J Netw Syst Manage (2007) 15:383–399
DOI 10.1007/s10922-007-9071-z

Connecting IPvX Networks Over IPvY with a P2P Method

Xiaoxiang Leng · Jun Bi · Miao Zhang · Jianping Wu

Published online: 21 June 2007

Abstract During the transition from IPv4 to IPv6, the IPv6 islands inside the IPv4 networks need to communicate with each other and with the native IPv6 network. The drawback of existing IPv6 transition methods is that relay gateways become potential communication bottlenecks. In this paper, a new method—PS64—is presented to connect IPv6 islands together over IPv4 network and reduce the reliance on these relays by shifting the burden to edge gateways on each island. In this method, direct tunnels are set up between the IPv6 islands, and a P2P network is maintained between edge gateways of these islands to propagate information of tunnel end points. After describing the algorithm, we analyze the connectivity of the P2P network, the scalability of this algorithm, and present the prototype and experiments. The results of our analysis and experiments show that the proposed method is reliable, scalable and effective. Obviously, this technique works for all IPvX over IPvY situations.

Keywords IPv6 transition · Internet · Overlay · Tunnel

Introduction

Due to the rapid expansion of the Internet and the lack of unallocated global IPv4 addresses, the transition from IPv4 to IPv6 [1] has been proposed and becomes a worldwide trend as the Internet develops. Given the prevalence of the existing IPv4 network, the transition will need considerable time to complete. Some IPv4 subnets may, while having had their network infrastructure upgraded to support IPv6, still lack direct links to the native IPv6 network. That is, there will be isolated IPv6 islands inside the global IPv4 network. In the existing methods, these islands use IPv6-in-IPv4 tunnels [2] via relay gateways maintained by IPv6 service providers to

X. Leng (✉) · J. Bi · M. Zhang · J. Wu
Network Research Center, Tsinghua University, FIT Building 4-204, Beijing 100084, P.R. China
e-mail: xleng@netarchlab.tsinghua.edu.cn

join the native IPv6 network. However, in these scenarios, all traffic from an IPv6 island to another IPv6 island will be forwarded via the IPv6-relay gateways from the source island to the native IPv6 network, and then be forwarded back from the native IPv6 network to the destination island. Therefore, these relay gateways can potentially become the communication bottlenecks.

A peer-to-peer (P2P) [3] based algorithm PS64 is proposed in this paper, in which direct tunnels are built between isolated IPv6 islands to reduce the burden of IPv6-relay gateways maintained by service providers. A P2P overlay network is set up among the gateways of IPv6 islands to propagate TEP (tunnel end point) information (⟨IPv6 prefixes, IPv4 address of edge gateway⟩) and to set up the full mesh tunnels. When an IPv6 island dual-stack gateway receives a data packet sent to the IPv6 network, it firstly checks whether there is a direct tunnel to the destination address. For packets being transmitted, the shortcut tunnels between the source and destination islands are assigned higher route priority than tunnels to relay gateways of service providers. Since the situation of IPv4 over IPv6 is mostly the same as IPv6 over IPv4, PS64 works for all IPvX over IPvY situations.

The rest of the paper is organized as follows. Section ''Problem Statement'' shows the problem statement; section ''The PS64 Algorithm''presents the PS64 algorithm; in section ''P2P Network Design'', the design of the P2P network used in PS64 is introduced; section ''Analysis of P2P Network Connectivity'' analyzes the connectivity of the P2P network by simulation; in section ''Scalability Analysis'', the scalability of the proposed algorithm is discussed by theoretical analysis, section ''Prototype and Experiments'' shows the prototype system modules and some experiments; section ''Future Work'' discusses future work and section ''Conclusion'' summarizes this paper.

Problem Statement

In the process of the deployment of IPv6, native IPv6 networks—both IPv6-only and dual stacks—will connect with each other to eventually become an IPv6 continent. The upgrade of IPv4 networks to support IPv6 will necessarily be a gradual process, due to the huge existing investment in IPv4. Native IPv4 networks and native IPv6 networks will coexist for a long time. During this period of coexistence, there are two methods for dual-stack hosts in an IPv4 continent to communicate with hosts in an IPv6 continent and to use IPv6 applications. The first way is to directly set up a tunnel—with the mechanisms ISATAP [4], Teredo [5], etc.—on the dual-stack host itself. The second way is to make the local network support both IPv4 and IPv6 and set up a tunnel on the edge gateway of the local network. In this second method, the local network becomes an isolated IPv6 island. When the internal information of local network needs to be protected, it is recommended that the second solution be used [6].

Existing methods for providing access service for IPv6 islands inside IPv4 networks include:

(1) Automatic tunnels, such as the 6to4 mechanism [7] where 6to4 addresses are used;

(2) Configured tunnels, such as IPv6 configured tunnels [2] where normal IPv6 addresses are used.

