

# Forming an IPv6-only Core for Today's Internet

Hong Zhang  
Department of Electronic Engineering  
Tsinghua University  
Beijing, 100084, P R China  
neilzh@ns.6test.edu.cn

Maoke Chen  
Network Research Center  
Tsinghua University  
Beijing, 100084, P R China  
cmk@ns.6test.edu.cn

## Abstract

Routing complexity of the core network is crucial to Internet scalability. IPv6 is believed to have better performance in multihoming than conventional IPv4. With the depolyment of native IPv6 backbone networks, a new door is opened to a better Internet, empowered by the combination of a well-aggregated IPv6-only core and the fertile application resources which still remain in IPv4 networks. To achieve this ideal, we propose a prefix-specific address mapping to realize a scenario in which an IPv6 globally routable prefix plays the role of a locator, while IPv4 host address is used as an identifier. To overcome the challenge of identifier/locator mapping lookups, we designed a distributed indexing service system that is self-organized. According to the numerical analysis based on global IPv4 and IPv6 route tables, customers multi-homed to 3 providers via IPv6 can get path diversity higher than those connected to however many providers via IPv4.

## Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Network communications; C.2.5 [Local and Wide-Area Networks]: Internet; C.4 [Performance of Systems]: Design studies

## General Terms

Design, Management, Performance, Experimentation, Standardization

## Keywords

Locator/ID separation, Multi-homing, IPv6, Addressing, Routing

## 1. INTRODUCTION

Scaling problems faced by the Internet routing system are attracting more and more attention from the community. The overloaded role of the IP address as both locator and

identifier is one of the root causes of this scalability problem[10]. Many novel architectures are designed by either keeping global route table unchanged, such as LISP[4], or adding extra information in the current BGP data structure, such as CRIO[14]. These approaches are generic, without special consideration of the protocol version of the Internet infrastructure.

On the other hand, IPv6 networks are widely deployed over the world, and IPv6-only backbones have emerged (such as CERNET2<sup>1</sup> in China and 6WiN<sup>2</sup> in Germany). At least up until now, the IPv6 global route table has been well aggregated. More importantly, the provider-dependent aggregatable (PA) IPv6 multihoming model has proved far superior to the provider-independent (PI) model of IPv4 [5]. On the other hand, current IPv6-only backbones are underutilized since most applications and content resources are still stored in IPv4 end systems. We wonder whether it is possible to re-organize today's Internet so that an IPv4 host address plays the role of identifier, while the IPv6 prefix plays that of locator, combining the good aggregation and multihoming properties of IPv6 and content resources in IPv4 networks.

This combination causes a scenario in which IPv6-only backbones are located in the core with IPv4 networks running as customers. IPv4 customers connected to IPv6 backbones require communication with each other, and with the outside world of the IPv4 Internet. Encapsulation techniques, like manually configured tunnels[7] and Tunnel Broker[6] can be used for IPv4-over-IPv6 communication, but they are not suitable for large-scale deployment, due to per-host level address mapping states maintained in tunnel servers, and bad behavior in causing route detours. Stateless tunneling approaches like 6to4[2], ISATAP[13] and Teredo[9] embed address mapping information into addresses themselves and eliminate state in the tunnel end points. However, they are all designed for IPv6-over-IPv4 tunneling and unsuitable for the case of reverse.

The authors' previous work[3] includes the design of a dedicated device called an "Edge Router", which is empowered with a novel, stateless address-mapping scheme. In this paper, we attempt to use Edge Router as the basic building block to build a new architecture for the IPv6-only Internet core. Here, core providers can be separated from the messing IPv4 global route tables without losing the ability to serve

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.

IPv6'07, August 31, 2007, Kyoto, Japan.

Copyright 2007 ACM 978-1-59593-790-2/07/0008 ...\$5.00.

<sup>1</sup>See <http://www.edu.cn/20060111/3170212.shtml> for an introduction to CNGI-CERNET2.

