CORE Metadata, citation and similar papers at core.ac.uk
Provided by ScholarBank@NUS

INVESTIGATION INTO PERFORMANCE OF IPV4 AND IPV6 TRANSITION MECHANISMS AND DISTRIBUTED NAT-PT IMPLEMENTATION

WANG WEI

NATIONAL UNIVERSITY OF SINGAPORE

2003

INVESTIGATION INTO PERFORMANCE OF IPV4 AND IPV6 TRANSITION MECHANISMS AND DISTRIBUTED NAT-PT IMPLEMENTATION

WANG WEI

(B.S. Nanjing University)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE
2003

To my parents

Acknowledgements

It has been more than one and a half year since I begun my research project on IPv4 to IPv6 transition mechanisms. Over this period of time, many people have contributed valuable help and advice in the course of my work.

First I would like to thank to my supervisor, Associate Professor A.L. Ananda, for his advice and technical direction. I am grateful that he made time in his busy schedule to give some important instruction of my research.

I must also thank to Wang Kai, former Research Assistant of Center of Internet Research, School of Computing, for his patience and guidance. His personal support and suggestion provided me with a precious learning and practicing experience in many unexpected areas besides the simple academic exercise I thought I was to embark on so long ago.

Last but not the least, I would like to thank Lai Zit Seng, Michael Yuan, Dai Yifan, Shao Tao, Zhang Xiaofeng, Dou Qingfeng, Shao Ning, Aurbind Shama, Venkatesh S. Obanaik, and many other members of CIR, who have aided me in one way or another. Without your kind assistance, I could not finish this project smoothly.

Contents

List of Figures

List of Tables

Summary

The limited size and structure of the Internet address space of current IP protocol, or IPv4, has caused difficulties in coping with the explosive increase in the number of Internet users. IPv6 is a feasible solution for the problems, which provides sufficient address space and brings on many other improvements as well. To achieve interaction between IPv4 and IPv6, some solutions have been proposed, each of them has its specific applicable scenario. Current widely applied IPv4 networks and services could not be completely replaced with IPv6 overnight, so efficient interworking between IPv4 and IPv6 is crucial for smooth transition from IPv4 to IPv6. Researching on network performance under these specific transition mechanisms is significant.

So far, a variety of techniques have been identified and implemented, basically falling into three categories: dual stack techniques, tunneling techniques and translation techniques. Compared with the most direct technique, dual stack, other two transition mechanisms theoretically result in performance decline in a way. Investigating end-to-end network performance under these mechanisms can help us precisely evaluate these special transition implementations respectively. In this thesis, we conduct TCP performance testing of three kinds of typical transition mechanisms, i.e. IPv6 over IPv4 configured tunneling, IPv4 over IPv6 configured tunneling, and NAT-PT connecting IPv4 and IPv6, which have representative application scenarios in different IPv4 and IPv6 transition phases.

To process quantitative analysis of effect on performance of each transition technique, we introduce a new criterion – transition efficiency. According to Our experiment results, each technique does induce performance decline, but their effect degrees are not uniform. Tunneling techniques generally present better performance than translation techniques. We analysed some factors that probably result in performance difference among these translation techniques and suggested some proposals for performance improvement. These results will lead to a better understanding of the theoretical and empirical properties of IPv4 and IPv6 integration technique from a comprehensive perspective.

According to our test results, NAT-PT is a comparatively less efficient transition solution. Meanwhile, translation technique has to track the sessions it supports and mandates so that inbound and outbound datagrams pertaining to a session have to traverse the same NAT-PT node, which further aggravates the network bottleneck and turns to be single point of failure. In this thesis, we improve NAT-PT performance by transforming the centralized system into a distributed system. As a distributed system, it has many advantages, such as higher reliability, load balancing, and convenient system augment.

We also conduct a set of experiments to compare TCP performance of distributed NAT-PT and centralized NAT-PT. Our experimental results show that, although distributed NAT-PT consumes some resource to implement mapping table synchronization which unavoidably affects translation performance, the effect is limited which results in 1% – 12% performance reduction. With load balance of network traffic, the overall performance of distributed NAT-PT presents obvious improvement over the original centralized NAT-PT system. In addition, this type of distributed system is convenient to combine with our existing dynamic IPv4 over IPv6 tunneling system – TwinGlass – and thus provides an integrated transition solution for future IPv4 to IPv6 migration.

Chapter 1 Introduction

1.1 A brief history of the Internet Development

The Internet, which plays an important role in our life, grew out of early attempts to link computers and enables them to share information and use common applications. The history of the Internet can be divided roughly into three phases. Figure 1.1 illustrates the three phases.

1.1.1 The Computer Age

The first development phase began in the U.S. in 1969 with the connection of four nodes in ARPANET (Advanced Research Projects Agency Network). The goal of this project was to connect computers and enable users to share applications and resources.