The 6to4 mechanism is an automatic way to connect isolated IPv6 sites (domains/hosts) attached to an IPv4 network that has no native IPv6 support. A host in a 6to4 site uses a 6to4 IPv6 address (2002:IPv4 Address::/80) as the communication identifier. When the IPv6 packet goes through the 6to4 router, the IPv4 address of the tunnel end point can be found within the 6to4 address and a tunnel is formed automatically. Without explicit tunnel setup, 6to4 technology is a simple solution for isolated IPv6 islands to communicate with each other and to access the IPv6 continent with the help of 6to4 relays. The automatic tunnels between 6to4 networks can effectively mitigate the burden of 6to4 relays. This method does, however, pose significant management challenges for ISPs, and there are also security problems with 6to4 brought about by the automatic tunneling mechanism [8].

With configured tunnels using normal IPv6 addresses, the isolated islands can be treated as natural extensions of the IPv6 continent. The manual configuration makes this kind of tunnel easy to control, and the security problems are greatly diminished when compared with 6to4. For the purpose of decreasing the configuration overhead, the model of a Tunnel Broker [9] is often used to manage the configured tunnels automatically.

A Tunnel Broker is a mechanism to automatically set up tunnels. Permanent IPv6 addresses are registered at the Tunnel Broker for the IPv6 hosts/islands in the IPv4 network. Then an IPv6-in-IPv4 tunnel is set up with a Tunnel Server to form the IPv6 connectivity.

Some other transition mechanisms can also be used to provide IPv6 access service for the IPv6 islands inside IPv4 networks and are discussed as follows:

In Wu et al. [10], a MP-BGP based mechanism (4over6) has been presented to the IETF softwire WG to connect isolated IPv4 networks together over IPv6 backbone. In this method, MP-BGP is extended to propagate TEP information inside the IPv6 backbone and to set up full mesh tunnels between border routers of isolated IPv4 networks. However, this method can only be used on the border routers (AFBR) of some large area IPv4 networks with MP-BGP support, and cannot be widely deployed on the edge gateways of isolated IPv4 islands. Furthermore, the BGP peers for one router are configured manually and cannot change automatically with the change of the network environment.

6PE [11] is an IPv6 transition method based on Multi Protocol Label Switching (MPLS) technology. The IPv6 islands over a (MPLS)-enabled IPv4 cloud can be connected together with the help of the IPv6 Provider Edge routers (6PE), which are Dual Stack in order to connect to IPv6 islands and to the MPLS core, which is only required to run IPv4 MPLS. The reachability information is propagated by MP-BGP and the dynamically established IPv4-signaled MPLS Label Switched Paths (LSPs) can be used for data transmitting. However, in this method, MP-BGP must be supported as 4over6, and the dependence on 6PE routers makes it hard to widely deploy this approach on the gateways of IPv6 islands.

In Zhou and van Renesse [12], P2P technology is utilized to build the IPv6 backbone over the existing IPv4 network. It directly uses a DHT-based [13] P2P network to forward the IPv6 data packets. There are some drawbacks for this scheme. Firstly, DHT is usually used for exact matching, and is not suitable for the longest matching in packet forwarding. Secondly, there is a problem in the performance of forwarding lookup on the distributed storage P2P network.

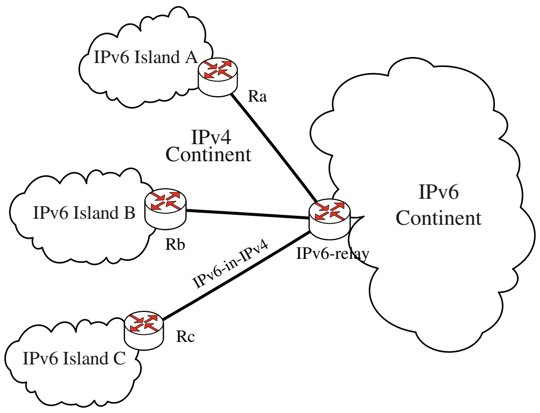
From the analysis above, we can see that the configured tunneling mechanisms with normal addresses (such as IPv6 configured tunnel, Tunnel Broker) are the most suitable methods to connect IPv6 islands with native IPv6 networks. However, these mechanisms generally use IPv6-relay gateways to provide the access service. As shown in Fig. 1, all the traffic from IPv6 islands will be forwarded by the relay gateways, even if the traffic is between IPv6 islands. The relay gateways can potentially become communication bottlenecks. Since the existing optimization algorithms are mainly focusing on the service providers, the only way to increase overall performance is to increase the performance or the number of relay gateways, which could be a costly exercise.

The PS64 Algorithm

In this paper, we present a new algorithm PS64 optimizing for the mostly used transition mechanisms (e.g., IPv6 configured tunnel, Tunnel Broker) under the scenario of IPv6 islands to reduce the burden of IPv6-relay gateways and provide high performance IPv6 access service. Besides the service providers, we also consider the customers—the edge gateways of the isolated islands. Direct tunnels are set up between IPv6 islands to transmit the traffic between these islands. When an IPv6 island gateway receives a data packet with a destination address in another IPv6 island, it directly forwards this packet to the destination island gateway through the IPv4 network. Otherwise, it sends the packet to the relay gateway of the service provider.

