

An End-Middle-End Approach to Connection Establishment *

Saikat Guha
Cornell University, Ithaca
saikat@cs.cornell.edu

Paul Francis
Cornell University, Ithaca
francis@cs.cornell.edu

ABSTRACT

We argue that the current model for flow establishment in the Internet: DNS Names, IP addresses, and transport ports, is inadequate due to problems that go beyond the small IPv4 address space and resulting NAT boxes. Even where global addresses exist, firewalls cannot glean enough information about a flow from packet headers, and so often err, typically by being over-conservative: disallowing flows that might otherwise be allowed. This paper presents a novel architecture, protocol design, and implementation, for flow establishment in the Internet. The architecture, called NUTSS, takes into account the combined policies of endpoints and network providers. While NUTSS borrows liberally from other proposals (URI-like naming, signaling to manage ephemeral IPv4 or IPv6 data flows), NUTSS is unique in that it couples overlay signaling with data-path signaling. NUTSS requires no changes to existing network protocols, and combined with recent NAT traversal techniques, works with IPv4 and existing NAT/firewalls. This paper describes NUTSS and shows how it satisfies a wide range of "end-middle-end" network requirements, including access control, middlebox steering, multi-homing, mobility, and protocol negotiation.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Protocol architecture

General Terms

Design, Security

Keywords

NUTSS, End-Middle-End, Off-path, On-Path, Signaling

1. INTRODUCTION

The Internet was designed to provide a small but critical set of transport services:

1. *User-friendly naming* of all Internet hosts (through DNS).
2. *Network-level identification* of all Internet hosts (through the IP address) and best-effort delivery of datagrams to identified hosts.
3. *Identification of the application* on the host that should receive a given packet (through the port number).

Implicit among these services was the idea that applications would individually take care of access control. The Internet[1] would deliver transmitted packets to the target application, and it was up to the application to decide whether to accept or reject the packet. A further implication of this approach is that there is no danger in asking an application to process an incoming packet. The application is assumed to be competent to look inside the packet and decide whether or not to accept it. Industry recognized in the early 90's that this approach was wrong: DoS attacks can overwhelm an application, and because of either bugs or just poor design, applications are incapable of securing themselves with certainty. The industry answer to this problem was the firewall, which effectively enunciated a fourth critical requirement for Internet transport service:

4. *Blocking of unwanted packets* before they reach the target application (through packet filters in firewalls).

Of course it is well-known that the Internet today is ill-equipped to satisfy these four core requirements. The IP address shortage prevents all hosts from being identifiable in the network. Port numbers do not adequately identify applications anywhere outside of the OS that created the socket. As a result, firewalls cannot be certain what application is behind a given port number, use costly deep packet inspection, and often err on the side of caution (preventing flows that might otherwise be acceptable).

The firewall compromised the E2E nature of the Internet architecture by placing a point of control outside of the end host. While this development was widely viewed as negative [23], we and others [58] believe that it is not only inevitable, but necessary and largely positive. A primary reason for this is the fact that there may be multiple legitimate stakeholders in a given packet flow—the end user, the corporate IT department, or the ISP—each with their own policies. The E2E nature of the Internet does not easily accommodate these policies. Another reason, however, is that sometimes it is simply economically expedient to deploy a function in the middle, even if it might ultimately be better done at the ends. Today there are often good reasons to want to route packets through middleboxes other than firewalls, for instance virus scanners, web caches, traffic shapers, performance enhancing proxies, protocol translators (IPv4 to IPv6) and so on. These middleboxes sometimes interrupt

*This work is funded in part by the Cisco CRP program

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCOMM'07, August 27–31, 2007, Kyoto, Japan.
Copyright 2007 ACM 978-1-59593-713-1/07/0008 ...$5.00.

[1] By "Internet", we mean the naming and transport services provided by IP addresses, ports, and DNS for today's "fixed" Internet (including wireless access to the wired Internet). Sensor networks and MANETs that perform their own naming and routing separate from the Internet are not included in this definition.

E2E semantics. The legitimate rise of middleboxes leads to another requirement:

5. *Explicit negotiation of middlebox usage* between the endpoints and networks in the middle, including the ability to steer packets through middleboxes not otherwise on the data-path between source and destination.

We refer to this set of five requirements as the *End-Middle-End* (EME) naming and addressing problem. Together they constitute what we consider to be the absolute minimum set of requirements that the modern Internet should satisfy. Put another way, a new standard sockets interface, and the networking infrastructure that supports it, should at a minimum satisfy the above requirements.

This paper presents an architecture and protocol, called NUTSS, that satisfies these core EME naming and addressing requirements. Specifically, NUTSS names endpoint applications with user-friendly names, and uses signaling protocols to dynamically and securely do late binding of named endpoints to ephemeral 5-tuple (addresses, ports, and protocol) transport flows. Unlike previous architectures [20, 58], transport flows in NUTSS are ephemeral and renegotiated using *both off-path* (an overlay off of the data-path) *and on-path* (on the IP data path) signaling protocols when required. This is in stark contrast to SIP [44] (off-path only) and RSVP [11] (on-path only), neither of which solves these core problems.

A simplified NUTSS connection establishment is described as follows. An initiating host transmits a signaling message containing source and destination name, and the name of an application. Using these names as the basis for routing, this message traverses *off-path* policy-aware boxes near both ends, where authentication is done and decisions are made to allow or disallow the connection. Once allowed, ephemeral addresses, ports, and firewall-traversal tokens are conveyed to both ends. Using the learned address as the basis for routing, this information is then used by an *on-path* signaling protocol to establish a data connection through firewalls. The firewall uses the secure tokens as capabilities to allow or disallow flows. It is these tokens that couple the off-path and on-path signaling phases. If the connection breaks, for instance because of mobility or firewall crashing, NUTSS can retry the on-path signaling using the addresses and tokens previously obtained, or failing that, fall back on off-path signaling using the names to re-establish the data flow.

NUTSS does more than satisfy the core EME requirements listed above. By using names as stable unique identifiers, and binding them late to 5-tuple flows as explored in much recent work [35, 49], NUTSS separates identification from network location, thus supporting network mobility and host and site multi-homing. Finally, NUTSS signaling allows endpoints and middleboxes to negotiate the protocol stack used in the data-path. This can be used not only to negotiate transport (UDP, TCP, SCTP, etc.) and security (IPsec, TLS, SSH, etc.), but different network layers as well (IPv6, IPNL [15], TRIAD [20], HIP [35], i3 [49], DoA [58], etc.). The ability to negotiate protocols as well as middleboxes creates a framework for managing multiple network layers created through virtual routers and switches, for instance as proposed for the GENI infrastructure [19]. Indeed, this very flexibility is exploited by NUTSS to provide itself with an incremental deployment path (Section 2.6).

Up to this point, we have asserted that NUTSS satisfies contemporary EME requirements without changes to existing network protocols. Indeed, we can make a stronger assertion: that *any* new network protocol benefits tremendously from a name-based signaling approach like NUTSS. This claim flies in the face of recent self-certifying, identity-based architectures [35, 49, 57, 58, 28], which suggest not only that flat identities can serve as the basis of network or content identities, but in some cases go so far as to suggest that there is no need for a single global user-friendly name space [57, 28]. Rather, a wide range of ad hoc mechanisms, such as search engines and HTML links, can be used to "discover" identifiers.

Our difficulty with these architectures derives mainly from the fourth EME requirement—that unwanted packets must be blocked before they reach the application, ideally in the network. This requires, among other things, that access control policy (e.g. ACLs) be configured in middleboxes. Today firewall vendors strive to build devices that may be configured using user-friendly names ("BitTorrent", or "ftp"), and that can filter on aggregates such as DNS zones or IP prefixes [8]. Flat identifiers are neither user-friendly nor aggregatable, and therefore are not well-suited to serve as the basis for ACL configuration. This is an issue that the proponents of identity-based approaches have not addressed, in spite of the fact that they recognize middleboxes as being no longer harmful, and incorporate mechanisms to steer packets through them [49, 58, 28]. There must be a globally-understood user-friendly namespace that identifies endpoints (applications, services, users, etc.), as well as a way to bind those names to the addresses, ports, and identifiers of data packets (collectively referred to here as "addressing material").

A key issue, then, is how to bind names to the addressing material. Both TRIAD [20] and IPNL [15], which use DNS names as user-friendly host identifiers, bind those names to network addresses by carrying both names and addresses in data packets. These schemes literally treat names as addresses in the sense that network routers run routing algorithms on the names themselves, and bind these to numerical addresses primarily as an optimization for the address lookup function. Both name-routed and address-routed packets follow the data path (in other words, are routed on-path).

While neither TRIAD nor IPNL sought to solve the middlebox problem, one can imagine extending them to do so, for instance by extending their host names with user, application, and service names, and by authenticating those extended names. Even so, we find on-path approaches to be less attractive than off-path approaches that use overlays to do name-based routing. On-path approaches are both overly constraining and overly intrusive. They are constraining in that they force the name-based access control policy decision to be made on-path. They are intrusive in that they force all routers to participate in a name-based routing algorithm that, in the case of TRIAD, may scale poorly, or in the case of IPNL, requires a DNS lookup at packet forwarding time.

An overlay approach to name-based routing, by contrast, allows the access control policy decision to be made anywhere in the Internet. In particular, it allows access control to be widely replicated and therefore more resilient to flash crowds or DoS attacks [40]. DNS, of course, is a name-based routing overlay, and certainly much of its success may be attributed to the fact that it is decoupled from on-path routing and is therefore easier to deploy. The problem with DNS in the EME context is that it is not at all designed to do access control. DNS is not aware of who is making a DNS query, and is typically not aware of the purpose of the query (i.e. which application the query is for). Indeed, current use of dynamic DNS [55] reveals private location information about mobile users, making it possible for instance to follow their travel itineraries [22]. Merely confirming the existence of a valid name to an unauthorized user can be considered a breach of privacy defined as contextual integrity [37].

