

Forwarding IPv4 Traffics in Pure IPv6 Backbone with Stateless Address Mapping

Maoke Chen, Xing Li

CERNET, Network Research Center
Tsinghua University
Beijing, P R China
{mk, xing}@cernet.edu.cn

Ang Li

Dept. of Electronic Engineering
Tsinghua University
Beijing, P R China
auron@ns.6test.edu.cn

Yong Cui

Dept. of Computer Science
Tsinghua University
Beijing, P R China
cy@csnet1.cs.tsinghua.edu.cn

Abstract—As the IPv6 networks rapidly deployed, the transition problem has been changed. Because the transition of applications is a steady process with a far longer duration in comparison to the transition of infrastructure, providers cannot give up the duty of serving IPv4 traffics for the time being. On the other hand, operating a dual-stack backbone is highly costing for large-scale deployment of new networks. In this paper, a new technique of forwarding IPv4 traffics in IPv6-only backbone is developed. Rather than the ever-existing tunneling approaches, the proposed one is automatic and stateless for end nodes, without explicit tunneling. The route entries for delivering IPv4 traffics are well aggregated on the borders of IPv6 autonomous systems. This makes the technique suitable for large-scale deployment in an inter-domain networking environment.

Keywords—IPv6; transition; automatic tunneling; routing

I. INTRODUCTION

Transition is one of the most significant problems in deploying an emerging network technology. Since the IP plays the role of engine in the Internet architecture, migrating networks from IPv4 to IPv6 is very difficult and complicated. In the past decade after the IPv6 protocol specification was defined, a series of techniques were developed to solve the issues of transition, including both translation and tunneling approaches [1-8]. Among them, Tunnel Broker [6] and 6to4 [7,8] are highlighted in the real deployment because of their ability of handling tunnel requests automatically.

However, these do not terminate all the obstacles of transition. Since native IPv6 networks are rapidly deployed over the world, the problem is changing unconsciously. In the early stage of IPv6 deployment, there are IPv6 islands dispersed among the ocean of IPv4 networks and therefore connectivity of IPv6 clouds is the major issue. Currently, native IPv6 infrastructures are established in large-scale and the connectivity becomes less serious. On the other hand, however, the applications and resources are still concentrated in the IPv4 networks. Although the IPv6 programming interface has been well defined since several years ago [9, 10], the application porting process are far slow than what is expected. Theoretically, porting an application from IPv4 to IPv6-enabled version is not a big burden. But this is often untrue in practice. Besides the socket interface should be modified, data structures and message formats, once involving IP addresses, should be reshaped as well. This often challenges the compatibility of an application protocol and its implementation

details. Even if programmers are fully encouraged to do the porting, there is still an obstacle at migrating the running services. Anyway, at this stage, interoperability of applications in dual-protocol environment has, replacing connectivity, become the sharpest issue of the IPv4-IPv6 transition.

Dual-stack is a feasible solution for application transition but not suitable for large-scale deployment of new networks, esp. in developing countries where the scale of next generation Internet might be as tens or hundreds times as that of current. Firstly, as the IPv4 address space has been half and more exhausted, it is almost impossible to get such a huge amount of address resources for new deployments. E.g. for the next generation Internet project in China, if each citizen would hold one permanent global address in average, then about 73 blocks of class A are needed. Secondly, a single-stack backbone is more economical than a dual-stack one. Single-stack routers consume less processing power and memory, and single-stack networks are less costing in operation and maintenance. Thirdly, the funding bodies, e.g. government, ministries and etc., sees deployment of IPv6 as a national strategy and single-stack obviously much more encourages the transition from traditional IPv4 to new IPv6 Internet.

Therefore, carrying IPv4 traffics in pure IPv6 backbone is a significant problem of current stage of transition. The generic encapsulation for IPv6 stack has been defined [11]. Tunnel broker approach can be applied for the IPv4-over-IPv6 case without any difficulty to accelerate the effect of access service provision. However, there is not a scalable solution as is stateless to end-systems, just like the 6to4 for IPv6-over-IPv4 case. Furthermore, if not all the tunnel brokers communicate each other and exchange the route information, they may cause IPv4 routing non-optimized and cause a great waste of network resources. However, meshed tunneling among the brokers is not scalable too. Lack of a scalable approach hampers the deployment of IPv6-only backbones.

In this paper, a new technique of forwarding IPv4 traffics in pure IPv6 backbone is proposed. In order to provide a transparent environment for users to run their IPv4 applications and to have accesses to the existing IPv4 content services, IPv4 packets are mapped into IPv6 (or vice versa) automatically at the edge of the IPv6 backbone, with the help of BGP routing tables. The rest parts of the paper are organized as follows. Section II discusses the functional requirements to the architecture design. Section III illustrates the basic design, which is suitable for transition in an intra-domain environment.

The basic design is just a small step because it does not much more than what a full-meshed tunnel server group has ever done, but it is an extensible foundation. In section IV, some critical issues on scalability and performance of the approach are analyzed in order to extend the functionality of the design. A case for migration to native IPv6 applications is presented in section V. And finally, the paper is concluded with remarks on unsolved problems.

II. FUNCTIONAL REQUIREMENTS

Here, the functional requirements of the new transition architecture are enumerated in order that the concepts in the design are presented. In the following discussion, we will pay attention to the way of IPv6-over-IPv4 techniques meeting each of the requirements, and try to learn experiences or lessons from their successes and failures.

A. Transparency

A transition scheme that requires end users to do much will not be successful. The best way should make application programs in end nodes unaware of the existence of transition, and even make operating systems of the end nodes unaware of this too. In other words, the transition system is transparent to the end nodes where users run their legacy IPv4 applications. Furthermore, we hope the transparency is available in a scope as large as possible. Tunnel approaches of IPv6-to-IPv4, such as Tunnel Broker and 6to4, do support the transparency for any scope theoretically, but, in practice, they are successfully applied for end users only.

To achieve the transparency, it is necessary to make all the activities of transition occurring at the boundaries between the IPv6 backbone and the subnets where IPv4 traffics happen inevitably.

B. Lightweight

If the transition scheme introduces too many states in the networks, then it must be unscalable. Actually, using the tunnel interfaces to implement the encapsulation in IPv6 packets [11] can solve the problem of carrying IPv4 traffics in IPv6 networks. However, this approach needs manual configuration and tunnel links should be established before the communication among IPv4 end systems.

The 6to4 approach presents a good paradigm of designing a lightweight transition scheme, for the case of IPv6-over-IPv4. Actually, the 6to4 is stateless, profiting from embedding the IPv4 addresses into IPv6 prefixes and the specified address head "2002::" [7]. It is inspired to make an automatic addressing rule in order that the requirement of lightweight is achieved.

