i3$ identifier in the forwarding table, updating the packet header and internal state (if needed), and forwarding to the next hop. Therefore, a lookup takes time $t = (\tau + \mu)O(\log N)$ or, in other words, time $t \leq \alpha(\tau + \mu) \log N$ for some positive constant α and for all large N. Considering the w.h.p. case, we assume that $t = \alpha(\tau + \mu) \log N$. The reasonable high-probability bound is $\mu \sim 0.5 \dots 1.0$ ms

By experimental results of Stoica *et al.* [18] the average lookup time can be estimated as $\frac{\tau+\mu}{2} \log N$. Taking $\alpha > \frac{1}{2}$ makes the bound more conservative, and $\alpha = 1$ is a high-probability bound.

$Hi3$ inherits cryptographic operations from HIP that load end-hosts and/or $i3$ nodes. The operations include the following.

(1) Solving a cryptographic puzzle in the HIP base exchange when C receives an R1 packet and generates an I2 packet with a correct puzzle solution.

(2) Checking the puzzle solution and authenticating the client. This is a consequence of the previous step.

A HIP responder (server S) can delegate a part of these operations to $i3$. In the optimized association setup, the $i3$ node $S3$ having the public trigger of the server S (Figure 1(b)) caches precomputed R1 packets of S. It is able to send R1 packets to initiators in response to I1 packets. Let μ_{pr} be the duration of this operation (Figure 3(b)). A high-probability bound is $\mu_{\text{pr}} \sim 100 \dots 200$ ms.

All other operations are less time consuming. Delegation of responding with an R1 to $i3$ is comparable with a cost μ of forwarding a packet. According to [1], the cost of a trigger insertion in $i3$ (even with checking trigger constraints) is $20 \dots 40$ μs and can also be bounded w.h.p. by μ.

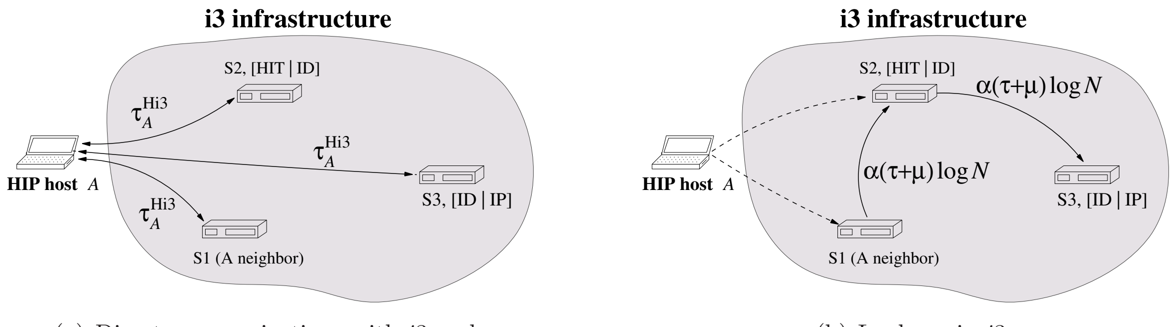
3.2.3 Cost of a request

Requests to the control plane can be classified in two groups: $O(1)$- and $O(\log N)$-*requests* depending on whether Chord routing is required.

$O(1)$-requests The requests involve a constant-bounded number of $i3$ nodes regardless of the $i3$ size N. The requests for HITs refreshment, update, and removal can be implemented with direct IP-based communication to the corresponding $i3$ node. However, when a HIT refreshment becomes the HIT re-insertion or when A loses direct connectivity with its peer, $i3$ lookups are needed and the request becomes the $O(\log N)$-type.

$O(1)$-requests load two $i3$ nodes having a public and a private trigger. Exactly one node is needed for an IP address update in $i3$ and in the rare case when the public trigger and the private one are stored at the same node.

