A Layered Naming Architecture for the Internet

Hari Balakrishnan,* Karthik Lakshminarayanan,† Sylvia Ratnasamy,‡
Scott Shenker,†§ Ion Stoica,† Michael Walfish*

April 2004

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 and (2) facilitate mobility and provide an elegant way to integrate middleboxes 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.

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, 6, 7, 9, 37, 41, 48, 49]). 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 the relatively incremental 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 are not only unnecessary, they are irrelevant; for some of these issues, naming holds the key.

Naming in the current Internet is quite primitive, as there are only two global namespaces, DNS names and IP addresses, both of which are tied to a pre-existing structure (domains and network topology, respectively). The paucity and rigidity of these namespaces are responsible for a variety of architectural ills. For instance, the Internet is now widely used to access services and data, yet the Internet's host-centric naming treats data and services as second-class network citizens. Also, users and system administrators often resort to architecturally suspect middleboxes—such as NATs/NAPTs, firewalls and transparent caches—because they cannot get similar functionality within the architecture. As we argue later, the solution to middleboxes lies in more flexible naming.

To remedy these and other architectural problems, in this paper we propose a new layered naming system that has four layers: DNS names and other user-level descriptors, service identifiers (SIDs), endpoint identifiers (EIDs), and IP addresses or other forwarding directives. Our focus is on synthesis, not innovation, and so there is little new in what we say. This naming hierarchy is nothing more than a particular realization of Saltzer's taxonomy of network elements [40], in which he identified users/services (our SIDs), hosts (our EIDs), network attachment points (IP addresses), and paths.[2] Moreover, the architecture we build around these namespaces is shamelessly scavenged from the literature. From the Host Identification Protocol (HIP) proposal [30, 31] we borrow the idea of decoupling the transport and networking layers. From the Internet Indirection Infrastructure (i3) work [46] we borrow the idea of host-directed indirection. From the paper on Semantic-Free Referencing (SFR) [52] we borrow the idea that the service identifiers and endpoint identifier namespaces be flat, and from HIP

*MIT, {hari,mwalfish}@csail.mit.edu
†UC Berkeley, {karthik, istoica}@cs.berkeley.edu
‡Intel Research, Berkeley, sylvia@intel-research.net
§ICSI, shenker@icsi.berkeley.edu

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

[2] Since we don't consider aspects of the architecture that require router involvement, we don't address the issue of naming paths.

again we borrow the idea that these flat identifiers should be used for cryptographic authentication.

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, 21, 36, 38, 47] for background on DHTs), and so we borrow from that literature as well. In addition to these antecedents, there are several other proposals, such as TRIAD [18] and IPNL [17], that share our goals and some of our mechanisms.

Thus, this work is a pastiche of borrowed elements; our contribution is certainly not their invention but is instead their synthesis into a coherent architecture. We present our three guiding design principles in Section 2 and outline the resulting architecture and its properties in Section 3. A key aspect of the design is the use of flat names, and we discuss the issues associated with them in Section 4. We cover 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 derive our architecture from three design principles. In what follows, we use the terms *data* and *services* to refer, respectively, to files and processes that can be remotely retrieved or invoked by a client.

2.1 Names and Protocols

We begin with a design principle that addresses the role of naming 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 forces an application to resolve service and data names down to an IP address, thereby binding the application request to a particular location (as expressed by an IP address). This resolution violates Principle #1 twice over: it binds data and services to a particular end-host and, even worse, it binds them to the network location of that end-host. As we argue below, rectifying this double violation requires the introduction of two (and only two) new naming layers.

Principle #1 requires that applications be able to refer to data and services with persistent names that aren't tied to the endpoint hosting the data or service. We therefore claim that a class of names called service identifiers (SIDs) should exist that give applications exactly this ability. Although SIDs can refer to services or data, for convenience we use only the term service.

Similarly, transport protocols exchange data between two endpoints, and their network locations are irrelevant to the basic semantics of transport. Only at the IP layer is the IP address naturally part of the protocol semantics, which is exactly network-address to network-address delivery. 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, so the connection cannot gracefully accommodate address changes.[3] Principle #1 suggests that transport protocols should be able to refer to endpoints in a manner independent of their IP address.[4] We thus propose a class of names, endpoint identifiers (EIDs),[5] that identify hosts (or, more generally, network entities) without reference to their network location.