In the years that followed, more nodes were connected and new protocols such as FTP (File Transfer Protocol) were developed. The first paper to describe TCP (Transmission Control Protocol), the basis of today's Internet, was published by Vinton Cerf et al. in 1974 [1]. In 1983, ARPANET adopted and standardized the U.S. Department of Defense's (DoD) TCP/IP protocols, which became the de facto protocols for the Internet. All the agency's computers were then linked to each other using similar protocols, and the same TCP/IP interface started to be used in ordinary Personal Computers (PC).

1.1.2 Information Retrieval

As the Internet grew, several new protocol and mechanisms were introduced. New ways to present information, such as Hyper Text Markup Language (HTML), were invented, and browsers that interpreted HTML made information retrieval easy for everyone. The Internet became more popular and was eventually opened up for commercial use in the mid-1990s.

This resulted in an increased demand for IP address, as many government, commercial, nonprofit and volunteer organizations set up their own websites to provide easy access to information about their products and activities. Internet service providers also began to offer services to individuals, putting further pressure on the available addresses.

1.1.3 Person-to-person Communications

The dramatic growth of wireless communication has driven the third phase in the development of the Internet and the need for a new IP version. Mobile access to the Internet already enables web browsing and email services for mobile users. The Mobile Internet adds even greater capabilities for services requiring person-to-person data connectivity over mobile phones and other devices. The General Packet Radio System (GPRS) [2], Third Generation Mobile Telecommunication (3G) [3], and other packet-based mobile networks, all provide permanent IP-based connections. As these services proliferate, the need for IP addresses is growing accordingly.

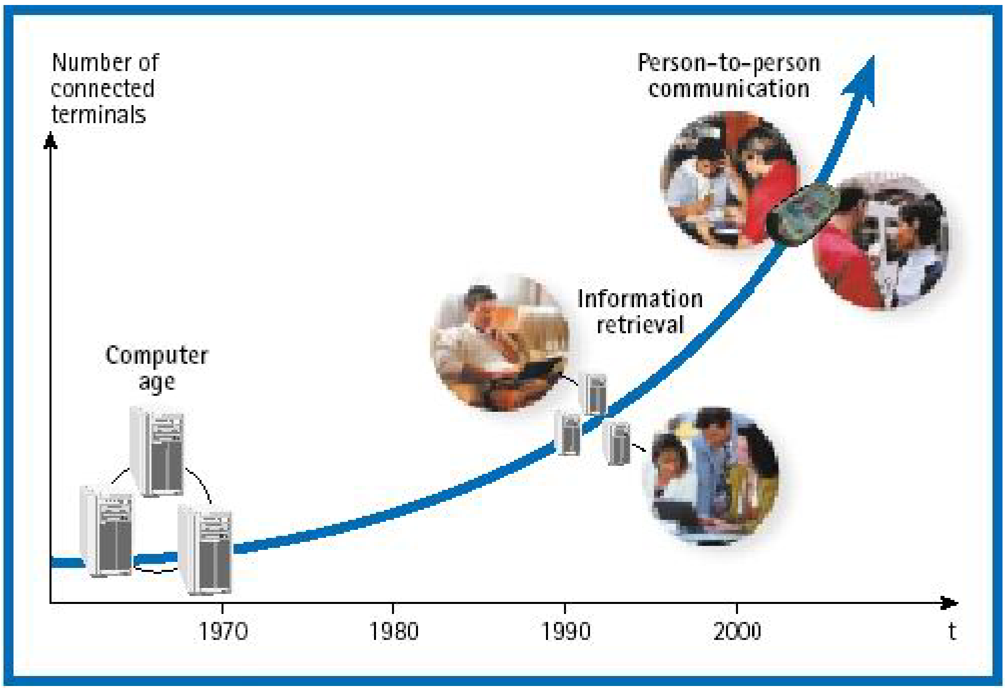

Figure 1.1. Internet History: From computer communication to person-to-person communication.

From the course of the Internet development, we can obviously notice that current Internet Protocol is deficient in terms of address capability. Actually, there are other facets that need to be improved in IPv4. We take a close look at these issues in the next section.

1.2 New Trends and Requirements for IP

First published in 1980's, IPv4 has been remarkably resilient in spite of its age, but it is beginning to encounter problems.

As stated in the last section, more and more nodes are connecting into the Internet community. The available addresses for the new nodes are becoming less and will eventually deplete in the near future. Dynamic Host Configuration Protocol (DHCP) [4] and Network Address Translation (NAT) [5] may alleviate the shortage problem in a way, but these are not ultimate solutions. Furthermore, implementing these mechanisms in network community brings on new side effects simultaneously. IP mobility, for example, could not be supported successfully under NAT addressing scheme: Mobile IP requires a permanent global IP address for each device, and thus could not apply widely in IPv4 address scheme – a shortage of globally routable IPv4 address and the use of private IPv4 address with NAT hampers Mobile IPv4 deployment in many cases.