Another widely deployed name-based routing overlay is SIP [44], which is used for media (voice or video) flow establishment. For the purposes of EME requirements, SIP is at least better than DNS

in that it carries the identity of both endpoints and allows them to be authenticated. Furthermore, SIP enables a powerful set of features including mobility, rich naming of users and endpoints, discovery, the ability to negotiate different protocols, independence from underlying transport, and the creation of infrastructure to support it all. Nevertheless, SIP itself is not designed to couple the off-path access control policy decision with on-path access control enforcement. Industry has tried to address this shortcoming in two ways. One is to implement SIP in the firewall itself [9]. This approach does not work in all cases, because the name-routed signaling path may differ from the address-routed data path. For instance, consider a dual-homed site with combined firewall/SIP servers $F1$ and $F2$. The signaling path may traverse $F1$, which authorizes the flow and allows access for the associated addressing material. The subsequent data path, however, may traverse $F2$, which has not authorized the flow.

The other way is to define a protocol that allows the SIP server to coordinate with the firewall [32, 50]. This approach suffers from a similar problem which may be solved in a brute-force fashion by having the SIP server enable a given flow in all possible firewalls that the data flow may traverse. While in the common case (a dual-homed site) this may be reasonable if inefficient, it becomes unworkable in scenarios where there are many firewalls. For instance, a widely replicated distributed firewall addressed as an IP anycast group might have hundreds or thousands of firewalls [14].

The key contribution of this paper is the design of NUTSS, a protocol that satisfies the core EME requirements through the novel combination of dual signaling—the explicit coupling of off-path name-routed signaling with on-path address-routed signaling to establish ephemeral 5-tuple flows. It is this novel coupling that overcomes the disconnect between name-based routing and IP routing that plagues previous approaches. NUTSS works with existing data protocol stacks (IPv4 or IPv6), and includes an incremental deployment path that initially requires no changes to NAT boxes. As with other architectures that separate location from identity, NUTSS facilitates mobility and multi-homing. Besides describing the design of NUTSS, this paper presents a proof-of-concept implementation and deployment of NUTSS and examines whether SIP [44] is appropriate as the off-path signaling protocol for NUTSS.

The remainder of this paper is structured as follows: Section 2 describes the basic NUTSS architecture. Section 3 presents various use cases and minor extensions to the basic architecture. Section 4 reports on our proof-of-concept implementation of NUTSS. Section 5 discusses related work. Section 6 offers concluding thoughts and directions for future work.

2. NUTSS ARCHITECTURE

This section starts with a brief overview of the NUTSS architecture, followed by a detailed description of NUTSS.

2.1 NUTSS Overview

In NUTSS, named *endpoints* may be applications or services, and may be associated with individual users or endhosts. The names are user-friendly, long-term stable, and location-independent. When an endpoint application wishes to establish a data flow with another endpoint, it opens a NUTSS socket using the names only (and not IP addresses) as endpoint identifiers. This triggers an end-to-end name-based signaling exchange that authenticates the endpoints and establishes the state necessary to transmit a 5-tuple (source and destination IP address, source and destination port, and IP protocol) *data flow* end-to-end via a series of middleboxes, including NATs and firewalls. In addition to the 5-tuple parameters and NAT mappings normally required by flows, this state also includes authorization tokens needed to traverse middleboxes that do access control.

There are two components in NUTSS, *P-boxes* and *M-boxes* (for policy-box and middlebox respectively). P-boxes and M-boxes are deployed in the network as well as in endhosts. Networks that enforce policies, such as access control or steering policies, must deploy P-boxes and M-boxes. P-boxes form an overlay over which name-routed signaling messages are carried end-to-end. Data flows (or just *flows* for short) do not traverse P-boxes. Flows do, on the other hand, traverse M-boxes, either because the M-box is deployed on the IP path between endpoints (as with a firewall), or because the signaling has negotiated to steer a flow through an M-box (for instance, an anonymizer). P-boxes make policy decisions about richly-named flows: whether to allow or disallow them, whether to steer them through M-boxes, whether to require encryption, and so on. M-boxes enforce the policy decisions made by an associated P-box.

Signaling messages may traverse P-boxes or M-boxes. They traverse P-boxes, routed by name, when no IP address is known for the destination, or when the security tokens needed to traverse M-boxes have not been obtained. Signaling through P-boxes is referred to as *name-routed* signaling. Otherwise, signaling messages naturally traverse M-boxes, routed by the destination IP address obtained during name-routed signaling (called *address-routed* signaling). Because a name-routed P-box overlay path always exists between endpoints, even for endpoints behind NAT boxes, there is always a way to signal another endpoint to establish a flow (policy permitting).

There is a bidirectional coupling that exists between name-routed and address-routed signaling, which exists by virtue of shared information (keys and addresses) between P-boxes and their associated M-boxes. This coupling is necessary to overcome the unavoidable lack of coordination between name-based overlay routing and IP-based address routing. Specifically, P-boxes convey secure tokens to endpoints during name-routed signaling, which are then carried in address-routed signaling to traverse M-boxes. If an unapproved flow is attempted through an M-box, the M-box *refers* the sending endpoint to a P-box that may authorize the flow.

2.2 Naming and Access Control

Endpoint names in NUTSS are $(user, domain, service)$ 3-tuples. The $user$ is a user-friendly identifier that is not globally unique (e.g. bob). The $domain$ is a globally-unique, user-friendly, hierarchical DNS name (e.g. acme.org). Together the user and domain identify the *principal* that is considered to own the endpoint; the user may be NULL, in which case the domain effectively identifies a machine. The $service$ is a globally-unique, user-friendly identifier for the service provided by the endpoint (e.g. ftpd for an FTP-server). Names are independent of network location.

Access control policy is defined in terms of names. Wildcards are permitted in policy definitions. A wildcard service name $*$ matches all services run by a particular principal (e.g. $(bob, acme.org, *)$), while a wildcard user name matches all principals in that domain. Furthermore, as domains are organized hierarchically, a wildcard prefix in the domain name matches all subdomains below that domain (e.g. $(*, *.cs.acme.org, *)$).

NUTSS relies on existing mechanisms to authenticate endpoint identities. Standard protocols, such as public-key signatures or challenge-response protocols (e.g. DIAMETER [7]), over the name-routed path are used to authenticate principals. Similarly, services can be authenticated if the necessary hardware and software support is present at endpoints. For instance, [46] proposes an architecture that leverages trusted hardware [51] to measure the integrity of

Parameters
E : $(user, domain, service)$ - Endpoint name
A : $address$ - Network address to reach endpoint
P : $port$ - Transport port for data flow
τ : $(token, nexthop)$ - address-routing state
ρ : (E_P, A_P) - Referral to P-Box

Name-routed messages (sent to P-Box)
REGISTER(E, A)
Register a name-route (wildcards OK).
FLOWNEGOTIATE$(E_{src}, E_{dst}, A_{src}, \tau_{1...n})$
Use name-routed signaling to negotiate address-routed path.
P-Boxes add τ_i, and modify A_x to effective address $A_{x'}$

Address-routed messages (sent through M-Box)
$\rho =$ FLOWINIT$(A_{self}, A_{peer'}, P_{self}, \tau_{1...n})$
Use address-routed signaling to initialize data path.
An M-Box may refer to additional P-Boxes to contact
M-Boxes modify P_x to effective port $P_{x'}$
$\rho =$ SEND$(A_{self} : P_{self}, A_{peer'} : P_{peer'}, data)$
Send data packet

Access Control (sent to P-Box)
DISALLOW(E_{dst}, E_{src})
ALLOW(E_{dst}, E_{src})
Add/remove filters for destination (wildcards OK).

Table 1: NUTSS API for establishing flows and controlling access

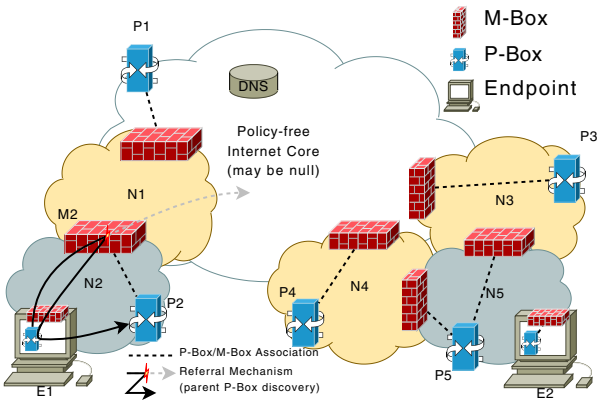


Figure 1: Network topology and referral mechanism. Network $N5$ is multi-homed.

the software-stack. As such, authentication is not further addressed in this paper.

2.3 Name-routed Signaling

We now discuss how NUTSS creates a name-routed path between endpoints. Our goal in creating this path is to incorporate the policy of networks on the data path between the endpoints. As mentioned, this is accomplished through policy-aware P-Boxes that, by design, form a name-routing tree[2] (rooted at the DNS). Endpoints form the leaves of the tree such that the path between two endpoints along the tree passes through P-Boxes designated by networks topologically between the two endpoints.