These two new naming layers which have been induced by Principle #1 require two additional layers of name resolution: from SIDs to EIDs, and from EIDs to IP addresses. The resolution of SIDs into EIDs and of EIDs into IP addresses should occur no sooner than required, so that the consequent bindings are accurate and appropriate. If resolution occurs prematurely, then transport cannot adjust to location changes, and applications cannot adapt to service and data migrations.

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 referent.

> 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: scalable routing requires that IP addresses reflect network topology, and DNS names, though more flexible, nonetheless reflect administrative structure. This arrangement works well for DNS's original purpose, naming hosts, because most machines with a given domain name are owned by, controlled by, or otherwise related

[3] One can introduce a session layer that manages different transport connections and hides address changes from the application [22, 43].

[4] As observed in [10], this endpoint, instead of one physical host, could be a distributed network *entity*.

[5] Of course, the idea of endpoint identifiers and their desirability has long been accepted Internet lore: see, for example, [28].

to the entity that oversees the domain's records. Consequently, changes in hosts' addresses can easily be reflected in DNS records.

However, the Internet has been increasingly used to access data and services, rather than particular hosts, and in the absence of any other global naming scheme, DNS names have been used for services (*e.g.*, mail servers) or data (*e.g.*, URLs). As has long been noted in the URN literature [23, 44, 45], DNS-based names for data are inherently ephemeral. Data does not naturally conform to domain boundaries and when data is replicated on, or moved to, hosts outside the originating domain, the original DNS record will typically not reflect its new location.[6] The same can be said of services, though one might argue that services are less peripatetic than data.

Thus, no namespace currently exists that can persistently name data and services. Some in the URN community have proposed introducing a new namespace and resolution mechanism [11] for each *genre* (*e.g.*, ISBN numbers would have a 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 [52], and Open Network Handles [34], 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 therefore does not impose any restrictions on referenced elements, thereby ensuring universal compliance with Principle #2. In this paper we adopt this second approach, using a flat namespace for SIDs and EIDs.

2.3 Resolution and Indirection

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, a SID's location would be an EID—usually an (EID, transport, port) triple—and an EID's location would be an IP address.[7] However, we think this typical definition is too restrictive and instead adopt the following more general design principle.

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

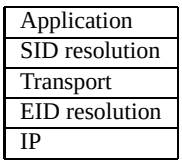
In any logical network connection, the initiator (at any level, *e.g.*, a human requesting a Web page or an endpoint initiating a transport connection) intends to connect with a destination entity. In our case, for example, applications connect with SIDs. However, the destination entity may not want to handle the connection directly, preferring instead to direct the connection to a chosen *delegate*. This kind of indirection does not alter essential trust relationships (if you trust an entity, you trust its delegates) nor does it interfere with established protocol semantics, as we will see when we describe the details of such delegation in Section 3. One can view delegation as a simple, special case of the kind of distributed network element envisioned in [10]; that is, the destination and its delegate are part of the same logical element even if they are physically distinct.

While Principle #3 might seem esoteric at first, it is crucial to the overall architecture. This form of indirection, when an entity delegates a connection to another entity, provides much flexibility. As we describe next, indirection allows the architecture to gracefully incorporate *intermediaries*, which we define as cleaner and more flexible versions of middleboxes. Delegated indirection also allows a small degree of protection against denial-of-service attacks and a primitive form of route selection.

3 Architecture

3.1 Overview

Principle #1 led us to claim that applications should bind to SIDs and transport protocols should bind to EIDs. Thus, there must be a layer between applications and transport that translates between SIDs and EIDs. Similarly, there must be a layer between transport and IP that translates between EIDs and IP addresses. We will call these *resolution* layers (though they do more than simply resolve identifiers).[8] This results in the following layered architecture:



We start by describing how this architecture works in the generic case. Later we will complicate the story somewhat but, because of space limitations, this description will be quite superficial.

[6] Of course, the problems of using hostname/pathname URLs for naming data go far deeper than this; see [44, 45, 52] and citations therein for a more complete discussion.

[7] Resolving a SID can also return metadata (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 a SID. For simplicity, our examples here illustrate only SIDs abstracting services, not data.

[8] Here we focus on how these layers fit into an overall architecture; in the next section we discuss how the actual resolution might be done.

