A Layered Naming Architecture for the Internet

Hari Balakrishnan[*] hari@csail.mit.edu

Karthik Lakshminarayanan[†] karthik@cs.berkeley.edu

Sylvia Ratnasamy[‡] sylvia@intel-research.net

Scott Shenker[†§] shenker@icsi.berkeley.edu

Ion Stoica[†] istoica@cs.berkeley.edu

Michael Walfish[*] mwalfish@csail.mit.edu

ABSTRACT

Currently the Internet has only one level of name resolution, DNS, which converts user-level domain names into IP addresses. In this paper we borrow liberally from the literature to argue that there should be three levels of name resolution: from user-level descriptors to service identifiers; from service identifiers to endpoint identifiers; and from endpoint identifiers to IP addresses. These additional levels of naming and resolution (1) allow services and data to be first class Internet objects (in that they can be directly and persistently named), (2) seamlessly accommodate mobility and multi-homing and (3) integrate middleboxes (such as NATs and firewalls) into the Internet architecture. We further argue that flat names are a natural choice for the service and endpoint identifiers. Hence, this architecture requires scalable resolution of flat names, a capability that distributed hash tables (DHTs) can provide.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Internet*; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed databases*

General Terms

Design

Keywords

Naming, Internet architecture, name resolution, distributed hash tables, middleboxes, global identifiers

[*]MIT Computer Science and Artificial Intelligence Lab

[†]UC Berkeley, Computer Science Division

[‡]Intel Research, Berkeley

[§]International Computer Science Institute (ICSI)

SIGCOMM'04, Aug. 30–Sept. 3, 2004, Portland, Oregon, USA.

1. INTRODUCTION

Despite its tremendous success, the Internet architecture is widely acknowledged to be far from ideal, and the Internet's increasing ubiquity and importance have made its flaws all the more evident and urgent. The case for architectural change has never been stronger, as witnessed by the burgeoning set of architectural critiques and counter-proposals emerging from the research community (*e.g.*, [2, 5–8, 41, 45, 55, 56]). Ironically, the growth that motivated these proposals now makes their success unlikely: the sheer size of the Internet's installed router infrastructure renders significant changes to IP almost impossible. The decade-long struggle to deploy IPv6 should give any aspiring network architect pause.

Rather than attempt the Sisyphean task of modifying routers, we focus on improving a more malleable facet of the architecture: naming.[1] Although this restriction in focus prevents us from addressing issues that inherently involve routers (such as complete denial-of-service protection, fine-grained host-control over routing, and quality-of-service) there are many issues for which changes to IP would be irrelevant—and for which changes to the naming architecture would be crucial.

The current Internet has only two global namespaces, DNS names and IP addresses, both of which are tied to pre-existing structures (administrative domains and network topology, respectively). The rigidity and paucity of these namespaces are responsible for a variety of architectural ills. For instance, the Internet is now widely used by applications to gain access to services (processes that are remotely invoked by clients, such as Web servers) and data (files, streams, etc.), yet the Internet does not have a mechanism for directly and persistently naming data and services. Instead, both are named relative to the hosts on which they reside. Using DNS to name data overloads the names and rigidly associates them with specific domains or network locations, making it inconvenient to move service instances and data, as well as to replicate them [23, 36, 50, 51, 59]. In this sense, the Internet's current host-centric naming treats data and services as second-class network citizens.

In addition, users and system administrators often resort to architecturally suspect middleboxes—such as NATs/NAPTs [52], firewalls and transparent caches—because they cannot get similar functionality within the architecture. The well-known architectural problems posed by today's middleboxes include violating IP semantics and making the Internet application-specific;

[1]Of course, our naming proposal requires alterations to host software and, as we discuss later, a new name resolution infrastructure. These alterations are a significant deployment barrier but not one as unyielding as changing the router infrastructure. We will return to this issue in Section 6.

see [15, 31, 60] for details.

To remedy these and other architectural problems, in this paper we revisit the issue of naming. We begin by describing four general design principles about the nature and use of names. While these principles are seemingly innocuous, they are routinely violated in today's Internet. We claim that adherence to these principles requires a naming framework with four layers: user-level descriptors such as search keywords, e-mail addresses, etc.; service identifiers (SIDs); endpoint identifiers (EIDs); and IP addresses or other forwarding directives.[2] We then propose an architecture that makes essential use of these namespaces. This architecture has the following benefits: (1) services and data become "first-class" Internet objects, in that they are named independent of network location or DNS domain and thus can freely migrate or be replicated across host and administrative boundaries, (2) mobility and multi-homing of hosts can be gracefully accommodated, and (3) network-layer and application-layer middleboxes (which we rechristen "intermediaries") can be interposed on the data path between two communicating endpoints.

Our principles, naming framework, and architecture rely heavily on existing proposals. From Nimrod [7] and the Host Identification Protocol (HIP) proposal [32, 33, 35], we borrow the idea of decoupling the transport and networking layers to address mobility and multi-homing. From the Unmanaged Internet Protocol (UIP) proposal [14], we borrow the idea of using this same decoupling to address problems that result from private addressing realms, such as those created by NATs. From the Internet Indirection Infrastructure (i3) [53], we borrow the idea of source-directed indirection. From Semantic-Free Referencing (SFR) [59], we borrow the idea that the service identifier namespace be *flat* (meaning that the identifiers are unstructured and not overloaded with any semantics about the object being named, *e.g.*, a flat identifier might be a number chosen uniformly at random from $[0, 2^{128}-1]$), and from HIP and UIP again, we borrow the idea that the endpoint identifiers be flat. Our proposal thus requires a name resolution infrastructure that can scalably resolve flat names. Distributed hash tables (DHTs) represent one possible solution to this resolution problem (see [3, 40, 42, 54, 61] for background on DHTs), and so we borrow from that literature as well.

Thus, this work is a pastiche of borrowed elements; our contribution is both the distillation of some basic principles and their synthesis into a coherent architecture. We present our four basic design principles in Section 2, followed by a description of the architecture and its benefits in Section 3. A key aspect of the proposal is flat names, and we discuss the issues associated with them in Section 4. We survey related work in Section 5, and in Section 6 we conclude with a brief discussion.

2. DESIGN PRINCIPLES

> *Those are my principles, and if you don't like them... well, I have others.*
>
> Groucho Marx

We now present four basic design principles that we feel are essential to the nature and use of Internet names.

[2]This naming hierarchy is nothing more than a particular realization of Saltzer's taxonomy of network elements [44], in which he identified users/services (our SIDs), hosts (our EIDs), network attachment points (IP addresses), and paths. Since we don't consider aspects of the architecture that require router involvement, we don't address the issue of naming paths.

2.1 Names and Protocols

Our first design principle addresses the role of names in protocols.