A lookup operation is inserted into the flow for packet forwarding of IPv6 island edge gateways. The steps of the proposed algorithm are described as follows:

Fig. 1 Scenario of IPv6 islands



(1) Form a Tunnel Table on each IPv6 island gateway by exchanging TEP information (⟨IPv6 prefixes, IPv4 address of edge gateway⟩) between the gateways.
(2) When an island gateway receives an IPv6 data packet, firstly, it examines the Tunnel Table to find out whether there is a direct tunnel to the destination. If yes, it then sends this packet directly to the IPv4 address specified by the Tunnel Table; Otherwise, it then forwards this packet to the relay gateway of IPv6 service provider.

The PS64 algorithm is based on that for configured tunnels, inheriting the advantages for control, management and security. In addition, since the IPv6 islands are mostly upgraded from IPv6 subnets, the service and applications inside the IPv6 islands will also upgrade to support IPv6. Furthermore, the nearby subnets may have the same network environment, hence the IPv6 islands may come into several clusters. With the principle of locality, the traffic forwarded by the relay gateways will be mostly restricted between the IPv6 islands. Therefore, as with 6to4, the direct tunnels between IPv6 islands can effectively reduce the burden on the relay gateways and greatly increase the access performance of the isolated IPv6 islands.

In contrast to 6to4, there are no special addresses used in these islands. This makes it impossible for the island gateways to get the TEP information from the destination address of a data packet. How a gateway of IPv6 an island gets the TEP information of all the other IPv6 islands is the key consideration in this algorithm. One solution is to set up a central server to hold all the information, send it to each island gateway and to propagate the updates. However, this method is not scalable, and will make the central server a weak point of the system. Another solution is to build a P2P overlay network between the edge gateways of IPv6 islands. The TEP information is propagated and updated by the P2P network. In the PS64 algorithm, we adopt this second option.

The architecture of PS64 is shown in Fig. 2. Ra, Rb, ... are edge gateways of IPv6 islands. A P2P network is maintained by them to share TEP information with each other. With the TEP information, shortcut tunnels are set up on each gateway for transferring data packets between these islands. Since the 4over6 and 6over4 situations are mostly the same, the PS64 algorithm can work for all IPvX over IPvY environments. In other words, PS64 is a general method to connect IPvX networks together over IPvY.

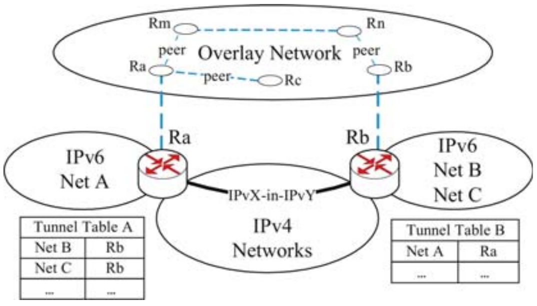


Fig. 2 The PS64 algorithm

After presenting the PS64 algorithm, we describe the design and the connectivity analysis of the P2P network; analyze the scalability of the proposed algorithm and show the prototype modules and experiments in the following parts.

P2P Network Design

The Overlay Network

A P2P network is set up to propagate TEP information among the IPv4 island gateways. Each IPv4 island edge gateway broadcasts its TEP information inside the P2P network and gets the other gateways' TEP information from its peers.

We have chosen to use an Unstructured P2P network where each node chooses its neighbors randomly. Since there is no special structure to this P2P network, it is necessary to find a way to keep the nodes inside this P2P network connected with each other. This will be discussed in section ''Analysis of P2P Network Connectivity''.

A bootstrap node, which any of the edge nodes can be, is enough to form a pure P2P network. However, as an IPv6 access service, some policies for control and management should be easily introduced in the future and, as a result, this bootstrap node can only be the one provided by the service provider. A Registration Server is set up as a tracker to help the gateways of IPv6 islands to join the P2P network gradually. The server then knows the IPv6 addresses of all nodes in the P2P network. For future control and management extension, it has to discover the changes of the P2P network, either from the removal or failure of nodes or from the changes of IPv6 addresses. Fortunately, these are already contained within the information spread inside the P2P network. We put the Registration Server into the P2P network as a normal node so that these changes are easily noticed by the update scheme of the P2P network.

The robustness of the registration scheme is a key issue. The failure of the Registration Server may make it impossible for a new node to join the P2P network. PS64 uses multiple A-records for the same domain name of the Registration Server in the DNS. In this way, several Registration Servers can be set up by the service provider, all with the same domain name. For each DNS lookup, a randomly chosen IP address of these Registration Servers is sent back, and makes this registration scheme more reliable.

Similar to some routing protocols, such as OSPF [14], the flooding scheme is used to broadcast the TEP information inside the P2P network. The TEP information is synchronized by a sequence number that is increased for each update and included in the header of each packet. When a node receives an update packet from one of its neighbors, it refreshes the related items in its local database, and then forwards the packet to all the other neighbors. Although flooding can be a waste of network resources, it's a reliable scheme to spread TEP information to all nodes in the P2P network.

The introduction above is the overview of the design of P2P network. Next, the process of new subscription and failure detection will be described.

