Implementing IPv6 as a Peer-to-Peer Overlay Network[*]

Lidong Zhou   Robbert van Renesse   Michael Marsh

Dept. of Computer Science, Cornell University, Ithaca, NY 14853

E-mail: {ldzhou,rvr,mmarsh}@cs.cornell.edu

Abstract

This paper proposes to implement an IPv6 routing infrastructure as a self-organizing overlay network on top of the current IPv4 infrastructure. The overlay network builds upon a distributed IPv6 edge router with a master/slave architecture. We show how different slaves can be constructed to tunnel through NATs and firewalls, as well as to improve the robustness of the routing infrastructure and to provide efficient and resilient implementations for features such as multicast, anycast, and mobile IP, using currently available peer-to-peer (P2P) protocols. The resulting IPv6 overlay network would restore the end-to-end property of the original Internet, support evolution and dynamic updating of the protocols running on the overlay network, make available IPv6 and the associated features to network applications immediately, and provide an ideal underlying infrastructure for P2P applications, without changing networking hardware and software in the core Internet.

1. Introduction

IPv6 [7], when deployed on a large scale, would solve many current networking problems. Its 128-bit addresses would adequately solve the address shortage problem. This in turn would eliminate the currently widespread use of Network Address Translation (NAT) [21] that destroys the end-to-end property of the Internet—the loss of the end-to-end property consequently prevents the use of IPSec [15] and complicates the deployment of true peer-to-peer (P2P) applications [16].

Unfortunately, adoption of IPv6 has been slow. This is because NAT has already solved many of the security and address shortage problems, thus supporting network applications without requiring the deployment of new networking hardware and software. As a result, there is little incentive to switch. Nevertheless, many popular operating systems do support IPv6 already.
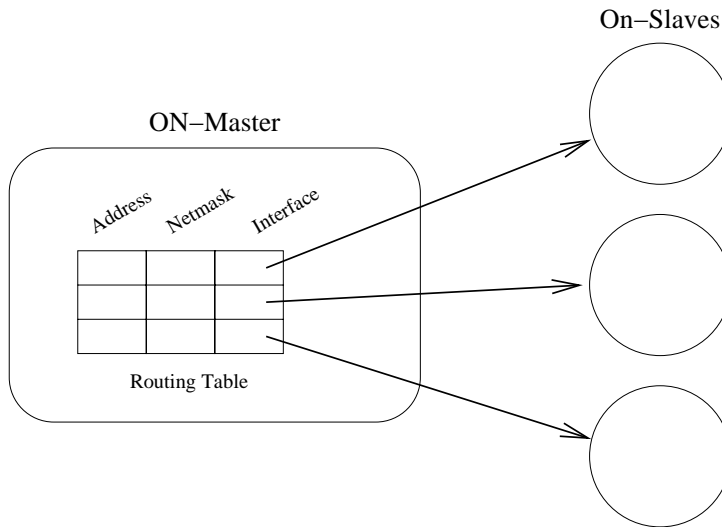
In this paper, we propose to implement an IPv6 routing infrastructure as an Overlay Network (ON) on top of the current IPv4 + NAT infrastructure using existing P2P protocols such as Chord [22] and Astrolabe [24]. We believe our ON would have the following advantages:

- it would allow network applications to adopt IPv6 immediately;

- it would support IPSec, multicast, anycast, and mobile IP;

- it would support P2P applications without the need for special-purpose Application Level Gateways (ALGs) and non-standard P2P protocols such as used both in JXTA [10] and Groove [16];

- it would not require any network hardware or software changes (as a routing infrastructure, IPv4 works well—why change it?);

- it would be easily evolvable.

We intend this ON to be reliable, efficient, and integrate seamlessly with existing or future deployments of a native IPv6 network. While the 6bone [1] is also an ON, the 6bone has been designed primarily as a testbed for IPv6 protocols. Unlike the 6bone, which requires manually configured tunnels, our proposed ON is designed to be self-organizing.

On–Slaves

ON–Master

Address Netmask Interface

Routing Table

Figure 1. Router Architecture.

Our proposition is interesting to the P2P community for two reasons. First, our ON *is* an application of P2P protocols. Second, our ON *supports* P2P protocols and applications, as it eliminates problems with NAT. This paper focuses on the first reason and describes the architecture of our proposed solution, as well as how current P2P protocols can be exploited to implement the multicast, anycast, and mobile IP features of IPv6.