Consider an application a on a given host that wishes to access a service represented by the SID s. It hands s to the SID resolution layer, which contacts the resolution infrastructure (one realization of which we describe in Section 4) and is handed back one or more (EID,transport,port) triples, where each triple represents an instance of the desired service. The SID resolution layer then invokes the desired transport protocol, using the EIDs of a and the resolution of s as the source and destination in the transport protocol. If the EID becomes unreachable, the SID resolution layer can attempt to contact another EID triple if more than one was returned. If all fail, the SID resolution layer re-resolves the SID to check for new triples.

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 (the multiplicity reflecting multihomed or cloned hosts).[9] The EID resolution layer uses one of these IP addresses in calling IP (the source IP address is the IP address of the sending host). If the host is unreachable, another IP address can be used if more than one is returned.[10] If none of the previously returned IP addresses works, the EID resolution layer re-resolves the EID in case the corresponding IP addresses have changed.

The architecture's most obvious property is that it provides *automatic and seamless rebinding* for both services and hosts. (Other proposals are explicitly designed for this rebinding too, including HIP, TRIAD [18], UIP [14], FARA [9] and IPNL [17]; with the exception of TRIAD and FARA, which we discuss in detail in Section 5, none of the proposals incorporates a notion of rebinding for services.) If a service migrates from one host to another, the SID resolution layer will re-resolve the SID. Similarly, if an endpoint changes its IP address, then the EID resolution layer will re-resolve the EID to find the new IP address. This re-resolution could conceivably occur on each packet but more likely will be invoked only when a failure is detected. As explained in [30, 31, 32], rebinding at the EID layer enables continuous operation in the presence of mobile or renumbered hosts and provides smooth failover for multihomed hosts; we direct the reader to these references for more details.

The other important property, *delegated binding*, isn't explicit in the generic case above. We now explain how this property enables intermediaries and why they might be useful.

[9] Our architecture also allows for an EID to resolve to another EID thus requiring multiple rounds of resolution before the EID is resolved down to an IP address. Likewise, the resolution of a SID to another SID is also possible.

[10] 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.

3.2 Delegated Bindings and Intermediaries

As above, we start with a basic case and then discuss complications. Delegation at the SID layer would be when a service s, running on a host h, rather than listing the EID of h in the resolution service, instead lists the EID of some other endpoint o. s would have to establish state (through some protocol that is outside the scope of our discussion) at o so that o knows how to handle packets destined for s. o could be, for example, an application-level gateway. Hosts trying to contact s will have their transport connections terminated at host o. This gateway can inspect the payload, and then decide to forward it or not depending on its contents. Note that this use requires a *logical SID header* that can carry the destination SID (and a source SID—the purpose of which we explain below); this field can be used by the receiving end-host, o, to demultiplex the request and associate it with the forwarding state. We will call hosts such as o *application-level* intermediaries.

Delegation at the EID layer works similarly; a host with EID e can insert the IP address 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, so that when packets arrive at the delegated host they can be forwarded. As with SIDs, this scheme requires that packets carry a *logical EID header* including the destination EID. The intermediary uses this information to make a forwarding decision. This type of intermediary is *network-level*; it forwards packets but does not inspect the layers above the EID.

The mechanism described above is sufficient to support standard network-level intermediaries (NATs/NAPTs, VPNs, and firewalls) cleanly and coherently. For example, depending on the scenario and the security assumptions, such an 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 prespecified EIDs. Other VPN scenarios exist; we do not cover them here.

While one obviously does not need our architecture to implement the types of intermediaries we have described (indeed, these intermediaries exist today), our architecture offers three key benefits relative to the status quo. First, the intermediaries do not violate protocol semantics; they only terminate transport connections or inspect packet payloads 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, some 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 natural network path of the connection. Under this design, for 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 host to authenticate each other), so that the firewall would know where to forward packets.

One could also use this form of indirection to provide a (small) degree of protection against denial-of-service (DoS) attacks. A server can shield itself from attackers by placing a forwarding intermediary between itself and untrusted clients. This allows the server to hide its IP address from these untrusted clients and to control its incoming traffic by installing traffic filters at the forwarding intermediary. This approach is identical in spirit to the overlay DoS protection schemes proposed in SOS [25], Mayday [1], and by Lakshminarayanan *et al.* [26]; our point here is merely to illustrate how their basic techniques can be implemented within our architecture. Of course, if attackers discover a server's IP address, they can launch a direct DoS attack, which our architecture can't prevent since it does not change current routers. However, having all incoming packets directed through intermediaries would make router-level packet filtering much easier.