Another problem is the more and more complicated routing tables especially in the backbone routers. With the rapidly increasing number of joining nodes, the size of routing tables explodes accordingly. Network maintenance becomes a tough assignment, which may even turn to be a nightmare. How to simplify address configuration and network management is a critical matter for continuous growing of the Internet, which is now spreading into the less developed world and drawing many autonomous and intelligent devices into the system.

The third facet results from the new applications, such as Quality of Service (QoS) and mobility. Although IPv4 can provide some remedial resolutions to meet such application requirement, it yet remains an inefficient and defective protocol that could hardly fulfill more complicated requirement for future applications. IPv4 was devised more than 20 years ago after all. As many famous quotes, like "I think there is a world market for maybe five computers" by Thomas Watson in 1943 or "640K should be enough for anybody" by Bill Gates in 1981, has proved to be wrong and even sound ridiculous nowadays, it is not likely to expect perfect matching between the twenty-year-old protocol and the current application requirements.

Hence, the network world started hunting for a new addressing mechanism. It was destined to find a specific future direction for the replacement of the current version of IP, and the feasible result should be IPv6.

1.3 Advances of IPv6

IPv6 [6] is short for “Internet Protocol Version 6”. It is designed by the IETF to solve many of the problems of the current version of IPv4 with regard to address depletion, security, auto-configuration, extensibility, mobility, and more. Its use will also expand the capabilities of the Internet to enable a variety of valuable and exciting scenarios, including peer-to-peer communication and mobile applications. Instead of being derived from an entire brand new concept, IPv6 is evolution – not revolution- because it changes implementation details but the basic concepts remain the same, and thus it helps the Internet to scale to new users and new services. This kind of concept similarity is also helpful for smooth transition from IPv4 to IPv6, eventually leading the Internet into the IPv6 dominated world. The following sections explain some benefits of IPv6.

1.3.1 Scalability

IPv6 address [7] has 128-bit address space, which is 4 times wider in bits compared with IPv4’s 32-bit address space, allowing a wide variety of different devices to be allocated their own global IP addresses. The enormous number of IP addresses makes many new exciting application and service possible, such as peer-to-peer communication, mobile IP and more.

1.3.2 Clearer specification and optimization

IPv6 follows good practices of IPv4, while rejecting minor flaws and obsolete items. IPv6 streamlines and enhances the basic header layout of the IP packet by omitting optional items and just retaining key components for the common use. Figure 1.2 shows the IPv4 and IPv6 packet header comparison. This simplified packet structure is a major improvement over IPv4 and will help offset the bandwidth cost of the longer IPv6 address fields. Meanwhile, the fewer fields and fixed length of the IPv6 header enable the implementation of simple hardware based routers. Unlike IPv4, IPv6 does not fragment packets when they are routed, further decreasing the routing burden. These improvements mean IPv6 can handle the exponential growth of Internet traffic in a cost-effective manner.