2.3.1 Network Topology

NUTSS models the Internet topology as policy-aware edge networks connected by a policy-free core (Figure 1). The policy-free core (or just *core* for short) is defined as the set of interconnected networks that do not assert middlebox policies and so do not deploy P-Boxes. This model reflects the current Internet: networks with firewalls today correspond to (policy-aware) edge networks, and networks without firewalls correspond to the (policy-free) core. Edge networks may comprise smaller networks that are separate administrative entities. Each network designates one logical P-Box (potentially multiple physical instances), which may be located either inside or outside that network (e.g. in the figure, network $N2$ designates P-Box $P2$ and $N1$ designated $P1$). A P-Box for a network not connected directly to the core has a *parent P-Box*. The parent P-Box is the P-Box for an adjacent network through which the former connects to the core ($P1$ is $P2$'s parent). A P-Box for a multi-homed network ($P5$) has multiple parents ($P3, P4$).

The network administrator associates M-Boxes with the P-Box for the network. M-Boxes are typically, though not always, deployed at the network boundary (e.g. $M2$). P-Boxes use standard cryptographic mechanisms (shared symmetric keys, or public keys) to pass confidential messages to the M-Box via untrusted third-parties. To facilitate deploying many M-Boxes, a P-Box need not know the addresses of the M-Boxes (except for M-Boxes that must be explicitly addressed e.g. NATs). M-Boxes, on the other hand, are all configured with the name and address of their associated P-Box. The P-Box and M-Box may optionally be co-located in the same physical package.

Endhosts have a resident P-Box and M-Box (Figure 1). NUTSS primitives (Table 1) are initially sent by endpoints to their local in-host P-Box and M-Box, and from there, to other P-Boxes and M-Boxes.

NUTSS assumes the presence of the DNS (or a similar name-resolution service) in the core. For each domain, the DNS contains the addresses of one or more *contact P-Boxes* for that domain. The contact P-Box is the outermost P-Box through which the endpoints in that domain can be reached. For example, in Figure 2, endpoints from acme.org typically register with $P2$; the DNS administrator for acme.org lists $P1$ as the contact P-Box for his domain as $P1$ can reach those endpoints through its child $P2$. Contact P-boxes must be globally addressable.

Note that the core may in fact be null, as long as the core freely allows packets between contact P-boxes. The rationale for exploiting DNS in the core is similar to that of IPNL [15]: it allows NUTSS to scale globally without changing the Internet core and without requiring new name-based routing protocols as do TRIAD [20] and DONA [28].

2.3.2 Discovery

A P-Box discovers it's parent P-Box through the M-Box referral mechanism mentioned earlier. The child P-Box (e.g. $E1$'s in-host P-Box) sends an address-routed message to a public address. The message contains any authorization tokens needed to clear the M-Boxes for the originating network/host (generated by the P-Box itself), but does not contain authorization tokens for the parent network ($N2$); the parent network's M-Box ($M2$) therefore blocks the message and responds with the name and address of the parent P-Box ($P2$). An advantage of using normal address-routed messages for P-Box discovery is that if P-Boxes and M-Boxes are added (even mid-flow), for instance if a site becomes multi-homed, they can be discovered via normal operation of the protocol.

2.3.3 Name-Route Creation

The REGISTER message, sent by endpoints through P-Boxes of networks connecting it to the core, creates a (reverse) route from the DNS to the endpoint through P-Boxes designated by the middle. The process is described as follows: endpoint E with network

[2]In the presence of multi-homed networks, this is a directed acyclic graph

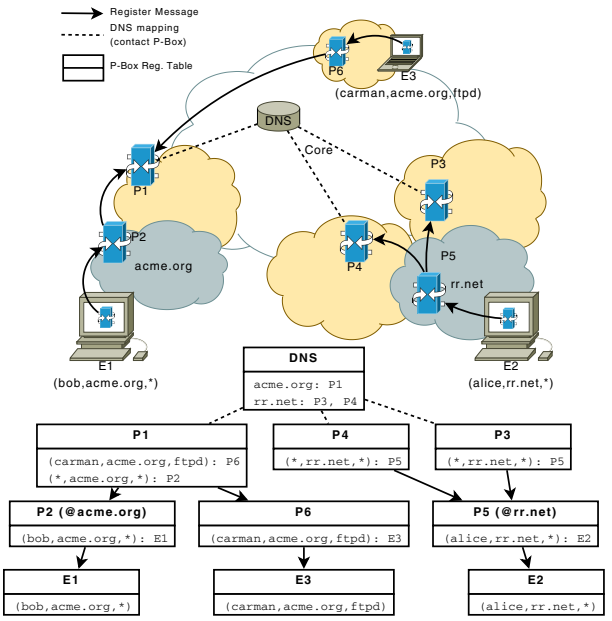


Figure 2: Endpoint registration, and name-routing state created. Network $N5$ is multi-homed. Endpoint $E3$ is roaming.

Algorithm 1 PROCESSREIGSTER(E,A)

Require: E is endpoint name (EU,ED,ES)
Require: A is next-hop address to E
Require: E has been authenticated, can be reached through A, and is authorized to register as per local policy.
Ensure: Name-routed path from contact P-Box for ED to E exists

```
1: UPDATEREGISTRATIONTABLE(E,A)
2: AL ← GETLOCALADDRESS()
3: FWDTO ← GETPARENTPBOXADDRESSES()
4: if ISEMPTY(FWDTO) then
5:    FWDTO ← GETCONTACTPBOXADDRESSESFOR(ED)
6:    if CONTAINS(FWDTO, AL) then
7:       return
8:    end if
9: end if
10: MSG ← new REGISTER(E,AL)
11: for all AP in FWDTO do
12:    SENDTO(MSG, AP)
13: end for
```

address A that wishes to accept flows sends the REGISTER(E, A) message to the local P-Box (Figure 2). When a P-Box receives a REGISTER message (Algorithm 1), it adds the mapping to its local *registration table* (assuming the endpoint is authenticated and authorized to register with that P-Box). If the P-Box has any parent P-Boxes, the P-Box propagates a mapping between the endpoint's name and the P-Box's own address to all the parents. This process is repeated recursively until the REGISTER reaches the core. For instance, in Figure 2, $E1$'s registration is forwarded by his in-host P-Box to $P2$ then to $P1$, $E2$'s registration to $P5$ then to both $P3$, $P4$, and $E3$'s registration to $P6$. Now, if the outermost P-Box is a contact P-Box registered for E's domain, then the registration process terminates as the reverse route from the DNS to the endpoint is complete (e.g. for $E1, E2$). Otherwise, to complete the route the message is forwarded one last hop to the contact P-Boxes for that domain (e.g. for $E3$); this second case is typically encountered by roaming endpoints.

As an optimization, wildcards in REGISTER messages are used

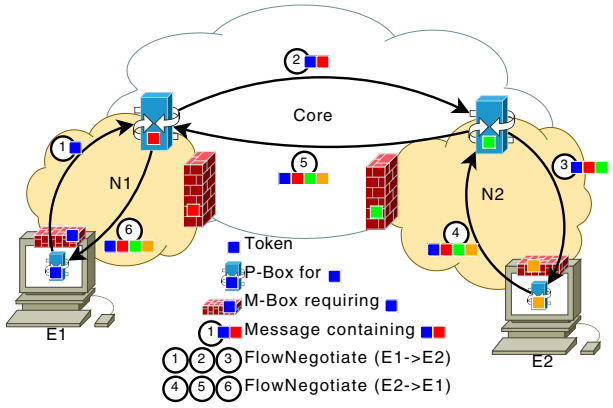


Figure 3: Flow negotiation over name-routed signaling.

to register default routes. A principal can register a default route for all services owned to point to his primary endhost, while a domain (or sub-domain) administrator can register a default route for all endpoints in that domain (or sub-domain) to go through a P-Box he administers. During name-routing, the most specific registration is used (i.e. a route for the endpoint is preferred over a route for the principal, which is preferred over a route for the longest matching domain portion).

2.3.4 *Access Control*

Flow requests may be rejected by P-Boxes in the network in one of two ways. First, the lack of a registration for a given service or principal will cause a P-Box to reject a flow request for that service or principal. Second, an endpoint or P-Box administrator may specify that flow requests for registered names be additionally filtered by the name of the requester, either as a whitelist or a blacklist.

These filters are installed in much the same way as name-routes. An endpoint E_{dst} that wishes to disallow flow requests from E_{src} sends the DISALLOW(E_{dst}, E_{src}) message to the local P-Box; wildcards can be used in either endpoint name to broaden the scope of the filter. A P-Box administrator may likewise do a DISALLOW(E_{dst}, E_{src}) at its P-Box. Either way, P-Boxes may forward the filter up the name-routing tree (as with REGISTER messages), but unlike REGISTER messages, the filter message does not need to bubble up all the way to the top. The filter should nevertheless go beyond the local (in-host) P-Box to allow for in-network filtering. How to resolve conflicting filters is a matter of local policy.

2.3.5 *Name-Routing*

Name-routing is performed over the tree-overlay created by P-Boxes and endpoints in the registration process. An endpoint E_{src} that wishes to initiate a flow with E_{dst} sends a FLOWNEGOTIATE $(E_{src}, E_{dst}, A_{src}, [\,])$ message to its local P-Box. E_{src} and E_{dst} are the endpoint names (no wildcards allowed), and A_{src} is the network address of the initiator. The P-Box authorizes the flow based on installed filters and local network policy. If authorized, the P-Box forwards the message towards the destination as illustrated in Algorithm 2: if the local registration table has an entry matching E_{dst}, the message is forwarded to the associated address. If no matching entry exists and the P-Box has a parent P-Box, the message is forwarded to the parent. If no parent P-Box exists (outermost P-Box), the message is forwarded to a contact P-Box for the destination domain. Local policy may be consulted to pick one or more of many candidate P-Boxes to forward to (e.g. for multi-homed networks).