> Principle #1: *Names should bind protocols only to the relevant aspects of the underlying structure; binding protocols to irrelevant details unnecessarily limits flexibility and functionality.*

This seemingly innocuous principle is routinely violated in today's architecture. When applications request a service or data, they care only about the identity (for service) or content (for data) of the object they requested; the particular end-host servicing a request is immaterial. However, today's DNS-based names for services and data (*e.g.*, URLs like `http://abc.org/dog.jpg`) force applications to resolve service and data names down to an IP address (*e.g.*, to fetch the data named by the URL above, the Web browser itself, rather than a lower level software module, has to learn the IP address represented by `abc.org`), thereby binding the application request to a particular network location, as expressed by an IP address. This resolution violates Principle #1 twice over: it binds data and services to particular end-hosts—and, even worse, to the network locations of those end-hosts. (In the rest of this paper, for brevity, we mostly use the term "service" to mean "service and data.")

Rectifying this double violation requires the introduction of two (and only two) new naming layers. First, Principle #1 implies that applications be able to refer to services with persistent names that aren't tied to the endpoint hosting the service. We therefore claim that a class of names called *service identifiers (SIDs)* should exist that give applications exactly this ability. We think that humans and the software they use should get these SIDs as the output of various mapping services that take as input *user-level descriptors*. By user-level descriptors, we mean handles in various formats that humans can exchange (*e.g.*, search queries, e-mail addresses). See [37, 58, 59] for discussion about such mapping services.

Second, transport protocols exchange data between two endpoints, and the network locations of the endpoints are irrelevant to the basic semantics of transport. Only at the IP layer is the IP address naturally part of the protocol semantics of best-effort packet delivery between network-layer addresses. Today, however, the semantics of IP are wound into the transport layers. For example, hosts name TCP connections by a quadruple that includes two IP addresses. As a result, a TCP connection fails when the IP address of an endpoint changes,[3] and a TCP connection on a multi-homed endpoint cannot use more than one of the IP addresses at a time. Principle #1 suggests that transport protocols should be able to refer to endpoints in a manner independent of their IP address or network topology. We thus adopt—from previous work, as mentioned in Section 1—the idea of a topologically independent *endpoint identifier (EID)* that uniquely identifies a host.

These two new naming layers that have been motivated by Principle #1 require two additional layers of name resolution: from SIDs to EIDs and from EIDs to IP addresses. To interact with a

[3]One solution to this problem, Mobile IP [34], treats the mobile host's "home" IP address as a permanent identifier and relies on IP-layer packet interception and redirection. Another solution is migrating TCP connections "in-band" [48]. A third is allowing connections to break but using a session layer to re-initiate broken connections [47, 49], giving applications the abstraction of an uninterrupted connection. These three solutions work *around* the fundamental issue: endpoints are named by topological identifiers (IP addresses). None directly addresses the architectural shortcoming.

service (*e.g.*, a Web server), the application initiates a *communication session* whose destination is named by the service's SID. When an application resolves that SID, it gets one or more EIDs that identify the end-hosts that run the service. The session will typically involve one or more transport-layer (*e.g.*, TCP) connections between the client and service EIDs. Before invoking IP, the transport layer resolves the EID to the current set of IP addresses to which the EID is attached.

A crucial property of this layering is that the resolution of a SID to the eventual set of IP addresses for the communication session does not happen prematurely. More concretely, applications generally deal with SIDs (after perhaps resolving user-level descriptors), transport protocols generally deal with (or bind to) EIDs, and only IP itself deals with IP addresses. The resulting bindings are thus accurate and appropriate even in the face of host mobility and service migration. For instance, if the EID-to-IP mapping changes, then the transport layer can re-initiate an EID lookup to rebind [35, 48]. If a service moves, or is copied, to another location, a new SID lookup provides the current SID-to-EID bindings; if a service were to move while a session were in progress, the application might initiate such a lookup to continue the session.

2.2 Namespaces and Network Elements

Principle #1 concerned how names should relate to protocols. Our second design principle discusses how names should relate to their referents. When users care about the identity of an object rather than its location, the object's name should be *persistent* in that it remains unchanged even when the object's location changes.

> Principle #2: *Names, if they are to be persistent, should not impose arbitrary restrictions on the elements to which they refer.*

The two current global namespaces, IP addresses and DNS names, are each closely tied to an underlying structure. Achieving scalable routing requires that IP addresses reflect network topology. DNS names, though more flexible, nonetheless reflect administrative structure.

As has been noted in the URN literature [23, 50, 51] and by others [4, 37, 59], DNS-based names for data are inherently ephemeral. Data does not naturally conform to domain boundaries, so data is likely to be replicated on, or moved to, hosts outside the originating domain. When this replication or movement happens, existing references to the data become invalid; *e.g.*, if the file `dog.jpg` moves to `def.org` from `abc.org`, existing Web links that reference `abc.org` are now useless. (For a more complete discussion of the problems of hostname/pathname URLs, see [50, 51, 59] and citations therein.) The same difficulty arises when services move and there are pre-existing pointers to those services (*e.g.*, when a popular FTP server encodes a DNS name that is no longer correct), though one might argue that services are less peripatetic than data.

Thus, no namespace currently exists that can persistently name data and services. Some of the URN literature proposes a new namespace and resolution mechanism [11] for each *genre* (*e.g.*, ISBN numbers would have their own canonical resolver). Partitioning allows resolution to scale since different resolver types can incorporate *genre*-specific knowledge, but then adherence to Principle #2 depends on an accurate mapping of elements to genres and on an element's never changing genres. In contrast, the Globe project [4], Semantic-Free Referencing [59], and Open Network Handles [37] take an entirely different approach: they advocate a single new *flat* namespace that can serve all present and future network elements. A flat namespace has no inherent structure and so does not impose any restrictions on referenced elements, ensuring universal compliance with Principle #2. In this paper, we adopt this second approach, using a flat namespace for SIDs and EIDs.

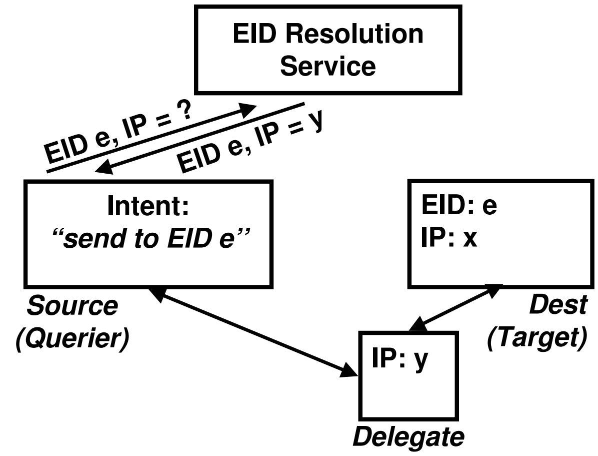

Figure 1: EID-level delegation. A source queries on a given EID and is given the IP address of a delegate. The source could also be given the delegate's EID or multiple EIDs (not shown).