C. Extensibility in Scope

In the 6to4 address mapping, an IPv4 address can be easily embedded into an IPv6 prefix which gives hints about encapsulation and tunneling to the stack. In IPv4-over-IPv6 case, one cannot embed an IPv6 address to an IPv4 one to provide the same hints since the IPv4 address is predetermined. However, for an intra-domain environment, a well-known

common IPv6 prefix can be applied before the IPv4 address and the formed IPv6 subnet identifier can be routed within the domain.

We hope the proposed scheme working not only in intra-domain scope but also in inter-domain environment. To meet this requirement, a prefix discovery methodology should be implemented. It decides which IPv6 prefix would be appended prior to an IPv4 address or an IPv4 subnet identifier. That is, a prefix discovery mechanism is indispensable.

D. Moderate Impact to Performance

Previous theoretical and simulation works have indicated that tunneling often significantly degrades the performance of the infrastructure [12, 13]. For IPv6-over-IPv4 cases, users blindly select tunnel broker services to get their IPv6 connections but what they have selected might be very far in IPv4 hop metric. The 6to4 technique makes a logically full-mesh potential topology among all the IPv4 addresses. However, due to the anycast-based 6to4 relays [8], there is also the problem of blind tunneling. Traffics from a 6to4 end point might be tunneled by a relay far away and then return to the native IPv6 network near to the sender. This might cause a great waste of IPv4 network resources.

This waste might be acceptable to the community at the stage where IPv6 traffics are very slow. But this is not the case for current and future. Therefore, the IPv4-over-IPv6 transition should pay much attention to the performance issue and try to find a way of synergy among the transition points. Routing information is much helpful for achieving this requirement.

E. Migration

The word "transition" means a process of migration from native IPv4 network and IPv4 application to mixed things and finally to native and pure IPv6. Therefore, a good transition technique does not only solve the problem of current stage but also conquers the difficulty of every stage in migrating to the completely native IPv6 Internet.

To the sense of address migration, the Tunnel Broker approach is superior to the 6to4. Because 6to4's /48 prefixes cannot be aggregated into regular IPv6 prefixes and it might be renumbered when native IPv6 is deployed. For the case of IPv4-over-IPv6, if special IPv6 address blocks are employed, they should be merged into the aggregators at the boundary of each autonomous system (or briefly, AS later).

III. THE BASIC DESIGN

To meet the requirements above, we design the architecture in such a way: firstly, a founding element – the IPv4-IPv6 edge router is defined to enable IPv4-over-IPv6 transition in an intra-domain environment, and then the design is extended according to further analyses, focusing on the issues related to prefix discovery.

Through out this paper, it is assumed that an Internet service provider who would like to apply this approach holds at least one IPv6 prefix not smaller than /32. A part of a /32 prefix is utilized to do the address mapping.

Fig. 1 presents an overview of the basic scheme that is designed: an autonomous system has a pure IPv6 backbone, which is running any of the intra-domain routing protocols. The autonomous system holds one or more IPv6 prefixes. Several IPv4 subnets (maybe they are dual-stack) are attached and each of them may contain one or more IPv4 prefixes. With a publicly agreed addressing scheme, the routers at the edge between IPv4 subnet and IPv6 backbone can perform proper routing and header processing. Arrows in the figure show the detail of routing information announcements.

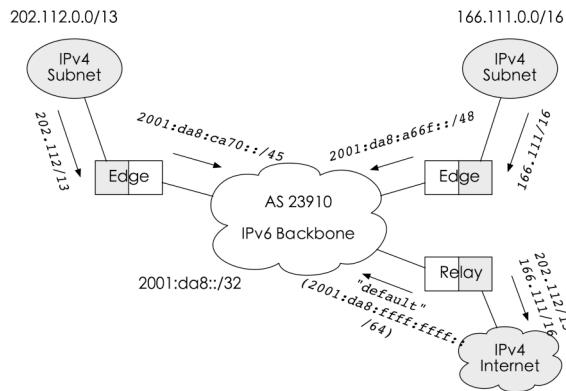


Figure 1. Intra-domain IPv4-over-IPv6 transition

A. Address Mapping

It is simple to map an IPv4 prefix into an IPv6 one by append the IPv4 network id to the IPv6 aggregation of the autonomous system. The length of the generated prefix is the sum of the length of IPv6 aggregation and that of the IPv4 network identifier.

This rule of addressing is deployed without difficulty because the address assignment is fully controlled by the autonomous system itself, which is aware of all the IPv4 prefixes under its administration. On the other side, assigning prefixes to native IPv6 networks should avoid covering these blocks generated with IPv4 network identifiers. It's equally to say that the IPv6 aggregation of the autonomous domain may contain all the IPv4 blocks belonging to this domain.

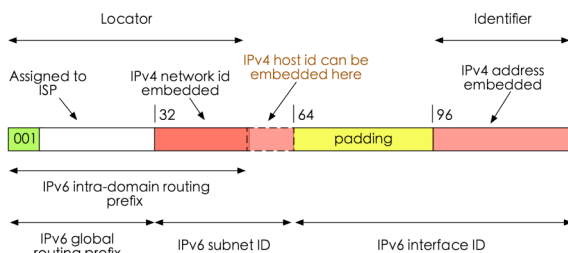


Figure 2. Intra-domain IPv4-over-IPv6 transition

With the address-mapping rule commonly known within the autonomous domain, IPv4 source and destination addresses

in a packet can be translated into their IPv6 version without any explicitly stored states, as described in Fig. 2.

Tab. I shows some examples for the address mapping illustrated by Fig. 1.

TABLE I. SOME ADDRESS MAPPING EXAMPLES

IPv4 prefix/address	IPv6 prefix/address
202.112.0.0/13	2001:da8:ca70::/45
166.111.0.0/16	2001:da8:a66f::/48
166.111.172.1	2001:da8:a66f:ac01::a66f:ac01

It is important to emphasize that only the IPv4 addresses held by the AS can be mapped, or the mapping would run out the most space of an IPv6 /32 block. Therefore a special prefix for IPv4 default route must be defined, representing all the other IPv4 networks outside of the AS. The special mapped prefix could be defined with, for instance, "0xFFFFFFFF" at its second word of 32 bits. However, in such a case, the original IPv4 address is lost in the first 64 bits and then we embed the complete IPv4 address at the end of its mapped IPv6 version. In order that we have a common convention, the last 32 bits of a mapped address represents the original IPv4 address, while the second 32bit-word in it – the route entity.