<sup>2</sup>cf. [http://www.6win.de/6WiN/6WiN\\_Aufbau.php?lang=en](http://www.6win.de/6WiN/6WiN_Aufbau.php?lang=en), the 6WiN Design and Implementation.

today’s Internet applications and users. Minimizing states in the core is a crucial challenge. This challenge can be overcome through a scalable, global mapping lookup service in an overlay network among the core providers. We designed the mapping lookup service in a self-organized fashion, without the need for a top-down deployment paradigm.

The remaining parts of the paper are organized as follows. In Section 2, we give an overview of the architecture components — Edge Router, prefix-specific mapping, and mapping lookup — with highlights on the design considerations. Then, Section 3 focuses on the mechanisms of the self-organized mapping lookup service system. In Section 5, with the hypothesis that the Internet core is formed with current IPv6 providers organized in the proposed architecture, we provide a numerical analysis to show the benefits. Finally, we conclude this paper with brief summary of future works.

## 2. THE ARCHITECTURE

Throughout this paper, the term “core” refers to the set of Internet providers. Combining the advantages of IPv6 global routing and IPv4 application resources, the direct aim of this architecture design is to enable this “core” to serve the whole Internet as it is today, providing not only IPv6 native services, but also IPv4 connectivity.

Three major components — the Edge Router, prefix-specific address mapping, and prefix mapping lookup service — form the basis of the architecture.

### 2.1 Edge Router

Edge Routers (herein as ER) are deployed at boundaries of core domains and their customers. To simplify the discussion, we treat each customer as connected to its provider via a single ER. Theoretically, encapsulation is enough for the requirement of IPv4-over-IPv6 tunneling, but we have designed the Edge Router as a kind of translator, to support IPv4-IPv6 mutual communication as well as combine both tunneling and translation functionalities into a single system. We believe this way encourages much faster adoption of IPv6. Packet translation of ER essentially follows the SIIT scheme[11], but uses the new address mapping rather than IPv4-mapped addresses[8]. ER keeps IPv4 header information with little exception, as SIIT has discussed, and therefore can also be used to form a tunnel.

For convenience, the terms “entry ER” and “entry provider” pertaining to a packet refer to where the packet enters the IPv6-only core, while the term “exit” refers to where the packet exits. For dual stack destination customers, whether the exit ER does translation from IPv6 to IPv4 depends on whether the destination matches a regular IPv6 route or an entry for mapping.

A prototype of the ER translator with current mapping rules (cf. Section 2.2) has been implemented by the authors, as a patched Linux kernel, available at the following URL:

<http://v6s.6test.edu.cn/impl/>

### 2.2 Prefix-specific Address Mapping

Arbitrarily mapping addresses requires the addition of states in boxes converting the packets, while mapping by a specific rule enables the system to have less state, or even no state at all. A special prefix approach is not suitable for large-scale deployment in the Internet with IPv6 core. For instance, if a customer network has IPv4 address 3.3.0.0/16,

then it can be mapped either to ::FFFF:3.3.0.0/112 (as an IPv4-mapped address in [8]), or to 2002:303::/48 (as in 6to4[2]). Obviously, it doesn’t make sense for the customer to negotiate with every potential transit provider to let them accept the route for these more-specific entries. Nonetheless, if such advertisements were permitted and accepted, then the IPv6 global route table would be messed by these mapped entries. Since keeping the global IPv6 route table well-aggregated is very important, we have designed a “prefix-specific mapping” mechanism.

The format for mapped IPv6 addresses is very simple, originating from a set of engineering considerations. First, it is assumed that the provider has a /32 IPv6 prefix. Fixed IPv6 prefixes will simplify the mapping lookup. Having a /32 prefix will not be too rigorous a requirement for IPv6 core providers, according to current IPv6 Address Allocation and Assignment Policy<sup>3</sup>. Second, the provider will allocate a special block of addresses dedicated for IPv4 mapping. A fixed mid-fix “FF” is defined for this purpose. Hence 1/256 of a provider’s address space is reserved for IPv4. This is a moderate and acceptable amount, but a consensus should be made via standardization in order to achieve global-wise deployment. Third, a provider needs to route the mapped prefixes of its customers (which are a more-specific entries) within its own IGP realm. Embedding an IPv4 address at the leftmost bits behind the provider prefix and the mid-fix makes the mapped IPv6 subnetwork prefix as short as possible. Therefore the IPv6 address mapped from IPv4 can be denoted with a colonial IPv6 address format:

< Provider’s /32 prefix>:FF<IPv4 address>::

Basically, the mapping is unique, but for multihoming cases, one IPv4 address block may be mapped to more than one IPv6 prefix. Therefore, we attach a value in [0, 1] to each mapping entry to identify its weight.