2.3 Resolution and Delegation

Our first two design principles concerned the role of names. Our third addresses how these names are resolved. The typical definition of "resolving a name" is mapping a name to its underlying "location". In our case, an SID's "location" would usually be an (EID, transport, port) triple[4] and an EID's location would be an IP address. However, we think this typical definition is too restrictive and instead adopt the following more general notion of resolution.

> Principle #3: *A network entity should be able to direct resolutions of its name not only to its own location, but also to the locations or names of chosen delegates.*

In any logical network connection, the initiator at any level (*e.g.*, a human requesting a Web page or an endpoint initiating a transport connection) intends to connect to a destination entity. In our case, for example, transport protocol entities connect to destination EIDs. However, the destination entity may not want to handle the connection directly, preferring instead to direct the connection to a chosen *delegate*, as shown in Figure 1. This kind of delegation neither alters essential trust relationships (if you trust an entity, you trust its delegates), nor interferes with established protocol semantics, as will be seen when we describe the details of such delegation in Section 3.2.[5]

While the recipient-controlled delegation in Principle #3 might seem esoteric at first, it is crucial to the overall architecture. As we describe in Section 3, delegation allows the architecture to gracefully incorporate *intermediaries*, which we define as cleaner and more flexible versions of middleboxes. Delegation also yields some

[4]Resolving an SID can also return meta-data (such as a pathname on a Web server) in addition to the "location", thereby allowing data (in this case a Web page) to be named by an SID.

[5]Recipient-controlled delegation could accommodate the kind of distributed network element envisioned in [9]; that is, the destination and its delegate could be part of the same logical element even if they are physically distinct.

protection against denial-of-service attacks, as discussed in Section 3.2.

2.4 Sequences of Destinations

In traditional IP routing, the routing protocol is responsible for choosing the packet's path through the network. However, there have been many *source routing* proposals in which sources are given the power to specify the path or, in the case of *loose source routing*, a few points along the path. We believe that this ability should be available not just at the routing layer (which is not our concern here) but also at the endpoint and service layers.

More specifically, the abstraction of sending to a destination should be generalized to allow sources to indicate that their packets should traverse a series of endpoints (specified by a sequence of EIDs) or that their communications, the granularity of which we address later, traverse a series of services (specified by a sequence of SIDs). Such abstractions would generalize the notion of a destination to a sequence of destinations. Note that since these various destinations are not specified at the IP layer, but rather at the endpoint and service layers, these intermediate points do not merely forward the packets but may act on them in non-trivial ways.

Combining this sentiment with Principle #3 suggests that endpoints and services should be able to have their names resolve not just to a single location but more generally to a sequence of identifiers (either IP addresses or EIDs). In this way, both senders and receivers could loosely dictate the paths of packets sent from them or destined for them. This idea is captured in our fourth, and final, design principle.

> Principle #4: *Destinations, as specified by sources and also by the resolution of SIDs and EIDs, should be generalizable to sequences of destinations.*

3. ARCHITECTURE

We should first note that our belief in the general principles above is deeper than our conviction about any of the architectural details that follow. The description below is intended to illustrate how to achieve the benefits that flow from these principles, but one should view this architecture merely as an existence proof that the general principles can be realized, not as their definitive embodiment. In fact, many of the details here arose during an implementation effort, described in [60].

The four general principles led us to claim that (1) two additional sets of names (SIDs and EIDs) should exist, (2) these names should be flat, (3) the architecture should support delegation as a basic primitive, and (4) destinations, whether specified by the source or receiver, can, in fact, be sequences of destinations. In this section, we present an architecture that results from these claims, first focusing on the consequences of SIDs and EIDs (Section 3.1), then on delegation (Section 3.2). We defer discussing the consequences of flat names to Section 4, and we incorporate the notion of sequences in our description of delegation. In the process of describing the architecture, we note how it yields the three benefits stated in Section 1, namely making services and data first-class objects, support for mobility, and graceful incorporation of intermediaries.

3.1 EIDs and SIDs

We start by describing how this architecture works in the basic case. The discussion of intermediaries in Section 3.2 will complicate the story. Although we will refer to SIDs and EIDs throughout Section 3, not every application will require both SIDs and EIDs. The two mechanisms are logically distinct and need not be coupled.

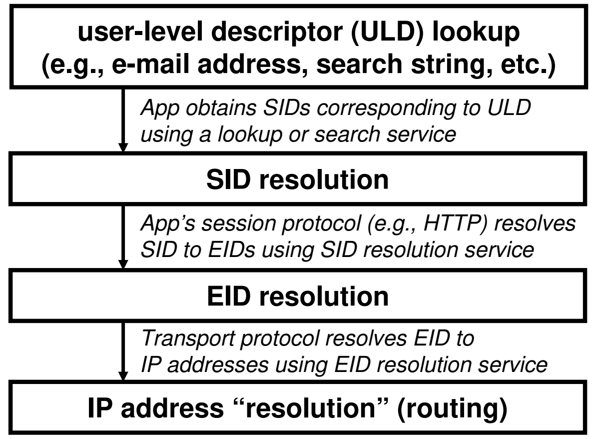

Figure 2: The naming layers.

Principle #1 led us to claim that applications should bind to SIDs and transport protocols should bind to EIDs. Thus, applications must use a layer between them and transport that resolves SIDs to EIDs, and similarly, transport protocols must use a layer between them and IP that translates between EIDs and IP addresses. We will call the layers *resolution* layers, though they do more than simply resolve identifiers. These layers could be separate libraries or software that is part of the application or transport protocol. The resolution layers result in the naming architecture depicted in Figure 2. We now give more details on how these resolution layers are used, focusing on how they fit into the overall architecture. We defer the mechanics of resolution to Section 4.

In what follows, we assume that humans and the applications under their control have already used an auxiliary mapping service (*e.g.*, a search engine) to map a user-level descriptor (*e.g.*, a search query) to an SID. As a result, we will not discuss user-level descriptors and will instead assume that applications have in hand an SID representing a service or data.

SID resolution: Consider an SID-aware application, a, running on a given host, h, and say that a wishes to gain access to a service or data represented by an SID s. The application hands s to the SID resolution layer, which contacts the resolution infrastructure (one realization of which is described in Section 4) and is handed back one or more (EID,transport,port)[6] triples, where each triple represents an instance of the desired service. Following the approach in [59], if the SID abstracts a data item, not just a service, then the SID resolution layer would also receive, for each triple, an application directive. For example, if s represented a Web server, then the triple returned might be (EID of the Web server, TCP, port 80). If s represented a Web page, not just a Web server, then a pathname on the Web server might also be returned.