Algorithm 2 PROCESSFLOWNEGOTIATE(ES,ED,AS,T)

Require: ES is source endpoint
Require: ED is destination endpoint (EDU,EDD,EDS)
Require: AS is effective source address
Require: T is address-routing state $\{\tau_{1...n}\}$
Require: ES is authenticated and authorized to contact ED
Ensure: Endpoints acquire address-routing information needed

```
if DISALLOWEDBYFILTER(ED,ES) then
    return false
end if
if EXISTSINREGISTRATIONTABLE(ED) then
    FWDTO ← REGISTEREDADDRESS(ED)
else if HAVEPARENTPBOX() then
    FWDTO ← SELECTPARENTPBOXADDRESS()
else
    FWDTO ← SELECTCONTACTPBOXADDRESSFOR(EDD)
end if
TOK ← CREATEAUTHTOKEN()
if BEHINDNAT(AS) or EXPLICITMBOX() then
    AS' ← GETMBOXEXTERNALADDRESS()
else
    AS' ← AS
end if
T' ← T ∪ {(TOK,AS)}
MSG ← new FLOWNEGOTIATE(ES,ED,AS',T')
SENDTO(MSG, FWDTO)
```

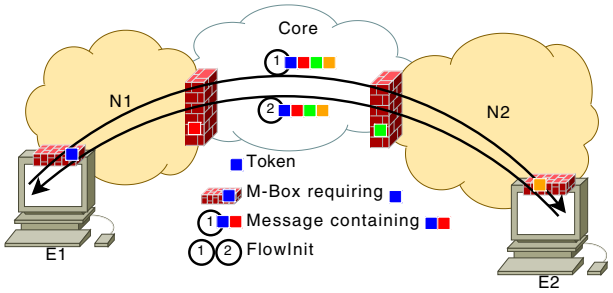


Figure 4: Flow initialization over address-routed signaling (performed after flow negotiation in Figure 3).

Before forwarding the FLOWNEGOTIATE, the P-Box modifies it by adding $\tau_i : (token, nexthop)$, which is the state needed by endpoints and M-Boxes to initialize the address-routed path. τ contains an authorization $token$, which is a nonce signed by the P-Box. If the A_{src} advertised by the endpoint is behind a NAT M-Box, or if the M-Box terminates the address-routed flow (e.g. application level M-Boxes that must be explicitly addressed), the P-Box replaces A_{src} with the address of the M-Box — this is the address that the remote endpoint should send packets to. In such cases, the M-Box will, however, eventually need the original A_{src} for address-routing of processed packets; for this purpose, the P-Box uses the $nexthop$ field in τ to communicate A_{src} to the M-Box. This addition of tokens is illustrated in Figure 3 where each P-Box enroute adds a token required by its M-Box.

When the destination receives the FLOWNEGOTIATE, it learns the effective address of the initiator and a set of tokens $\tau_{1...n}$ that it needs to initialize its data path. The destination name-routes its own address (A_{dst}) and the acquired tokens $\tau_{1...n}$ back to the initiator in a FLOWNEGOTIATE message, which allows the initiator to learn the destination's effective address and tokens.

2.4 Address-routed Messages

Endpoints use the peer address and $\tau_{1...n}$ acquired over name-routed signaling to initialize the address-routed path. The initialization installs any necessary per-flow state in M-Boxes enroute. The initialization process is described as follows: both endpoints address-route a FLOWINIT($A_{self}, A_{peer'}, P_{self}, \tau_{1...n}$) message to the remote endpoint; the message is sent to the peer's effective address $A_{peer'}$ over IP from the local source address A_{self}. P_{self} is the local transport port allocated for the flow, and $\tau_{1...n}$ are the tokens accumulated in the FLOWNEGOTIATE. The message is naturally routed through M-Boxes for networks on the IP-path between the endpoints as shown in Figure 4.

At each M-Box, the message is checked for the presence of a τ_i with a valid authorization token for that M-Box. If found, the message is forwarded to the next-hop as per normal IP routing. If an M-Boxes requires additional state to forward the message (e.g. NATs), the M-Box initializes this state from the $nexthop$ field in τ_i. Port-translating NAT M-Boxes also translate the advertised port P_{self} for outbound messages as per normal NAT operation; this allows the remote endpoint to learn the effective port to use. Once both endpoints have sent FLOWINIT messages, application data can flow along the address-routed path.

As mentioned earlier, if a M-Box receives a message without a valid authorization token, the M-Box responds with a REFERRAL message for its associated P-Box (Algorithm 3). The only exception is a message sent to the associated P-Box, as the P-Box must by default be reachable from both inside and outside that network to route new name-routed messages.

Note that M-Boxes, in general, are not explicitly addressed. This is needed so IP routers retain the ability to route around network failures (particularly around a failed M-Box). If a M-Box fails, the IP route may fail over to another M-Box in the same network; the second M-Box generates a referral for the first data packet routed through it (due to lack of flow state). In such cases, the endpoint attempts to re-initialize the address-routed flow through the new M-Box with the tokens it used to initialize the first M-Box; this is likely to succeed and data flow can resume immediately. In cases where the IP route fails over to a different network altogether (with potentially different flow policies), the original set of tokens is insufficient and the endpoint must re-negotiate the flow over name-routed signaling through the referred P-Box before re-initializing the new address-routed path.

Algorithm 3 PROCESSPACKET(P)

Require: P is an address-routed packet
Ensure: Only authorized flow packets can pass

```
if FOREXISTINGFLOW(P) or FORMYPBOX(P) then
    FORWARDPACKET(P)
    return
end if
if PACKETISFLOWINIT(P) then
    for all τ_i in P do
        if ISVALIDAUTHTOKENFORME(τ_i) then
            if IAMANAT(P) then
                FWDTO ← GETNEXTHOPIN(τ_i)
                CREATENATSTATE(P, FWDTO)
            end if
            FORWARDPACKET(P)
            return
        end if
    end for
end if
RESPONDWITHREFERRAL(P)
```

2.5 Security Considerations

P-Boxes, M-Boxes, referrals, tokens, names and name-routed messages are new elements for attackers to attack, and through

them, attack flow establishment. We now discuss how the architecture defends against these new attacks.

NUTSS brings Akamai-like protection to all endpoints. NUTSS allows for massive replication of P-Boxes and M-Boxes by being flexible about (and dynamically discovering) where they are placed in the network. Furthermore, the NUTSS token mechanism can be co-opted by approaches, such as capabilities [62], to provide DDoS protection to endhosts. While this approach is similar to that taken by Akamai [1], NUTSS operates at the network layer and need not rely a single large proxy provider. NUTSS assumes the presence of external DDoS protection mechanisms [30, 4, 5, 61, 62, 27, 2, 26] to protect P-Boxes and M-Boxes at the IP level. Other than that, standard defenses (crypto-puzzles [59], CAPTCHAs [56] etc.) delivered over the name-routed path apply against resource exhaustion attacks.

We assume that standard authentication protocols on the name-routed path are used by P-Boxes and endpoints to establish trust in each other. P-Box to P-Box communication may be secured with keys exchanged out-of-band when possible (e.g. when establishing customer-provider relationships, or stored in DNS). NUTSS does not mandate the mechanism for establishing trust. As today, trust can be established through reputation-based "webs-of-trust" [64], mutually trusted certificate authorities [54], trusted hardware [51], trust in domains that have good security practices through [7], and so on as per individual preference.

Another target for attack is the authorization token used to couple the name-route to the address-route. An eavesdropper may attempt to use the token generated for legitimate endpoints. A small alteration in how tokens are handled protects tokens against eavesdroppers. The token is never sent in the clear: P-Boxes append three copies of the token in τ, one encrypted for each endpoint, and one encrypted for the M-Box. Endpoints sign FLOWINIT messages with their copy and include the encrypted M-Box copy within. M-Boxes decrypt the token and use it to verify the signature to establish that the endpoint sending the packet possesses the token.

A malicious P-Box (or M-Box) can, at worst, deny service to an endpoint behind it. Note, however, that a malicious P-Box not on the name-routed path between the endpoint and its contact P-Box cannot fake a registration, nor can a malicious P-Box redirect flows to malicious endpoints; authentication protocols along the name-routed path prevent it. Malicious M-Boxes may attempt to redirect FLOWINIT messages to an alternate (malicious) destination, however, without access to the tokens possessed by the intended destination, the alternate destination cannot complete the initialization process in the reverse direction. The only time an address-routed path can be diverted without authorization tokens is if *every* M-Box between the two endpoints is compromised — including, in particular, the in-host M-Box of the non-malicious endpoint.

A malicious endpoint may attempt to abuse its network privileges; the middle can, in response, contain such endpoints at the cost of additional name-routed messages. For instance, an endpoint can attempt to replay legitimately acquired tokens to initialize paths to multiple destinations only one of which is explicitly authorized. This is possible because, *by default*, tokens are bound to named flows and not to ephemeral addresses (to allow for some mobility); P-Boxes may however choose to bind the token to the addresses from which the token can be used, limit the time or number of data bytes the token is valid for, or in extreme cases, make the token single-use by requiring M-Boxes to notify the P-Box of each use. The cost of restricting tokens to granularities finer than flows is additional name-routed signaling each time the address-route breaks trivially (e.g. M-Box reboots).

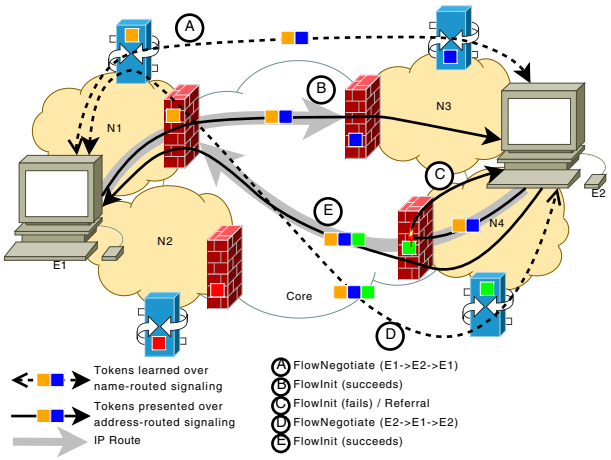


Figure 5: Asymmetric routing example. $E1$ and $E2$ are multi-homed. All M-Boxes perform NAT. IP routing is asymmetric.

