P6P: A Peer-to-Peer Approach to Internet Infrastructure

Lidong Zhou
Microsoft Research Silicon Valley
Email: lidongz@microsoft.com

Robbert van Renesse
Dept. of Computer Science, Cornell University
Email: rvr@cs.cornell.edu

***Abstract*— P6P is a new, incrementally deployable networking infrastructure that resolves the growing tensions between the Internet routing infrastructure and the end sites of the Internet. P6P decouples the two through a P2P overlay network formed by the edge routers. P6P brings the benefits of IPv6 directly to end hosts, solving the major headache of IPv6 deployment as well as those of ISP switching, multihoming, and dynamic addressing.**

P6P is a potential killer-app for P2P protocols and could have profound implication for the future Internet; various existing P2P protocols can be retrofitted into P6P to provide features such as routing robustness and multicast. The paper describes the P6P design and architecture, addresses the security and performance concerns, and shows simulation results that support its feasibility.

I. Introduction

The current Internet has been torn with tensions between the inertia of *core Internet*, which forms the public routing infrastructure, and the ever increasing demands from local *end sites*. While IPv6 was proposed to provide end hosts with a large address space for truly end-to-end connectivity and with better support for features such as multicast, anycast, and mobility, the core Internet has been defying the switch to IPv6.

This paper presents P6P, a new networking infrastructure that alleviates the tension between the core Internet and the end sites. P6P hinges on the decoupling of addresses as *identifiers* from addresses as *locators* for routing. The decoupling creates a clean separation of end sites from the core Internet routing infrastructure, as well as an isolation of the transport layer from the network layer. More specifically, P6P provides end hosts with IPv6 capabilities, while preserving IPv4 for core Internet routing; the transport layer for end hosts uses IPv6 addresses as end-to-end identifiers, while the core Internet uses IPv4 addresses for routing packets. The use of persistent unique identifiers for end hosts restores the end-to-end connectivity, thereby simplifying deployment of IPSec and P2P applications. Perhaps most importantly, P6P can be incrementally deployed and uses the support for IPv6 networking already available in all major operating system platforms.

The mapping from identifiers to locators is accomplished through a P2P overlay network formed by *edge routers* that connect end sites to the core Internet. Persistent identifiers are isolated from any changes to how the site connects to the core Internet through the mapping updates in the overlay. Therefore, P6P shields local sites from ISP switching/multihoming. The overlay can use existing distributed hash table (DHT) protocols (*e.g.*, Chord [1]) to achieve scalability.

P6P is a promising candidate as a P2P killer app. P6P targets concrete and fundamental networking problems in the current Internet and has the potential to shape the future of the Internet. A full deployment of P6P requires the kind of scalability that many P2P protocols are designed to achieve. Because P6P builds upon relatively stable edge routers, P6P circumvents the problem of churn in many P2P protocols. P6P provides a unified framework in which various P2P efforts (*e.g.*, for providing application-level multicast) can be integrated.

Section II presents the architecture of P6P. Section III describes P6P tunnel routing protocol, how P6P accommodates ISP switching, and how security can be incorporated into P6P routing. A preliminary performance evaluation through simulation is presented in Section IV. Alternative design choices, nested deployment of P6P, and the support for multihoming, multicast, and robustness are the topic of Section V. We discuss related work in Section VI and conclude in Section VII.

II. P6P Architecture

P6P assigns IPv6 addresses, referred to as *P6P addresses*, to hosts in each end site, referred to as a *P6P site*. P6P addresses are *identifiers that are permanently assigned to hosts (or interfaces) in IPv6 sites.* Each site has a unique site identifier, which is a common and location-independent 48-bit prefix of the P6P addresses in the site, possibly assigned by IANA (Internet Assigned Numbers Authority), and is distinguishable from the site identifiers within native IPv6 addresses.