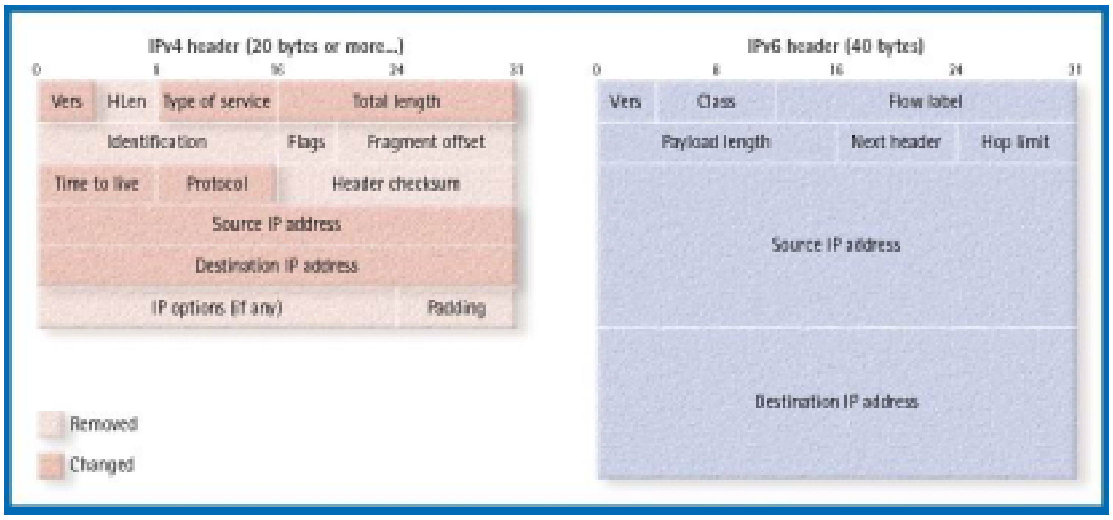

Figure 1.2. IPv4 and IPv6 packet header comparison

1.3.3 Autoconfiguration

IPv6 enables autoconfiguration [8] of different type of addresses for host interfaces. Besides aggregatable global unicast address, which is the default IPv6 address type, there are three more types of addresses for host interfaces, including on-site addresses, broadcast addresses and mobility addresses. A node first identifies its link-local address using a neighbor discovery mechanism. Once this is achieved, another mechanism, which is essentially a plug and play feature requiring no manual intervention from users or operators, acquires the unique global routable address. Compared with stateful mechanism such DHCP, this process requires less operations and maintenance effort.

1.3.4 Mobility

Mobile IP [9] provides users the freedom to roam beyond their home subnet while consistently maintaining their home IP address. Mobile IP enables transparent routing of IP datagrams to mobile users during their movement, so that data sessions can be initiated to them while they roam; it also enables sessions to be maintained in spite of physical movement between points of attachment to the Internet or other networks. The integrated IP layer mobility enabled by the mobile IPv6 protocol will offer crucial advantages, especially as the number of mobile terminals continues to grow. Although a similar mobile protocol exists in the IPv4 world, there is one fundamental difference: mobile IPv4 cannot cope with a large number of terminals. Figure 1.3 illustrates logical concept of IPv6 mobility.

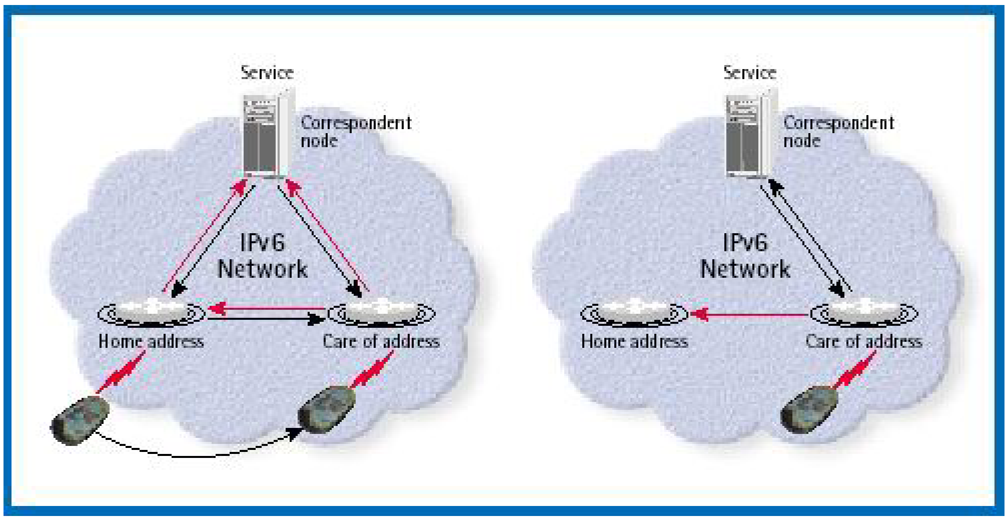

Figure 1.3. Mobile IPv6.

1.3.5 Qos Consideration

IPv6 provides Quality of Service (QoS) function with added advantages in the area of service differentiation. These benefits derive from a field called a flow label. This 20-bit long field, embedded in IPv6 header, enables the identification and differentiated treatment of any IP address flow in the intermediate nodes. Although the exact use of the flow label has not yet been standardized, it may support new pricing systems based on the level of service and other criteria in the future. Additionally, IPv6 also helps improve quality of service in several other ways, mainly by enabling always-on connections, preventing service breaks and enhancing network performance. Better network and service quality, in turn, raise the expectation of value, improve customer satisfaction and generate a higher return on relationship.

1.3.6 Security

IPSec [10] is a framework of open standards developed by the IETF that provide security for transmission of sensitive information over unprotected networks such as the Internet. With IPSec, data can be sent across a public network without fear of observation, modification, or spoofing. IPSec functionality is essentially identical in both IPv6 and iPv4; however, IPSec in IPv6 can be developed from end-to-end – data may be encrypted along the entire path between a source node and destination node. (Typically, IPSec in IPv4 is deployed between border routers of separate networks.) IPv6 includes security in the basic spec. It includes encryption of packets (ESP: Encapsulated Security Payload) and authentication of the sender of packets (AH: Authentication Header).