$O(\log N)$-requests $O(\log N)$-requests use $i3$ lookups and, therefore, affect $O(\log N)$ nodes. The following requests belong to this group: pure and optimized association setup, recovering



(a) Direct communications with $i3$ nodes (b) Lookups in $i3$

Figure 4: Transmission costs of a packet in $Hi3$.

the double-jump, and HIT (re-)insertion.

Unlike the $O(1)$-requests, the $O(\log N)$-ones are "deep", involving many nodes for a large N. However, $\frac{\log N}{N} \to 0$ when $N \to \infty$ and the relative cost is small in a large-size infrastructure.

3.3 Latency estimates

Each request consists of one or more packets (e.g., the association setup requires four HIP base exchange packets I1, R1, I2, and R2). Denote this number k. A packet *enters* $i3$ when it is received by a first-hop $i3$ node and it *leaves* $i3$ when it is forwarded by a last-hop node to the destination end-host. Define τ^{Hi3} and τ^{out} as the time for a packet to be *inside* and *outside* $i3$, respectively. The *request latency* is $L = k(\tau^{\text{Hi3}} + \tau^{\text{out}}) = T^{\text{Hi3}} + T^{\text{out}}$, where τ^{Hi3} and τ^{out} are averaged over all k packets of the request, $T^{\text{Hi3}} = k\tau^{\text{Hi3}}$ and $T^{\text{out}} = k\tau^{\text{out}}$ are *internal* and *external* request latencies.

There are three types of $i3$ nodes that may communicate directly with a HIP host A, see Figure 4(a). The node $S1$ is an arbitrary $i3$ node that A contacts. 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. According to Section 3.2, the one-way trip time to these nodes is τ_A^{Hi3}.

A request to the $Hi3$ control plane uses $i3$ lookups in the following cases (Figure 4(b)). (1) An arbitrary $i3$ node ($S1$ in our case) is requested by A to forward a message to the public trigger location. (2) The node $S2$, which stores the public trigger, looks up the private trigger. The cost of such a lookup is $\alpha(\tau + \mu)\log N$. The latency estimates are summarized in Table 1 and explained below.

Pure association setup There are $k = 4$ packets. Each of them travels to $i3$ nodes and back. HIP cryptographic processing of the cost μ_{pr} is needed in both end-hosts: R1/I2 (initiator) and I2/R2 (responder). Therefore, $T^{\text{out}} = 4\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + 4\tau_S^{\text{Hi3}}$. The two first packets (I1 and R1) use two $i3$ lookups each. For two last packets (I2 and R2) one lookup is sufficient. In total, there are six lookups and $T^{\text{Hi3}} = 6\alpha(\tau + \mu)\log N$.

Table 1: Latency estimates of requests to the *Hi*3 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, $A \in \{C, S\}$	2	$2\alpha(\tau+\mu)\log N$	$2\tau_A^{\text{Hi3}}$
HIT refreshment/removal, $A \in \{C, S\}$	4	$2(\tau+\mu)$	$4\tau_A^{\text{Hi3}}$

Optimized association setup The I1/R1 processing is delegated to the *i*3 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 *i*3 for time τ_C^{Hi3}. The I2 packet leaves *i*3 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 *i*3 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. There are two lookups, $T^{\text{Hi3}} = 2\alpha(\tau+\mu)\log N$ and $T^{\text{out}} = 3\tau_C^{\text{Hi3}} + 2\mu_{\text{pr}} + \tau_S^{\text{Hi3}} + \tau_{SC}$.

Location update (IP update) When a host A, either C or S, changes its location, the private trigger update is performed $[\text{ID}_A \mid \text{IP}_A] \to [\text{ID}_A \mid \text{IP}'_A]$, where IP'_A is a new A's address. Two packets are involved ($k = 2$), an update request by A and the response. The request packet travels directly from A to the *i*3 node storing the private trigger, and the node replies back. The external latency is $T^{\text{out}} = 2\tau_A^{\text{Hi3}}$. The internal latency $\tau + \mu$ includes the update and forward operations.