Note that P6P is not intended to solve the problems of Mobile IP routing and addressing. The proposed solutions for Mobile IP (*e.g.*, [2]) should work as well with P6P as with native IPv6 and are orthogonal to the work described in this paper. While P6P site identifiers

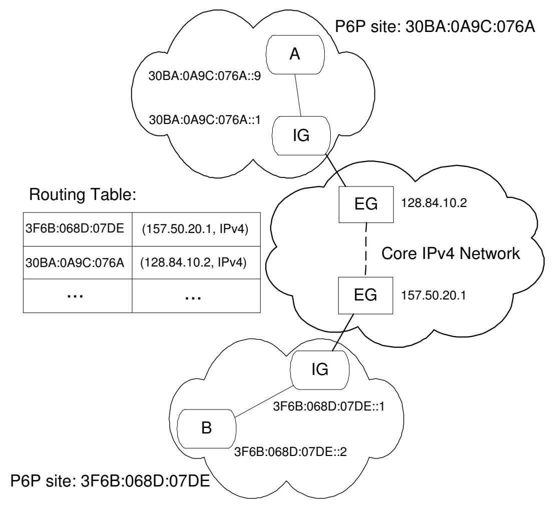

Fig. 1. An example of two P6P sites connected to a core IPv4 network.

are location-independent, P6P addresses are not, as they are tied to their sites. Having P6P addresses be location-independent would no longer make them aggregatable, severely complicating scalability. Also, it would no longer be possible to use the existing IPv6 protocols available for networking within sites.

Each edge router in P6P consists logically of two types of components: *Internal Gateway* (IG) and *External Gateway* (EG). An IG forwards P6P packets between a P6P site and its EG. The IG has its own P6P address, and hosts in the site use it as their default gateway. Each P6P site has to be connected to at least one IG, but multiple IGs may share a single EG. EG components are attached to the core Internet and exchange encapsulated P6P packets through tunnels set up between them.

When an EG receives a P6P packet from an IG, the EG retrieves the P6P destination address from the packet. For routing, each EG maintains a routing table that maps a site identifier to a set of *address records*, each containing the type of protocol used to tunnel P6P packets and the corresponding protocol address of the peer EG. Such protocols may include IPv4, UDP/IPv4, or even native IPv6—a valid option for sites concerned about IPv6 renumbering. Figure 1 shows an example of P6P configuration. The routing table of each EG is populated and maintained by the overlay formed by the EGs, as described in the next section.

III. P6P Routing

The P6P architecture requires a way to map P6P addresses to address records. Unlike site identifiers in standard IPv6 unicast addresses, P6P site identifiers are not location-dependent, and thus a non-hierarchical mapping is necessary between site identifiers and EGs. This section describes one protocol for implementing such a mapping based on a DHT. Alternatives are discussed in Section V-A. Due to space limitation, we omit error handling and other details.

A. Basic Protocol

Each EG runs a DHT agent that implements the DHT routing protocol. While in theory it would be possible to use a DHT to find the EG's address records corresponding to a P6P site identifier, doing so for each packet would be too expensive. Instead, each EG maintains a routing table that caches a subset of the mapping. Changes to the mapping are infrequent and are dealt with in Section III-B. The routing table of an EG initially contains only an entry for its own site identifier. The address records in this entry correspond to the set of protocols that this EG supports, and the set of ISPs the EG is connected to. Let $\mathcal{DHT}(x)$ be the EG that the site identifier x maps to in the DHT. The EG then uses the DHT agent to send an `INSTALL` message containing its local P6P site identifier id and its address records to $\mathcal{DHT}(id)$. On receipt of an `INSTALL` message, an EG copies the routing entry in the `INSTALL` message into its local routing table.