### 2.3 Prefix Mapping Lookup Service

Once our architecture is deployed at a large-scale, one ER needs to know the proper IPv6 prefix for any destination, as well as the corresponding weight. There must be a global prefix mapping lookup service deployed within the IPv6-only core, or more exactly speaking, among the providers who are contributing /32 IPv6 prefix(es) for customer services.

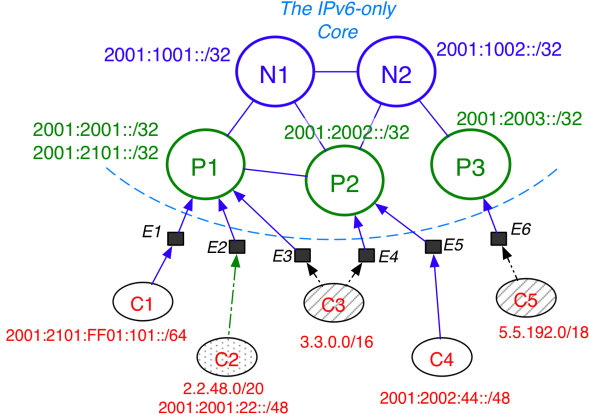
For this purpose, one dedicated server for prefix mapping lookup is set up in each involved provider, which we call an MLS server. Considering the infeasibility of a top-down deployment, we make the MLS servers self-organized into an overlay network for distributed mapping index. Each MLS server keeps a copy of the MLS server list, where one can find out in which MLS server the mapping for a certain IPv4 address block is authoritatively stored. When an ER needs to find a mapping, it will firstly contact the local MLS server which has the same /32 prefix as the ER and a pre-defined interface ID (e.g. ::1:4664) and then it will get either the final answer or a referee to which it can make the next query. Logically, one provider deploys one MLS server for each /32 prefix.

Details of the MLS overlay will be discussed later in Section 3.

<sup>3</sup>The newest version of the document can be found at major RIR’s web site, e.g. <http://www.apnic.net/docs/policy/ipv6-address-policy.html>. See Section 5.1 for allocation policy for providers.

## 2.4 Example

To clarify the working principles of the architecture, an example network is depicted by Fig. 1. Each provider has one or more /32 IPv6 prefixes while customers may have either IPv4, IPv6, or both addresses. Providers involved in customer service are marked with “Px” while others who do only transit service with “Nx”.



**Figure 1: Example of Customers Served by IPv6-only Core**

Edge routers announce mapped IPv6 routes to providers for each customer having IPv4 addresses or IPv6 addresses mapped from early IPv4 blocks. Providers only announce /32 (or less specific) routes to others in the core. The Table 1 lists related route advertisements in comparison with the same topology in IPv4.

Direction	Advertised Route Entry, Weight	Original IPv4
E1 → P1	2001:2101:FF01:101::/64 (1.0)	1.1.1.0/24
E2 → P1	2001:2001:FF02:230::/60 (1.0) 2001:2001:22::/48 (1.0)	2.2.48.0/20
E3 → P1	2001:2001:FF03:300::/56 (0.7)	3.3.0.0/16
E4 → P2	2001:2002:FF03:300::/56 (0.3)	3.3.0.0/16
E5 → P2	2001:2002:44::/48 (1.0)	N/A
E6 → P3	2001:2003:FF05:5c0::/58 (1.0)	5.5.192.0/18
P1 → others	2001:2001::/32 2001:2101::/32	1.1.1.0/24 2.2.48.0/20 3.3.0.0/16
P2 → others	2001:2002::/32	3.3.0.0/16
P3 → others	2001:2003::/32	5.5.192.0/18

**Table 1: Route Advertisement for Example Network**

Customer C1 has an /64 IPv6 subnetwork ID mapped from its previous IPv4 network address 1.1.1.0/24, reflecting the stage that almost all of its users are mostly visiting IPv6 but content services in it are still frequently visited by other IPv4 users. On the other hand, hosts in C1 may possibly need cross-protocol communication to IPv4 world via DNS names, therefore C1’s caching name server should be extended to automatically do appropriate A-to-AAAA type conversion. The name server must retrieve the mapping lookup service as well, in a manner identical to that of ER.