New Subscription

In PS64, a Registration Server is set up to help the IPv6 island gateways to join the P2P network. When the Registration Server receives a subscription request from a newly arrived gateway, it randomly chooses 20 of all existing nodes and sends their IPv4 addresses to the sender of the subscription request. The new node then gets the TEP information of all the islands corresponding to the nodes specified by the received IPv4 addresses and forms a Tunnel Table for data forwarding. The steps of a new subscription are described as follows:

(1) *Register*. The newly arrived node registers at the Registration Server, and gets a list of IPv4 addresses for 20 existing nodes.
(2) *Make overlay neighborhoods*. The new node tries to make overlay neighborhoods with the nodes specified by the received IPv4 addresses.
(3) *Exchange TEP information*. The new node gets the TEP information of all other islands from its neighbors. Meanwhile, it spreads its TEP information inside the P2P network.
(4) *Form a Tunnel Table*. With the TEP information received, the new node sets up a Tunnel Table for data transmission.

Failure Detection

It is common to maintain the overlay neighborhoods of the P2P network by sending keep-alive messages periodically. When a node detects that it has not received such keep-alive message from its neighbor for a certain time, it broadcasts the failure notice to the whole P2P network through its live neighbors.

Furthermore, in order to increase the stability of the P2P network, we set the minimal degrees d for each node to be 5. When a node finds the count of its live neighbors is less than d, it randomly chooses some nodes from the TEP information stored locally and tries to make new overlay neighborhoods until the count of its live neighbors reaches d.

Analysis of P2P Network Connectivity

Since there is no special structure maintained with the P2P network used in PS64, the leaving or failure of some nodes may cause the P2P network to split into multiple disconnected subnets. In this section, the connectivity of the P2P network will be analyzed. It is hoped to find an effective way to keep the P2P network connected.

In 1964, Paul Baran did some research on distributed communication networks, which consist of unreliable nodes and unreliable links. From the viewpoints of Node Destruction and Link Destruction, his simulation and analysis shows that, with a certain node degree, the distributed network can provide a reliable communication service under a given probability of node failure and/or link failure [15]. Afterward, the situation of Link Destruction is proved by the theorem about sharp threshold in a random graph that ''If G is a k-connected graph of n vertices and the edge probability $p(n)$ satisfies $p(n) \geq c\log(n/k)$ for a large enough absolute constant $c > 0$, then a.s. the random sub-graph G_p is connected'' [16].

Furthermore, the reliability analysis of the random multicast protocol Gossip shows the sharp result that if there are n nodes, and each node gossips to $\log n + k$ other nodes on average, then the probability that everyone gets the message converges to $exp(exp(-\lambda))$ [17].

This research gives us inspiration in analyzing the connectivity of the P2P network. The major difference between a physical network and a P2P overlay network is that in a P2P network, the neighbor nodes for a node are not fixed. When a node finds one of its neighbors unreachable, it can set up a new neighborhood with another live node. Thus the probability of disconnection in a P2P network may be much lower than in a physical network.

As described in section ''New Subscription'', a simple Random Recovery Scheme is used to avoid the disconnection of the P2P network by setting a minimal degree d for each node and randomly choosing new neighbors for recovery. The effectiveness of this recovery scheme will be analyzed by simulation. The feasible scope of the variable d to keep the P2P network connected will be discussed. When simulating, we have a basic assumption that the physical network underlying the P2P network is robust and reliable. In this way, the behavior of the P2P network is independent of the physical networks. We developed a special tool for this simulation. During the simulation, the following two kinds of failures are considered:

(1) *Single Node Destruction (SND)*. Each node in the P2P network is destroyed with a certain probability. The probability for each node is independent of the probabilities for other nodes.
(2) *Single Virtual Link Destruction (SVLD)*. Each virtual link between two overlay neighbors is destructed with a certain independent probability. An example of SVLD would be transport timeout or de-convergence of routing after some physical failure.

The situation of Single Node Destruction, Single Virtual Link Destruction and the combination of SND and SVLD will be simulated in the following parts. The steps of simulation are as follows:

(1) Randomly build a N-node network where the nodes join the network gradually with minimal degree d;
(2) Destroy each node or virtual link independently with a certain probability;
(3) Rebuild the network with the Random Recovery Scheme;
(4) Check the connectivity of the rebuilt network;
(5) Loop step (1–4) for 100,000 times to get the probability of disconnection.

Single Node Destruction (SND)

In this situation, unlike the physical network, along with the increasing probability of SND, the probability of disconnection of the P2P network can be restricted to a low level with the Random Recovery Scheme.