2. Architecture

Our IPv6 ON is implemented at or near the *edge* of the *Core Internet* (CI). The CI consists of those machines that are not residing behind NAT boxes or firewall routers and that have static IPv4 addresses. Its edge consists of the end-hosts on the core Internet, and includes server and workstation machines, as well as NAT boxes and firewall routers, but not internal routers. We will call the collection of machines behind a NAT box or a firewall router a *site*. We assume that all servers and workstations support IPv6 already, and thus all they need is an IPv6 routing infrastructure.

How to construct such an ON poses some interesting and challenging research questions. The objectives are as follows:

- **Self-organization:** The ON requires no centralized administration and minimum maintenance, even as the set of participants grows or changes.

- **Robustness:** The ON is resilient to certain broken routes through adaptation.

- **IPv6 Features:** The ON supports IPv6 multicast, anycast, mobility, and IPSec.

Our technology builds on a *distributed IPv6 edge router*. This router has a master/slave architecture, as depicted in Figure 1. It consists of a single *ON-master*, and several *ON-slaves*. Each ON-slave is responsible for part of the IPv6 address space. The ON-master receives messages from ON-slaves, and routes messages to ON-slaves based on its local routing table. ON-slaves can be added or removed dynamically at any time. The ON-master implements the usual longest prefix matching routing policy.

Our architecture is agnostic about where an ON-slave resides with respect to the ON-master. An ON-slave could reside on a different machine behind a NAT box or a firewall router, in which case communication between the ON-slaves and the ON-master could be through HTTP, where the ON-master acts as a web server. The first HTTP request from the ON-slave reports its IPv6 address range to the ON-master, who can then update its routing table accordingly. (Similarly, if a connection to an ON-slave breaks, the ON-master removes the corresponding entry from its routing table.) An ON-slave could also run on the same machine as the ON-master, in which case IPC, which incurs less overhead, could be used for communication between the two.

We envision a large collection of ON-slaves, for example:

- **Gateway Slaves:**

  A Gateway Slave typically resides within a site—it acts as an IPv6 gateway for the other machines within the site, but it may also be deployed on the CI to act as an IPv6 gateway for IPv6 hosts on the CI. (In the latter case, the Gateway Slave can be deployed on the

same machine as the ON-master.) If the Gateway Slave is deployed within a site, then the connection to the ON-master is from inside a firewall or NAT box to the outside. There is almost always some way of establishing such a connection. Our current implementation uses HTTP 1.1 over IPv4, which can run through an HTTP proxy if necessary.

- **DHT Slaves:**

  A DHT Slave tunnels IPv6 messages over IPv4 to other DHT Slaves. For this, the DHT Slave maps IPv6 addresses to IPv4 addresses using a Distributed Hash Table protocol [19, 22, 20, 25]. In fact, we will use a different Slave for each DHT implementation, and so the IPv6 address range can be subdivided among the different protocols.

  The way the mapping works is as follows. First, each DHT Slave installs its local IPv6 aggregate address in the DHT. To look up an IPv6 address, a DHT Slave first attempts to look the address up as is. If this fails, the Slave zeroes the last set bit in the address, and makes another attempt. This continues until the address is all zeroes, at which point the Slave gives up. If successful, the Slave caches the mapping for some limited amount of time. (Actually, failed mappings are also cached, but for a smaller amount of time.)

  Note that this technique also supports mobile IP [18] in a trivial way, simply by updating the DHT for the moving address.

- **Multicast Slaves:**

  Another class of ON-slaves is responsible for multicasting messages. There are many so-called *Application-Level Multicast* protocols available for this [5, 9, 14, 3, 4, 17, 26]). Similar to point-to-point routing, the multicast address space can be subdivided among the various protocols. In order to fully support IPv6 multicast, the Gateway Slave is required to implement Multicast Listener Discovery [6]. As IPv4 multicast is poorly supported in today's IPv4 network, we plan to support IPv4 multicast in our architecture as well.

- **Anycast Slaves:** Anycast is a routing facility where an address is mapped to the most appropriate receiver. For example, all DNS servers could share the same address, and messages would be routed to the nearest one. Astrolabe [24] provides a framework for keeping track of and managing resources in a large-scale network. This framework can serve as a basis for supporting anycast, where a request needs to be routed to one of a set of possible servers that is most appropriate based on certain metrics.