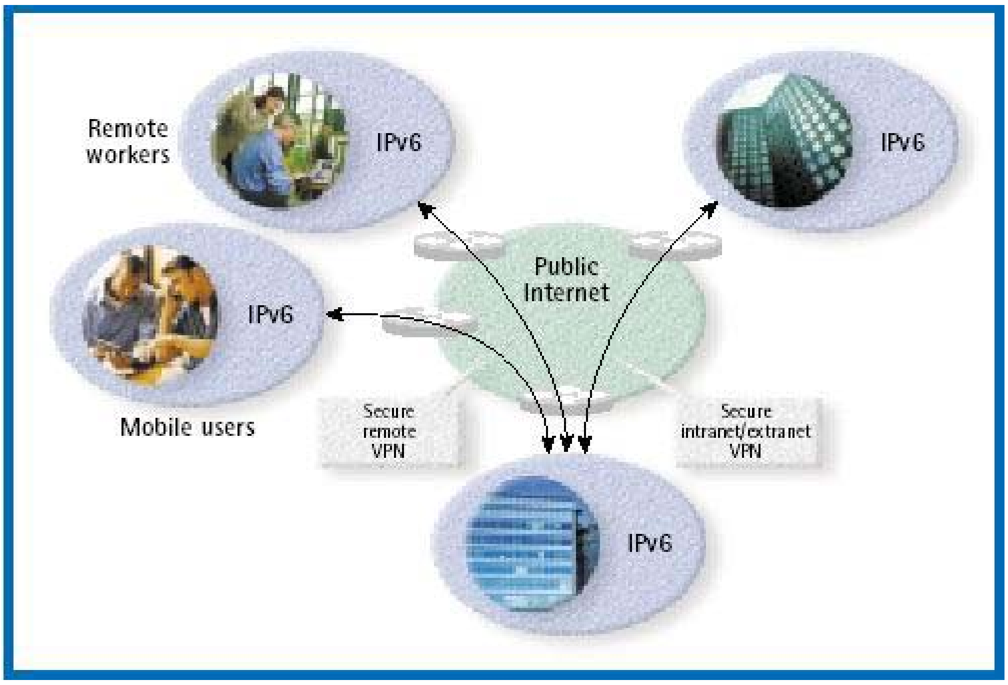

Figure 1.4. Secure VPN with IPSec forIPv6

1.4 *Thesis Objectives*

As IPv4 applications and services could not be replaced by IPv6 ones overnight, integration and coexistence with IPv4 is a prerequisite to enable the smooth transition of current network to IPv6. So far, a variety of techniques, which will be discussed in Chapter 2, have been identified and implemented, basically falling into three categories: dual stack techniques, tunneling techniques and translation techniques.

Compared with the most direct technique, dual stack, other two transition mechanisms theoretically result in performance decline in a way. Researching on network performance under these specific transition mechanisms is significant. In this project, our objectives includes the following:

- Investigating end-to-end network performance of three typical transition mechanisms.

- Analyzing possible factors that lead to performance difference among these techniques
- Implementing enhancement on certain transition mechanism.

1.5 Thesis Contributions

In this thesis, we conduct TCP performance testing of three kinds of typical transition mechanisms, i.e. IPv6 over IPv4 configured tunneling, IPv4 over IPv6 configured tunneling, and NAT-PT connecting IPv4 and IPv6, which have representative application scenarios in different IPv4 and IPv6 transition phases. Our experiment results show that, although each technique does induce performance decline, the effect degrees are not uniform. Tunneling techniques generally present better performance than translation techniques. We analyze some factors that probably result in performance difference among these translation techniques and suggest some proposals for performance improvement. These results will lead to a better understanding of the theoretical and empirical properties of IPv4 and IPv6 integration technique from a comprehensive perspective.

According to our test results, NAT-PT is a comparatively less efficient transition solution. In this thesis, we improve NAT-PT performance by transforming centralized system into distributed system. As distributed systems possess many advantages over centralized systems, our distributed NAT-PT has advantages over single node NAT-PT system, such as higher reliability, load balancing, and incremental growth. We also conduct a couple of experiments to test TCP performance of distributed NAT-PT and centralized NAT-PT. Our experimental results show that, although distributed NAT-PT consumes some resource to implement mapping table synchronization which unavoidably affects translation performance, the effect is limited which results in 1% – 12% performance reduction. In addition, this type of distributed system is convenient to combine with our existing dynamic IPv4 over IPv6 tunneling system – TwinGlass – and thus provides an integrated transition solution for future IPv4 to IPv6 migration.

1.6 Thesis Walkthrough

The remainder of this thesis is organized as follows.

Chapter 2 introduces prospective three main phases of IPv4 to IPv6 transition period, followed by detailed description of various identified and implemented transition techniques.

Chapter 3 explains on motivation and implementation issues of improvement for certain transition technique solution.