The functions we next describe might be abstracted by an application-independent library. However, since the library would be under a's control, and since some applications might want different behavior and thus elect not to use the library, we will describe the actions as performed by a, not by the library. Given

[6] Since EIDs are not required, the triple could be (id,transport,port), where id is another host identifier, such as a DNS name or an IP address. However, as we have noted, such usage would not cope with host mobility.

a triple of the kind mentioned above, a would communicate with the specified EID using the specified transport protocol and port number (or other transport-specific information). The transport protocols, now bound to EIDs (instead of IP addresses), would use h's EID as the source EID and the one from the triple as the destination EID. Depending on the application semantics, a might use multiple triples for simultaneous connections, or it might use multiple triples as backups in case the current connection failed. If all of the triples failed, a could re-invoke the SID resolution layer to re-resolve s to check for new triples.

EID resolution: The transport protocol prepares one or more packets to send, which it passes down to the EID resolution layer. The EID resolution layer resolves the destination EID into one or more IP addresses. (Multiple IP addresses could arise for multi-homed hosts and also when a logical endpoint represented a collection of physical machines, each with its own IP address.) When handing control to the IP layer, the EID resolution layer uses one of the returned IP addresses as the destination IP address, and the source IP address is that of the sending host. If the destination host is unreachable, the EID layer can use another IP address if it received more than one from the resolution step.[7] If none of the previously returned IP addresses works, the EID resolution layer re-resolves the EID in case the corresponding destination IP addresses have changed.

Where do these identifiers go? Packets are logically destined for endpoints, which are identified by EIDs. Hence, we imagine that EIDs would be carried in packets to identify the packet's logical endpoint. In Section 3.2, we say why the sender must put the destination EID in the packet. See [60] for a more complete description of one instantiation of this architecture at the EID level.

The case of SIDs is conceptually identical. SIDs name services or data, and so the SID must often be carried in band, like the EID. However, the SID is not required to be in every packet but rather in each *logical piece of data* being communicated between sender and recipient. The actual location of SIDs in data streams would vary by application and by what the SID is being used for. For example, the SID corresponding to a given SMTP server might be carried in an e-mail header. Similarly, the SID corresponding to an HTTP Web proxy might be in the HTTP header. If an SID named a Web page, the SID might again be somewhere in the HTTP header. We will use the term *application data unit* [10] to mean "a coherent unit of data transmitted between applications." In the examples above, the e-mail and HTTP requests were the ADUs.

Benefits: As explained in Sections 2.1 and 2.2, naming data and services with SIDs overcomes the problems of using DNS-based URLs for that purpose. Naming endpoints with EIDs provides natural solutions to mobility and multi-homing: if an endpoint identified by EID e changes its IP address, then the EID resolution layer on a peer of the endpoint will re-resolve e to find the new IP address.[8] As explained in [32, 33, 35], this rebinding enables continuous operation in the presence of mobile or renumbered hosts and provides smooth failover for multi-homed hosts; we direct the reader to these references for more details.

[7] We envision, as in HIP, using explicit end-to-end signaling for expected address changes and using EID resolution layer keepalives to detect unexpected address changes or other failures.

[8] This re-resolution could conceivably occur on each packet but more likely will be invoked only when the EID layer on the peer detects failure.

3.2 Delegated Bindings and Intermediaries

The other major aspect of our architecture, delegation, is dictated by principle #3. In this section we explain the mechanics of delegation, and then describe how delegation provides support for intermediaries.

Delegation: At the EID layer, a host with EID e can insert the IP address or EID of a different host in the resolution infrastructure. As a result, when a third host establishes a transport connection to e, its packets actually go to the delegated host. The host identified by e must establish state at the delegated host—through some protocol outside the scope of our discussion—so that when packets arrive at the delegated host they can be forwarded. The intermediary uses the destination EID, which is carried in every packet, to determine the intended ultimate recipient of the packet. This type of intermediary is *network-level*, in that it is a delegate for an endpoint, not a service.

At the SID layer, the mechanism is similar: a service s, running on a host h, rather than listing the EID of h in the resolution infrastructure, instead lists the EID of some other endpoint, o. (s could also list an SID instead of an EID, and this SID would map to an EID.) s would have to establish state at o so that o would know how to handle ADUs destined for s. o could be, for example, an application-level gateway: hosts trying to contact s would have their connections terminated at endpoint o, and the gateway would inspect the ADUs, and then make a decision about whether to forward them. The reason that s's SID must be in the data stream near or in the ADU is to let o know which service is the logical destination of the ADU: o might be a gateway for other services s', s'', etc. We will call endpoints such as o *application-level* intermediaries and give examples of such intermediaries below.

In accordance with Principle #4, at the SID (resp., EID) level, receiving entities could express the fact that more than one intermediary should be involved: services (resp., endpoints) could list in the resolution infrastructure a *sequence* of SIDs (resp., EIDs). Each of these identifiers represents an intermediary that the receiver wants the ADU (resp., packet) to visit on the way to the final destination.

However, Principle #4 also applies to the source, so our architecture allows sources to specify a sequence of EIDs or SIDs to be traversed, via the well-known mechanism of stacked identifiers (used by i3 [53] and others). One can think of these waypoints as source-controlled (as opposed to receiver-controlled) intermediaries; the source can express that it wants one or more intermediaries to *send* on its behalf, just as the destination can express through the resolution of its EID that it wants one or more intermediaries to *receive* on its behalf. An intermediary, which is assumed to be a chosen delegate of either sender or receiver, can also make decisions on behalf of the delegator (which might include pushing additional identifiers onto the destination stack). These two mechanisms, sender- and receiver-controlled indirection, are not exclusive: when both entities specify intermediaries, the source creates the actual sequence of intermediaries by concatenating its desired sequence to the sequence specified by the receiver (which is returned in the resolution step).

When the receiver and sender switch roles, the original receiver may need to resolve the original sender's EID to determine the path back to the sender. The same thing could occur at the SID level, which might require introducing the notion of a *source SID*.

Example use: At the EID level, the delegation mechanism described above—in which an endpoint inserts into the resolution infrastructure a map from its EID to the IP address of a delegate—can support standard network-level intermediaries (NATs/NAPTs, VPNs [19], and firewalls) cleanly and coherently. Depending on the scenario and the security assumptions, the intermediary may be configured for no access control if it is only doing NAT, for some access control if it acts as a firewall that allows only certain ports, or for much more stringent access control if it acts as a VPN box, logically interposed between a private network and the global Internet and only accepting packets from pre-specified EIDs.

To expand slightly on the firewall example, all hosts belonging to an institution could logically reside behind its network-level firewall; each such host would list the firewall's EID in the resolution infrastructure, and then would send to the firewall their own EID (and possibly IP address and security information allowing the firewall and the endpoint to authenticate each other), so that the firewall would know where to forward packets. This approach is detailed in [60].