Thus the locator and identifier for IPv4 node are well separated in the IPv6 world. The IPv4 host id can be embedded as well (this is convenient for the implementation of header mapping) but it is not necessary.

In Fig. 1, as an example, address 61.149.249.44 should be mapped to 2001:da8:ffff:ffff::3d95:f92c in IPv6, because it is not held by AS23910.

B. Routing

1) Routing for IPv4 subnets

It is unavoidable to inject the prefixes mapped with the IPv4 network identifiers into the backbone's IPv6 routing table, or the routers cannot find where the other side of the peer is connected. Within a single autonomous system, IGP is carrying necessary information for routing the IPv4 over IPv6.

If the IPv4 subnets are all isolated, then it's enough to have their routing information in the IPv6 backbone with the address mapping and the route injection. At the early stage of transition, however, there might be a parallel routing infrastructure for the IPv4 networks, which is also connected with the subnets. In this case, the backbone should tell each IPv4 subnet that it could achieve other subnets, in order that the integrity of IPv4 routing is well maintained. This means the route mapping should be bi-directional.

2) Relaying for IPv4 Internet out of the AS

For IPv4 networks outside the autonomous domain, there are only a special IPv6 prefixes mapped within the aggregation, i.e. that one with "0xFFFFFFFF" at second 32bit-word. They are routed with default gateways, which play the role of IPv6-to-IPv4 relays.

A relay is a special edge router. If there is not any relay deployed, then the connectivity of IPv4 subnets to the IPv4 Internet is limited by what the AS holds in the IPv4 address space. Therefore, the basic scheme proposed in this section is only suitable for intra-AS cases.

C. Edge Router

The routers at the edges of IPv4 subnets and the IPv6 backbone perform the functionality of mapping addresses for packets and translating routing information between IPv6 and IPv4.

An edge router is a dual-stack node, extended with the ability of translating routing messages and modifying packet IP versions between IPv6 and IPv4.

1) Route mapping

For the IPv6 backbone, the edge router plays the role of an access gateway that provides connectivity for a certain IPv6 block, by announcing a prefix with certain length. The backbone does not and cannot care whether such an access router connects a real IPv6 network or an IPv4 network in truth. On the other hand, the IPv4 subnet that this edge router connects receives IPv4 routing messages from it, without knowing whether the routing messages are actually learned from the IPv6 backbone routers or not. The edge router is permanently mapping the IPv4 forwarding table into the IPv6 one and vice-versa.

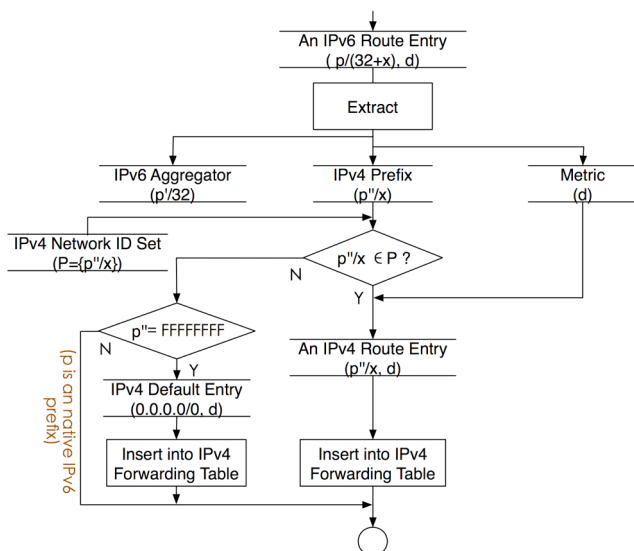


Figure 3. The flow chart of mapping IPv6 entry into IPv4 route table

Similar to the addressing scheme, it is remarkable that the mapping should be done only for the IPv4 address blocks that are held by the autonomous system. Because the aggregation contains IPv6 blocks, among which there are many entries related to native IPv6 networks. If these prefixes for native IPv6 sections were translated at the edge routers and then the translated IPv4 prefix were announced into the IPv4 route table, then the IPv4 routing would be messed. Therefore, each

edge should know a complete list of IPv4 network identifiers, which really belong to the autonomous domain. Fig. 3 describes the process of the route mapping from IPv6 to IPv4.

The reverse mapping is almost the same but all the IPv4 networks outside the AS should be mapped to a special prefix, i.e. by appending "0xFFFFFFFF" to the IPv6 prefix, in our specification. Note that the metric of an IPv4 route entry is mapped into the IPv6 backbone route entry without change. However, the semantics of hop metric will be influenced with the way of packet header manipulation.

2) Packet forwarding

An edge router manipulates the IP headers for the packets passing through it from IPv6 to IPv4 or vice versa. There are two possible ways of header manipulation: encapsulation and translation. The former one is like what a tunnel end point does while the latter is similar to what a NAT-PT gateway does.

In the encapsulation paradigm, the IPv6 network is treated as a link of the IPv4. Therefore, passing through the IPv6 backbone is treated as a hop in the view of an IPv4 packet. Here we have encountered a problem with the route metric. As is previously mentioned, the metric information in an IPv4 route entry can be mapped into its IPv6 counterpart. The metric, if it is measured in hop count, is increased normally in the IPv6 routing process. However, when an encapsulated packet comes out of the IPv6 backbone and returns to the IPv4 realm, only 1 is decreased in the TTL field of the IPv4 header. This is the normal case in any tunnel approaches, but makes the metric mapping from IPv4 to IPv6 routes is meaningless.

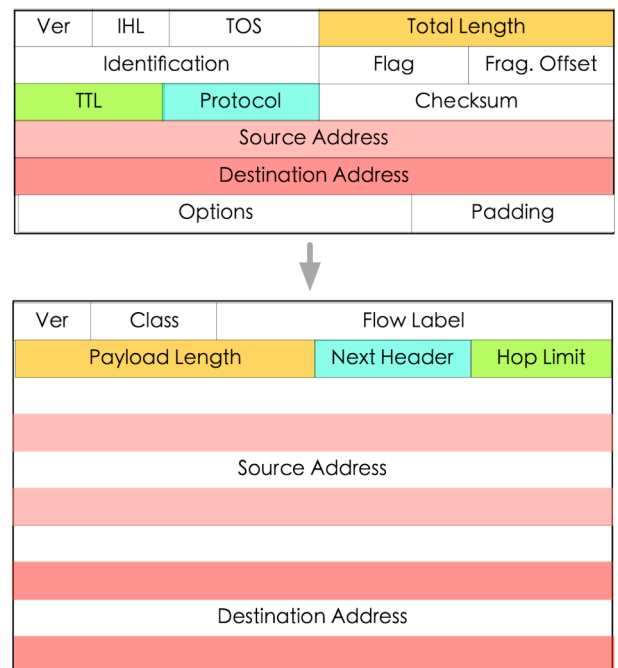


Figure 4. Mapping the header from IPv4 to IPv6

To eliminate the semantic inconsistency between TTL decrement and the route metric, either we have all the IPv6 backbone routers aware of whether a prefix is mapped from an IPv4 network id, or we establish an explicit tunnel between

each pair of edge routers. The former solution is almost impossible, making the IPv4 routing not transparent to the IPv6 backbone routing, while the latter one is not scalable when the edge routers are changed. Full-meshed tunneling does not support steady migration process that the edges of IPv4 and IPv6 are farther and farther from the backbone and get nearer to the end nodes.