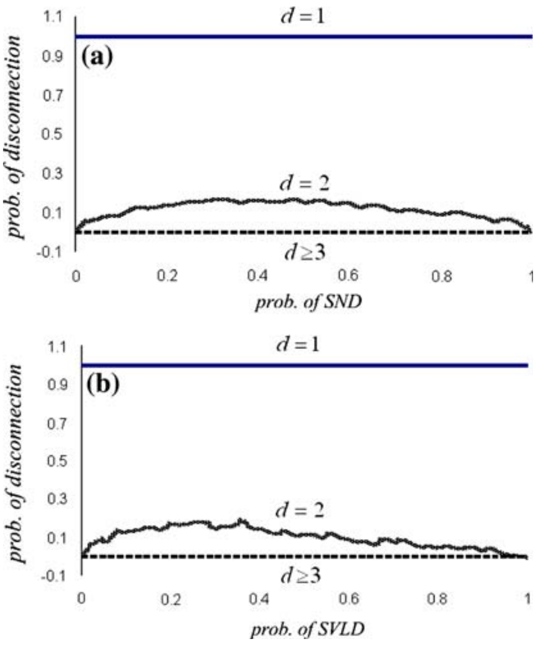


Fig. 3 Simulation results of SND or SVLD, N = 10,000

Figure 3a shows the surprising simulation result of SND with node number N = 10,000. The probability of disconnection stays less than 0.2 when d is 2. And this probability decreases to less than 10^{-5} when $d \geq 3$, whatever the probability of SND is. We present the following further analysis: Assume, for example, a probability of SND = 99%. After the SND occurs, the number of active nodes will be 10000*(1–99%) = 100. If the Random Recovery Scheme is sufficient to keep the 10000-node P2P network connected, it is much easier to keep a 100-node P2P network connected. In other words, the probability of keeping the P2P network connected is close to 1 as long as d is equal to or greater than 3.

Single Virtual Link Destruction (SVLD)

Similar to SND, when the virtual links are destructed separately, the Random Recovery Scheme also can prevent disconnection of the P2P network effectively. As shown in Fig. 3b, the probability of keeping the P2P network connected is again close to 1 when $d \geq 3$.

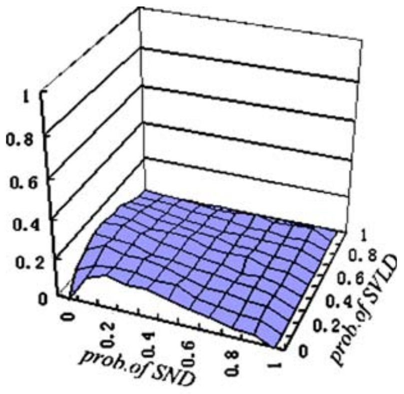
SND + SVLD

As shown in Fig. 4a and b, in the situation of both SND and SVLD, similarly, the P2P network can keep connected when $d \geq 3$.

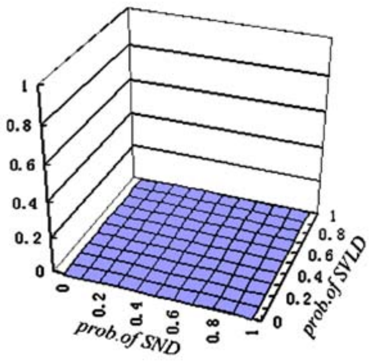
Influence of Node Number N

A threshold C is defined to indicate the minimal value of d that keeps the disconnection probability of the P2P network less than 10^{-5}. In other words, when

Fig. 4 Simulation results of SND + SVLD, N = 10,000



(a) d=2



(b) $d \geqslant 3$

$d \geq C$, the P2P network can keep connected whatever the probability of SND or SVLD. In the simulation above, C is equal to 3. When $d \geq 3$, the probability of keeping the P2P network connected is more than 0.99999, independently of SND or SVLD. With the node number N varying from 1 to 10,000, the relationship between C and N is shown in Fig. 5. With increasing N, C grows quickly to 3 and stabilizes.

Summary

From the analysis above, when the node number is less than 10,000, the P2P network used in PS64 can keep connected as long as the minimal degree of each node d is equal to or more than 3. For some limitations of the simulation environment, the situation with more than 10,000 nodes is difficult to simulate. From the analysis above, we can infer that when d is equal to or larger than 3, the P2P network can keep connected no matter what the node number is.

In the PS64 algorithm, d is set to 5. This should be enough to prevent disconnection of the P2P network.

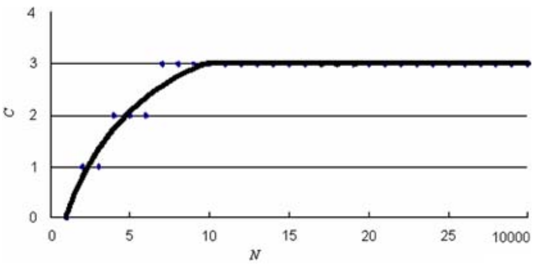


Fig. 5 Relationship between C and N

Scalability Analysis

During the transition from IPv4 to IPv6, isolated IPv6 islands will exist widely in the current IPv4 networks. The PS64 algorithm must scale to support many (i.e., 10,000) IPv6 islands.

On the data plane, the forwarding performance with such a huge number of tunnels on each gateway should be considered. There are already plenty of studies on the area of fast data forwarding. A trie-tree [18] structure is used to form the Tunnel Table and to limit the overhead of prefix lookup to a constant value.

On the control plane, the scalability of the P2P network used in PS64 should be discussed. A flooding-based Unstructured P2P network (i.e., Gnutella [19]) may not be scalable because of the heavy flooding overhead caused by transient peers. Fortunately, the edge gateways of IPv6 islands are much more stable, so the P2P network used in PS64 can support many more nodes. A hash-based structure is used to hold the TEP information of all the other gateways. This limits the maintenance overhead for each update of TEP information to almost a constant value. Since receiving an update packet does not consume many CPU cycles, the flooding overhead of the proposed P2P network scheme is mostly the cost of network bandwidth.