Double-jump When C has discovered the double-jump by a timeout it sends the first packet of the HIP update exchange to *i*3. As for the location update, $k = 2$. We assume that C remembers the IP address of an *i*3 node storing HIT_S. The first packet crosses *i*3 with one lookup with additional latency outside of *i*3 of $\tau_C^{\text{Hi3}} + \tau_S^{\text{Hi3}}$. Then, S sends the response packet that travels directly to C in time τ_{SC}.

HIT (re-)insertion Host A sends a request packet to an *i*3 node. Two *i*3 lookups are performed: routing to a node to insert the public trigger and routing to a node to insert a private trigger. Hence, $T^{\text{Hi3}} = 2\alpha(\tau+\mu)\log N$ and $T^{\text{out}} = 2\tau_A^{\text{Hi3}}$.

HIT refreshment and removal Two requests are used by a host to refresh or remove the triggers. In the worst case, both requests run sequentially and the latency is doubled. For each request the internal *i*3 latency is $\tau + \mu$ and the external latency is $2\tau_A^{\text{Hi3}}$.

4 Workload model

The preceding section described existing request types to the *Hi*3 control plane. Scalability analysis needs to consider scenarios with many end hosts involved so that *Hi*3 is highly loaded. In this section, we present several such scenarios and derive estimates for the control plane workload.

4.1 General workload pattern

A request of a given type loads several *i*3 nodes; denote this number r. The request is either $O(1)$- or $O(\log N)$-type and $r = c$ or $r = c \log N$, where c is a constant relative to N different for each request type.

Consider a set of H hosts sending requests of a given type to the control plane. Let λ be the corresponding rate (requests per time unit) that is uniform for all hosts. Then, λH is the total number of requests sent by the hosts per time unit.

Let us define the workload W of an *i*3 node as

$$W = \frac{\lambda H r}{N}, \tag{1}$$

Several examples for (1) are presented in Table 2. A parameter $0 \le P_{\mathrm{us}} < 1$ gives the probability that a location update event is a double-jump.

Table 2: Workload estimates for requests to the *Hi*3 control plane. The workload is generated by H hosts at the rate of λ per host.

Request type	**Rate,** λ	**#(*i*3 nodes),** r	**Workload,** W
Pure association setup	λ_{s}	$6\alpha \log N$	$W_{\mathrm{s}} = \dfrac{6\alpha\lambda_{\mathrm{s}} H \log N}{N}$
Optimized association setup	λ_{so}	$2\alpha \log N$	$W_{\mathrm{so}} = \dfrac{2\alpha\lambda_{\mathrm{so}} H \log N}{N}$
Location update	λ_{u}	1	$W_{\mathrm{u}} = \dfrac{\lambda_{\mathrm{u}} H}{N}$
Double-jump	$\lambda_{\mathrm{u}} P_{\mathrm{us}}$	$\alpha \log N$	$W_{\mathrm{us}} = \dfrac{2\alpha\lambda_{\mathrm{u}} P_{\mathrm{us}} H \log N}{N}$
HIT insertion	λ_{i}	$2\alpha \log N$	$W_{\mathrm{i}} = \dfrac{2\alpha\lambda_{\mathrm{i}} H \log N}{N}$
HIT refreshment/removal	λ_{r}	2	$W_{\mathrm{r}} = \dfrac{\lambda_{\mathrm{r}} H}{N}$