Application-level multicast is another example of a service that, similar to DoS protection, requires setting up an infrastructure of forwarding intermediaries, though in this case between a source and multiple receivers. However, for multicast, when a destination EID resolves to multiple IP addresses, the intermediary forwards the packet to all of them; this requires a multicast option associated with the resolution of EIDs.[11]

As described so far, our delegated binding mechanism allows for simple forms of indirect forwarding. With one modification, this mechanism can support source routing at the SID or EID layers.[12] Such host control over routing is useful for hosts who wish to specify intermediaries on the return path, *e.g.*, a handheld PDA (or its network provider) that needs its Web replies sent first to a machine that transforms HTML from Web servers into a pared-down version suitable for the PDA's browser.

Our modification, as in i3, is to use stacks of SID (or EID) identifiers to encode a path at the SID (or EID) level, and then to use these stacks of identifiers as the source and destination fields in the logical packet header. A source host can thus control its outbound path by placing a stack of identifiers in the SID (or EID) destination field. Similarly, it can control its inbound path (*i.e.*, the path back from the remote endpoint) by placing an identifier stack in the SID (or EID) source field sent to the remote service (or endpoint); the destination uses the source field as the destination field in the return packet thereby allowing the source to control its return path. Moreover, intermediaries can push SIDs (EIDs) onto the destination stack; doing so has the semantics of permitting an intermediary to say "send this here before it goes there". We note that, in analogy with transport-level ports today, a host application may use an *ephemeral* SID for its own identifier.

This section described, superficially, how our architecture can be used to provide simple and dynamic rebinding, indirect forwarding through intermediaries, and end-host-controlled source routing over an infrastructure of forwarding intermediaries. We leave many fine points unresolved: examples include the details of the signaling protocols required to set up state at intermediaries, details of the software and API to interpret and create stacks of identifiers, and the implications of this layering for host software. One question that particularly interests us is how congestion control fits in this layering scheme; since congestion control is typically associated with a *path*, and paths change when IP addresses change, will the seamless rebinding make congestion control much harder to manage? We have not addressed security in this paper, but it is addressed in HIP [30, 31], and we can inherit many of the mechanisms therein.

Putting aside these questions, we now turn 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 [34], and SFR [52]. Here we discuss only two particularly troubling aspects of flat names: they are hard to resolve and they aren't human-readable.

4.1 Resolution

DNS achieves scalability through hierarchy. It has been an assumption, often implicit, that such structure was necessary for scalable resolution. As a result, most

[11] Note that if IP Multicast were ubiquitously deployed, our architecture would trivially support it; an EID would merely resolve to the IP Multicast group address.

[12] Note that this level of route control is much coarser than the AS-level control in NIRA [53]. Our mechanism only enables the equivalent of loose source routing through a set of intermediaries.

architectural 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 points.

DHTs arose in the context of peer-to-peer (P2P) systems, but an unmanaged and untrusted P2P system would not be suitable 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. Also, DNS's hierarchical delegation naturally ensures each name is unique and controlled by the relevant authority. With flat names, these goals are harder to achieve but certainly not impossible. Several mechanisms exist for global uniqueness (see [29, 39] for example) and also for achieving administrative locality [24, 29, 52]. 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.* [51, 52]).

DHTs' typical resolution time—$O(\log n)$ for an n node system—would be unacceptable for most name resolutions particularly 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, [19, 20]. Second, a DHT-based resolution infrastructure can be designed using local proxies [52], local replication [24] or two-layered resolvers [29] that enable entries written within the local network to be accessed with local hops; these schemes also provide fate-sharing in that if an organization is disconnected from the rest of the Internet it can still access the entries written locally. See [29, 52] 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 should users 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 pay for lookups and for storing, likely a flat fee for a reasonable number of accesses. The various RSPs would have mutual "peering" relationships whereby they exchange updates, much as the tier-1 ISPs all interconnect today. Since each RSP would be judged by how well they served their clients, they would have incentives to process requests honestly. Moreover, one could use proposals like [8] to achieve a greater level of trust.

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 only make 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 with 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 can be offered by many 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 bitstring gives no such assurance. Here, bitstrings could be accompanied by metadata which includes cryptographic statements like "Authority A says that this SID points to the newspaper New York Times." Again, these authorities would not be part of the infrastructure but part of a competitive market of SID authenticators.

In addition, as in HIP, the SIDs and EIDs could be cryptographically strong; *e.g.*, they could derived by hashing a public key, making authentication of services and endpoints more convenient.