For the simplicity, the service type and options are cast away by the translation. They are rarely used in today's Internet.

For the fragment information carried by the fields of identification, flags and offset, it is reasonable to have them mapped into IPv6 fragment option header without any loss. However, in practice, only few (far less than 1% in average) datagrams are fragmented in the real Internet [16], we just omit fragment-related fields to get simplicity and less decrement in path MTU – the fragment option header takes more 8 bytes in IPv6 datagram. Omitting fragment information makes large UDP datagram unable to be integrally transferred. Fortunately, however, most real UDP-based protocols such as DNS and RIP have limited the size of message carried by UDP quite small, e.g. not more than 512 bytes. For TCP-based applications, this is not a problem, because TCP utilizes path MTU to determine its maximum segment size (MSS). Anyway, it won't be difficult to translate fragment-related fields into IPv6 fragment option header, if it is thought really necessary.

Whether the fragment information is kept or not, path MTU is a considerable issue, because IPv6 header is longer than IPv4 one. The path MTU discovery mechanism doesn't work unless the ICMP/ICMPv6 translation [3] is added into the design. This is trivial when the edge routers are deployed in backbone where link MTU is large enough. Therefore, we don't include it in the basic design.

Note that either the IPv4 source or the destination address is sometimes copied twice at the translation. This conforms to the addressing scheme for the design. The first copy plays the role of locator while the second - the role of identifier. The two copies are identical if and only if the IPv4 address belongs to the AS. Therefore the translation process should check the list of IPv4 blocks held by the AS. If there is an entry for the IPv4 address in the list, it copies this address twice; if not, then the translator copies "0xFFFFFFFF" into the second 32-bit word and the IPv4 address into the forth, in the corresponding IPv6 address field.

Because the packet TTL is translated into Hop Limit, its decrement and the route metric are kept consistent.

D. Remarks

Mapping is the main idea of the proposed approach: mapping of addresses, of route entries and of packet headers. By the help of automatic mapping, the scheme does not introduce states of virtual links into the networks as the previous tunnel approaches do. This is obtained at the cost of memory and CPU times in IPv6 backbone routers in order to store and process the routes of mapped-from-IPv4 prefixes. The cost is moderate and acceptable in an intra-domain environment. To this sense, the proposed design is preferable

than meshed IPv4-over-IPv6 tunneling in the IPv6 backbone to some extent. Furthermore, as we will illustrate in the next section, the design is much superior to tunneling when being extended in more complicated circumstances.

It is reasonable to suspect that the translation might cause significant degradation to the network performance. In order to evaluate such an impact, as well as to verify the availability of the scenario, we made a prototype implementation based on Linux kernel, where the forwarding table is extended to contain the address mapping information, and then we did basic experiments in a simplest network configuration.

E. In-lab Experiments

The experiment environment is configured as is described in Fig. 5, which could be seen as the simplest configuration.

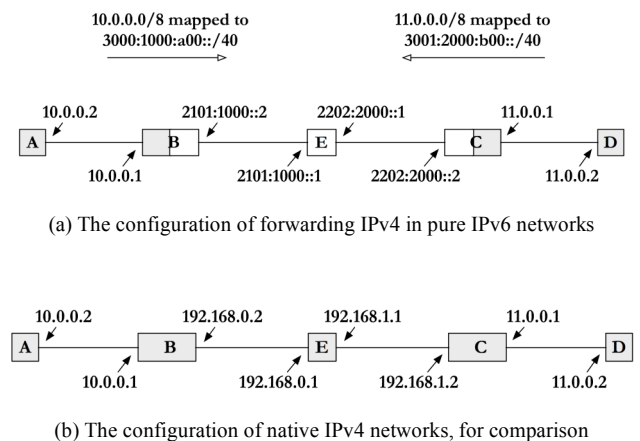


Figure 5. Configurations of in-lab experiment environment

In the environment, 5 computers are connected one by one over 100Mbps direct Ethernet links. They can be configured in two ways alternatively: forwarding IPv4 traffic through pure IPv6 networks, see Fig. 5(a); or all the nodes are running in native IPv4 mode as is shown in Fig. 5(b).

In the case of Fig. 5(a), There are two native IPv4 networks, 10.0.0.0/8 and 11.0.0.0/8, separately connected to pure IPv6 network sections, via edge routers B and C that have been enabled with our modified Linux kernel. Host A (10.0.0.2) and D (11.0.0.2) does not have IPv6 functionality enabled at all, and they would like to communicate each other by IPv4 protocol only. The host A's default gateway is configured as 10.0.0.1 while D's is 11.0.0.1.

Because what is concerned is the header translation's impact to performance, it is assumed that the address mapping and route mapping information has been learned and well installed. The two IPv4 network ids, 10.0.0.0/8 and 11.0.0.0/8 are mapped into IPv6 prefixes, 3000:1000:a00::/40 and 3001:2000:b00::/40, respectively. Native IPv6 router E isolates two native IPv6 network sections, 2101:1000::/64 and 2202:2000::/64. In the IPv6 stack, the edge router B knows that destination 3001:2000::/32 could be reached via 2101:1000::1, while C knows 3000:1000::/32 via 2202:2000::1; in the IPv4

stack, B knows 11.0.0.0/8 reachable via IPv6 networks while C knows 10.0.0.0/8 reachable too.

1) Verifying the Functionality

Firstly, the functionality of the boxes B and C is verified by most well known applications. We have tested echo over TCP and over UDP, Telnet, FTP, HTTP, Windows File Sharing and Remote Desktop services. When A plays the role of client and D does that of server, the services are all well working.

Note that servers sending large messages require setting host A and/or host D's link MTU to 1480 bytes. A 1480-byte IPv4 packet will become a 1500-byte IPv6 packet after the translation and therefore one should manually set the link MTU to path MTU, whose discovery is not enabled in our header translations. FTP, HTTP, File Sharing, Remote Desktop are some typical cases which do not work without such a setting.