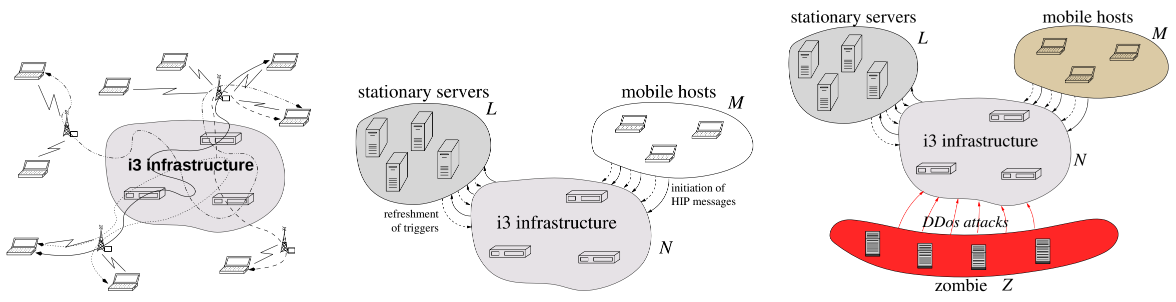
Total workload W_{tot} is defined as the sum of each request type workload. If the hosts use an optimized association setup (W_{so}), location update (W_{u}), and HIT insertion (W_{i}), then $W_{\mathrm{tot}} = W_{\mathrm{so}} + W_{\mathrm{u}} + W_{\mathrm{i}}$. Some important cases are presented in Section 4.2.

A node involved in serving the request either forwards a HIP packet (primary requests) or manages a HIT-based trigger (auxiliary requests). According to the assumptions in Section 3.2, the processing cost is bounded w.h.p. by μ. Hence, r can also be interpreted as the number of μ-long operations per a request.

More detailed analysis could differentiate operations in an $i3$ node using different values for the processing costs. In general $\mu = \mu(N)$ since the local state in a node grows with the size N of a DHT-based infrastructure. Measuring the workload in units of μ is CPU-oriented. It is possible to take into account the link bandwidth; it is a subject of further work.

4.2 Workload scenarios

Let us consider several simple scenarios of $Hi3$ use based on the workload model.



(a) Mobile peers

(b) Mobile hosts and stationary servers

(c) Presence of zombie

Figure 5: Workload scenarios.

4.2.1 Mobile peers

Let M hosts (mobile peers) use $Hi3$ for communication, see Figure 5(a). All hosts generate requests for association setups (pure and optimized) and location updates. This leads to the following estimate of total workload:

$$W_{\mathrm{tot}} = W_{\mathrm{s}} + W_{\mathrm{so}} + W_{\mathrm{u}} + W_{\mathrm{us}} = \alpha \frac{\lambda_{\mathrm{mob}} M}{N} \log N + \frac{\lambda_{\mathrm{u}} M}{N}\,, \tag{2}$$

where $\lambda_{\mathrm{mob}} = 6\lambda_{\mathrm{s}} + 2\lambda_{\mathrm{so}} + \lambda_{\mathrm{u}} P_{\mathrm{us}}$.

For simplicity we do not consider requests of HIT insertion and refreshment. These requests are rare compared to the other requests. Instead, we can use larger values for λ_{mob} and λ_{u} to avoid underestimating the workload.

4.2.2 Mobile hosts and stationary Internet servers

In this scenario, M mobile hosts and L stationary Internet servers (Figure 5(b)) are considered separately. A mobile host communicates with servers and with other mobile hosts as in the previous scenario. The servers do not initiate communication and do not change their location, but perform HIT insertion and refreshment. Together, all $M + L$ hosts generate the following workload:

$$W_{\mathrm{tot}} = \alpha \frac{\lambda_{\mathrm{mob}} M + 2\lambda_{\mathrm{i}} L}{N} \log N + \frac{\lambda_{\mathrm{u}} M + 2\lambda_{\mathrm{r}} L}{N}\,. \tag{3}$$

The Eq. (2) is a particular case of (3) when $L = 0$.

4.2.3 DDoS attacks from zombies

We extend the previous scenario to model DDoS attacks to the control plane. A set of Z zombie hosts generates workload to $Hi3$ in a distributed denial-of-service (DDoS) attack as shown in Figure 5(c). The goal of the zombie set is to overload $i3$.