For contrast, customer C4 has only a natively assigned IPv6 prefix, possibly due to exhaustion of IPv4 address space. Edge Router E5 is a completely regular IPv6 router that does not help users in C4 to have accesses to IPv4-only

networks, unless NAT-PT or similar stateful facilities are utilized. C2 is dual-stack, while C3 and C5 run IPv4 only.

## 2.5 Path Selection in Multihoming Environment

Because our idea is motivated from the demand of scaling routing in multihoming environment, it is worthy to have this separated subsection to discuss on the case of multihoming in the example network.

Customer C3 in Fig. 1 connects via E3 and E4 to providers P1 and P2 respectively. With the help of the attached policy-based weight value, this IPv4 network can gain the same path diversity as if it were an IPv6 customer, connected to providers with a provider-dependent aggregatable (PA) prefix[5]. For instance, when host 5.5.192.7 sends a packet to 3.3.8.8, the packet arrives at Edge Router E6,

which finds that network 3.3.0.0/16 can be mapped into 2001:2001:FF03:300::/56 with weight 0.7, or 2001:2002:FF03:300::/56 with weight 0.3.

Then E6 can translate the destination address to

2001:2001:FF03:308:800:: with higher probability while to 2001:2002:FF03:308:800:: less probably. Hence the inbound traffic load for C3 is tuned.

However, randomly flapping prefixes and path changes for inbound packets will cause packet reordering, which is commonly considered harmful to TCP performance [1]. The criteria and algorithm for weight assignment and usage is a work-in-progress.

Multiple address mappings, also provide customers with network failure tolerance. For example, if the path via E3 to C3 is broken, then an ICMPv6 “destination unreachable” message will arrive at E6 (actually E6 is going to translate the message to the corresponding ICMP version and send it further to the source) notifying it of the failure, E6 can select the another mapping, and subsequent packets will be translated with provider P2’s prefix and routed via E4 arrive at the destination.

## 3. SELF-ORGANIZED OVERLAY FOR PREFIX MAPPING LOOKUP

Similar to LISP[4] variants, it is possible to implement the mapping lookup with extended ICMPv6 messages, or a DNS or DHT system. Considering the demands of load balancing, the lowest impact to IPv6 global routing, and ease of incremental deployment, we design the prefix mapping lookup service as a self-organized distributed indexing system.

The design based on a simple intuition that, a new provider with /32 IPv6 prefix involved, new MLS servers will join the system and share the load of maintenance for mappings in the entire IPv6 address space. New MLS servers always take half of the load from the most heavily loaded server, with the load of queries balanced adaptively. Relationship of load transfer makes MLS servers organized into a tree-like overlay network. We denote the tree as graph  $G(V, E)$ . Each MLS server  $S_i$  has three basic attributes: its IPv6 address, also denoted with  $S_i$ ; the prefix  $X_i$  it take care of mapping for; and its neighborhood,

$$A_i \triangleq \{(X_j, S_j) \mid (i, j) \in E \text{ or } j = i\}$$

### 3.1 Mapping Lookup

```

FindPref( $x, i$ ) {
  If ( $x \wedge X_i == X_i$ ) Then {
    If ( $S_i$  has matched mapping entries for  $x$ , i.e. the  $p(x)$ )
  Then
    Return all the entries  $\{p(x), w\}$ ;
    Else Return NULL;
  }
  Find  $j \in A_i$ , so that  $X_j$  mostly matches  $x$ 
  Return FindPref( $x, j$ );
}

```

**Figure 2: Algorithm for Retrieving MLS Overlay**

The mapping lookup problem is thus: given the target IPv4 address  $x$ , find the location in which  $x$ 's prefix mapping  $p(x)$  is stored. The algorithm, shown in Fig. 2, starts from a randomly selected MLS server  $S_i$ , which is handling the mapping for IPv4 prefix  $X_i$ , and recursively applies the longest match technique to get the target. The symbol " $\wedge$ " represents the operator of bitwise AND.