Fig. 6 shows (a) a real IPv4 packet captured on the host D, which just receives a TCP echo packet from host A, and (b) its translated version captured on router E. (The router E is a FreeBSD box and then its tcpdump default style is quite different from the Linux case). The observation shows the basic design of address mapping and header translation has been well implemented.

```
14:46:46.684782 IP (tos 0x0, ttl 61, id 0,
offset 0, flags [none], proto: TCP (6), length:
308) 10.0.0.2.56156 > 11.0.0.2.7: P 1:257(256) ack
1 win 65535 <nop,nop,timestamp 341980155 0>
```

```
0x0000: 4500 0134 0000 0000 3d06 67c1 0a00 0002
0x0010: 0b00 0002 db5c 0007 4bf4 e110 4728 3e24
0x0020: 8018 ffff 6e83 0000 0101 080a 1462 33fb
0x0030: 0000 0000 3838 3838 3838 3838 3838 3838
0x0040: 3838 3838 3838 3838 3838 3838 3838 3838
...
```

(a) An IPv4 packet captured on host D

```
14:45:55.493228 3000:1000:a00:2::a00:2.64476 >
3001:2000:b00:2::b00:2.7: P 1:257(256) ack 1 win
65535 <nop,nop,timestamp 341986833 0> (len 288,
hlen 63)
```

```
6000 0000 0120 063f 3000 1000 0a00 0002
0000 0000 0a00 0002 3001 2000 0b00 0002
0000 0000 0b00 0002 fbdc 0007 ec73 b7f7
da6c 3d06 8018 ffff 2a60 0000 0101 080a
1462 4e11 0000 0000 3838 3838 3838 3838
...
```

(b) Corresponding IPv6 version captured on router E

Figure 6. Observing header translation in TCP echo sessions

2) Measuring the performance

Secondly, we concentrate our attention on the performance issue. The measurement is produced in the configuration as Fig. 5(a) first and then as in Fig. 5(b). The comparison between the two cases tells whether the translation causes network performance down heavily.

The following table (Tab. II) summarizes the measurement results. The bandwidth was tested with the `ttcp` tool, and a 16MB data was transferred from host A to host D in each configuration. For the round-trip delay measurement, a flood ping is employed with 10000 trials of sending 1000 bytes data

unit. The measurement result shows that the impact of header translation to the edge router's performance is ignorable.

TABLE II. PERFORMANCE COMPARISON

Parameter	IPv4 across IPv6 (Case Fig. 5(a))	Native IPv4 (Case Fig. 5(b))
Bandwidth	9122.49 KB/s	9117.42 KB/s
RTT Delay	1.097 ms	1.066 ms
RTT Std Dev.	0.104 ms	0.104 ms

IV. EXTENSIONS TO INTER-DOMAIN ENVIRONMENT

The basic design is focused on the intra-domain environment. When the scheme is extended to fit the requirements of forwarding IPv4 traffic in an inter-domain environment, a prefix discovery mechanism is necessary in order for edge routers to determine which prefix should be employed for the address mapping. The key idea of the prefix discovery is utilizing the AS number to map an IPv4 network id to an IPv6 one.

On the other hand, for an inter-domain routing environment, BGP [17] provides useful information on the origin of any network prefix. Utilizing this information can prevent us from introducing any extra list of network identifiers held by any autonomous system.

In this section, we extend the scheme in two models: 1) stub autonomous systems are connected to an IPv6 backbone, as customer networks of the latter, and only have accesses to IPv4 hosts within the scope controlled by the provider; 2) transit autonomous systems are inter-connected with each other. The prefix discovery mechanism is the key component for the latter case. Before closing the section, we discuss the issues of partial deployment and of selecting prefix among those held by an AS if there are many of them.

A. Customer IPv4 To Pure IPv6 Provider

Now we consider a circumstance a little more complicated than the basic one. A set of dual-stack or IPv4-only customer networks is connected to the IPv6 backbone, no matter whether there is or is not an ever-established IPv4 backbone in the same AS. Each customer network has an AS number and peers with the backbone via EBGp for IPv4, and possible for IPv6 too. The backbone border gateways speak IBGP for IPv6.

Fig. 7 summarizes the scenario where access networks send their IPv4 traffics to each another via the pure-IPv6 backbone of their service provider. In the Figure, two IPv4 native ASes are attached to the provider IPv6 AS with the help of edge routers, while the remaining IPv4 Internet is attached with a relay router. EBGp peers are established between edge routers and their customers' BGP routers, while IBGP peers are utilized between relay routers and their counterparts in IPv4 Internet. Inside the realm of backbone, however, IPv6 IBGP peers are deployed just like inside a general autonomous system to keep routing information consistent.

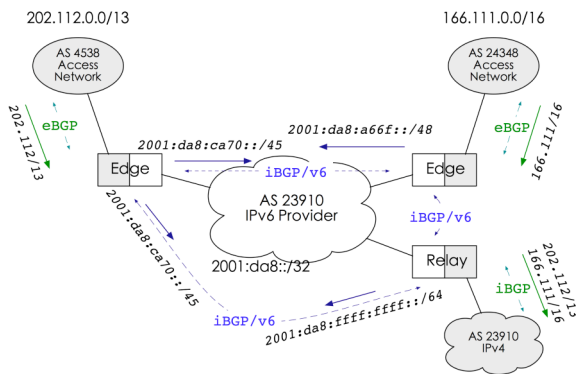


Figure 7. IPv4-over-IPv6 transition for access networks

1) Addressing

The addressing scheme is similar to that in the basic design. However, though the access networks now have independent AS numbers and possibly hold IPv6 prefixes allocated by the provider, only the specified /32-prefix of the provider is employed for address mapping. The customers might have no IPv6 prefix short enough. For a prefix with length other than 32, embedding the IPv4 network identifier might form a locator longer than 64 bits. It might challenge greatly to the current implementations of IPv6 router.

It is also defined that a special prefix appended with “0xFFFFFFFF” is correspondent to network identifiers elsewhere out of the scope that the provider serves.

2) Route mapping and edge router behavior

Although the mapped prefixes belong to the aggregation of the provider, they are companied with an AS path in the BGP table learned by the provider’s backbone. This is as it were that a mapped IPv6 prefix is allocated to an access AS, even if the latter has no IPv6 facilities at all.

Therefore the route mapping process can be implemented as a BGP extension, which manipulates the mapping between the IPv4 RIB (Routing Information Base) and the IPv6 one.

Now the behavior of the edge routers has changed in comparison to the basic design. They cease to do businesses with the IGP directly but work over BGP and leave the interoperation between BGP and IGP running regularly as in today’s Internet.