Obviously, a zombie prefers $O(\log N)$-requests since these requests add more load to $i3$. Thus, a zombie can (i) initiate the association setup with many other hosts via $Hi3$ and/or (ii) insert many HITs to $i3$. We skip the double-jump request as the the same load can be achieved easier by (i) and/or (ii).

An initiation of the association setup can be implemented by a zombie as follows (1) send an I1 packet to $i3$, (2) receive an R1 packet from $i3$, (3) immediately reply with an I2 packet containing a wrong puzzle solution.

Overall, this generates five (for a pure setup) or two (for an optimized setup) $i3$ lookups. Let λ_{zs}, λ_{zso} and λ_{zi} be request rates for a pure association setup, optimized association setup, and HIT insertion by a zombie, respectively. Together all zombies generate the following injurious workload

$$W_{\text{bad}} = W_{\text{zs}} + W_{\text{zso}} + W_{\text{zi}} = \alpha \frac{\lambda_{\text{z}} Z}{N} \log N \,, \tag{4}$$

where $\lambda_{\text{z}} = 5\lambda_{\text{zs}} + 2\lambda_{\text{zso}} + 2\lambda_{\text{zi}}$.

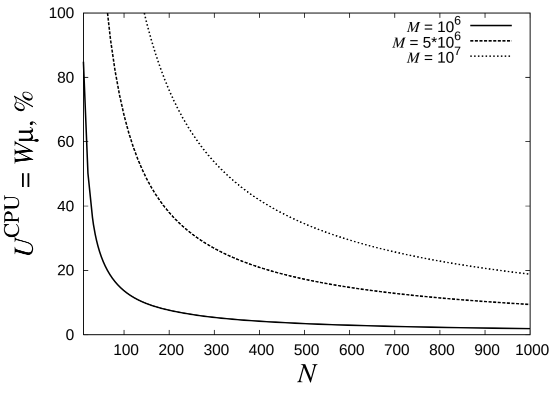
5 Scalability analysis

In this section, we analyze the scalability of the control plane for workload scenarios described in the previous section. The general idea is to estimate the utilization of the control plane depending on the number of $i3$ nodes N. We obtain an estimate of N that provides a certain utilization level for given workload and internal request processing latency.

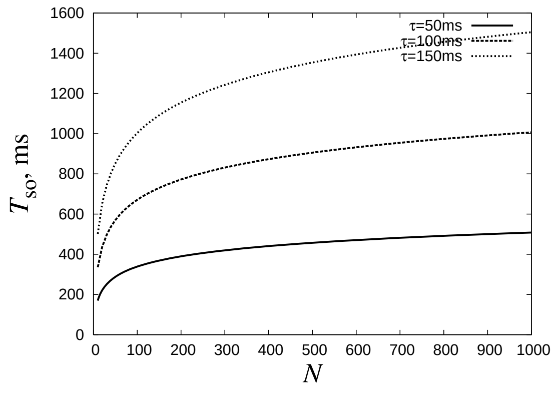
5.1 The utilization/latency trade-off

The utilization of an $i3$ node can be estimated as $U = U^{\text{CPU}} = W\mu > 0$, i.e., what fraction of time a node spends for $Hi3$-related processing. If $U > 1$ then the infrastructure is overloaded by requests. Similarly, the utilization of $i3$ bandwidth is $U' = U^{\text{COM}} = W\tau > 0$, where τ is the one-trip time between two arbitrary $i3$ nodes (see Section 3.2). Each node involved in a request sends a packet after processing for μ milliseconds. The overload happens when $U' > B$ for a some throughput threshold B.

Intuitively, the greater N is, the less the utilization is. This fact is supported by the workload model, see Eq. (1). On the other hand, increasing N increases the internal latency T^{Hi3} as shown in Table 1. The trade-off between the utilization and latency of the infrastructure is an interesting fact that we consider next. We assume $\mu = 1$ ms and $\alpha = 1/2$ based on the measurements in [6, 18].