When an EG receives a P6P packet from its IG, the EG first checks whether it already has a mapping for the site identifier id of the destination P6P address in its routing table. If so, the EG selects an appropriate address record and sends the packet accordingly. If not, the EG sends a `LOOKUP` request to $\mathcal{DHT}(id)$ using the DHT. The `LOOKUP` request contains the site identifier and the set of local address records for returning the response. If the site exists, the receiving EG should have a mapping for the site identifier, and returns the entire entry inside an `INSTALL` message. This response is not sent using the DHT, but directly over the core network using the return address in the `LOOKUP` request.

An important optimization is for each EG to piggyback a (limited) number of entries from its routing table on each routing message it sends. The receiver merges these entries into its own routing table. In order to be effective, it is important that the sender selects good entries from its routing table. Random entries are likely not to correspond to popular sites. We currently use the K most recently looked up entries in the routing table. Although it is possible to piggyback on encapsulated data packets as well, we only piggyback on `INSTALL` and `LOOKUP` messages. We also piggyback the local entry of the routing table on `LOOKUP` messages so that P6P response packets can be routed without an additional `LOOKUP`.

B. Updating Address Records

So far we have assumed that the address records of a site do not change. Isolating end sites from the core Internet shields a site from changes in how the site connects to the core Internet (e.g., due to switching ISPs or adding ISPs for multihoming.) In these infrequent cases, address records must be updated. Thus, a routing table in an EG should be considered a *cache* of mappings from P6P to core addresses; P6P must balance freshness and overhead.

Fortunately, it is likely that, when a customer switches ISPs or gets a new address from an ISP to replace its old one, ISPs offer a grace period, during which the customer can continue to receive packets on the old address. Thus out-of-date address records are likely to remain valid for some time, affording P6P some time for updates. We call this time $T_{transition}$.

P6P routing table entries have version numbers for controlling replacement. Routing table entries maintain the time at which a new version of an address record is installed. Each EG increments the version number of its own mapping every $T_{refresh}$ seconds as well as whenever its address record changes, and installs the new mapping (using an `INSTALL` request). When routing a P6P packet, an EG looks up a mapping as before. If the mapping is older than some constant T_{expire} or does not exist, the EG sends a `LOOKUP` request using the DHT. If the mapping exists, whether old or new, the EG sends the packet across the core network using the address record in the entry.

The following should hold:

$$T_{refresh} < T_{expire} < T_{transition}$$

$T_{transition}$ is expected to be at least on the order of days, and we are currently using $T_{refresh} = 15$ minutes and $T_{expire} = 30$ minutes in our prototype.

C. Security

In the P6P routing protocol, as described this far, it would be easy for an adversary to hijack a P6P site identifier simply by installing a DHT entry for the P6P site identifier with any address records of choice. The problem is resolved through public key cryptography.

Entries in the routing table consist of a pair of X.509 certificates [3]: an *owner certificate* and a *map certificate*. The owner certificate establishes the owner of a P6P site identifier. It contains the P6P site identifier and a public key. The corresponding private key is held only by the owner of the P6P site identifier. We expect the owner certificate to be signed by the provider of P6P site identifiers (i.e., IANA). A site obtains this certificate when applying for a P6P site identifier along with the private key to be used for signing mappings.

The map certificate establishes a mapping of a P6P site identifier to address records. It is signed using the private key of the owner of the P6P site identifier. The map certificate also contains the mapping's version number, which is used for updates but also prevents replay attacks.

`INSTALL` messages contain these two certificates for each mapping. For each received mapping, an EG should check both certificates in case the mapping was previously unknown, or is an update for a currently installed mapping.

The solution scales well, as each EG only needs to have an owner certificate for its P6P site identifier, the corresponding private key (for signing new map certificates), and the public key of IANA.

Finally, the DHT itself has to be secure. P6P implements its own integrity through the use of X.509 certificates, but relies on the DHT for its availability. Note that the reachability of a destination host hinges not only on the connectivity between the source EG and the destination EG, but also on the availability of the mapping from the site identifier of the destination to the address records of the destination EG. The work in [4], for example, offers the needed solutions.