2.6 Incremental Deployment

We now describe how the NUTSS architecture can be realized in three incremental phases. The goal of the deployment strategy is to create incentives for applications and networks to adopt NUTSS while keeping costs low.

In the first phase, only endpoint applications are made NUTSS-aware; this involves performing name-routed and address-routed signaling during connection establishment but does not require any changes to networks. A third-party provides a public P-Box that the application can use. Endpoints benefit from architectural support for mobility, ability to traverse legacy NATs (the "killer-app" use-case of NUTSS as described in the next section), and end-to-end (but not end-middle-end) access control. In Section 4, we report on our implementation and deployment of this first phase.

In the second phase, the middle is gradually made name-aware. This is accomplished by individual networks deploying a P-Box. Endpoints behind these networks are configured to use the P-Box (in the same way that DNS resolvers are configured today i.e. through DHCP). The need for configuration is temporary until networks deploy M-Boxes in the third phase allowing the referral mechanism to operate. Networks benefit by gaining insight into, and weak access control over, flows carried by the network.

In the third and final phase, networks replace legacy middleboxes with NUTSS-aware M-Boxes. M-Boxes allow networks to enforce access control policies, and control network use in multi-homed settings. The need for legacy NAT traversal and P-Box configuration introduced in the first two deployment phases is eliminated. If the network still has some legacy (non-NUTSS-aware) endpoints that were not upgraded in the first phase, the M-Boxes are made aware of them so the M-Boxes can allow them through.

2.7 An Example: Asymmetric Routing through Firewalls

We end this section with an example that demonstrates the need to couple name-routed and address-routed signaling, and describes how existing approaches fail in this case. The example involves a scenario, shown in Figure 5, that may easily arise with site multi-homing. In this example endpoints E_1 and E_2 wish to communicate. Both endpoints are multi-homed; E_1 connects to the Internet through networks N_1 and N_2, and E_2 connects through N_3 and N_4. Each network N_i operates a NAT M-Box (M_i with exter-

#	From	To	Message
1.	E_1	P_1	FLOWNEGOTIATE$(E_1, E_2, A_{E_1}, [])$
2.	P_1	P_3	FLOWNEGOTIATE$(E_1, E_2, A_{M_1}, [\tau_1])$
3.	P_3	E_2	FLOWNEGOTIATE$(E_1, E_2, A_{M_1}, [\tau_1, \tau_3])$
4.	E_2	P_3	FLOWNEGOTIATE$(E_2, E_1, A_{E_2}, [\tau_1, \tau_3])$
5.	P_3	P_1	FLOWNEGOTIATE$(E_2, E_1, A_{M_3}, [\tau_1, \tau_3])$
6.	P_1	E_1	FLOWNEGOTIATE$(E_2, E_1, A_{M_3}, [\tau_1, \tau_3])$
7.	E_1	M_1	FLOWINIT$(A_{E_1}, A_{M_3}, P_{E_1}, [\tau_1, \tau_3])$
8.	M_1	M_3	FLOWINIT$(A_{M_1}, A_{M_3}, P_{M_1}, [\tau_1, \tau_3])$
9.	M_3	E_2	FLOWINIT$(A_{M_1}, A_{E_2}, P_{M_1}, [\tau_1, \tau_3])$
10.	E_2	M_4	FLOWINIT$(A_{E_2}, A_{M_1}, P_{E_2}, [\tau_1, \tau_3])$
11.	M_4	E_2	REFERRAL(P_4, A_{P_4})
12.	E_2	P_4	FLOWNEGOTIATE$(E_2, E_1, A_{E_2}, [\tau_1, \tau_3])$
13.	P_4	P_1	FLOWNEGOTIATE$(E_2, E_1, A_{M_4}, [\tau_1, \tau_3, \tau_4])$
14.	P_1	E_1	FLOWNEGOTIATE$(E_2, E_1, A_{M_4}, [\tau_1, \tau_3, \tau_4])$
15.	E_1	P_1	FLOWNEGOTIATE$(E_1, E_2, A_{E_1}, [\tau_1, \tau_3, \tau_4])$
16.	P_1	P_4	FLOWNEGOTIATE$(E_1, E_2, A_{M_1}, [\tau_1, \tau_3, \tau_4])$
17.	P_4	E_2	FLOWNEGOTIATE$(E_1, E_2, A_{M_1}, [\tau_1, \tau_3, \tau_4])$
18.	E_2	M_4	FLOWINIT$(A_{E_2}, A_{M_1}, P_{E_2}, [\tau_1, \tau_3, \tau_4])$
19.	M_4	M_1	FLOWINIT$(A_{M_4}, A_{M_1}, P_{M_4}, [\tau_1, \tau_3, \tau_4])$
20.	M_1	E_1	FLOWINIT$(A_{M_4}, A_{E_1}, P_{M_4}, [\tau_1, \tau_3, \tau_4])$

Table 2: Message-flow for asymmetric routing example.

nal address A_{M_i}) and an associated P-Box (P_i). Inside the multi-homed networks, IP routing results in asymmetric paths — packets from E_1 to A_{M_3} and A_{M_4} are routed through N_1 and N_2 respectively, while packets from E_2 to A_{M_1} and A_{M_2} are routed through N_4 and N_3.

NUTSS establishes an end-middle-end path as follows (Table 2). After registration state is created, E_1's FLOWNEGOTIATE is exchanged with E_2 through P_1 and P_3 (say). In the process E_1 learns A_{M_3} and E_2 learns A_{M_1} as the other side's effective address, along with the tokens needed (messages #1–6 in the table, arrow $\vec{A}$ in the figure). E_1's FLOWINIT to E_2 succeeds (#7–9, $\vec{B}$), however, E_2's FLOWINIT, IP routed through M_4, fails due to the lack of the necessary token resulting in a referral to P_4 (#10–11, $\vec{C}$). E_2 resumes name-routed negotiation through P_4, and both endpoints acquire tokens for M_4 (#12–17, $\vec{D}$). E_2 successfully re-attempts the FLOWINIT with the newly acquired tokens (#18–20, $\vec{E}$). As a side-effect, E_1 learns A_{M_4} as an alternate effective address for E_2 that can be used as a failover (once initialized).

In comparison, existing approaches fail to establish a path. As one might expect, any approach that relies solely on address-routed signaling (e.g. TCP/IP, HIP [35]) simply cannot signal through the facing NATs due to both endpoints having private addresses. Relaying application data through public proxies (e.g. i3 [49]) is sub-optimal as public proxies are potential network bottlenecks. Approaches that use name-routed signaling before address-routed signaling (e.g. DONA [28], i3+DoA [58], SIP+STUN [42, 45]) but do not strongly couple the two fail to recover when the name-routed path does not coincide with the address-routed path (i.e. unexpectedly encountering M_4 above).

Note that the default path discovered by NUTSS is asymmetric owing to the underlying asymmetric IP routing. If this asymmetry is undesirable, the P-Box can use explicit M-Box addressing whereby P_3 changes the address advertised in the FLOWNEGOTIATE (#3) from A_{M_1} to A_{M_3} (and stores A_{M_1} in τ_3); E_2 learns A_{M_3} as the effective address for E_1. E_2's FLOWINIT (#10) in this case is addressed to A_{M_3} instead of A_{M_1}. The message is address-routed to M_3, which validates τ_3 and NATs the message to A_{M_1}, which in turn NATs the message to E_1 completing the initialization. The resulting path is symmetric despite the underlying asymmetry.

3. USING AND EXTENDING NUTSS

This section supplies a number of scenarios that serve to elucidate the operation of NUTSS. Some of these scenarios require minor extensions to the basic architecture described in the previous section. While we should note that each of these scenarios may be handled by one or another existing technology, taken together they demonstrate the breadth of NUTSS and its ability to serve as the foundation for a wide variety of important Internet features.

3.1 Mobility

Mobility in NUTSS follows naturally from the basic connection establishment mechanism. A mobile endpoint registers with the P-Box at the new network address. Once registration state is installed in the intermediate P-Boxes, FLOWNEGOTIATE messages are routed to the new location. An added option to the REGISTER message can be used to explicitly expunge the previous registration state if it is no longer valid. Data transfer for already established flows is suspended while the endpoint is disconnected. Upon re-joining, the endpoint attempts to re-initialize the suspended flow from the new address using the existing tokens; if the initialization succeeds, for instance mobility inside the same network where the new M-Box can use the same token as the old M-Box, data flow can be resumed immediately. Otherwise, data flow is resumed after the name-based path is re-established, the flow re-negotiated, and the address-routed path re-initialized with new tokens.

3.2 Legacy NAT Traversal

Endpoints use name-based routing as a generic signaling mechanism to conduct legacy NAT traversal as proposed in [21]. In the presence of legacy M-Boxes without an associated P-Box, endpoints use a configured third-party P-Box service on the Internet. Endpoints advertise their public address and port in FLOWNEGOTIATE messages. To learn their public address and port, endpoints use a public service like STUN [45]. While key architectural components (tokens, referrals etc.) are not used in this particular case, legacy NAT traversal is a killer-app for endpoints; being able to support legacy NAT traversal creates incentives for endpoints to implement NUTSS, thus bootstrapping deployment.