Assume the duration time that each node (island gateway) stays connected to the P2P network independently has an exponential distribution with parameter λ. Within one second, a single node is destroyed with probability $1 - e^{-\lambda}$. When the node number of the P2P network is N, the number of nodes failed in one second is expected to be $N(1 - e^{-\lambda})$. And the expected size S of each update packet flooding inside the P2P network is $S_0N(1 - e^{-\lambda})$, where S_0 indicates the size of each failure notice.

According to the founding process of the P2P network with minimal degree d, the neighbor number of node i is expected to be:

$$E(n_i) = \begin{cases} d + \sum\limits_{l=d+2}^{N} \frac{d}{l-1}, & 0 < i \le d+1 \\ d + \sum\limits_{l=i+1}^{N} \frac{d}{l-1}, & d+1 < i \le N \end{cases} \tag{6.1}$$

From Eq. 6.1, we can see that the first $d + 1$ nodes have the maximum neighbor number on average. These nodes may have more trouble with flooding. We consider the worst case that one of the first $d + 1$ nodes receives the update packet from all its neighbors at the same time. Meanwhile, this node should forward the packet to all its neighbors after receiving it for the first time. In this case, the cost of network bandwidth of this node is expected to be:

$$C = 2S \max_{0<i\leq N}(E(n_i)) = 2dS_0N\left(1 + \sum_{l=d+2}^{N}\frac{1}{l-1}\right)\left(1 - e^{-\lambda}\right) \tag{6.2}$$

Suppose that the bandwidths of the nodes are independent, and each can most spend 10% of its total bandwidth B on flooding control packets, we have the restriction:

$$C \leq B \times 10\% \tag{6.3}$$

When B = 10 Mb/s, S_0 = 256 bits, d = 5, λ = 1/1,800 (the duration time of each node is 0.5 h on average), from Eqs. 6.2 to 6.3, the maximum node number that the P2P network can support in the worst case is 6.5×10^4. This result shows that the P2P network used in PS64 scales enough to meet the scalability requirements during IPv6 transition.

Prototype and Experiments

PS64 Prototype

Up to now, the architecture and protocol details of PS64 have been described. A Linux-based prototype system has also been developed. As shown in Fig. 6, our implementation on the edge gateways of IPv6 islands has four functional modules.

(1) P2P System;
(2) RT Control;
(3) Tunnel Table;
(4) Virtual Interface.

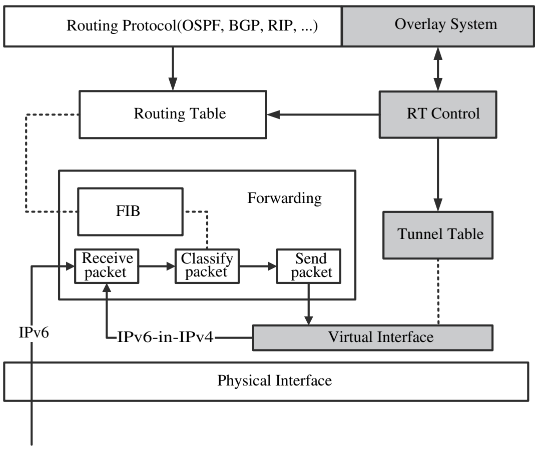


Fig. 6 Modules of PS64 prototype

The P2P System module processes configuration commands that define the information of tunnel end point of local islands edge gateways and gets the TEP information of other islands from an overlay P2P network.

The RT Control module receives the TEP information from the P2P System, forms the Tunnel Table, and inserts the forwarding information into the Routing Table of the gateway in order to assign the direct tunnels with higher route precedence than other routes and to make the traffic to the destination islands all forwarded to the Virtual Interface.

The Tunnel Table module holds the information of direct tunnels between IPv6 islands, and points out the IPv4 address of the destination gateway for the Virtual Interface when forwarding packets.

When the Virtual Interface receives a data packet to a destination IPv6 island, it looks up an entry in the Tunnel Table to find out the IPv4 address of the remote island gateway, encapsulates this packet with an IPv4 header, and sends it back to the forwarding scheme of the local gateway.

Experiments

With this prototype system, the PS64 algorithm has already been deployed on four isolated IPv6 island gateways inside TUNET (a campus IPv4 network). Each island has some inner IPv6 users and services (i.e., web, ftp, media steaming, etc), and is connected with the same relay gateway through an IPv6-in-IPv4 tunnel to access the native IPv6 network—CERNET2 (a national wide IPv6 backbone), as show in Fig. 7. Both, the island and relay gateways, have a 100 Mbps Fast Ethernet link with