Once an IPv6 BGP route entry comes to the edge router, the router will check whether it is originated from an access AS who is a customer of the provider. The check is done with the IPv6 RIB and the list of customer autonomous systems. If the answer is yes, then the entry will be translated into corresponding IPv4 version and injected into the IPv4 RIB with the same AS path information, setting the “network” attribute with itself. If the answer is no, then the entry is a regular IPv6 route and there is no need to do anything.

For some implementations of BGP multi-protocol extensions [18], it is not a problem to configure a pair of IPv6 speaker advertising routes for IPv4 address family (e.g. see [19, 20]). However these advertisements should cooperate with

static routes or with “set next-hop” phases in route-map, because the advertised next-hop is unreachable in IPv4.

Unfortunately, the latter solution does not fit the requirements of the reverse mapping, from IPv4 to IPv6. Therefore, it is unavoidable to extend the edge router with new functionality. To this sense, we do not recommend to apply the configuration of advertising IPv4 routes over IPv6 peers.

3) Route mapping for the Internet and the relay behavior

The relay router is also joining the IPv6 IBGP conversation instead of running any IGP, i.e. edge routers and relay routers here are all BGP speakers. On the other hand, unlike common edge routers, it is peered with the IPv4 backbone of the same AS, via one or more IPv4 IBGP sessions. The relay translates the learned IPv6 prefixes that originate from the mapped network identifiers of the customers, back into its original IPv4 version, and advertises them to the IPv4 backbone.

Meanwhile, the relay router advertises the special prefix with “0xFFFFFFFF” for all the IPv4 networks outside the scope of the provider’s administration.

It is also feasible to configure the relay speaking IPv4 routes with other edges via IPv6. In such a configuration, however, a full-mesh peering topology should be established among all the edges and all the native IPv4 IBGP speakers within the same AS, or the relay should be running as a reflector. This is also not recommended for the same reason mentioned above.

It is possible that there are more than one relays. Anyway, they are advertising the same prefix for the outside world.

Some edge routers or relays might be peered with other native IPv6 service providers. On these EBGPeers, the route should be well aggregated to prevent the more specific leaking from the realm under the control of the provider.

4) Header mapping

Relays and edge routers are doing the header mapping as is described in Fig. 4. Correspondingly, the information of what IPv4 network identifiers are served by the provider is now learned from the BGP RIB instead of being retrieved from a publicly distributed address list. Nonetheless, we can do a cache of such a list from the RIB, in order to accelerate the processing speed.

Indeed, the BGP RIB provides a way of automatically distributing the list. Actually, the scenario in Fig. 5 can be seen as an alternative for that in Fig. 1, as long as each IPv4 subnet are configured with BGP and assigned with a private AS number.

The rest part of the header manipulation process is as same as in the basic design.

B. IPv4 Access Networks and Pure IPv6 Providers

In the previous subsections, we employ a special prefix to avoid mapping the IPv4 route table explosion into IPv6. Only those IPv4 network identifiers that belong to the provider are mapped correspondingly while the others are merged into the special prefix.

It is a safe solution at the current stage, when almost each of the Internet service providers is running at least one native IPv4 backbone and some of them running a dual-stack one or separated IPv6 and IPv4 backbones. In the solution, all the IPv4 traffics are only changed to IPv6 within the provider's backbone. They must be reverted to IPv4 before they are going to leave the autonomous system.

It is interesting to ask further: is it possible to realize the transition among native IPv6 internets? If so, it is not necessary to transfer the IPv6 packets on the borders between different providers' domains.

Here, we are trying to extend the address mapping scheme and the functionality of the relay router, in order to achieve such a goal.

Fig. 8 summarizes the idea in such an environment, where two providers are inter-connected by native IPv6 link, and both do support the proposed mechanism. In Fig. 8, the symbol "E" represents edge router connecting an access network to a provider backbone while the "R" represents relay router, a special kind of edge router between two provider backbones. Note that, in this new model, ends in IPv4 internet would treat the IPv6 backbone as an alternative option of routing their packets, and IPv4 EBGP is introduced between the two ASes even though there is not any IPv4 traffic. Therefore, things are quite different under the view of IPv4 as under the view of IPv6. The major aspects of the figure are described in detail as follows.

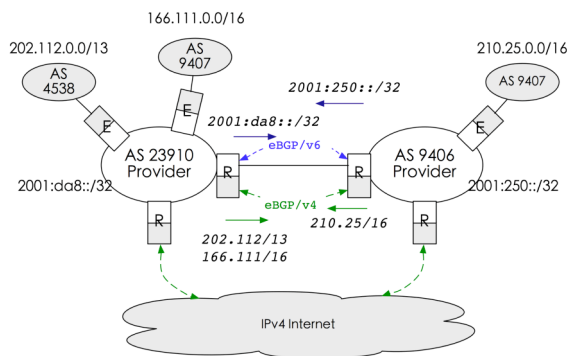


Figure 8. Transition in inter-domain environment

1) Inter-domain address mapping – prefix discovery

Because the BGP RIB contains the origin AS information for each IPv4 network, it is possible to discover the proper prefix for any destination IPv4 address. We have the following algorithm as is described in Fig. 9.

If the algorithm returns a valid value, then it should be the mapped prefix. It is possible that the autonomous system holding the IPv4 address does not hold any IPv6 prefix and therefore it must be sent back into the IPv4 world. In such a case, the special prefix as is described before will be utilized.

If the algorithm returns NULL, it means no AS information is found in the routing table, and the mapping is failed definitely. Thus the corresponding network is unreachable.

2) Forwarding behavior

A packet will be translated with header mapping by the first edge router it encounters from the access network into the backbone of one provider AS. Now the source (destination) address field in the IPv6 header will be generated with the source (destination) provider AS's IPv6 prefix and the corresponding IPv4 address. The prefix discovery algorithm is applied when doing the header mapping.

Input: IPv4 network id, q, with length x; current AS's prefix p^*. Output: mapped IPv6 prefix, p, with length $32 + y$. 1. Retrieve in IPv4 RIB, find the entry for q, with maximum matching; 2. If it fails, go to step 9.; else find AS number a which is the origin of the q/x in the found entry; 3. Retrieve in IPv6 RIB, find the set of all the prefixes originating from a, say $P = \{(p, y)\}$, and record all the corresponding AS paths; 4. If $P = \emptyset$ go to step 8.; else check in IPv6 RIB and find each /32-prefix for every $p \in P$; 5. If there are not any such /32-prefix, go to step 8.; else find the smallest prefix among them, say p''; 6. Form p with p'' and q; 7. Return p; 8. Take the first 32 bits of p^* and form p with p^* and 0xFFFFFFFF, $y=32$; Return p; 9. Return NULL.