3.3 Endpoint-Imposed Middleboxes

The NUTSS architecture as discussed focuses on the ability of the middle to impose middleboxes. Endpoints too, can impose middleboxes on a per flow basis. We outline one method as follows. The initiating endpoint imposes a middlebox (e.g. anonymizing proxy) by sending the FLOWNEGOTIATE to the middlebox and having it proxy both name-routed and address-routed messages. The endpoint accepting a flow imposes a middlebox (e.g. virus scanner) by returning the address of the M-Box instead of its own address in the FLOWNEGOTIATE; in addition, the endpoint appends a τ_{dst} that contains its own name. The initiator initializes the address-routed path to the intended middlebox. The middlebox recovers the name of the intended destination from τ_{dst}, negotiates the path to that destination and proxies processed data. Endpoints chain multiple middleboxes by adding a τ_i for each link.

3.4 Application-Level Anycast

Multiple endpoints REGISTER different addresses for the same name; P-Boxes store all address mappings in their local registration table and choose one to forward name-routed messages to. The choice can be based on local network policy, or policy specified

by each endpoint at registration time (e.g. round-robin, primary-backup and so on). This is in contrast to i3 (where the middle cannot specify policy), and Oasis [18] (where the policy specified by the middle can be sidestepped at flow establishment).

While the approach above works for cases where a single FLOWNEGOTIATE can acquire all the tokens needed, a small modification is needed if multiple FLOWNEGOTIATEs are needed to acquire all tokens as there is no guarantee that subsequent messages will be name-routed to the same instance. To rectify this lack of affinity, we add an additional $instance$ component to the name making it a 4-tuple. The $instance$ field uniquely identifies an endpoint in a group of anycast endpoints. The instance name may be picked by the application to be user-friendly. For instance, an application may detect other instances and use the hostname to differentiate itself, or if appropriate may ask the user to name each instance, e.g. home and work. REGISTER messages contain the $instance$ as part of the name. An application can elect to send a FLOWNEGOTIATE to a specific instance (set in E_{dst}), or to any instance (instance name of $*$); P-Boxes use the destination instance name to route to a matching endpoint. An endpoint, however, must always include its own instance name in FLOWNEGOTIATE messages that it generates so the other endpoint can learn its unicast name.

3.5 Negotiating Multicast

NUTSS can be used to support several forms of multicast. The basic idea is to use the 4-tuple names that define a group of endpoints defined in the previous section (3.4) for multicast instead of anycast by transmitting messages to all members rather than only one. There are several possible variants, depending on how large the multicast group is, and whether IP multicast is available. For instance, for small-scale multicast, endpoints could simply establish point-to-point flows with all other members, and individually replicast packets onto all flows. If IP multicast is available, then similar to SIP, the IP multicast address may signaled through P-boxes, and members may join that multicast group. Otherwise, the rendezvous point and group name for an application multicast protocol ([25, 53], among many others) may be conveyed through P-boxes, and endpoints can join the appropriate application multicast group. Finally, P-Boxes and M-Boxes can participate in carving out IP multicast islands in overlay based approaches [63].

3.6 Default-Off

NUTSS can be used to implement the default-off paradigm [5] without requiring changes to the public IP routing core; this is accomplished by disallowing name-routed messages between all endpoints by default. Endpoints must explicitly enable specific flows with ALLOW messages. A common concern is how non Internet-savvy users make use of this paradigm without the default quickly regressing to default-on. We believe that the application must ultimately involve itself in selecting an appropriate policy. For example, remote-desktop-like applications can elect to be default-off, while BitTorrent like applications can be default-on. Over time, one could well imagine applications evolving to be slightly more sophisticated. For example, a given BitTorrent endpoint could use the name of a given torrent as its instance name (Section 3.4) and indicate to the P-Box to filter on that.

3.7 Protocol Negotiation

FLOWNEGOTIATE messages can be used to negotiate the entire protocol stack with involvement from the middle. A general protocol negotiation mechanism would enhance the evolvability of the Internet, and could be used to manage multiple network layers created through virtual routers and switches, for instance as proposed for the GENI infrastructure [19]. In addition to addresses and tokens, endpoints would advertise supported protocol stacks including available transports, network layers, security protocols, tunneling protocols and so on, and how to layer them. For instance, a web-browser may advertise: 1) HTTP-TCP-IPv4, 2) HTTP-TCP-IPsec-IPv6, 3) HTTP-TLS-SCTP-IPv4 etc. P-Boxes remove advertised stacks if the network cannot support it (e.g. #3 if the M-Box does not support SCTP) or if the stack violates network policy (e.g. #1 if policy requires TLS or IPsec).

3.8 Optimizations

One of the main concerns with NUTSS is the added latency required for establishing data flows. Here we discuss three optimizations that may alleviate this concern. First, is to piggyback application data on to signaling messages in order to expedite initial data transfer. With appropriate changes to the networking stack in end-host OS's, this piggybacking could conceivably include protocol handshakes such as HIP, IPsec, TCP, and HTTP, potentially resulting in an overall reduction in data exchange latency as compared with today's protocol operation.

The second optimization is combining the FLOWNEGOTIATE and FLOWINIT when the P-Box and M-Box are co-located (likely for small networks) to initialize the data path sooner. The FLOWINIT in such cases may contain incomplete information regarding tokens and the remote address. The P-Box fills in the token and uses it to initialize the M-Box flow state. Note that the embedded FLOWINIT is piggybacked with the FLOWNEGOTIATE along the name-route so the remote address is not needed for address-routing; however, if the remote addresses is needed for per-flow state in the M-Box, the P-Box waits until the FLOWNEGOTIATE in the reverse direction before initializing the M-Box.

A third optimization couples NUTSS with Self-Certifying Identifiers (SC-ID) in the protocol stack, for instance HIP [35], in order to eliminate the need for additional signaling during IP-path changing events like mobility or middlebox failover. The idea is to include the SC-ID in FLOWINIT messages, and to transmit multiple FLOWINIT messages in parallel to M-boxes. In this way, failover M-boxes (for instance) will have pre-authorized the SC-ID, and can forward data packets immediately upon receiving them. Indeed, this approach can be used to establish multiple parallel data flows through the network, for instance to increase throughput.

4. IMPLEMENTATION

To test the feasibility of NUTSS, we implemented a library that adds NUTSS support to endpoints, and implemented an M-Box used for legacy NAT traversal that we deployed on Planetlab. While the implementation did not uncover any unexpected issues, it did help us iron out the design. Using off-the-shelf software and existing infrastructure, our implementation enables end-middle-end communication (including name-based connection establishment in applications, legacy NAT traversal, mobility, default-off behavior and application-level anycast) in many cases requiring little to no modifications to existing applications.

Our implementation uses SIP [44] for name-routing. While other name-routed signaling protocols (e.g. Jabber/XMPP [47]) may be used, we chose SIP because of its maturity and support in commercial hardware. At the same time, our choice allows us to assess what subset of SIP is most important for NUTSS.

P-Boxes (SER) and Access Control (CPL): We chose to base name-routed components on off-the-shelf commercial software in order to facilitate the second phase of deployment (upgrading networks with support for name-routing). P-Boxes in our implementation are (as yet) unmodified SIP Express Router (SER) [17] proxies.

NUTSS name: $(user, domain, service, instance)$
SIP URI encoding: user@domain;srv=service;uuid=instance

Socket API	NUTSS Primitive	SIP Counterpart
nbind	REGISTER	REGISTER
nsetpolicy	ALLOW/DISALLOW	re-REGISTER (w/ CPL)
nconnect	FLOWNEGOTIATE	INVITE
naccept	FLOWNEGOTIATE	200 OK
nsend/nrecv	FLOWINIT (one-time)	-
nclose	-	BYE

Table 3: Mapping from socket operations to NUTSS primitives, and NUTSS primitives to SIP messages used.

Policy definitions (for ALLOW/DISALLOW messages and domain policy) are compiled (manually, at present, using CPLEd [16]) into the Call Processing Language (CPL) [29], a declarative language used for user-specified VoIP policy and call routing.

Name-routed messages in NUTSS are encoded as SIP messages (Table 3 lists the mapping). Source and destination endpoint names are encoded in SIP header fields (From:, To:), and advertised addresses and tokens are encoded in the body of the SIP message. The messages are (optionally) signed using S/MIME certificates [41]. Address-routed messages are normal TCP/IP messages; the library inserts the FLOWINIT message into the 5-tuple data flow in front of application data.

M-Boxes: While our implementation supports legacy NATs, we implemented a NUTSS-aware M-Box that performs TURN-like [43] relaying of application data to assist endpoints behind particularly poorly behaved NATs [21] to communicate through them. To allow for an unmodified SER proxy, our M-Box includes a shim P-Box that generates tokens for the M-Box; the SER proxy coupled with our shim in series perform the coupling between name-routing and address-routing. The token itself is a 32-bit nonce, one copy of which is sent to the endpoint and another exchanged in-memory between the shim P-Box and M-Box that is used for validating the impending data path.

Endpoints: Endpoint support is implemented as a userspace library for Linux and Windows applications. The library consists of roughly 10K lines of C code and relies on a number of external libraries including eXosip2 [3] for SIP support and OpenSSL [39] for data security. Our library has two interfaces. The first interface offers NUTSS equivalents of the socket API including an `AF_NUTSS` socket address family, and a `sockaddr_ns` structure that encodes the user, domain and application, and optionally, the instance name, as strings. In order to use this interface, applications must be modified accordingly; however, as the API is similar to BSD sockets only minor modifications are needed—mostly confined to populating the address structure with names instead of IP addresses and ports.