One could also use EID-level delegation to provide some protection against denial-of-service (DoS) attacks. A server could shield itself from attackers by placing a forwarding intermediary between itself and untrusted clients and by installing traffic filters at the forwarding intermediary. This approach is identical in spirit to the overlay DoS protection schemes proposed in SOS [25] and Mayday [1]; our point here is merely to illustrate how their basic techniques can be implemented within our architecture. Of course, an attacker could launch a DoS attack by sending packets directly to a server's IP address, which our architecture cannot prevent since it leaves alone current routers. However, having all incoming packets directed through the same intermediary would simplify router-level packet filtering.

At the SID level, the delegation mechanism allows the owners of services and data items to invoke application-level proxies. For example, say that a given e-mail user, user@domain, wants to receive e-mail from an SMTP mail server after having it first scanned for spam and viruses at a third-party site specializing in this task. To achieve this functionality today, the administrator of "domain" makes the MX record of "domain" resolve to the third-party site. This approach is limited, however: first, an e-mail address cannot map to more than one intermediary, and, second, different e-mail addresses in the same domain are forced to resolve to the same mail server (though this limitation could be overcome by deploying the MB resource record [30], which works at the granularity of e-mail addresses).

SIDs can address these limitations. Before continuing, we emphasize that the following details give one possibility; better approaches, also based on SIDs, likely exist. Using SIDs, the owner of the address could insert into an auxiliary mapping service (one such service being DNS with a record type mapping e-mail addresses to SIDs) the mapping from user@domain (which functions here as a user-level descriptor) to a single SID, s, which identifies the e-mail account. The owner of the address would also have inserted into the SID resolution infrastructure a map from s to a sequence of destinations, $[s_1, s_2]$, where s_1 identifies the third-party virus filtering service and s_2 identifies the user's SMTP server. Observe at this point that the services can change administrative domains or be mobile or multi-homed, and as long as the mapping from the s_i to the corresponding EIDs is correct, the owner of the address need not get involved.

To send e-mail to user@domain, a mail agent would first resolve the user-level descriptor to get s, then resolve s to get $[s_1, s_2]$, and then send the e-mail to the endpoint represented by s_1 (which requires resolving s_1 to get an EID). Once the e-mail arrived at the third-party service, the third-party service would resolve s_2 to an endpoint and then send the e-mail message. The pseudo-code in Figure 3 gives more detail. Of course, for this approach to work, e-mail agents would have to be "SID-aware". In general, taking advantage of SIDs requires changes to application software, as discussed in Section 1.

```
// M is the message to be sent
// s1 is SID of virus filter
// s2 is SID of mail server
send_message()
{
    SID s <-- lookup(user@domain);
    SID_Sequence [s1 s2] <--- sid_resolve(s);
    send_email(s2|M, s1); // send "s2|M" to s1
}

send_email(s2|M, s1)
{
    EID e1 <-- sid_resolve(s1);
    tcp_connect(e1);  // note: TCP sees EIDs
    tcp_send(s2|M, eid1);
}

// s1 now sends cleaned message M' to s2
forward_cleaned_email(M', s2) {
    EID e2 <-- sid_resolve(s2);
    tcp_connect(e2); // note: TCP sees EIDs
    tcp_send(M', e2);
}
```

Figure 3: Pseudo-code showing SID-level delegation in e-mail example.

Other examples of application-level intermediaries include those for the Web. Owners of Web servers or particular Web objects could direct the resolution of the appropriate SID to a cache or to a sequence of transcoders with the result that Web clients requesting the given objects would be directed to this sequence of intermediaries. Senders could also invoke proxies by using SIDs. Both sender- and receiver-invoked intermediaries here would require changes to HTTP.

Although much of the functionality provided by our intermediaries can be achieved with today's middleboxes, we think intermediaries are a better approach, for several reasons. First, the intermediaries do not violate layering principles or protocol semantics; they only inspect packets or ADUs explicitly addressed to them. Second, they are explicitly invoked by endpoints (at the network level) or services (at the application level); no endpoint is forced to send its traffic through intermediaries. Of course, people may still deploy architecturally suspect middleboxes that impose their will on endpoints. Our point here is that these middleboxes are no longer necessary to achieve much of the same functionality. Third, because intermediaries are explicitly requested and globally addressed, they need not lie on the IP routing path between logical source and logical destination.

The purpose of this section was to illustrate, in general terms, how one might build an architecture around the principles of Section 2. We leave many fine points unresolved, such as the signaling protocols required to set up state at intermediaries, the software and API to interpret and create stacks of identifiers, and the detailed implications of this layered naming for host software.

Also, we have not discussed security. For now, we make only the following brief comments. Broadly speaking, the security issues of our proposal exist at the network and the application levels; at both levels, we believe our proposal makes things worse and better. At the network level, decoupling location and identity means that using IP routing to send a packet to a given *location* (via IP) no longer means that the packet is going to the host with the intended *identity* (EID). On the other hand, because EIDs are flat, they can hold cryptographic meaning; *e.g.*, the identifiers could be derived by hashing a public key. As a result, two communicating parties, given each other's identifiers, can authenticate each other in a way that they could not if hosts were identified only by IP address. HIP [32, 33, 35] and UIP [14] are premised on these facts—HIP is exactly designed to address security issues—and we can inherit many of the mechanisms therein. Moreover, as discussed above, we think our proposal provides a set of primitives that endpoints can use for network-level protection. At the application level, tradeoffs again arise. Our proposal sacrifices some convenience—one cannot tell by looking at an SID whether it corresponds to one's "intended" target—but achieves stronger security properties, since one's computer *can* tell by looking at an SID or EID whether the accompanying meta-data is correct. These issues are discussed in more depth in Section 4.2 and in [37, 59].

We now turn to an issue we've ignored until now: how can one effectively handle a flat namespace?

4. COPING WITH FLAT NAMES

As we argued in Section 2, flat names are uniquely able to provide persistence for all uses. However, flat names also pose significant problems. Several systems have been designed to meet these challenges, such as the Globe project [4], Open Network Handles [37], and SFR [59]. Here we discuss two troubling aspects of flat names: they are hard to resolve and they aren't human-readable. We discuss these issues in turn.

4.1 Resolution

DNS achieves scalability through hierarchy. It has been an assumption, often implicit, that scalable resolution required such structure. As a result, most network architecture proposals shied away from requiring new global namespaces. The advent of distributed hash tables (DHTs) suggests that flat namespaces can indeed be scalably resolved with a resilient, self-organizing, and extensible distributed infrastructure. The literature on DHTs is large and rapidly growing, so we don't review the technical details here. However, we note the following challenges, and possible remediations, that come from our use of DHTs (or of any other flat-namespace resolution method).

DHTs arose in the context of peer-to-peer (P2P) systems, but an unmanaged and untrusted P2P system would be unsuitable for a crucial piece of the Internet infrastructure. Instead, we envision a well-managed, distributed collection of machines providing the name resolution service using a DHT or other flat namespace resolution algorithm.