IV. Evaluation

We developed a simulation to evaluate the P6P routing protocol. The performance of P6P tunneling in a steady state with no routing table misses is relatively well understood because similar techniques are widely used (*e.g.*, in IPv6 transition mechanisms listed in Section VI). So, our evaluation focuses on the cases of routing table misses that necessitate `LOOKUP` requests before packets are tunneled. In particular, we were interested in the ratio of routing table misses, as well as in the distribution of the P6P routing load across the EGs, as functions of N, the number of P6P sites, and K, the number of piggybacked entries on `LOOKUP` and `INSTALL` messages. We ignore DHT churn at this time because we expect the EGs to be fairly stable, although we do intend to study the impact of churn on P6P performance in the near future.

The simulation runs 1000 rounds, and each run has N microrounds, where N is the number of sites. In each microround, a random source site is chosen, as is a random destination site, according to a Zipf distribution (the sites are ranked according to their incoming packet rates.) Next, a packet is sent from a randomly chosen address within the source site to a randomly chosen address within the destination site. Expiration times and failures are not modeled in these simulations.

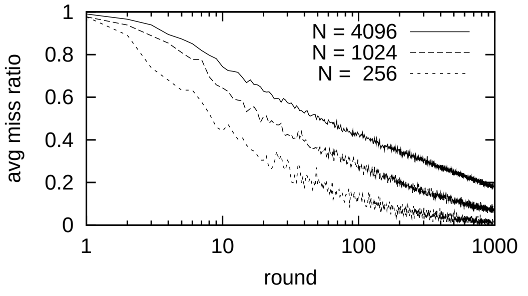

Fig. 2. Average miss ratio as a function of round number for various N. $K = 8$.

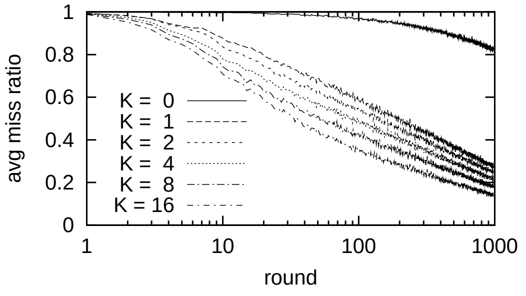

Fig. 3. Average miss ratio as a function of round number for various values of K. $N = 4096$.

In the first set of experiments, we fixed K to 8. Figure 2 shows the average *miss ratio* (*i.e.*, the number of routing table misses divided by N), as a function of the round number (time, if you will) for various N. Note that the x-axis has a log scale, and so the average miss ratio appears to decrease approximately logarithmically with time until about 90% of routing requests (including all of the most popular ones) can be filled from the local routing table. After that the decrease slows down, as LOOKUP requests still occur for unpopular sites.

In the next set of experiments we used 4096 sites, and varied K, the number of piggybacked routing table entries on routing protocol messages. In Figure 3 we show the average miss ratio as a function of round number. We see that even if $K = 1$, piggybacking improves the protocol considerably compared to not piggybacking. However, increasing K only gradually improves efficiency.

To see how load grows as a function of N, we look at the average number of LOOKUP requests received divided by N in the 100^{th} round of the simulation. In Figure 4, we plot the average load over all sites as a function of N. For $K > 0$, the load appears to grow approximately logarithmically with N, which indicates that the protocol scales well. (For $K = 0$, the load is high while the effectiveness is low.) The load decreases logarithmically with K, and as it comes at the price of larger protocol messages, choosing a large K is not cost-effective. The load is still high in round 100, but as the load reduces in later rounds, the tendencies as a function of N and K appear to remain the same (not shown in this paper).