5 Related Work

As noted in the Introduction, our work borrows most 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, in fact, that here we can only present the most superficial of overviews.

Saltzer [40] was one among many [16, 28, 42] who made fine distinctions between network elements; the most common, and least practiced, of these distinctions is between a host's identifier and its address (see [27] for a comprehensive discussion of this topic). This distinction has been embedded in two more recent proposals: UIP [14] and Peernet [13]. 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.

Creating location-independent, persistent names, and an accompanying infrastructure for resolution, has long been the goal of the URN literature [23, 44, 45]. In addition, the Open Network Handles work [33, 34] argues for flat, unfriendly domain names for Web *sites*. The Globe project [4, 51] envisions a single infrastructure for mapping from (possibly human-unfriendly) persistent object identifiers to current locations.

There are an increasing number of proposals for radically new network architectures. These include earlier proposals like PIP [15], IPv6 [12], Dynamic Networks [35], Active Networks [48], Nimrod [7], and more recent proposals like Smart Packets [41], Network Pointers [49], Role-Based Network Architecture [5], and Ephemeral State Processing [6]. Each of these proposals shares at least some of our goals here, but they are all *ab initio* designs that would (in their full glory) require significant modifications to all network elements. More importantly, these proposals have substantially different goals than we do here.

Four other proposals deserve special mention. The authors of TRIAD [18] 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 also shares many of our motivations. IPNL is intended to make renumbering easier, create separate end-host identifiers, and leave 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 [9] presents a novel organization of network architecture concepts. While many of its goals are similar to ours, one directly conflicts: FARA deliberately avoids creating new global namespaces whereas we strongly advocate creating one new flat global namespace. A detailed comparison of our approaches is outside the scope of this paper but merits further consideration.

P6P [50, 54] 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).

6 Discussion

This paper proclaims three 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-level or network-level intermediaries. Mobility would be seamless, and there would be modest, but by no means complete, protection against denial-of-service attacks and host route control.

While we believe in our proposal, the details are less important than two 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 these 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.

Of course, our proposal also faces serious hurdles. In-

corporating these 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. While this is all good news, 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.

References

[1] D. G. Andersen. Mayday: Distributed filtering for Internet Services. In *4rd USENIX Symposium on Internet Technologies and Systems (USITS '03)*, Seattle, WA, 2003.

[2] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet denial-of-service with capabilities. In *2nd ACM Workshop on Hot Topics in Networks*, 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 Workshop on Hot Topics in Networks*, 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, Aug 1996. RFC 1992.

[8] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *5th USENIX Symposium on Operating Systems Design and Implementation (OSDI '02)*, Boston, MA, 2002.

[9] 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.

[10] 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.

[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 International Workshop on Peer-to-Peer Systems*, Berkeley, CA, Mar. 2003.

[14] B. Ford. Unmanaged Internet protocol: taming the edge network management crisis. In *2nd ACM Workshop on Hot Topics in Networks*, Cambridge, MA, Nov. 2003.

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

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

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

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

[19] 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, 2004.

[20] 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 International Workshop on Peer-to-Peer Systems*, Berkeley, CA, Feb. 2003.

[21] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao. Distributed object location in a dynamic network. In *14th ACM Symposium on Parallel Algorithms and Architectures*, Aug. 2002.

[22] C. Huitema. Multi-homed TCP. Internet Draft, May 1995. (expired).

[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, Nov. 2000.

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

[26] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica. Taming IP packet flooding attacks. In *2nd ACM Workshop on Hot Topics in Networks*, Cambridge, MA, Nov. 2003.

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

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

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

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

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

[32] 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, Feb 2003.

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

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

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

[36] 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.

[37] T. Roscoe, S. Hand, R. Isaacs, R. Mortier, and P. Jardetzky. Predicate routing: Enabling controlled networking. In *1st ACM Workshop on Hot Topics in Networks*, Princeton, NJ, Oct. 2002.

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

[39] 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, Oct. 2001.

[40] 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, August 1993.

[41] 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.

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

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

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

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

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

[47] 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.

[48] 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.

[49] C. Tschudin and R. Gold. Network Pointers. In *1st ACM Workshop on Hot Topics in Networks*, Princeton, NJ, Oct. 2002.

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

[51] 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, Jan. 1998.

[52] 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, 2004.

[53] X. Yang. NIRA: A new Internet routing architecture. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Karlsruhe, Germany, Aug. 2003.

[54] 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.