DNS's hierarchical delegation naturally ensures each name is unique and controlled by the relevant authority; flat names make these goals harder, but not impossible, to achieve. Several mechanisms exist for global uniqueness (see [29, 43] for example). Data integrity (*i.e.*, ensuring that no one else can change the resolution of an entity's name) is also challenging but possible (see, *e.g.* [37, 58, 59]).

DHTs' typical resolution time—$O(\log n)$ for an n node system—would be unacceptable for most name resolutions, particularly in comparison to DNS, since DNS often returns results from a local name server. This latency issue can be addressed on two levels. First, many DHT-style routing algorithms, either by design or through caching, have far better than $O(\log n)$ performance; see, for example, [21, 22, 39]. Second, a DHT-based resolution infrastructure can be designed using local proxies [59], local replication [24] or two-layered resolvers [29] that enable hosts within a local network to find local instances of entries written from within the network; these schemes also provide fate-sharing in that if an organization is disconnected from the rest of the Internet, its hosts can still gain access to entries written locally. See [29, 59] for a detailed explanation of these issues.

One advantage of the DNS infrastructure is that it has a built-in economic and trust model: domains provide their own name servers. The central facilities required (the root servers) are minimal and inexpensive. In contrast, our resolution infrastructure does not have the "pay-for-your-own" model, as names are stored at essentially random nodes. Our model raises the questions of who will pay and why users should trust the infrastructure. It would be foolhardy to predict the eventual economic model of such an infrastructure, but one could easily envision a future in which resolution service providers (RSPs) form a competitive yet cooperating commercial market much like current ISPs. Customers could pay for lookups and for storing, likely a flat fee for a reasonable number of accesses. The various RSPs would have mutual "peering" relationships to exchange updates, much as the tier-1 ISPs all interconnect today. Since each RSP would be judged by how well it served its customers, the RSPs would have incentives to process requests honestly.

4.2 Living in an Opaque World

More troubling than the performance and economic issues is the lack of semantics in the names themselves. A flat namespace is highly versatile but provides no user-readable hints. Although this fact poses little challenge for EIDs, which are replacing almost equally opaque IP addresses, difficulty arises when dealing with data and services for which the human-readability of URLs has been crucial. This issue is addressed at length (in somewhat different ways) in the various proposals mentioned above, so here we make only two comments. The first relates to how users obtain an SID. Users often find URLs through search engines rather than directly typing them into a browser; search engines could continue to perform the same function were services and data identified by SIDs. Moreover, third-parties could offer directory services mapping human-readable *canonical names* to SIDs. The advantage of these canonical names is that they are not part of the infrastructure and thus can be offered by multiple competing entities.

Our second comment is that users need some assurance that the SID they have in hand points to the intended target. A URL like `http://www.nytimes.com` provides hints (sometimes false) about its target but an opaque bit string gives no such assurance. Here, bit strings could be accompanied by meta-data that includes cryptographic statements like "Authority A says that this SID points to the newspaper New York Times." Again, authorities like Authority A would not be part of the resolution infrastructure but instead part of a competitive market of SID authenticators.

In addition, embedding cryptographic meaning in the identifiers—*e.g.*, by deriving an identifier from a collision-resistant hash of a public key [28]—allows users to verify that the output of the resolution step is the "correct" result for the given identifier.

5. RELATED WORK

Our work, as noted in Section 1, borrows heavily from three projects—HIP, SFR, and i3—and can be seen as synthesizing these works, each of which has a narrower goal, into a larger whole. However, many other works describe related ideas—so many that here we can only present a superficial overview.

Saltzer [44] was one among many [8,9,17,26,27,46] to make fine distinctions among network elements; the most common, and least practiced, of these distinctions is between a host's identifier and its address (see [26] for a comprehensive discussion of this topic). This distinction is embedded in two recent proposals: Peernet [13] and UIP [14] (from which we also borrow the EID mechanism). Both use overlays with DHT-inspired routing algorithms: Peernet serves mobile networks, and UIP seeks to interconnect heterogeneous networks, using all nodes in the network as routers. Our approach differs from UIP's in that while we look for mechanisms to accommodate middleboxes, UIP's overlay of peers makes certain classes of middleboxes, like NATs, transparent.

The Internet Indirection Infrastructure (i3) [53] uses flat identifiers and supports intermediaries and service composition. The chief difference between i3 and what we describe here is when the binding between identifier (SID or EID) and IP address is done. i3 uses late binding, having each packet (in the general case) sent to the resolution infrastructure, whereas the approach here uses early binding with the lookup occurring before packets are transmitted. However, the distinction between these approaches is blurred when i3 uses extensive caching or when our approach re-resolves on every packet.

Creating location-independent and persistent names for objects, and an accompanying infrastructure for resolution, has long been the goal of the URN literature [23, 50, 51]. In addition, the Open Network Handles work [36, 37] argues for flat, unfriendly domain names for Web *sites*. The Globe project [4, 58] envisions a single infrastructure for mapping from (possibly human-unfriendly) persistent object identifiers to current locations.

There are an increasing number of proposals that range from architectural enhancements to radical refactorings. These include earlier proposals like PIP [16], IPv6 [12], Dynamic Networks [38], Active Networks [55], Nimrod [7], and more recent proposals like Smart Packets [45], Network Pointers [56], Predicate Routing [41], Role-Based Network Architecture [5], and Ephemeral State Processing [6]. While each of these proposals shares at least some of our goals, they all differ in two respects: first, they would (in their full glory) require significant modifications to all network elements, not just hosts and middleboxes. Second, while some of these proposals are intended to obsolete middleboxes, none is intended to *accommodate* them.

Five other proposals deserve special mention. The authors of TRIAD [20] share nearly all of our motivations. They observe that data should be first-class objects in the modern Internet, capable of being addressed, and they, like many others, create location-independent end-host identifiers. The technical details of TRIAD's solution and our own are quite different: in TRIAD, the resolution step and the routing step are conflated, thereby improving latency, and at the shim layer between IP and transport, they use forward and reverse tokens that record the path taken, instead of stacks. However, the main difference between our proposals is that identifiers in TRIAD, both of hosts and data, are derived from domain names, and indeed, the TRIAD approach relies on the semantics and hierarchy of domain names to aggregate routes to content names. As we hold the conviction that persistent names ought to be flat, and as we have two layers of such names, our technical problems differ from those of TRIAD (and vice-versa).

IPNL [18] also shares many of our motivations. It creates separate end-host identifiers and leaves the core IPv4 routing infrastructure untouched. Under IPNL, the end-host identifiers are domain names, though the authors acknowledge that a flat, cryptographically strong identifier, as in HIP, may be preferable for security reasons.