Chapter 4 describes major parts of our testbed infrastructure and experiment design principle. Then presents our experimental results by groups.

Chapter 5 discusses the experimental results, suggesting some factors which may result in performance difference among these transition mechanisms.

Chapter 6 summarizes the work that has been done in this project, and finally draws our conclusion.

Chapter 2 IPv4 and IPv6 Transition Mechanisms

IPv6, proposed as the substitute for IPv4, fixes the problem of limited address number in IPv4. It also adds many improvements such as auto-configuration, security and mobility. Migrating from IPv4 to IPv6 in an instant is impossible because of huge size of the Internet and of the great number of IPv4 users. Moreover, many organizations are becoming more and more dependent on the Internet for their daily work, and they therefore cannot tolerate downtime for the replacement of the IP protocol. As a result, there will not be one special day on which IPv4 will be turned off and IPv6 turned on. As current IPv4 network and services will exist for quite a long time, the transition period will be lengthy. We can roughly divide the period into three phases.

2.1 IPv4 and IPv6 Transition Phases

Figure 2.1 gives a simple picture of the transition phases. These are described from a private network point of view but the principles are also applicable for other network types.

The starting position (the IPv4 world) is the network supporting only IPv4. All the terminals connected to the Internet are native IPv4 equipment. Network Address Translators (NATs) may be used due to the limited amount of available public IP addresses.