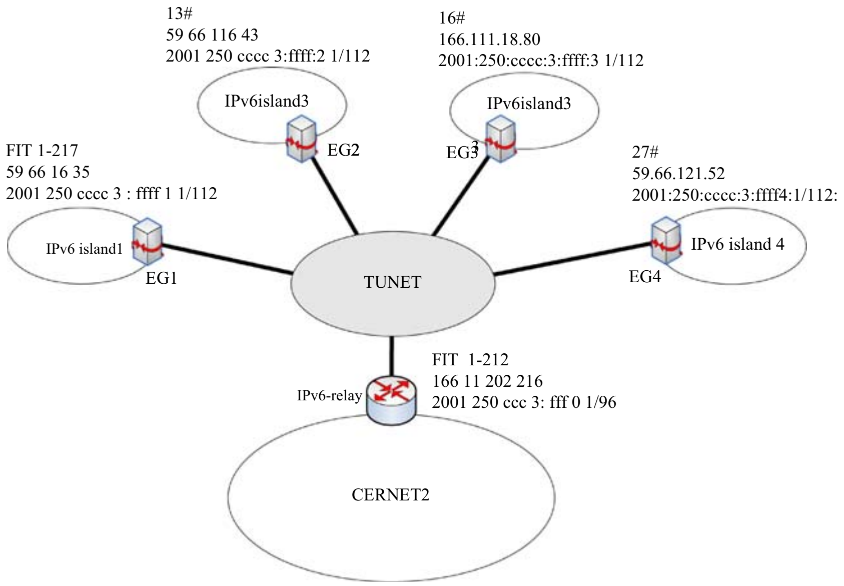


Fig.7 Experiment topology for PS64

the IPv6 network each. We performed the following two experiments to validate the effect of PS64 in reducing the load of the relay gateway and increasing the access performance of isolated islands.

Experiment 1

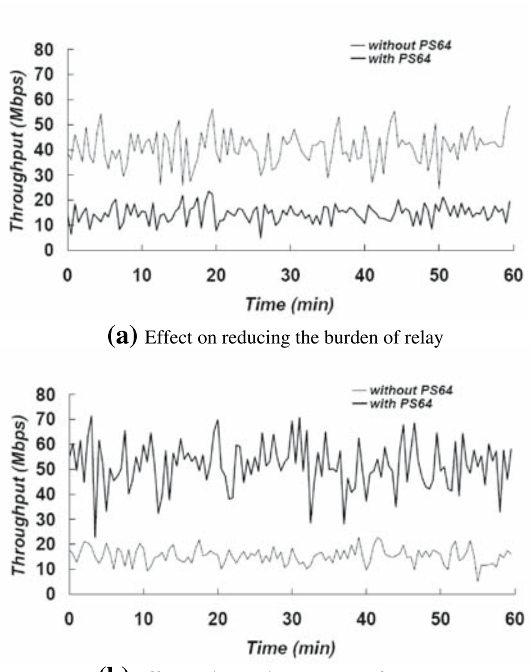
In this experiment, we limit the bandwidth of IPv6 island gateways to 10 Mbps. This makes the bottleneck of this system not the relay but the island edge gateways. That means no matter whether PS64 is used, the throughput of island gateways will be limited to 10 Mbps. In this way, we can clearly notice the effect of PS64 on reducing the load of the relay gateway.

We measured the throughput of the relay gateway with/without PS64 used on the island gateways for one hour. As shown in Fig. 8a, the throughput has an average bit rate of about 40 Mbps without PS64 and only about 15 Mbps with PS64. This shows that the PS64 algorithm can greatly reduce the load on the relay. Besides, from this experiment, we can see that only 37.5% of the traffic is outside the scope of the IPv4 islands, which is consistent with our assumption in section ''The PS64 Algorithm'' that ''the traffic forwarded by the relay gateways will be mostly restricted between IPv6 islands''.

Experiment 2

In this experiment, we focus on the throughput of the IPv4 island gateways with no limit at all. The bit rate of one island gateway in one hour is shown in Fig. 8b. From

Fig. 8 Experiment results: Exp.1 and Exp.2



(a) Effect on reducing the burden of relay

(b) Effect on increasing access performance this result, we can see that the throughput changes from about 16 Mbps to 58 Mbps in average when PS64 is in use. In other words, the PS64 algorithm can effectively increase the access performance of isolated islands.

Future Work

So far, the architecture and protocol details of PS64 have been designed and a prototype system has been developed. In the future, we will do some further study in the following directions:

(1) *Various kinds of P2P networks*. Which kind of P2P network is used in the PS64 algorithm is not a key aspect of this method. PS64 now uses an unstructured P2P network to propagate the TEP information. In the future, we would like to do more simulations and experiments to find out the different behaviors with different kinds of P2P networks, including DHTs.
(2) *Forwarding Policy*. PS64 currently increases the access performance of IPv6 island gateways by simply sending the packet between IPv6 islands through direct tunnels. However, since direct routing is not always better if links are congested, PS64 might make some routing performance worse in some situations. Some metric detection scheme should be introduced into PS64 to try to find the best routing for each packet.
(3) *Applicability extension*. The design and implementation of PS64 currently depends on IPv6-in-IPv4 tunneling technology. More work will be done to extend PS64 to support different types of encapsulation, such as GRE or IPv6-in-UDP-in-IPv4.
(4) *Security consideration*. We will also study on security enhancements of our algorithm. Since the edge gateways of isolated islands are sometimes lightweight devices and cannot support the heavy overhead of an end-to-end security scheme like IPSec between each pair of TEPs [20], the SPM [21] technology can be deployed to provide a lightweight security guarantee for data transmission. The security consideration about the P2P network will also be an important research topic.