FARA [8] is a meta-architecture that actual network architectures could "instantiate." In FARA, the basic unit of communication is the *entity* (analogous to our service), and packets are logically delivered from one entity to another, with no explicit invocation of the hosts underneath the entities or between them. The details of the exact mapping of our concepts to FARA's are beyond this paper's scope, but this mapping reveals that while our proposal could be viewed as an instantiation of, or consistent with, FARA, two aspects contradict FARA's spirit: first, FARA avoids notions of host identity. Second, FARA's AId (which identifies a connection between two entities) is supposed to have only local scope, whereas our analogous construct (the SID) is a global identifier.

P6P [57, 62] proposes a DHT-based infrastructure as a way to deploy IPv6: sites send IPv6 packets to their gateway DHT node, which treats the IPv6 destination address as a flat identifier, uses this identifier to look up the IPv4 address of a counterpart DHT gateway, and then sends the packet over traditional IPv4 to this counterpart, where the encapsulation is inverted and the packet is delivered to its destination. P6P shares many of our motivations but does not give hosts persistent names (if a site changes ISPs, all of the identifiers at the site change).

Mobile IP [34] creates host identifiers out of IP addresses, thereby separating location and identity, in some measure. Mobile IP also uses a form of delegation, but a limited one: when a given host is not in its home network, then an intermediary (called the home agent) must receive all packets logically destined for the host in question, and the intermediary is required to be in the host's home network.

6. DISCUSSION

This paper proclaims four design principles and derives from them a layered naming architecture that alleviates some of the Internet's current problems. Services and data could be named persistently yet flexibly, elevating them to first-class network elements. Middleboxes, long the bane of network architects, would be virtuously reincarnated as either application- or network-level intermediaries. Mobility would be seamless, and there would be modest, but by no means complete, protection against denial-of-service attacks.

While we believe in our proposal, the details are less important than three deeper messages we now emphasize. The first is that DHTs allow us, for the first time, to contemplate using flat namespaces in an architecture. While the transition to such namespaces is hardly painless, the payoff is profound. Once a flat namespace is established, it can be used to name anything. No longer will our old namespaces, DNS names and IP addresses, encumber network elements with their underlying structure. New applications will no longer face a Devil's choice between accepting the strictures of an existing but inappropriate namespace or bearing the overhead of creating a new one; instead, with a flat namespace, all new network elements can be effortlessly incorporated.

The second message is that the extra naming layers will shield applications from the underlying routers. One of the great frustrations of network architects is how quickly the Internet went from a flexible academic playground to an ossified commercial infrastructure. It feels, to many, as if a work-in-progress has been prematurely but permanently frozen in time. Perhaps one day significant changes will come to this infrastructure, or a general-purpose overlay will render it irrelevant. In the meantime, however, it seems crucial to insulate applications and protocols from this underlying infrastructure. Our layered naming architecture binds to IP addresses only at the lowest logical layer, thereby minimizing the extent to which the routing infrastructure constrains the protocols and applications above.

For a variety of reasons, including address exhaustion, attacks, and efficient delivery of content, there is a seemingly irreversible trend toward the interposition of functionality between communicating Internet endpoints. Currently, such functionality is implemented via middleboxes which are rightly criticized for violating the architecture and for reducing the Internet's flexibility. Our third message is that these adverse effects need not be the case. The concept of *delegation*, in which such functionality is explicitly invoked by the endpoints, allows interposition without violating the spirit of the end-to-end principle or the semantics of IP. Thus, we believe intermediaries retain the desired architectural purity and application flexibility, while achieving the aims of middleboxes.

Of course, our proposal faces serious hurdles. Incorporating the new naming layers requires significant changes to host software, both applications and protocols. Resolving these flat names requires a new resolution infrastructure. We do not underestimate the difficulty of making these changes; they are indeed massive challenges. However, both changes can occur incrementally. DHTs can be incrementally scaled, so in the beginning, when clients are few, the resolution infrastructure can be small; as demand grows, so can the size of the DHT. The host software can also be incrementally deployed; early adopters get a significant benefit, but they can remain (at least for a very long time) backward compatible with the old architecture. However, we don't mean to imply that deployment will be easy, only that it won't be impossible. This, unfortunately, is all one can hope for.

ACKNOWLEDGMENTS

We thank the anonymous reviewers, Anjali Gupta, and Eddie Kohler for their comments. This work was done as part of the IRIS project (`http://project-iris.net/`), supported by the National Science Foundation under Cooperative Agreement ANI-0225660. This work was also supported by NSF CAREER Award ANI-0133811, NSF ITR ANI-0205519, an NDSEG fellowship, and a Sloan Foundation Fellowship.

7. REFERENCES

[1] D. G. Andersen. Mayday: Distributed filtering for Internet Services. In *4th USENIX Symposium on Internet Technologies and Systems* , Seattle, WA, March 2003.

[2] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial-of-service with capabilities. In *2nd ACM Hotnets Workshop*, Cambridge, MA, Nov. 2003.

[3] H. Balakrishnan, M. F. Kaashoek, D. Karger, and R. Morris. Looking up data in P2P systems. *Communications of the ACM*, 46(2):43–48, Feb. 2003.

[4] G. Ballintijn, M. van Steen, and A. S. Tanenbaum. Scalable user-friendly resource names. *IEEE Internet Computing*, 5(5):20–27, 2001.

[5] R. Braden, T. Faber, and M. Handley. From protocol stack to protocol heap – role-based architecture. In *1st ACM Hotnets Workshop*, Princeton, NJ, Oct. 2002.

[6] K. L. Calvert, J. Griffioen, and S. Wen. Lightweight network support for scalable end-to-end services. In *ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[7] I. Castineyra, N. Chiappa, and M. Steenstrup. The Nimrod routing architecture, August 1996. RFC 1992.

[8] D. Clark, R. Braden, A. Falk, and V. Pingali. FARA: Reorganizing the addressing architecture. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Karlsruhe, Germany, Aug. 2003.

[9] D. Clark, K. Sollins, J. Wroclawski, and T. Faber. Addressing reality: An architectural response to demands on the evolving Internet. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Karlsruhe, Germany, Aug. 2003.

[10] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *ACM SIGCOMM*, Philadelphia, PA, August 1990.

[11] L. Daigle, D. van Gulik, R. Iannella, and P. Faltstrom. URN namespace definition mechanisms, June 1999. RFC 2611.

[12] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6), Dec. 1998. RFC 2460.

[13] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. PeerNet: Pushing peer-to-peer down the stack. In *2nd Intl. Workshop on Peer-to-Peer Systems*, Berkeley, CA, March 2003.

[14] B. Ford. Unmanaged Internet Protocol: taming the edge network management crisis. In *2nd ACM Hotnets Workshop*, Cambridge, MA, Nov. 2003.