Figure 9. Prefix discovery algorithm

When the packet is delivered from the source provider AS to the destination provider AS, it just follows the route for the well-aggregated IPv6 prefix whose length might be 32 or even shorter. The forwarding routers between these two providers do not know the packet is an IPv4-mapped one. They treat it just as a native IPv6 packet.

After the packet enters the realm of the destination provider, the access mechanism discussed in the previous subsection will take effect and will find the final destination.

Therefore, the forwarding action for the IPv4-mapped packets does not require that the border gateways between the two provider autonomous systems are dual-stack.

3) Routing

In the global IPv6 BGP table, all the prefixes are well aggregated and therefore there will not be those IPv4-mapped prefixes (which are more specifics) in the BGP view at inter-provider peers.

Although the scheme is working even without IPv4 forwarders between the two provider domains, there should be an IPv4 BGP session between them, in order that the IPv4 global routing table is integral. This session, as same as in the

previous case, can be running over IPv6 peers. However, if there are other inter-domain relays working between a provider AS and the other part of the IPv4 Internet, all the relays must run IPv4 IBGP sessions to exchange IPv4 global routing table.

Here, the relay routers do not perform the functionality of mapping all the outside IPv4 network identifiers into a single special IPv6 prefix. It just transfers the IPv4 global RIB to the edge routers for access, in order that they have the ability of applying the prefix discovery algorithm. On the other hand, because there must be IBGP sessions among all the relays and the edges in a provider AS, it is not necessary for the relay to translate the mapped version of the inside IPv4 network identifiers. In other words, a relay router is just a regular BGP speaker for both IPv6 and IPv4, standing between two provider domains.

4) Remarks on partial deployment issue

If all the neighboring providers have been enabled with the proposed scheme, then the transition will be working well. There will not be the case that a pure-IPv4 AS plays the role of transit for such two providers, because the AS path corresponding to the discovered prefix is found in the IPv6 RIB.

However, there would be a case where one or more pure IPv6 AS is standing on the path from one provider to another. This AS does not establish any IPv4 BGP peers at all, and accordingly it cannot transfer IPv4 routing table for the provider domains.

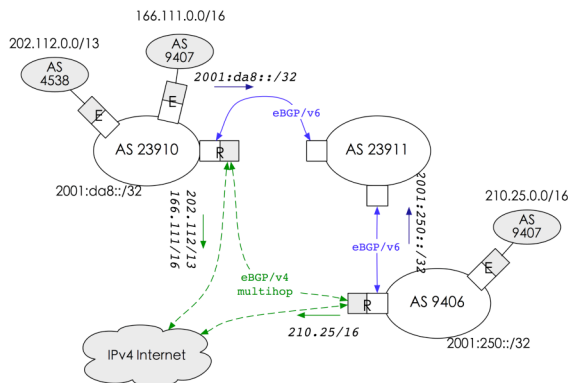


Figure 10. Transition in inter-domain environment: partial deployment

Therefore, the pair of autonomous systems that provide IPv4-over-IPv6 access service should establish an EBGP peer, directly or via a route server [21]. This configuration could be described with Fig. 10.

Unfortunately, however, this solution might encounter some non-technical obstacles. These might be the real reason of why there is not an ever-existing EBGP peer between the two providers.

5) Remarks on prefix selection issue

The prefix discovery algorithm will encounter an issue of prefix selection. If a provider hold more than one /32 blocks, it is unreasonable to require it reserve all the possibly mapped prefixes in each block. Only one block will be reserved for the

use of IPv4-over-IPv6 transition. If the prefix discovery algorithm makes a wrong prefix, then the packet delivery will be failed.

The algorithm described in Fig. 9 introduces a common rule for the addressing. It is assumed that all the providers might follow the rule that the numerically smallest IPv6 prefix in their subspaces of addresses will be employed. However, this is a very weak constraint because, from the IPv6 RIB, a remote one might learn only part of all the address prefixes held by a provider.

Therefore, the transition scheme is far less mature than the needs of practice. The issues just discussed need more research efforts.

V. MIGRATION TO NATIVE IPv6 APPLICATIONS: A CASE STUDY

Here we illustrate a migration process from native IPv4 access network to a complete native IPv6 application environment, in a real national network, in order to give a big picture on how the proposed scheme supports IPv4-IPv6 transition.

Fig. 11 is a simplified version for our inter-domain solution that is shown in Fig. 8. However, two native IPv6 access networks are added, and then the AS 24348, AS 24349 and AS 24350 represent 3 typical stages in migration, relatively.

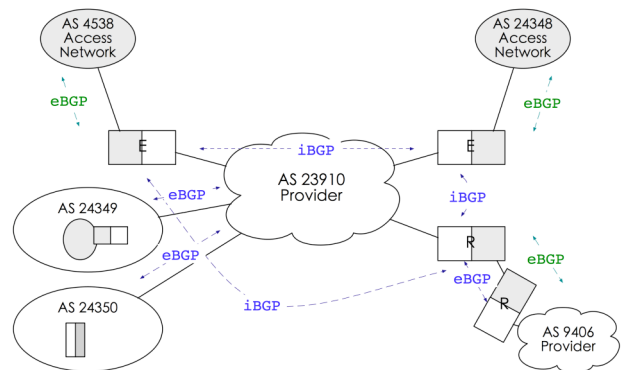


Figure 11. Migration with 3 steps

A. IPv4-only Access Network (AS 24348)

This is the case that we have ever discussed. The edge router is deployed on the boundary of the access network and the provider backbone. It announces to the backbone an IPv4-mapped IPv6 prefix for the whole access network. It translates headers for all packets coming in or going out of the access network.

B. Native IPv6 Access Network Containing a Small Group of IPv4 Subnets (AS 24349)

At this stage, the access network has contained a native IPv6 infrastructure. However, not all the network nodes have upgraded into IPv6 or not all the applications have been ported, and therefore there are still some small groups of subnets where it has to be to communicate with the Internet via IPv4.

The AS 24349 might be assigned with an IPv6 block. Besides, it can also heritage the prefix that assigned by the provider at the first stage. For instance, assume the IPv4 block of the access network is 210.29/16, then the corresponding IPv6 prefix would be 2001:da8:d21d::/48. The relay router of provider backbone will announce 210.29/16 to the IPv4 Internet while the regular border gateway of the access network advertises its native IPv6 prefix 2001:da8:d21d::/48.

The route mapping and header mapping will be performed within the scope of the access network AS 24349, by the help of the basic design.