The second interface to our library accommodates unmodified existing application binaries. This interface is available only for Linux applications. The library is pre-loaded into memory by the Linux dynamic loader's LD_PRELOAD functionality. This allows our library to transparently hijack libc socket calls in the application and redirect them to NUTSS equivalents. The local endpoint name is configured into environment variables by the user. The user also enters specially encoded hostnames into the legacy application. When the legacy application performs a `gethostbyname` call for the encoded hostname, the call is intercepted by our library, which decodes the NUTSS name and creates a mapping between the identifier and a fake IP address returned to the application. When the application later initiates a `connect` to the fake IP address, the library intercepts and initiates an `nconnect` to the associated name. Calls to other legacy BSD socket functions are handled similarly.

In order to encourage adoption, the NUTSS library transparently performs NAT traversal. After exchanging addresses and ports over name-routed signaling if the direct TCP connections (in both directions), and TCP hole-punching [21] fail, endpoints negotiate the use of a public relay (the TURN-like M-Box described earlier). M-Boxes are deployed on Planetlab hosts. The associated P-Box can be contacted through sip.nutss.net, which routes to the shim P-Box in a randomly selected M-Box. Endpoints acquire a token and transport address for the M-Box. Both endpoints connect to the M-Box and initialize the flow by sending a FLOWINIT message over the connection; the M-Box verifies the token and uses it to pair up the two connections.

As the M-Boxes in our Planetlab deployment do not lie on the IP data path between endpoints, we have not gathered sufficient experience with the referral mechanism.

We have successfully run a number of applications using both the legacy and non-legacy interfaces to our library while transparently incorporating endpoint policy, authentication, legacy NAT traversal and endpoint mobility. Our library works with client-server applications written to our API, as well as with many unmodified legacy applications (e.g. iperf, VNC, GNOME GUI desktop). The library is available for public download at nutss.net.

4.1 Findings

SIP Lessons Learned: We were surprised to find that although SIP was originally conceived for a very different purpose, it can be used to implement name-routing in NUTSS (with one minor modification). Admittedly SIP is rather heavy-weight for the purpose and we would prefer to use a leaner protocol. Nevertheless, given that SIP is deployed widely today and enjoys significant mindshare, there is a compelling case to be made for using a subset of it.

One aspect of SIP that requires special workarounds in our implementation is the lack of support for nested domains. A single REGISTER message only creates a single registration at the local P-Box and not the chain of registrations in the P-Boxes in front of the endpoint as required by NUTSS. While this limitation is not a concern in the first phase of deployment where a public P-Box is used, in the second phase it affects networks not connected directly to the core. A temporary brute-force workaround is for endpoints to explicitly create registrations for each link of the chain; however, this is not always possible due to firewall policy. A more permanent solution is to modify SIP with support for nested domains, and accordingly modify our SER proxy to forward the registrations to the parent P-Box.

Latency: Since ours is a proof-of-concept implementation of the NUTSS architecture, performance numbers are of little relevance as they relate only to our (perhaps simple) access control policy. Nevertheless, some brief comments on performance are worth making. We found that there is little added latency in establishing connections (less than 15ms) with P-Boxes deployed on the same network segment as the endpoints. This is because signaling in our particular setting added one name-routed round-trip (FLOWNEGOTIATE) and one address-routed round-trip (FLOWINIT). Quite predictably, when two nearby Planetlab nodes on the west coast use our public P-Box service deployed at Cornell, the connection establishment latency shoots up to 100–200ms due to name-routed messages having to make four coast-to-coast trips before the direct data can flow. The optimization suggested in Section 3.8 where data is piggybacked in signaling messages should allow initial data bytes to be exchanged in half that time while the address-routed path is established in the background. Our P-Box (SER proxy)

can route approximately 1200 name-routed messages per second on contemporary hardware (∼1050 with challenge-response authentication enabled). A single such P-Box can handle moderate sized sites, but load-balancers will be needed for large sites.

In real-world settings, interaction with multi-homing, complex access control policy, mid-flow reconfigurations, and mobility will make signaling more heavy-weight.

5. RELATED WORK

Several other Internet architectures have been proposed that wean away from 5-tuple addressing. TRIAD [20], IPNL [15], HIP [35], SHIM6 [38] and i3 [49] route datagrams based on URLs, FQDNs, host keys, hashes and flat identifiers respectively; these approaches advocate end-only control and require protocol-stack modifications at endhosts and middleboxes. NUTSS advocates control shared by both the end and the middle and uses a separate name-routed signaling phase that is strongly coupled to existing address-routed stacks. GMPLS [31], which doesn't involve endpoints, uses IP for "name"-routed signaling to negotiate the layer-2 path. Selnet [52], Plutarch [10], AVES [36], Metanet [60], SIP [44], UIA [13], and DONA [28] all involve the middle in resolving endpoint names to realm-specific addresses and routes. In order to provide complete end-middle-end connectivity, however, we believe the middle must play a yet larger role in blocking unwanted flows, and instantiating the address-routed path.

When it comes to blocking unwanted flows, one weakness of the E2E security model as mentioned is that not everyone who has a stake in security is empowered to provide that security. In this model, the middle has little say in what flows are allowed and must rely completely on the endpoints for the protection of the network itself. In select scenarios, in an enterprise for example, the IT department can enforce this control over the ends through software update and configuration management tools like Marimba [6]. In other cases, such as with DoA [58], endpoints can explicitly invoke security services provided by the middle. Such solutions, however, do not protect against malicious or compromised endpoints that may preempt the IT department's control and abuse the network.

An alternate solution is where the middle exerts direct control on flows with the help of a middlebox on the address-routed path. While middleboxes protect the network against uncooperative endpoints, they face the aforementioned problems which we repeat here: firewalls must infer malice based largely on the 5-tuple and costly deep packet inspection, D-WARD [34] infers malice based on deviations from a "normal traffic model", NATs protect only against drive-by intrusions from the outside, and VPNs cannot authenticate remote endpoints that are not VPN members. Ultimately, such middle-only approaches that cannot explicitly negotiate the intent of the endpoint rely on heuristics, which potentially block unmalicious flows.

When it comes to establishing the address-routed path, protocols such as UPnP [33] and Midcom [48] allow the endpoint to create state in the middle. A limitation in these approaches, however, is that the middle cannot participate in flow negotiation either to enforce network policy or to indicate whether the address-routed path and protocol stack chosen by the endpoints is even possible. Session Border Controllers [24] combine name-routing and address-routing in one box, but do so without endpoint knowledge or consent creating authentication and authorization hurdles.

6. SUMMARY AND FUTURE WORK

In this paper, we propose NUTSS, a name-based "end-middle-end" approach to establishing network flows through middleboxes like NATs and firewalls that takes into account policies of the ends and the middle. NUTSS is unique in that it couples both off-path and on-path signaling in order to overcome the weaknesses of either. NUTSS has an incentivized incremental deployment path, and enables a number of important features, including mobility, multi-homing, NAT traversal, negotiation of different network layers, multicast, and anycast.

Although this paper shows NUTSS to be a promising approach, the devil is in the details. Our partial proof-of-concept implementation notwithstanding, the most important next steps are to gain experience with NUTSS and to do more security analysis. Towards this end, we hope that the NAT traversal features of our implementation may serve as a "killer app" to drive deployment and experimentation.

Beyond this, there are a number of interesting research directions that we hope to explore. Due to lack of space, we only briefly list them here. Foremost among these is to explore the use of self-certifying identifiers with NUTSS, as discussed in Section 3.8. Another is the use of an alternative to DNS in the core, for instance a DHT or gossip protocol, as a means of name-based routing between contact P-boxes, in particular to avoid latencies associated with the DNS lookup. Finally, NUTSS appears to be a promising basis for advanced services at the sockets API such as Quality of Service, auditing and billing, publish-subscribe, and store-and-forward service for intermittently connected devices, for instance in DTNs [12].

Acknowledgemnts

The authors would like to thank David Wetherall (our shepherd), Melinda Shore, Scott Brim, Mark Baugher, and our anonymous reviewers for valuable comments and discussion on earlier versions of this document. Ariel Rabkin and Tyler Steele contributed to the NUTSS library.

7. REFERENCES

[1] AKAMAI TECHNOLOGIES, INC. Akamai: How it works.
[2] ANDERSEN, D. Mayday: Distributed filtering for internet services. In *Proceedings of the USITS '03* (Seattle, WA, Mar. 2003).
[3] ANTISIP SARL. The eXtended osip library.
[4] ARGYRAKI, K., AND CHERITON, D. R. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In *Proceedings of the 2005 USENIX Annual Technical Conference* (Anaheim, CA, Apr. 2005).
[5] BALLANI, H., CHAWATHE, Y., RATNASAMY, S., ROSCOE, T., AND SHENKER, S. Off by Default! In *Proceedings of the HotNets'05* (College Park, MD, Nov. 2005).
[6] BMC SOFTWARE. Marimba Product Line.
[7] CALHOUN, P. R., LOUGHNEY, J., ARKKO, J., GUTTMAN, E., AND ZORN, G. RFC 3588: Diameter Base Protocol, Sept. 2003.
[8] CISCO SYSTEMS, I. *Cisco IOS Security Configuration Guide (Release 12.4)*. Cisco Press, 2006, ch. Access Control Lists: Overview and Guidelines, pp. 429–436.
[9] CISCO SYSTEMS, I. *Cisco IOS Security Configuration Guide (Release 12.4)*. Cisco Press, 2006, ch. Firewall Support for SIP, pp. 587–600.
[10] CROWCROFT, J., HAND, S., MORTIER, R., ROSCOE, T., AND WARFIELD, A. Plutarch: An Argument for Network Pluralism. In *Proceedings of the SIGCOMM '03 Workshops* (Karlsruhe, Germany, Aug. 2003).
[11] (ED.), R. B., ZHANG, L., BERSON, S., HERZOG, S., AND JAMIN, S. RFC 2205: Resource ReSerVation Protocol (RSVP), Sept. 1997.
[12] FALL, K. A Delay-Tolerant Network Architecture for Challenged Internets. In *Proceedings of SIGCOMM '03* (Karlsruhe, Germany, Aug. 2003).
[13] FORD, B., STRAUSS, J., LESNIEWSKI-LAAS, C., RHEA, S., KAASHOEK, F., AND MORRIS, R. Persistent Personal Names for