The total number of MLS servers is expected not very big. Taking account of this, we may cache a complete list of all MLS servers in each of them. The cache can be updated whenever the selected  $S_i$  retrieves a remote one, not in the set  $A_i$  for the time being.

The algorithm for new mapping entry registration is quite similar to the lookup. The only concern is that the object, a mapping  $(X, p(X))$  for an IPv4 network prefix  $X$ , might (though not so likely) be a quite large block, even larger than that maintained by the target MLS server. In this case, more than one MLS server will be selected to store the same entry.

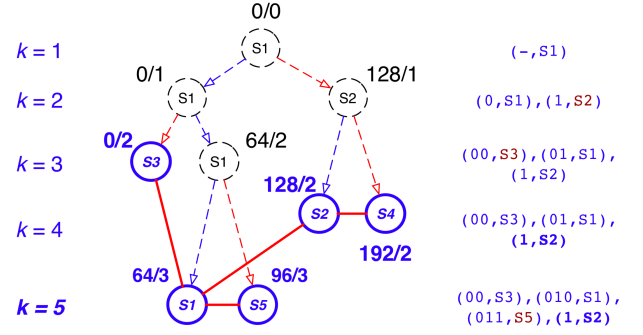
The mapping withdraw algorithm is identical to the registration procedure, except that the final step after finding the target is a removal instead of an addition.

### 3.2 Growth of MLS Overlay

As illustrated by the example in Fig. 3, the whole IPv4 address space is divided once every new MLS server joins the overlay. The CIDR notation is applied to mark every MLS server's work range  $X_i$ . Graph  $G$  is drawn using hard stroke lines. Dashed arrows represent load-splitting events. Let  $i$  represent the sequence number of MLS server  $S_i$  and, for convenience, treat  $i$  as the time that  $S_i$  joins the MLS overlay.

At the time  $k+1$ , the newcomer  $S_{k+1}$  will randomly select an MLS server, say  $S_{i_0}$  among  $S_1, S_2, \dots, S_k$ , start the join procedure, and it iteratively search along the tree  $G$  from  $S_{i_0}$  until it finds out who is currently maintaining the mapping for the biggest block, which we suppose to be  $S_{i^*}$ . Let  $X_{i^*}^{(k)}$  represent the block handled by  $S_{i^*}$  at the time  $k$ . The newcomer takes either upper or lower half of its current maintained block  $X_{i^*}^{(k)}$  into its own management and leaves the other half in the original  $S_{i^*}$ . This way, the prefix length of the updated  $X_{i^*}^{(k+1)}$  and  $X_{k+1}^{(k+1)}$  is identical, and equal to the length of  $X_{i^*}^{(k)}$  minus 1.

$S_1$ 's neighbor list,  $A_1$ , is listed on the right side of the figure. Obviously, the neighborhood is a *historical* accumulation of the load assignment relationship. Note, at  $k=4$ ,  $S_2$  splits its block, but only involves  $S_4$  and itself.  $S_1$  doesn't know the event for the time being and still stores the original neighbor list. Such a historical neighborhood quite works for the lookup algorithm. For instance, if a provider would



**Figure 3: Example of Splitting IPv4 Address Space to MLS Servers ( $S_1$ 's historical neighbor set is listed on the right)**

like to register a new mapping entry:  $193.245.244.0/20 \mapsto 2001:254::/32$ , and randomly selects  $S_1$  at the start of the registration, it will find  $S_2$  takes care of prefix  $128.0.0.0/1$  with longest match and retrieves  $S_2$ . However, currently  $S_2$  has split its original block and knows  $S_4$  is now responsible for the lower half set  $192.0.0.0/2$ , which matches the entry to be registered. And finally the provider will register this mapping at the MLS server  $S_4$ .

After such a lookup process,  $S_1$  obtains the known current status of  $S_2$  and  $S_5$ . It can update its neighbor list and involve  $S_5$ . Ideally, all MLS servers will build a full mesh overlay, i.e. a complete graph  $G$ , in this way. We suggest that they be allowed to obtain all information learned, rather than actively or periodically solicited or advertised in flooding. It is worth doing so if the total number of MLS servers is not very big.