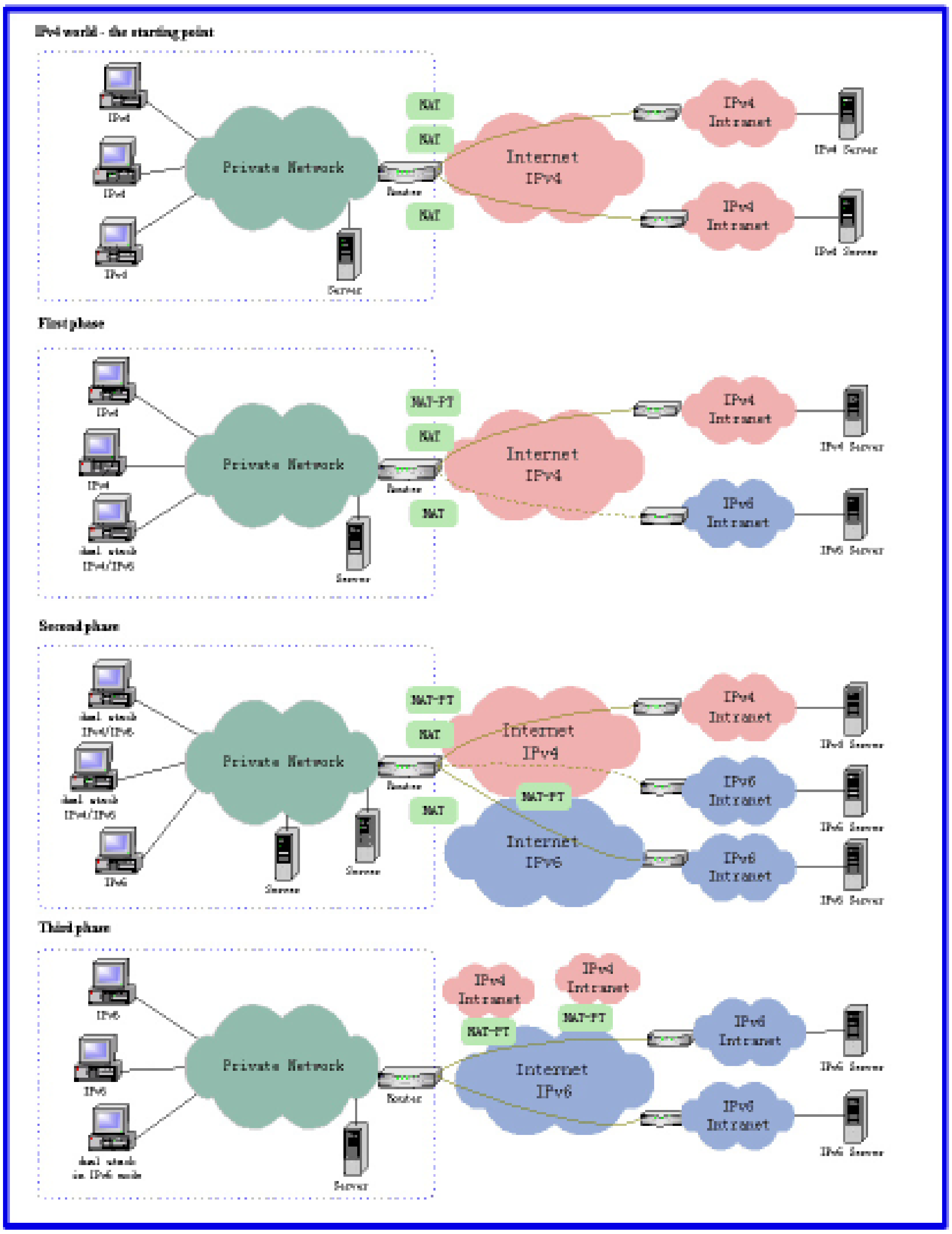

Figure 2.1. IPv4 to IPv6 transition phases

In the first phase, there are separate IPv6 islands in the network, connected by IPv4 Internet using automatic or configured "IPv6 in IPv4" tunneling. Some IPv6 services are provided to users within the private network in this phase. Other IPv6 connections, such as accessing a remote IPv6 server of another IPv6 network, are reached by configured or automatic IPv6 over IPv4 tunnels over the IPv4 Internet: conventional IPv4 services are provided to users having IPv4 or dual stack terminals. There can also be NATs in the operator network that deal with the limited pool of public IPv4 addresses by distributing temporary ones. Also translators such as NAT-PT can be installed in the operator network to perform the IPv4-IPv6 protocol translation.

In the second phase, IPv6 is widely deployed and numerous services are implemented on the IPv6 platform. IPv6 Internet has a wide deployment, but tunneling via IPv4 Internet is sometimes still needed as full connectivity. Implementing all new services on the IPv6 platform accelerates the IPv6 deployment. Mobile networks, for instance, help lead this development. Numerous conventional IPv4 services still exist and dual IPv4/IPv6 stacks are installed in many nodes.

In the third phases, IPv6 has achieved a dominant position. IPv6 Internet has global connectivity and all services work on the IPv6 platform. No dual stack functionality or addresses or protocol translators are vitally needed in the private network. This enables the simplification of the network architecture and leads to easier maintenance. Even in this phase, there are maybe some remnant IPv4 networks, which could access each other through automatic or configured IPv4 over IPv6 tunnels.

From above description, we notice that there are different transition requirements during different phases. Various types of techniques, in turn, should be implemented in corresponding scenarios. By far, a wide range of techniques have been identified and implemented, basically falling into three categories: Dual IPv4/IPv6 stacks in network elements and mobile terminals, tunneling whether automatic or configured, and IPv4-IPv6 protocol translators in the network. We elaborate on these three types of mechanisms respectively in the next few sections.

2.2 Dual Stack

Dual stack is also known as Dual IP layer. It is the most straightforward way for IPv6 nodes to remain compatible with IPv4-only node by providing a complete IPv4 implementation. IPv6 nodes that provide a complete IPv4 and IPv6 implementations are called “dual stack nodes”, which have the ability to send and receive both IPv4 and IPv6 packets. Dual stack is a preferred method on application’s servers. They can directly interoperate with IPv4 nodes using IPv4 packets, and also directly interoperate with IPv6 nodes using IPv6 packets. Choice of the IP version is based on name lookup or application preference.

As a technique for transition to IPv6, the dual IPv4 and IPv6 protocol stack technique enables gradual, one-by-one upgrades to applications running on nodes. Applications running on nodes are upgraded to make use of the IPv6 protocol stack. Applications that are not upgraded – they support only the IPv4 protocol stack – can coexist with upgraded applications on the same node. New and upgraded applications simply make use of both the IPv4 and IPv6 protocol stacks. (Figure 2.2.)

We illustrate dual stack mechanism with an example in Figure 2.3. An application that supports dual IPv4 and IPv6 protocol stacks requests all available addresses for the destination host name www.a.com from a DNS server. The DNS server replies with all available addresses, both IPv4 and IPv6 addresses, for www.a.com. The application chooses an address, mostly depending on the particular system, and connects the source node to the destination using the IPv4 or IPv6 protocol stack.