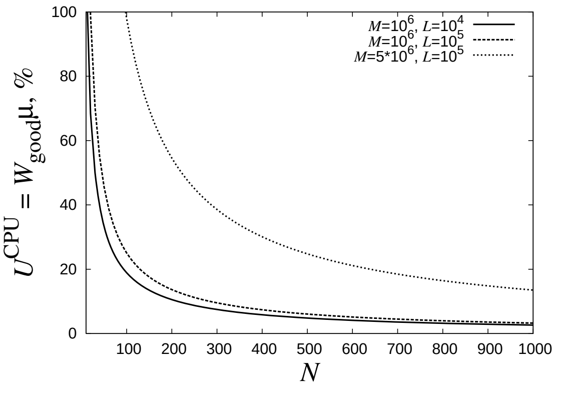
(a) CPU utilization U^{CPU} for several values of M



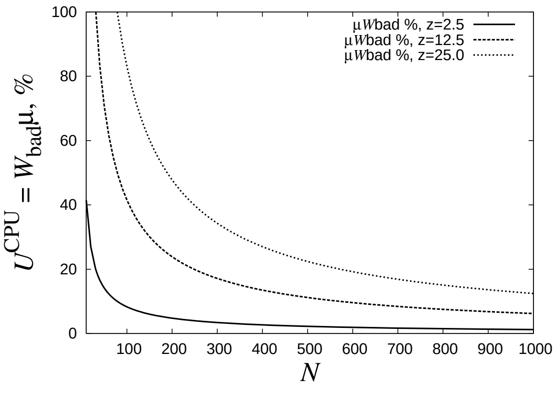
(b) Internal latency of the optimized association setup

Figure 6: The utilization/latency trade-off for the mobile peers scenario. $\alpha = 1/2$, $\lambda_{\text{s}} = \lambda_{\text{so}} = 30\ \text{min}^{-1}$, $\lambda_{\text{u}} = 5\ \text{min}^{-1}$, $P_{\text{us}} = 10^{-2}$.

Figure 6 presents the case when workload is generated in accordance to the mobile peers scenario, Eq. (2). The plot in Figure 6(a) shows a rapidly decreasing load with the growth of N (order of $\frac{\log N}{N}$). On the other hand, the internal latency increases slowly with the growth of N (order of $\log N$). Figure 6(b) shows the internal latency for the optimized association setup; the latency of a pure association setup, double jump and HIT insertion is proportional with coefficients 3, 1/2 and 1, respectively. The latency of the location update does not depend on N and equals $\tau + \mu \approx \tau$, i.e., $100 \ldots 200$ ms.



(a) Utilization by good workload



(b) Utilization by injurious workload

Figure 7: The utilization/latency trade-off in case of M mobile hosts, L servers, and Z zombie hosts. $\alpha = 1/2$, $\lambda_{\text{i}} = 1\ \text{h}^{-1}$, $\lambda_{\text{r}} = 30\ \text{s}^{-1}$, $\lambda_{\text{mob}} = 6\lambda_{\text{s}} + 2\lambda_{\text{so}} + \lambda_{\text{u}} P_{\text{us}}$, $\lambda_{\text{s}} = \lambda_{\text{so}} = 30\ \text{min}^{-1}$, $\lambda_{\text{u}} = 5\ \text{min}^{-1}$, $P_{\text{us}} = 10^{-2}$.

Figure 7 shows utilization for M mobile hosts and L stationary Internet servers. These two sets generate good workload W_{good} according to Eq. (3), as shown in Figure 7(a). Injurious utilization is plotted in Figure 7(b) for different values of $z = \lambda_z Z$.

5.2 Scalability problems

Rapidly decreasing workload and slowly increasing internal latency are attractive properties of the $Hi3$ control plane inherited from Chord [18]. It enables $Hi3$ to scale well, as shown in detail below.