Due to limited room in this paper, we have omitted a formal description of the algorithm for newcomer joining.

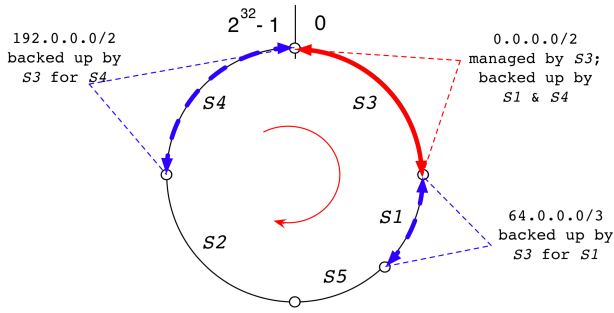
### 3.3 Redundancy

Up to now, one mapping entry for a certain IPv4 network ID is stored only once in the authoritative MLS server defined above. For redundancy considerations, we introduce a backup mechanism similar to the "leaf set" in the Pastry approach[12]. If we put the MLS servers into a circled space according to their managed address ranges, a new neighborhood appears as shown in Fig. 4. MLS server  $S_3$  maintains a backup copy for its neighbors in the circulated IPv4 address space. Meanwhile,  $S_4$  and  $S_1$ , as  $S_3$ 's neighbors, also back up  $S_3$ 's mapping entries in return.

Thus every MLS server has two backup servers. If we involve intermediate neighbors for backup, we can get more redundancy, but too much is not necessary. For lookup and registration algorithms, it is necessary to traverse the assignment tree  $G$  and find the partition of circulated IPv4 address space in advance. For instance, if an ER gets nothing from  $S_3$  or it doesn't know  $S_3$ 's address at all, it can find an uncovered range on the circle and send a query to the neighboring ranges' MLS servers in order to get the backup information.

## 4. PERFORMANCE CONSIDERATION

Making a retrieval in an overlay network before sending a packet will impact the packet delivery performance signifi-



**Figure 4: Address Ranges Manage and Backed Up by  $S_3$**

cantly. Fortunately, we have two potential weapons to fight this issue: a local cache mechanism in ER and MLS overlay relaying.

#### *Edge router local cache: .*

Since mapping entries are quite stable, we can avoid too-frequent mapping lookups by always caching retrieved entries. However, storing all mappings is not scalable because there are potentially as many as in the IPv4 global FIB. Considering that small number of destination addresses occupy large portion of traffics, the cache size can be significantly shrunk if only most frequently used mapping entries are saved. Furthermore, we can optimize the cache so that the highest ranked entries can be found most rapidly.

In order to check the feasibility of the mechanism where there only small amount of mapping entries are cached, we've made an analysis based on a data set of destination addresses in traffics that CERNET users visit global Internet, which was collected within 24 hours (on the day Apr. 1st) at one of the CERNET international links. In the result, if we have a local cache with 3000 entries, 80.2% destination addresses will be hit by the cache and, if we have the size up to 30000, then 99.5% can be found in the cache, without need of resolution in the MLS overlay.

Caching is suggested to be implemented together with the forwarding table, to benefit from the constantly fertilized route table optimization techniques.

#### *MLS overlay relaying: .*

Cache size is quite limited. Although only a fraction of destination addresses are not cached, once non-cached destination addresses appear in traffic, they will cause new mapping queries, which takes seconds to complete. On the other hand, when an ER is booted up, the cache will be steadily established over the first several or tens of seconds. In both cases, routers cannot provide enough buffers for burst traffic in durations of *seconds*. Therefore, it is important for the ER to deliver packets before the proper mapping entry is retrieved.

This idea for fast delivery is based on the fact that MLS overlay actually has knowledge of the provider whose MLS server  $S_{i^*}$  maintains the mapping  $p(x)$  for the final destination  $x$ . Because the ER can easily get the list of all MLS servers from its local MLS server, and the MLS server list is moderate in size, the ER can use the longest match to find  $S_{i^*}$ , its IPv6 address. Using  $S_{i^*}$ 's prefix  $p^*$  instead of  $p(x)$

to do the translation and deliver the packet first, ER gains a time slot to retrieve  $S_{i^*}$  remotely in order to find out the  $p(x)$ .