[15] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer (P2P) communication across middleboxes, October 2003. Internet draft `draft-ford-midcom-p2p-01.txt` (Work in progress).

[16] P. Francis. A near-term architecture for deploying PIP. *IEEE Network*, 7(6):30–27, 1993.

[17] P. Francis. *Addressing in Internetwork Protocols*. PhD thesis, University College London, UK, 1994.

[18] P. Francis and R. Gummadi. IPNL: A NAT-extended Internet architecture. In *ACM SIGCOMM*, San Diego, CA, Aug. 2001.

[19] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A framework for IP based virtual private networks, Feb. 2000. RFC 2764.

[20] M. Gritter and D. R. Cheriton. TRIAD: A new next-generation Internet architecture. `http://www-dsg.stanford.edu/triad/`, July 2000.

[21] A. Gupta, B. Liskov, and R. Rodrigues. Efficient routing for peer-to-peer overlays. In *1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, CA, March 2004.

[22] I. Gupta, K. Birman, P. Linka, A. Demers, and R. van Renesse. Building an efficient and stable P2P DHT through increased memory and background overhead. In *2nd Intl. Workshop on Peer-to-Peer Systems*, Berkeley, CA, Feb. 2003.

[23] International DOI Foundation. `http://www.doi.org/`.

[24] J. Kubiatowicz et al. Oceanstore: An architecture for global-scale persistent storage. In *9th ASPLOS*, Cambridge, MA, November 2000.

[25] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[26] E. Lear and R. Droms. What's in a name: Thoughts from the NSRG, September 2003. draft-irtf-nsrg-report-10, IETF draft (Work in Progress).

[27] C. Lynn. Endpoint Identifier Destination Option. Internet Draft, IETF, Nov. 1995. (expired).

[28] D. Mazières, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. In *17th ACM Symposium on Operating Systems Principles*, pages 124–139, Kiawah Island, SC, Dec. 1999.

[29] A. Mislove and P. Druschel. Providing administrative control and autonomy in peer-to-peer overlays. In *3rd Intl. Workshop on Peer-to-Peer Systems*, San Diego, CA, February 2004.

[30] P. Mockapetris. Domain Names – Implementation and Specification, November 1987. RFC 1035.

[31] K. Moore. Things that NATs break. http://www.cs.utk.edu/ moore/opinions/what-nats-break.html, as of June 2004.

[32] R. Moskowitz and P. Nikander. Host identity protocol architecture, September 2003. draft-moskowitz-hip-arch-05, IETF draft (Work in Progress).

[33] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host identity protocol, October 2003. draft-moskowitz-hip-08, IETF draft (Work in Progress).

[34] A. Myles, D. Johnson, and C. Perkins. A mobile host protocol supporting route optimization and authentication. *IEEE Journal on Selected Areas in Communications*, 13(5), June 1995.

[35] P. Nikander, J. Ylitalo, and J. Wall. Integrating security, mobility, and multi-homing in a HIP way. In *Network and Distributed Systems Security Symposium (NDSS '03)*, pages 87–99, San Diego, CA, February 2003.

[36] M. O'Donnell. Open network handles implemented in DNS, Sep. 2002. Internet Draft, draft-odonnell-onhs-imp-dns-00.txt.

[37] M. O'Donnell. A proposal to separate Internet handles from names. `http://people.cs.uchicago.edu/~odonnell/Citizen/Network_Identifiers/`, February 2003. submitted for publication.

[38] S. W. O'Malley and L. L. Peterson. A dynamic network architecture. *ACM Transactions on Computer Systems*, 10(2):110–143, May 1992.

[39] V. Ramasubramanian and E. G. Sirer. Beehive: $O(1)$ lookup performance for power-law query distributions in peer-to-peer overlays. In *1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, CA, March 2004.

[40] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM*, pages 161–172, San Diego, CA, August 2001.

[41] T. Roscoe, S. Hand, R. Isaacs, R. Mortier, and P. Jardetzky. Predicate routing: Enabling controlled networking. In *1st ACM Hotnets Workshop*, Princeton, NJ, Oct. 2002.

[42] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[43] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[44] J. Saltzer. On the naming and binding of network destinations. In P. Ravasio et al., editor, *Local Computer Networks*, pages 311–317. North-Holland Publishing Company, Amsterdam, 1982. Reprinted as RFC 1498, Aug 1993.

[45] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. D. Rockwell, and C. Partridge. Smart packets: applying active networks to network management. *ACM Transactions on Computer Systems*, 18(1):67–88, Feb. 2000.

[46] J. F. Shoch. Inter-network naming, addressing, and routing. In *17th IEEE Computer Society Conference (COMPCON '78)*, pages 72–79, Washington, DC, September 1978.

[47] A. C. Snoeren. *A Session-Based Architecture for Internet Mobility*. PhD thesis, Massachusetts Institute of Technology, December 2002.

[48] A. C. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Proc. ACM MOBICOM*, pages 155–166, 2000.

[49] A. C. Snoeren, H. Balakrishnan, and M. F. Kaashoek. Reconsidering Internet mobility. In *8th ACM Workshop on Hot Topics in Operating Systems*, Elmau, Germany, May 2001.

[50] K. Sollins. Architectural principles of uniform resource name resolution, January 1998. RFC 2276.

[51] K. Sollins and L. Masinter. Functional requirements for Uniform Resource Names, December 1994. RFC 1737.

[52] P. Srisuresh and K. Egevang. Traditional IP network address translator (Traditional NAT), January 2001. RFC 3022.

[53] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002.

[54] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking*, 11(1):17–32, Feb. 2003.

[55] D. L. Tennenhouse, J. M. Smith, D. Sincoskie, D. J. Wetherall, and G. J. Minden. A Survey of Active Network Research. *IEEE Communications Magazine*, 35(1):80–86, 1997.

[56] C. Tschudin and R. Gold. Network Pointers. In *1st ACM Hotnets Workshop*, Princeton, NJ, October 2002.

[57] R. van Renesse and L. Zhou. P6P: A peer-to-peer approach to Internet infrastructure. In *3rd Intl. Workshop on Peer-to-Peer Systems*, San Diego, CA, Mar. 2004.

[58] M. van Steen, F. J. Hauck, P. Homburg, and A. S. Tanenbaum. Locating objects in wide-area systems. *IEEE Communications Magazine*, 36(1):104–109, January 1998.

[59] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the Web from DNS. In *1st USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04)*, San Francisco, CA, March 2004.

[60] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. Technical Report TR/954, MIT CSAIL, June 2004.

[61] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, January 2004.

[62] L. Zhou, R. van Renesse, and M. Marsh. Implementing IPv6 as a peer-to-peer overlay network. In *Workshop on Reliable Peer-to-Peer Distributed Systems, 21st IEEE Symposium on Reliable Distributed Systems (SRDS '02)*, Suita, Japan, Oct. 2002.