- **Resiliency Slaves**: RON [2] is an IPv4 ON that improves the robustness of the connectivity of the core Internet. This technique may be exploited in an IPv6 ON as well by implementing the appropriate ON-slave.

- **Bridge Slaves**: Bridge Slaves connect our ON with other existing overlay or native networks, such as the 6bone. A Bridge Slave is often both a slave in our ON and a router in another network. Therefore, the Bridge Slave can move packets from one network to another through standard routing.

- **Management Slaves**: The dynamic addition and removal of ON-slaves are the responsibility of Management Slaves. The default entry of the routing table on an ON-master points to a Management Slave, so that the Management Slave is invoked when no ON-slave is found for a certain address. The Management Slaves could then look up the software for the appropriate ON-slave being requested, download the software, and install the ON-slave on-demand.

  The Management Slaves could form a software distribution overlay to facilitate the distribution, lookup, and management of ON-slave software. Existing techniques, such as dynamic loading [11] and gossiping [8] for distributing new protocols, could be used to implement these functions of the Management Slaves.

  The Management Slaves also removes ON-slaves when appropriate. For example, a Multicast Slave could be removed if no machine in the domains controlled by the ON-master is interested in using the multicast facility.

Figure 2 illustrates the architecture of our proposed IPv6 ON. A Gateway Slave connects a local site to the rest of the ON. The ON-master for the Gateway Slave maintains other ON-slaves. Every such ON-slave, jointly with the corresponding ON-slaves of other ON-masters, forms an overlay network. (In the case of Bridge Slaves, the ON-slaves are part of an existing overlay or native network.)

The large IPv6 address space makes it possible to have multiple overlay networks of ON-slaves to coexist in different address spaces. Consequently, our ON provides an ideal platform for testing and comparing a set of protocols of the same type; for example, different DHT protocols or different application-level multicast protocols.

The IPv6 addressing architecture [12] provides a convenient way to manage these different address spaces. Services such as multicast and anycast already have their own address spaces. Furthermore, different TLA IDs (Top-Level Aggregation Identifier) or NLA IDs (Next-Level Aggregation Identifier) [12] could be assigned to different ON-slave
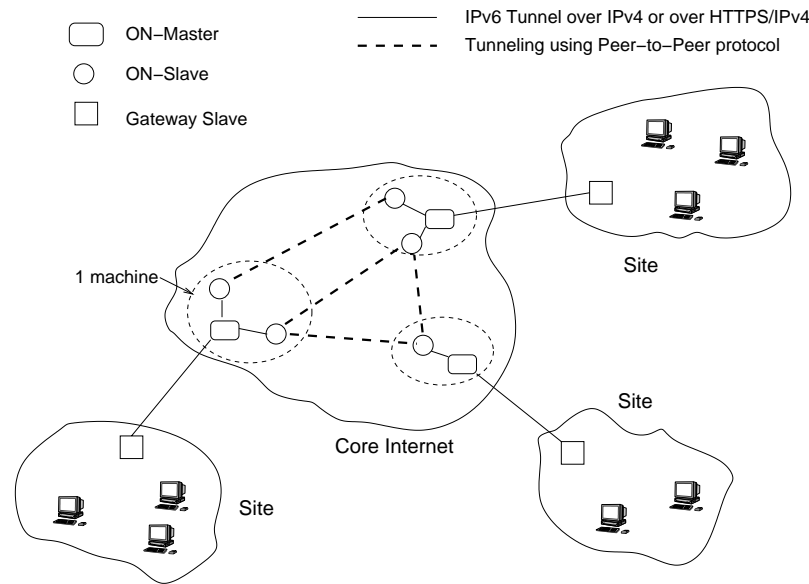
Figure 2. An IPv6 Overlay Network.

overlays.[1] Consequently, a local machine might have multiple IPv6 addresses (with different TLA IDs or NLA IDs), one for each ON-slave overlay in which it participates.

Note that we only intend to use these peer-to-peer protocols on our routers. As the set of routers is a relatively stable collection, many instability concerns that exist with these protocols when deployed on a dynamic set of workstations do not apply here. Furthermore, by not including in the ON the end hosts of each site, the proposed architecture preserves locality within a site, which would otherwise be lost.