Globally Connected Mobile Devices. In *Proceedings of the OSDI '06* (Seattle, WA, Nov. 2004).
[14] FRANCIS, P. Firebreak: An IP Perimeter Defense Architecture. Tech. Rep. cul.cis/TR2006-2060, Cornell University, Ithaca, NY, 2006.
[15] FRANCIS, P., AND GUMMADI, R. IPNL: A NAT-extended internet architecture. In *Proceedings of the SIGCOMM '01* (San Diego, CA, Aug. 2001).
[16] FRAUNHOFER FOKUS. CPLEd - A CPL Editor.
[17] FRAUNHOFER FOKUS. SIP Express Router.
[18] FREEDMAN, M. J., LAKSHMINARAYANAN, K., AND MAZIÈRES, D. OASIS: Anycast for Any Service. In *Proceedings of NSDI'06* (San Jose, CA, May 2006).
[19] GENI PLANNING GROUP. GENI: Global Environment for Network Innovations.
[20] GRITTER, M., AND CHERITON, D. R. An Architecture for Content Routing Support in the Internet. In *Proceedings of the USITS '01* (San Francisco, CA, Mar. 2001).
[21] GUHA, S., AND FRANCIS, P. Characterization and Measurement of TCP Traversal through NATs and Firewalls. In *Proceedings of the 2005 Internet Measurement Conference* (New Orleans, LA, Oct. 2005).
[22] GUHA, S., AND FRANCIS, P. Identity Trail: Covert Surveillance Using DNS. In *Proceedings of 7th Workshop on Privacy Enhancing Technologies* (Ottawa, Canada, June 2007).
[23] HAIN, T. RFC 2993: Architectural Implications of NAT, Nov. 2000.
[24] HAUTAKORPI, J., CAMARILLO, G., PENFIELD, R. F., HAWRYLYSHEN, A., AND BHATIA, M. Internet draft: Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments, Apr. 2007. Work in progress. draft-ietf-sipping-sbc-funcs-03.txt.
[25] HUA CHU, Y., RAO, S. G., SESHAN, S., AND ZHANG, H. A case for end system multicast. *IEEE Journal on Selected Areas in Communications 20*, 8 (Oct. 2002), 1456–1471.
[26] HUICI, F., AND HANDLEY, M. An Edge-to-Edge Filtering Architecture Against DoS. *ACM SIGCOMM Computer Communications Review 37*, 2 (Apr. 2007), 41–50.
[27] KEROMYTIS, A. D., MISRA, V., AND RUBENSTEIN, D. SOS: secure overlay services. *SIGCOMM Comput. Commun. Rev. 32*, 4 (2002), 61–72.
[28] KOPONEN, T., CHAWLA, M., CHUN, B.-G., ERMOLINSKIY, A., KIM, K. H., SHENKER, S., AND STOICA, I. A Data-Oriented (and Beyond) Network Architecture. In *Proceedings of SIGCOMM'07* (Kyoto, Japan, Aug. 2007).
[29] LENNOX, J., WU, X., AND SCHULZRINNE, H. RFC 3880: Call Processing Language (CPL): A Language for User Control of Internet Telephony Services, Oct. 2004.
[30] MAHAJAN, R., BELLOVIN, S. M., FLOYD, S., IOANNIDIS, J., PAXSON, V., AND SHENKER, S. Controlling High Bandwidth Aggregates in the Network. *ACM Computer Communications Review 32*, 3 (July 2002), 62–73.
[31] MANNIE, E. RFC 3945: Generalized Multi-Protocol Label Switching (GMPLS) Architecture, Oct. 2004.
[32] MARSHALL, W. RFC 3133: Private Session Initiation Protocol (SIP) Extensions for Media Authorization , Jan. 2003.
[33] MICROSOFT CORPORATION. UPnP – Universal Plug and Play Internet Gateway Device v1.01, Nov. 2001.
[34] MIRKOVIĆ, J., PRIER, G., AND REIHER, P. Attacking DDoS at the Source. In *Proceedings of ICNP'02* (Paris, France, Nov. 2002).
[35] MOSKOWITZ, R., AND NIKANDER, P. RFC 4423: Host Identity Protocol (HIP) Architecture, May 2006.
[36] NG, T. S. E., STOICA, I., AND ZHANG, H. A Waypoint Service Approach to Connect Heterogeneous Internet Address Spaces. In *Proceedings of USENIX Annual Technical Conference* (Monterey, CA, June 2002).
[37] NISSENBAUM, H. Privacy as Contextual Integrity. *Washington Law Review 79*, 1 (Feb. 2004), 119–158.
[38] NORDMARK, E., AND BAGNULO, M. Internet draft: Level 3 multihoming shim protocol, Nov. 2006. draft-ietf-shim6-proto-07.txt. Work in progress.
[39] OPENSSL TEAM. The Open Source toolkit for SSL/TLS.
[40] RAMASUBRAMANIAN, V., AND SIRER, E. G. CoDoNS: The Design and Implementation of a Next Generation Name Service for the Internet. In *Proceedings of SIGCOMM'04* (Portland, OR, August 2004).
[41] RAMSDELL, B. RFC 3851: Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification, July 2004.
[42] ROSENBERG, J. RFC 3856: A Presence Event Package for the Session Initiation Protocol (SIP), Aug. 2004.
[43] ROSENBERG, J., MAHY, R., AND HUITEMA, C. Internet draft: TURN – Traversal Using Relay NAT, Mar. 2006. Work in progress.
[44] ROSENBERG, J., SCHULZRINNE, H., CAMARILLO, G., JOHNSTON, A., PETERSON, J., SPARKS, R., HANDLEY, M., AND SCHOOLER, E. RFC 3261: SIP Session Initiation Protocol, June 2002.
[45] ROSENBERG, J., WEINBERGER, J., HUITEMA, C., AND MAHY, R. RFC 3489: STUN – Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs), Mar. 2003.
[46] SAILER, R., ZHANG, X., JAEGER, T., AND VAN DOORN, L. Design and Implementation of a TCG-based Integrity Measurement Architecture. In *Proceedings of 13th USENIX Security Symposium* (San Diego, CA, Aug. 2004), pp. 223–238.
[47] SAINT-ANDRE, P. RFC 3290: Extensible Messaging and Presence Protocol (XMPP): Core, Oct. 2004.
[48] STIEMERLING, M., QUITTEK, J., AND TAYLOR, T. MIDCOM Protocol Semantics, June 2004. Work in progress.
[49] STOICA, I., ADKINS, D., ZHUANG, S., SHENKER, S., AND SURANA, S. Internet Indirection Infrastructure. In *Proceedings of the SIGCOMM '02* (Pittsburgh, PA, Aug. 2002).
[50] TECHNICAL SPECIFICATION GROUP CORE NETWORK AND TERMINALS. 3GPP TS 29.207: Policy control over Go interface, Sept. 2005.
[51] TRUSTED COMPUTING GROUP. TPM Specification Version 1.2.
[52] TSCHUDIN, C., AND GOLD, R. SelNet: A Translating Underlay Network. Tech. Rep. 2003-020, Uppsala University, Uppsala, Sweden, Nov. 2001.
[53] VENKATARAMAN, V., FRANCISY, P., AND CALANDRINO, J. Chunkyspread: Multitree Unstructured Peer-to-Peer Multicast. In *Proceedings of the IPTPS '06* (Santa Barbara, CA, Feb. 2006).
[54] VERISIGN INC. Security (SSL Certificates), Communications, and Information Services.
[55] VIXIE, P., THOMSON, S., REKHTER, Y., AND BOUND, J. RFC 2136: Dynamic Updates in the Domain Name System, Dec. 1997.
[56] VON AHN, L., BLUM, M., HOPPER, N. J., AND LANGFORD, J. CAPTCHA: Using Hard AI Problems For Security. In *Proceedings of EUROCRYPT'03* (Warsaw, Poland, May 2003).
[57] WALFISH, M., BALAKRISHNAN, H., AND SHENKER, S. Untangling the Web from DNS. In *Proceedings of the NSDI '04* (San Francisco, CA, Mar. 2004).
[58] WALFISH, M., STRIBLING, J., KROHN, M., BALAKRISHNAN, H., MORRIS, R., , AND SHENKER, S. Middleboxes No Longer Considered Harmful. In *Proceedings of the OSDI '04* (San Francisco, CA, Dec. 2004).
[59] WANG, X., AND REITER, M. K. Defending Against Denial-of-Service Attacks with Puzzle Auctions. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy* (Washington, DC, USA, 2003), IEEE Computer Society, p. 78.
[60] WROCLAWSKI, J. The MetaNet: White Paper. In *Proceedings of Workshop on Research Directions for the Next Generation Internet* (Vienna, VA, May 1997).
[61] YAAR, A., PERRIG, A., AND SONG, D. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy* (Pittsburgh, PA, May 2004), pp. 130– 143.
[62] YANG, X., WETHERALL, D., AND ANDERSON, T. A DoS-limiting Network Architecture. In *Proceedings of the SIGCOMM '05* (Philadelphia, PA, Aug. 2005).
[63] ZHANG, B., WANG, W., JAMIN, S., MASSEY, D., AND ZHANG, L. Universal IP multicast delivery. *Computer Networks, special issue on Overlay Distribution Structures and their Applications 50*, 6 (Apr. 2006), 781–806.
[64] ZIMMERMANN, P. R. *The official PGP user's guide*. MIT Press, Cambridge, MA, 1995.