C. Native IPv6 Access Network Containing Separated Dual-Stack Hosts (AS 24350)

At this stage, all the hosts and routers in the access network have been upgraded to support IPv6. On the other hand, as the time goes on, more and more applications are ported to IPv6-enabled version, but this is a very long process. Furthermore, there might be much resources still stay on the IPv4 Internet. Therefore the IPv4 accessibility cannot be removed for the time being.

At this stage, edge routers are removed and their functionality of header mapping is replaced by end systems themselves. Route mapping is no more needed. Native IPv6 routing is directly serving end nodes.

For example, a node has IPv4 address 210.30.4.5 now has the corresponding automatically mapped IPv6 address 2001:da8:d21e:0405::d21e:0405. It is only connected with the network via IPv6 link. It is an isolated IPv4 island in the IPv6 ocean at all. It is running a service that some other one in another IPv4 access network might want to visit.

When an IPv6 packet arrives, the node will check the source address to determine whether an edge router somewhere maps it. If so, a local translation is performed and the translated IPv4 version will be sent to the IPv4 stack to process. Reversely, a packet related to IPv4-only applications will be generated from the IPv4 stack but be translated into the IPv6 version to send out.

When all the outgoing and incoming application traffics going through the edge routers have become IPv6 only, the transition process is over, and the whole network has been upgraded into IPv6 infrastructure and IPv6 application resources.

The stage 3 is most significantly important. Although the dual-stack solution is not preferred in our opinion, it works quite well in rough consensus. However, the stage 3 is not avoidable even for the dual-stack transition paradigm. Running two stacks over the world should not be a permanent operation and therefore there must be a moment that all the IPv4 internet services over the world will be stopped, but nobody can make sure that every application, every server at every corner of the world has been ported to IPv6 at that time. To complete the transition, the stage 3 is unavoidable, and here a working scheme has ever proposed.

VI. CONCLUSIONS

This paper introduces a new solution for routing IPv4 in IPv6 backbone without explicit tunneling. The details of the design and possible solutions are illustrated in 3 steps, from the simplest to the most complicated: a basic design for intra-domain environment, an access network transition scheme and an inter-domain solution. The prototyping is performed in the China Education and Research Network (the CERNET) under the framework of the China Next generation Internet (CNGI) project. Because the CNGI project is constructing a native IPv6 infrastructure as tens times large as the current CERNET, we believe that the new infrastructure cannot be dual-stack and therefore the research effort of this paper is significant. For encouraging the transition and migration from IPv4 to IPv6, esp. for the application migration, it is important to have an IPv4-over-IPv6 transition scheme. The works related to this paper are to do such experiments. These experiments will present more comprehensive results than the design and analyses.

The advantages and disadvantages of IPv4-over-IPv6 transition in comparison to dual-stack solution are still in debate. On one hand, there might be impossible for a developing country like China to deploy a new, dual-stack nation-wide internet, tens or hundreds times as large as current national infrastructure in scale. On the other hand, even in the dual-stack transition solution, a scheme of serving IPv4 applications with pure IPv6 networks is also necessary, unless people do not like to have the transition completed forever.

Some problems related to IPv4-over-IPv6 transition have not been mentioned yet. Here is an incomplete list for them:

- The multicast support;
- The mobility support;
- The impact of multi-homing;
- The QoS support and etc.

All of the problems will be studied in details only after the comprehensive experiments are completed.

ACKNOWLEDGMENT

We'd like to show our thanks to Prof. Jianping Wu, who, as the director of the CERNET, leads our research activities on the approach. Thanks to Dr. Ke Xu, Dr. Ming-wei Xu and other colleagues who contribute their ideas in discussions. A special thanks to Marc E. Fiuczynski, Princeton University, who met Maoke in European PlanetLab Workshop, discussing the possibility of IPv6-enabled PlanetLab, and then introduced his early works on IPv4-IPv6 header translation.

REFERENCES

- [1] R. Gilligan and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers," RFC 2893, Aug 2000.
- [2] G. Tsirtsis, P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)," RFC2766, Feb 2000.
- [3] E. Nordmark, "Stateless IP/ICMP Translation Algorithm (SIIT)," RFC 2765, Feb 2000.

- [4] H. Kitamura, "A SOCKS-based IPv6/IPv4 Gateway Mechanism," RFC 3089, Apr 2001.
- [5] J. Hagino and K. Yamamoto, "An IPv6-to-IPv4 Transport Relay Translator," RFC 3142, June 2001.
- [6] A. Durand, et al. "IPv6 Tunnel Broker," RFC3053, Jan 2001.
- [7] B. Carpenter, K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," RFC3056, Feb 2001.
- [8] C. Huitema, "An Anycast Prefix for 6to4 Relay Routers," RFC 3068, June 2001.
- [9] R. Gilligan, et al. "Basic Socket Interface Extensions for IPv6," RFC 3493, Feb 2003.
- [10] W. Stevens, et al. "Advanced Sockets Application Program Interface (API) for IPv6," RFC 3542, May 2003.
- [11] A. Conta, S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC2473, Dec 1998.
- [12] Maoke Chen and Xing Li, "Potential risks of deploying large scale overlay networks," Lecture Notes in Computer Science (LNCS) 3420, p.508-517, Apr 2005.
- [13] I. Raicu and S. Zeadally, "Evaluating IPv4 to IPv6 transition mechanisms", in Proceedings of ICT 2003, Feb 2003.
- [14] J. Postel, ed. "Internet Protocol," DARPA Internet Program Protocol Specification, RFC 791, Sep 1981.
- [15] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, Dec 1998.
- [16] K. Claffy, G. Miller and K. Thompson, "The Nature of the Beast: Recent Traffic Measurements from an Internet Backbone," in Proceedings of INET'98, Geneva, Switzerland, 21-24, July, 1998. (online version: http://www.isoc.org/inet98/proceedings/6g/6g_3.htm)
- [17] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, Mar 1995.
- [18] T. Bates et al. "Multiprotocol Extensions for BGP-4," RFC 2858, June 2000.
- [19] Cisco Systems Inc., "Implementing multiprotocol BGP for IPv6," in *Implementing IPv6 for Cisco IOS software*, p. 115-144.
- [20] T. Strauf and C. Schield, editors. "IPv6 cookbook for routing, DNS, intra-domain multicast, inter-domain multicast, security," 6NET Project Deliverable, 32603/WWU(JOIN)/DS/3.1.2v2/A1, Nov 2004.
- [21] D. Haskin, "A BGP/IDRP Route Server alternative to a full mesh routing," RFC 1863, Oct 1995.