Let us introduce a simple solution to the utilization/latency trade-off. Despite its simplicity and coarseness, it is useful for evaluating several important scalability problems. We show the estimates only for the CPU utilization; the throughput utilization is a subject of future work.

5.2.1 Estimating the needed $i3$ size

The problem is to find an interval for N values that keeps utilization reasonable for the workload presented in Section 4.2. For these values, the internal latency can be estimated using equations in Table 1.

Consider the mobile peers scenario with $\tau = 100$ ms. Require the latency of the optimized association setup to be at most 1 second[3] w.h.p. As shown in Figure 6(b), the $i3$ size can reach 1000 nodes without exceeding the latency bound. As shown in Figure 6(a), several hundred $i3$ nodes can serve several million mobile hosts within reasonable utilization bounds.

Consider the workload scenario shown in Figure 7(a) with L servers added. Increasing L from 10^4 to 10^5 increases utilization less compared to increasing M from 10^6 to $5 \cdot 10^6$. The result confirms that $Hi3$ is more resilient to the number of stationary servers.

5.2.2 Resistance to DDoS attacks

Below we present analysis for a zombie DDoS attack to the $Hi3$ control plane. Let a threshold $\overline{W}$ for total workload of an $i3$ node be fixed, e.g., $\overline{W}\mu = 50\%, 70\%, or\ 90\%$. We assume that $i3$ can handle the given workload if $W_{\text{good}} + W_{\text{bad}} \leq \overline{W}$, where W_{good} is good workload generated by $L + M$ legitimate hosts and W_{bad} is injurious workload from zombie DDoS attacks. According to Eq. (4) the critical value for the rate×size of the zombie set is

$$z = \lambda_z Z = \left(\overline{W} - W_{\text{good}}\right) \frac{N}{\alpha \log N} \tag{5}$$

The dependency is plotted in Figure 8 for several values of the threshold.

Let us estimate the size of the zombie set based on the critical value z. Assume that links between $i3$ nodes have much higher bandwidth compared with links between zombies and $i3$. In the worst case, a HIP packet has a size of several hundred bytes or approximately 0.5 KByte = 4 Kbits. If every zombie host has 1 Mbits/s access to $i3$ then $\lambda_{\text{r}} \leq 0.25$. For $Z = 100$ we have $\lambda_{\text{r}} Z \leq 25$. According to Figure 8, about 120...230 nodes can resist the attack with utilization at most 90...50%. For 1000 nodes, the maximal number of zombies is in the range of $400 \dots 700$.

For large N, $z = \text{rate} \times \text{size}$ grows a little slower than linearly. More precisely, $z = z(N) = a\frac{N}{\log N} + b \log N + c$, where a, b, and c are constants (see Eqs. (2) and (3)). Such behavior

[3] In this case, a pure association setup and a double-jump would take 3 and 0.5 seconds, respectively.

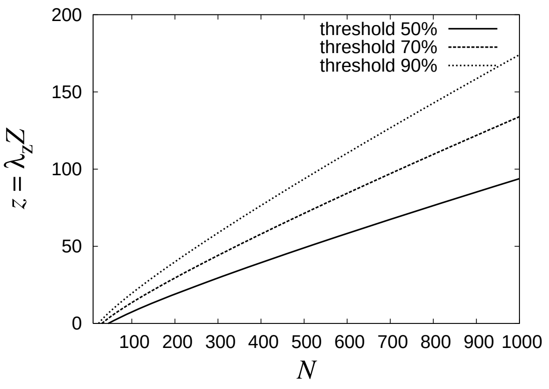


Figure 8: Critical values of the zombie rate×size product acording to Eq. (5), where W_{good} is defined by Eq. (3), $M = 10^6$, $L = 10^5$, $\alpha = 1/2$, $\lambda_{\text{i}} = 1\ \text{h}^{-1}$, $\lambda_{\text{r}} = 30\ \text{s}^{-1}$, $\lambda_{\text{mob}} = 6\lambda_{\text{s}} + 2\lambda_{\text{so}} + \lambda_{\text{u}} P_{\text{us}}$, $\lambda_{\text{s}} = \lambda_{\text{so}} = 30\ \text{min}^{-1}$, $\lambda_{\text{u}} = 5\ \text{min}^{-1}$, $P_{\text{us}} = 10^{-2}$.