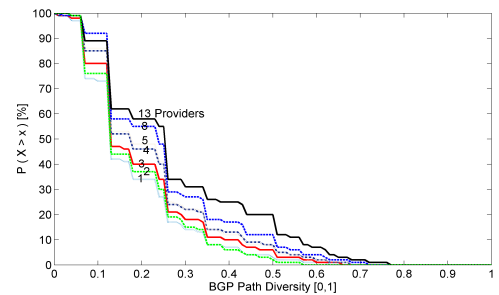
MLS overlay is typically used for prefix mapping lookup. Here, we utilize it to provide relays for packet delivery. Carrying traffic in the overlay network will cause some detours, which will be one of the drawbacks of our solution. Fortunately, once an ER enters equilibrium, the amount of mapping lookup will be quite small.

## 5. NUMERICAL ANALYSIS

What if today's customers who have connected to IPv6 providers cease being connected to these providers' IPv4 backbones? It is obviously expectable that a provider can stop its IPv4 backbone when all of its customers have done this. On the other hand, customers will benefit from the architecture in multihoming. In this section, we use current IPv4 and IPv6 AS-level topology data to make a theoretical deployment. Original data from the global IPv6 table is merged from AS1221 and AS6447's BGP tables<sup>4</sup>, while IPv4 AS relationships and AS adjacencies are obtained from CAIDA<sup>5</sup>.

There are 733 IPv6 domains, and among them 305 ASes currently provide IPv4 services as well. 3,991 IPv4 customers are connected to the Internet via these IPv6-enabled providers. If these providers stop their IPv4 backbones, they and their transit providers among the IPv6 domains form an IPv6-only core. Now imagine that they keep providing services to those IPv4 customers via Edge Routers.

In order to make a quantified evaluation for the advantages of the architecture, we calculate the path diversity as defined by De Launois [5]. Calculation with our approach applied is made for the network consisting of those 3,991 stub ASes (customers) and their 305 providers, in comparison between the theoretical deployment in which all involved providers are IPv6 only, with the reality of the current Internet in which the same providers are still running IPv4. The calculation results are shown in Fig. 6 and Fig. 5, respectively. The presentation method is identical to [5]. The curves are classified by number of providers of the destination customer, and the number is attached to each curve.



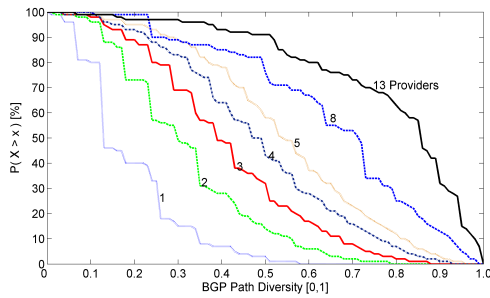
**Figure 5: Path Diversity for Customers Whose Providers Are Still IPv4, with PI multi-homing**

The results of calculation show that for those 3,991 customers, single-homed AS gains little in path diversity when

<sup>4</sup>Downloaded from <http://bgp.potaroo.net/>

<sup>5</sup>Downloaded from <http://as-rank.caida.org> and <http://sk-aslinks.caida.org/>, respectively.





**Figure 6: Path Diversity When Providers Run IPv6 Only, with PA multi-homing**

their providers migrate from IPv4 to IPv6-only. This is consistent to the intuition that current interconnectivity among IPv6 providers is far less rich than that among IPv4 providers. However, for dual-homed customers, it is shown that less than 40% among them actually have path diversity better than 0.2 in IPv4, while the percentage raises to more than 70% when the providers migrate to IPv6-only. This shows the advantage of PA multi-homing supported by IPv6. In other words, even today, when the IPv6 deployment is in the early stages, forming IPv6-only Internet core is quite attractive in term of providing better path diversity to customer networks.