The flexibility of our ON comes in part from the fact that our routers can run any protocols we deploy, especially with the help of Management Slaves. The design with Management Slaves is, to some extent, similar to Active Networks [23], which advocate dynamically programmable routers. However, an ON does circumvent some difficulties that Active Networks have faced. Most importantly, our ON does not require changes to, or overheads in the core Internet routers. Nonetheless, our ON does introduce some overhead due to tunneling between the ON-slaves and the ON-master. Implementation and detailed measurements are underway to quantify such costs.

3. Conclusion

Our proposed IPv6 Overlay Network implements a network with the following desirable features:

- **End-to-End:** Because of the large address space of IPv6, each node in the network can have a unique (IPv6) address, at least in space if not in time. The Internet's original end-to-end property can thus be restored. Mechanisms such as IPSec, whose operations hinge on the end-to-end property, are now possible, as are true peer-to-peer applications.

- **IPv6 features:** Applications can take advantage of IPv6 features, such as multicast, anycast, mobile IP, and IPSec. Applications developed using the IPv6 API can run not only on our ON, but may continue to run as IPv6 is deployed in various other forms.

- **Robustness:** Application-level routing techniques can exploit the redundancy in underlying connectivity to improve robustness.

- **Flexibility:** Active Networking-like technology allows the network to be upgraded or extended with new protocols, and is thus able to evolve quickly as new peer-to-peer routing technology emerges.

We believe all these feature can be implemented efficiently without changes to the existing IPv4 Internet.

---

[1]This is consistent with the standard practice. For example, the 6bone is assigned a TLA ID 0x1FFE [13].

References

[1] The 6bone. http://www.6bone.net.

[2] D. G. Andersen, H. Balakrishnan, M. Frans Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP'2001)*, pages 131–145, Chateau Lake Louise, Banff, Alberta, Canada, October 2001. ACM.

[3] Y. Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, University of California, Berkeley, December 2000.

[4] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the Internet using an overlay multicast architecture. In *Proceedings of ACM SIGCOMM*, pages 55–67, San Diego, CA USA, August 2001. ACM.

[5] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, pages 1–12, Santa Clara, CA USA, June 2000. ACM.

[6] S. Deering, W. Fenner, and B. Haberman. Multicast listener multicast (MLD) for IPv6. Request for Comments: 2710, October 1999.

[7] S. Deering and R. Hinden. Internet Protocol, version 6 (IPv6) specification. Request for Comments: 2460, December 1998.

[8] 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 *Proceedings of the 6th ACM Symposium on Principles of Distributed Computing*, pages 1–12, Vancouver, BC Canada, August 1987. ACM.

[9] P. Francis. Yoid: Extending the Internet multicast architecture. http://www.icir.org/yoid/docs/index.html, April 2000.

[10] L. Gong. JXTA: A network programming environment. *IEEE Internet Computing*, 5(3):88–95, May/June 2001.

[11] M. Hicks. *Dynamic Software Updating*. PhD thesis, Computer and Information Science Department, the University of Pennsylvania, August 2001.

[12] R. Hinden and S. Deering. IP version 6 addressing architecture. Request for Comments: 2373, July 1998.

[13] R. Hinden, R. Fink, and J. Postel. IPv6 testing address allocation. Request for Comments: 2471, December 1998.

[14] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th USENIX Symposium on Operating System Design and Implementation (OSDI'00)*, pages 197–212, San Diego, CA USA, October 2000. USENIX Association, IEEE TCOS, and ACM SIGOPS.

[15] S. Kent and B. Atkinson. Security architecture for the Internet protocol. Request for Comments: 2401, November 1998.

[16] A. Oram, editor. *Peer-To-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[17] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS)*, pages 49–60. USENIX Association, 2001.

[18] C. Perkins. IP mobility support. Request for Comments: 2002, October 1996.

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

[20] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, Nov. 2001.

[21] P. Srisuresh and K. Egevang. Traditional IP network address translator (traditional NAT). Request for Comments: 3022, January 2001.

[22] I. Stoica, R. Morris, D. Karger, M. Frans Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *ACM SIGCOMM 2001*, pages 149–160, San Diego, CA USA, August 2001. ACM.

[23] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[24] R. van Renesse and K. Birman. Scalable management and data mining using Astrolabe. In *Electronic Proceedings for the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA USA, 2002. http://www.cs.rice.edu/Conferences/IPTPS02/.

[25] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department, 2001.

[26] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide area data dissemination. In *Proceedings of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001)*, Port Jefferson, NY USA, June 2001. ACM.