is due to the Chord protocol where most requests require $O(\log N)$ hops. However, an *i*3 node can cache IP addresses of other nodes reducing the lookup time to $O(1)$. Although this enables *i*3 to be a one-hop network for repeating requests, it also complicates its operation in the presence of churn, when nodes enter and leave the Chord ring. Evaluation of such effects is left for future study.

6 Conclusions

In this paper, we presented analytical analysis of an Internet control plane created by integration of HIP and *i*3. Our workload model includes a conservative (w.h.p.) case of host requests, as well as the case of heavy load generated by a DDoS attack of zombies. The scalability analysis of the control plane outlines the trade-off in balancing the latency of the control infrastructure vs. decreasing the utilization of *i*3 servers.

Summarizing the results, the *Hi*3 control plane has the following scalability properties.

- The workload and the internal latency scale well: order $\frac{\log N}{N}$ of decreasing and order $\log N$ of increasing, respectively.
- A few *i*3 nodes ($N \sim 10^2$) is sufficient for a large set of HIP hosts ($M + L \sim 10^6 \ldots 10^7$).
- The resilience of *Hi*3 control plane to DDoS attacks behaves as $z(N) = O(\frac{N}{\log N})$, which is close to linear (proportional) behavior for large values of *i*3 size.
- For reasonable values of *i*3 size ($N \sim 10^2 \ldots 10^3$) the internal latency is satisfactory: at most a few seconds for rare requests such as association setup and simultaneous host mobility; for location update the internal latency does not depend on N and is at most a few hundred milliseconds (a substantial part of the total request latency is outside of *i*3).

The model is based on estimates that are high-probability bounds between typical values and the worst case. In practice, the trends of estimated parameters are more optimistic. For instance, when caching of Chord IDs is enabled in *i*3, the control plane is certainly more resilient to DDoS attacks.

At larger scales, performance evaluation using measurements or simulation can be prohibitively expensive or impossible. Even given its clear limitations, the simple model used in this paper is a suitable tool to study a new large-scale networking architecture before its wide deployment. In future work, we plan to extend our first approximation analysis to include details of the *Hi*3 infrastructure such as caching, and perform more extensive calibration of a model using measurements.

Acknowledgments

The authors thank Ion Stoica, Pekka Nikander, Teemu Koponen, and Abhinav Pathak for comments that helped to improve the quality of this paper.

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

[4] B. Ford. Unmanaged Internet Protocol: taming the edge network management crisis. *ACM Computer Communication Review*, 34(1):93–98, 2004.

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

[6] A. Gurtov and T. Koponen. Hi3 implementation for Linux, June 2005. Available at http://infrahip.hiit.fi and http://www.openhip.org.

[7] 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.

[8] 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.

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

[10] 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.

[11] D. Malkhi, M. Naor, and D. Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 183–192, New York, NY, USA, 2002. ACM Press.

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

[13] 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.

[14] 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.

[15] P. Nikander, J. Arkko, and B. Ohlman. Host identity indirection infrastructure (Hi3). In *Proc. of The Second Swedish National Computer Networking Workshop 2004 (SNCNW2004)*, Karlstad, Sweden, Nov. 2004.

[16] 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.

[17] 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.

[18] 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.

[19] H. Tschofenig, A. Nagarajan, V. Torvinen, J. Ylitalo, and J. Grimminger. NAT and firewall traversal for HIP: draft-tschofenig-hiprg-hip-natfw-traversal-02, July 2005. Work in progress. Expires in January 19, 2006.