## 6. CONCLUSIONS

We have proposed a new architecture for the Internet in which the IPv6 prefix plays the role of locator, while the IPv4 host address plays that of identifier. Prefix-specific address mapping is the key to a scalable solution that maintains aggregation in the IPv6 global route table. However, since finding the proper prefix mapping for a given IPv4 destination is a big challenge for the architecture, we propose a novel, self-organized overlay network to deal with the issue of prefix mapping lookup. To the authors' knowledge, it is an early concrete effort to utilize a self-organized overlay to solve routing-related problems.

Analysis shows a positive result that, if today's dual-stack providers cease IPv4 service, will give their customers much better path diversity with ever-existing multihoming. At the same time, the performance degradation caused by mapping is moderate and acceptable. These results may be a very encouraging for the customer adoption of IPv6.

Several issues related to the MLS overlay need to be further investigated. The mechanisms and algorithms for self-organization need to be proved and evaluated, and the security of the mapping registration and lookup needs to be carefully explored. Furthermore, the manner in which the weight for multi-homed mapping is utilized is also a very interesting problem. Future works will focus on these topics.

## Acknowledgments

We would like to thank Bruno Quoitin from Université Catholique de Louvain, who gave us comprehensive comments on our previous related work and, through deep discussion, inspired the authors very much. We are also grateful to Mr. Ang Li from University of California at Irvine, who made the first prototype for the Edge Router, when he was an under-

graduate in Tsinghua University. We thank Prof. Jianping Wu and Prof. Xing Li, proposed the strategic concept for building a pure IPv6 backbone, and the method of applying stateless address mapping to IPv4-IPv6 mutual communication. Finally, we'd like to thank Mr. Jed Schmidt for his careful proofreading for the paper.

## 7. REFERENCES

- [1] J. C. R. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Trans. Netw.*, 7(6):789–798, 1999.
- [2] B. Carpenter and K. Moore. Connection of IPv6 domains via IPv4 clouds. RFC 3056, Internet Engineering Task Force, Feb. 2001.
- [3] M. Chen, X. Li, A. Li, and Y. Cui. Forwarding IPv4 traffics in pure IPv6 backbone with stateless address mapping. In *IEEE/IFIP Network Operations & Management Symposium (NOMS 2006)*, pages 260–270, Vancouver, Canada, April 2006.
- [4] D. O. D. Farinacci, V. Fuller. Locator/ID Separation Protocol (LISP). draft-farinacci-lisp-00.txt, Jan. 2007. Work in progress.
- [5] C. de Launois, B. Quoitin, and O. Bonaventure. Leveraging network performance with IPv6 multihoming and multiple provider-dependent aggregatable prefixes. *Computer Networks*, 50(8):1145–1157, 2006.
- [6] A. Durand, P. Fasano, I. Guardini, and D. Lento. IPv6 tunnel broker. RFC 3053, Internet Engineering Task Force, Jan. 2001.
- [7] R. Gilligan and E. Nordmark. Transition mechanisms for IPv6 hosts and routers. RFC 4213, Internet Engineering Task Force, Oct. 2005.
- [8] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), Feb. 2006.
- [9] C. Huitema. Teredo: Tunneling IPv6 over UDP through network address translations (nats). RFC 4380, Internet Engineering Task Force, Feb. 2006.
- [10] D. Meyer, L. Zhang, and e. K. Fall. Report from the IAB Workshop on Routing and Addressing. draft-iab-raws-report-01.txt, Feb. 2007. Work in progress.
- [11] E. Nordmark. Stateless IP/ICMP translation algorithm (SIIT). RFC 2765, Internet Engineering Task Force, Feb. 2000.
- [12] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Nov. 2001.
- [13] F. L. Templin, T. Gleeson, M. Talwar, and D. Thaler. Intra-site automatic tunnel addressing protocol (isatap). RFC 4214, Internet Engineering Task Force, Oct. 2005.
- [14] X. Zhang, P. Francis, J. Wang, and K. Yoshida. Scaling ip routing with the core router-integrated overlay. In *Network Protocols, 2006. ICNP '06. Proceedings of the 2006 14th IEEE International Conference on*, pages 147–156, Fess parker's Doubletree, Santa Barbara, Ca, USA, Nov. 2006.