We were initially concerned that the P6P routing protocol might place an uneven load on sites because of the highly non-uniform Zipf distribution of site popularity. After all, all LOOKUP requests for the most popular site go to one particular other site (as selected by the DHT). This concern appears unfounded for

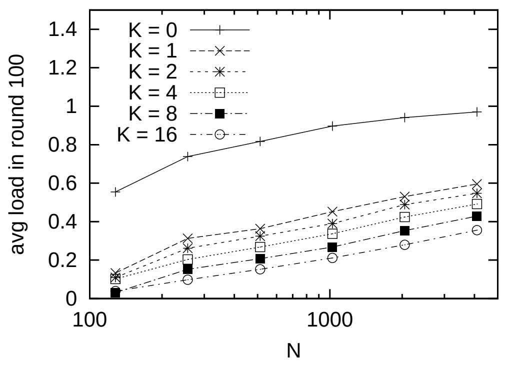

Fig. 4. Average load on EGs in round 100 as a function of N.

$K > 0$. Except for $K = 0$, the variance is low (not shown here), and thus piggybacking ensures that the load is distributed well among the EGs.

The simulation results are synthetic and the number of sites small, and a large-scale deployment will be necessary for further validation and for obtaining realistic latency measurements. Nonetheless, the simulation results indicate that the protocol appears to scale well in the number of sites. Piggybacking significantly improves the protocol's performance, as well as the distribution of load across the P6P routers. The amount of piggybacking can be small, as increasing the amount of piggybacking increases the size of protocol messages linearly while improving the performance of the protocol only logarithmically. Most packets can be routed immediately, while only the initial packets for unpopular sites require waiting for the P6P lookup protocol to finish. The simulation results may be pessimistic because locality of access is not modeled.

V. Discussion

A. Alternative Routing Protocols

The P6P routing protocol as described is one way of mapping P6P site identifiers to core addresses. It could be argued that the DHT technology at the core of P6P is not yet mature enough, and we only have experience with DHTs on relatively small scales.

Other possibilities for mapping would include the use of a directory service. For example, an EG can retrieve a set of address records by doing a reverse DNS lookup on the P6P site identifier. This would require the introduction of a new record type to DNS, but also some new protocol to replace the piggybacking of P6P routing in order to reduce the load on DNS. For example, the most popular mappings could be gossiped using an epidemic protocol [5].

Relying on and extending the existing DNS infrastructure could cause conflicts because the new functions might demand different system design tradeoffs than the original DNS functions. Alternatively, the overlay formed by the P6P EGs can provide a directory service of its own, making it possible to deploy P6P without relying on another infrastructure and to customize the directory service for P6P only.

B. Nested Deployment

P6P can be deployed within a site as well as on the Internet, resulting in a nested deployment of P6P. Within the site, IPv6 sub-sites would be connected through the site's private IPv4 network. A private DHT would tie the sub-sites together into a P6P network.[1] The site's IG would serve as the site's IPv6 exit router. This IG has to be connected to an internal P6P relay router that the sub-sites can tunnel to.

Two small modifications to P6P are necessary to make this work. First, instead of using the 48-bit site identifier, the key for the site's internal DHT would be made up from all or part of the subnet identifier in order to distinguish the various sub-sites. Secondly, internal EGs should route packets destined for local sub-sites using its DHT-managed routing tables, but packets destined for remote P6P sites, as well as native IPv6 addresses, should be routed to the site's exit router.

C. Multihoming, Multicast, and Robustness

P6P does not require cooperation of the core routers or of the ISPs to take advantage of multihoming. A site can simply attach its EG to multiple ISPs and list all connections in its set of address records. The address records can be extended to contain policies for indicating a primary link or for traffic engineering.

P6P can incorporate application-level multicast (*e.g.*, Narada [6] and Overcast [7]). We are currently developing an infrastructure that will provide applications with the illusion of IPv6 multicast by setting up point-to-point dissemination trees between EGs.

P6P can also be extended to support routing robustness by exploiting ideas from RON [8] and Detour [9].

VI. Related Work

The general idea of using P2P for implementing IPv6 appeared in [10], where the IPv6 addresses for the end hosts are not aggregated. P6P instead trades off generality for scalability by aggregating IPv6 addresses for each site. P6P further improves the scalability using piggybacking and addresses the security concern.