Conclusion

In this paper, we present a peer-to-peer based method PS64 to provide access service for isolated IPv6 islands inside the IPv4 networks. This algorithm distributes the traffic between IPv6 islands directly through the IPv4 network and mitigates the load of IPv6-relay gateways of service providers.

The connectivity of the P2P network used in PS64 and the scalability of the proposed algorithm has been analyzed. The simulation results show that the P2P network can keep connectivity with the Random Recovery Scheme; and the theoretical analysis indicates that the PS64 algorithm can meet the scalability requirements during IPv6 transition. In other words, the proposed algorithm is reliable and scalable. The results of the experiments carried out with a prototype system on four isolated IPv6 islands inside TUNET show that the proposed algorithm can greatly reduce the reliance of the relay gateways and increase the access performance of IPv6 islands.

Obviously, the PS64 approach works for all IPvX over IPvY situations. Without any special requirement, PS64 can be widely deployed on the gateways of IPvX islands to provide high performance IPvY access service.

References

1. Deering, S., Hinden, R.: Internet Protocol, version 6 (IPv6) specifications. RFC 2460, December 1998
2. Nordmark, E., Gilligan, R.: Basic Transition Mechanisms for IPv6 Hosts and Routers, RFC 4213, October 2005
3. Oram, A.: Peer-to-Peer: Harnessing the Power of Disruptive Technologies. O’Reilly Press (2001)
4. Templin, F., Gleeson, T., Talwar, M., Thaler, D.: Intra-site Automatic Tunnel Addressing Protocol (ISATAP), RFC 4214, October 2005
5. Huitema, C.: Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs), RFC 4380, February 2006
6. Davies, E., Krishnan, S., Savola, P.: IPv6 Transition/Co-existence Security Considerations, draft-ietf-v6ops-security-overview-06.txt, Work in Progress, October 22, 2006
7. Carpenter, B., Moore K.: Connection of IPv6 Domains via IPv4 Clouds, RFC 3056, February 2001
8. Savola, P., Patel, C.: Security Considerations for 6to4, RFC 3964, December 2004
9. Durand, A., Fasano, P., Lento, D.: IPv6 Tunnel Broker, RFC 3053, January 2001
10. Wu, J., Cui, Y., Li, X., Metz, C., Barber, S., Mohapatra, P., Scudder, J.: Softwire Mesh Framework, draft-ietf-softwire-mesh- framework-00. txt, Work in Progress, March 2007
11. De Clercq, J., Prevost, D.S., Le Faucheur, F.: Connecting IPv6 Islands over IPv4 MPLS using IPv6 Provider Edge Routers (6PE), RFC 4798, February, 2007
12. Zhou, L., van Renesse, R.: P6P: A Peer-to-Peer Approach to Internet Infrastructure, IPTPS’04 (2004)
13. Stoica, I., Morris, R.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, SIGCOM’01 (2001)
14. Moy, J.: OSPF Version 2, RFC 2328, April 1998
15. Baran, P.: On distributed communication networks. IEEE Trans. Commun. Syst. **CS-12**(1–2), 1–9 (1964)
16. Krivelevich, M., Sudakov, B., Vu, V.H.: A sharp threshold for network reliability, combinatorics. Probab. Comput. **11**, 465–474 (2002)
17. Kermarrec, A.-M., Massoulié, L.: Probabilistic reliable dissemination in large-scale systems. IEEE Trans. Parallel Distributed Syst. **14**, 248–258 (2003)
18. Ruiz-Sanchez, M.A.: Survey and taxonomy of IP address lookup algorithms. IEEE Network **15**, 8–23 (2001)
19. Guterman, J.: Gnutella to the Rescue? Not so Fast, Napster fiends. http://www.gnutella.wego.com, September 2000
20. Graveman, R., Parthasarathy, M., Savola, P., Tschofenig, H.: Using IPsec to Secure IPv6-in-IPv4 Tunnels, RFC 4891, May 2007
21. Bermlerand, A., Levy H.: Spoofing Prevention Method, INFOCOM’05, 2005

Author Biographies

Xiaoxiang Leng received his B.S. degree from Department of Computer Science and Technology in Tsinghua University in 2004. From the year 2004, he began to be a M.S. candidate at Tsinghua Network Research Center. His research interests include network architecture, routing protocols, P2P and IPv6.

Jun Bi received the B.S., M.S. and Ph.D. degrees in Computer Science from Tsinghua University, Beijing, China. From 1999 to 2003, he worked for Bell Laboratories Research and Bell Labs Advanced Technologies, New Jersey, USA. Currently he is a full professor and director of Network Architecture & IPv6 research laboratory, Network Research Center, Tsinghua University. His research interests include IPv6 next generation network architecture and protocols.

Miao Zhang received his Ph.D degree in CS from Tsinghua University in 2004. From 2004 to 2006, he worked as Assistant Professor at Tsinghua Network Research Center. His research interests were in congestion control, network architecture, network security, network management, IPv6, multicast. In 2006, he began to work for Sogou.com.

Jianping Wu received the B.S., M.S. and Ph.D. degrees in Computer Science from Tsinghua University, Beijing, China. Currently, he is a full professor in the Computer Science Department, Tsinghua University. He is also the director of the China Education and Research Network. His current research interests include computer network architectures, next generation Internet, and formal methods.