In [11], the advantages and disadvantages of separating identifiers and locators, two roles currently overloaded on IP addresses, are discussed in the context of the GSE proposal [12]. GSE proposes to split an IPv6 address into two parts, the first containing the locator, and the second the identifier. GSE routers rewrite the locator part of the IPv6 address as they forward a packet. The essential difference between GSE and P6P is that GSE is a header-rewriting technology (like NAT), while P6P is a layering technology. Therefore, in P6P the locators are not visible to hosts. Clean separation between the network and transport layers in P6P provides end hosts with truly end-to-end connectivity.

UIP (Unmanaged Internet Protocol) [13] also advocates the separation of naming and routing. However, UIP aims to facilitate ubiquitous network computing by designing a new Internet-independent routing infrastructure. As a consequence, it has to build up a DHT when nodes join, while P6P can use the existing Internet infrastructure for the construction of the DHT.

HIP (Host Identity Payload) is yet another proposal that supports the decoupling of internetworking from transport layer. The deployment of HIP requires a new protocol number assigned by IANA and changes to DNS for maintaining the mapping. In contrast, P6P does not require such dramatic changes to the current Internet infrastructure and can be incrementally deployed.

IPNL (IP Next Layer) [14] is an alternative proposal to IPv6. IPNL is layered on top of IPv4 with NAT-boxes, and uses DNS domain names as addresses, while using the structure of domain names as a way to route packets. IPNL achieves many of the same properties as P6P, but requires a large software development commitment. PeerNet [15] goes a step further and proposes to replace the IP layer with a P2P protocol that separates identifiers from addresses. The main objective is routing in wireless networks. Compared to IPNL and PeerNet, P6P deployment is relatively cheap because most operating systems support IPv6 stacks and many important applications have been ported.

[1]Smaller sites that do not require the scalability of a DHT might opt for a simpler mapping mechanism such as a directory.

I3 (Internet Indirection Infrastructure) [16] shares many of the same objectives with P6P, and also uses a DHT in order to separate identifiers from addresses and to support a wide variety of end-to-end communication options. I3 is an overlay, and with it comes a new API. P6P provides a standard IPv6 API, requires no changes to end hosts, and is unencumbered by churn.

The architecture of P6P is related to transitioning mechanisms such as 6to4 [17], ISATAP [18], and Teredo [19]. None of these provide a separation of identifiers and locators, nor the ability to add new communication services.

VII. Conclusion

P6P is an overlay routing architecture that, transparently to end-hosts, implements the IPv6 routing abstraction. P6P simplifies transition to IPv6, but also provides features that have value even in a core IPv6 network. These derive from the fact that P6P allows end-host addresses to be administered separately from the addresses used in the core routing infrastructure. Such features include the ability to renumber the core Internet, to switch ISPs, and to support multihoming, without reconfiguring entire sites, breaking connections, updating DNS records, mutual cooperation of ISPs, or increased load on core routers. P6P can be extended to support multicast and robust routing.

P6P is easy to deploy as it interoperates with the IPv6 stacks supported by all major operating systems. Moreover, P6P can be deployed incrementally without any changes to existing IPv6 routing protocols.

We present the key elements of a preliminary design and evaluation to demonstrate the feasibility of the approach. Various alternative design choices, optimizations, and other engineering/deployment details are yet to be explored fully. Even so, P6P has already shown promises as a killer-app for P2P protocols.

VIII. Acknowledgments

Piyoosh Jalan, Adrian Bozdog, and Michael A. Marsh made early contributions to this work. We would also like to thank the anonymous reviewers for their comments and suggestions that helped improve the paper. This work was funded in part by DARPA/AFRL-IFGA grant F30602-99-1-0532, and by the AFRL/Cornell Information Assurance Institute.

References

[1] I. Stoica, R. Morris, D. Karger, and M.F. Kaashoek, “Chord: A scalable peer-to-peer lookup service for Internet applications.,” In *Proc. of the ’01 Symp. on Communications Architectures & Protocols* [20].

[2] C. Perkins, “IP mobility support,” Oct. 1996, RFC 2002.

[3] CCITT, “Recommendation X.509: The Directory Authentication Framework,” 1988.

[4] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D.S. Wallach, “Secure routing for structured peer-to-peer overlay networks,” in *Proceedings of the 5th Symposium on Operating System Design and Implementation (OSDI’02)*, Boston, MA, Dec. 2002.

[5] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, “Epidemic algorithms for replicated database maintenance,” in *Proc. of the 6th ACM Symp. on Principles of Distributed Computing*, Vancouver, BC, Aug. 1987, pp. 1–12.

[6] Y. Chu, S.G. Rao, S. Seshan, and H. Zhang, “Enabling conferencing applications on the Internet using an overlay multicast architecture,” In *Proc. of the ’01 Symp. on Communications Architectures & Protocols* [20].

[7] J. Jannotti, D.K. Gifford, Johnson K.L., Kaashoek M.F., and J.W. O’Toole Jr., “Overcast: Reliable multicasting with an overlay network,” in *Proceedings of the 4th Symposium on Operating System Design and Implementation (OSDI’00)*, San Diego, CA, Oct. 2000.

[8] D.G. Andersen, H Balakrishnan, M.F. Kaashoek, and R. Morris, “Resilient Overlay Networks,” in *Proc. of the 18th ACM Symp. on Operating Systems Principles*, Banff, Canada, Oct. 2001, pp. 131–145.

[9] Andy Collins, “The Detour framework for packet rerouting,” M.S. thesis, University of Washington, Seattle, October 1998.

[10] 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, Proc. 21st IEEE Symposium on Reliable Distributed Systems*, Suita, Japan, Oct. 2002.

[11] M. Crawford, A. Mankin, T. Narten, J.W. Stewart, and L. Zhang, “Separating identifiers and locators in addresses: An analysis of the GSE proposal for IPv6,” Oct. 1999, Internet Draft, draft-ietf-ipngwg-esd-analysis-05.txt.

[12] M. O’Dell, “GSE–an alternate addressing architecture for IPv6,” Feb. 1997, Internet Draft, draft-ietf-ipngwg-gseaddr-00.txt.

[13] B. Ford, “Unmanaged Internet Protocol: Taming the edge network management crisis,” in *2nd Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[14] P. Francis and R. Gummadi, “IPNL: A NAT-extended Internet architecture,” In *Proc. of the ’01 Symp. on Communications Architectures & Protocols* [20].

[15] J. Eriksson, M. Faloutsos, and S. Krishnamurthy, “PeerNet: Pushing Peer-to-Peer down the stack,” in *Peer-to-Peer Systems—Second International Workshop (IPTPS’03)*, Cambridge, MA, Feb. 2003, vol. 2735 of *Lecture Notes on Computer Science*, pp. 268–277, Springer-Verlag.

[16] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, “Internet Indirection Infrastructure,” in *Proc. of ACM SIGCOMM’02*, Pittsburgh, PA, Aug. 2002.

[17] M. Holdrege and P. Srisuresh, “Connection of IPv6 domains via IPv4 clouds,” Feb. 2001, RFC 3056.

[18] F. Templin, “Intra-Site Automatic Tunnel Addressing Protocol (ISATAP),” Mar. 2001, Internet Draft, draft-ietf-ngtrans-isatap-00.txt.

[19] J. Huitema, “Teredo: Tunneling IPv6 over UDP through NATs,” Sept. 2002, Internet Draft, draft-ietf-ngtrans-shipworm-08.txt.

[20] ACM SIGCOMM, *Proc. of the ’01 Symp. on Communications Architectures & Protocols*, San Diego, CA, Aug. 2001.