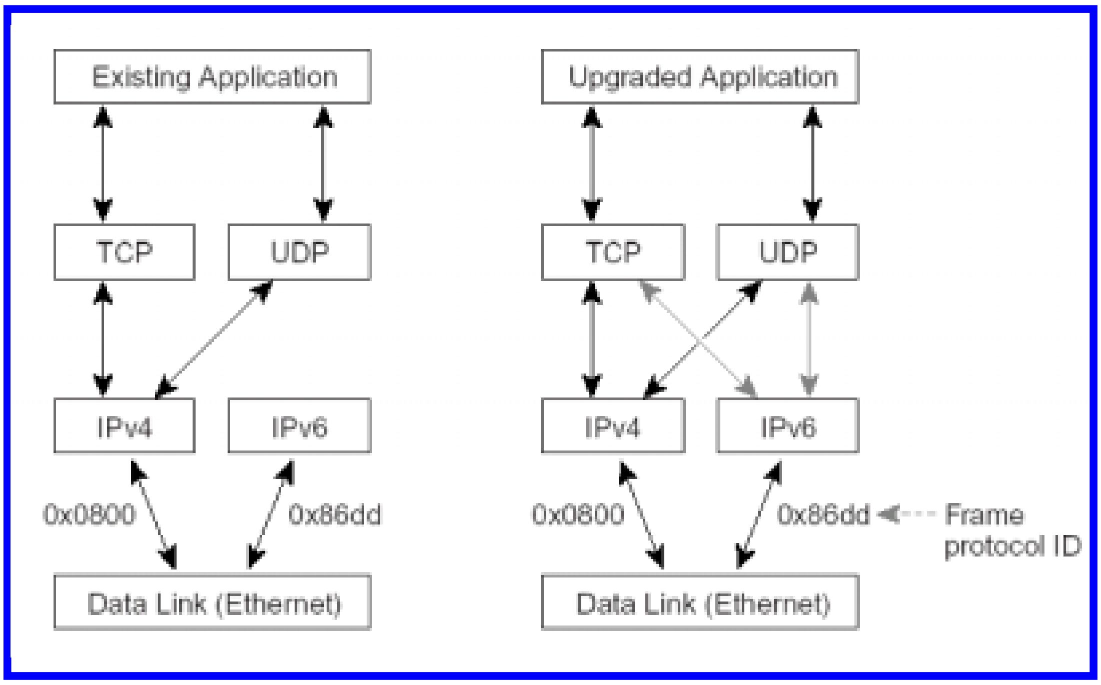

Figure 2.2. Dual IPv4 and IPv6 Protocol Stack Technique

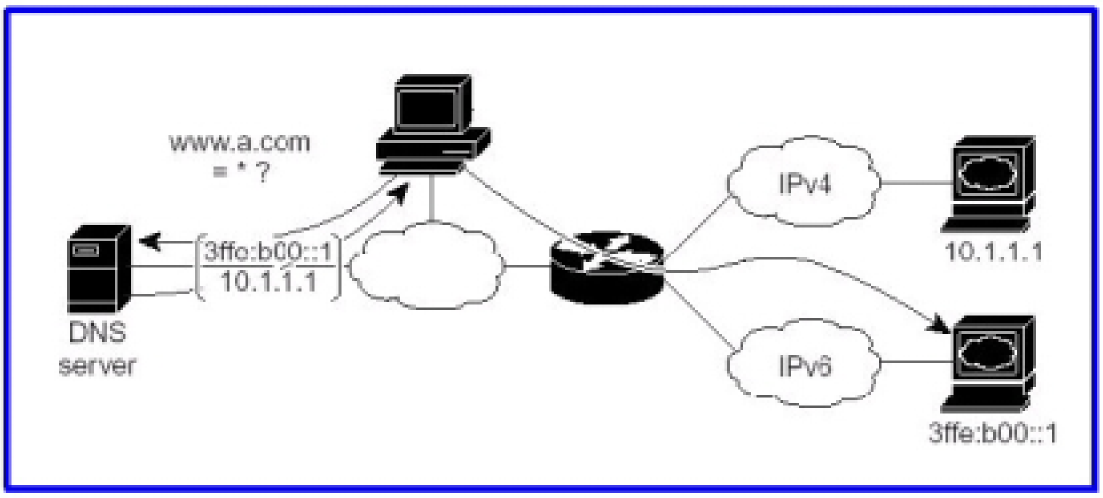

Figure 2.3. Dual IPv4 and IPv6 Protocol Stack Applications

Dual stack is the most straightforward method for the emerging new applications, which support both IPv6 and IPv4. With the ability of handling both of these two protocols, dual stack nodes can communicate with other nodes conveniently. Regarding most of existing old applications, however, it is extremely onerous to upgrade these applications and make them support IPv6. In some cases it is just an impossible mission. Besides, the new adding-on dual stack node should also be assigned an IPv4 address, which is unlikely feasible in case of the eventually exhausted available IPv4 addresses. Therefore, we have to seek other techniques. The following sections explain other two types of important transition techniques – tunnel and translator.

2.3 Tunnel

In most deployment scenarios, especially the first and second transition periods as described in section 2.1, the IPv6 routing infrastructure will be built up over time. While the IPv6 infrastructure is being deployed, the existing IPv4 routing infrastructure can remain functional, and can be used to carry IPv6 traffic. Tunnel provides a way to utilize an existing IPv4 routing infrastructure to carry IPv6 traffic. The same story happens in the third phase, when IPv6 has well developed and achieved a dominant position. Some IPv4 users and services, however, are still running and maybe will live forever. Tunnel technique presents its power in case of setting up connections between two isolated IPv4 islands through wide IPv6 sea. Tunnel provides a vital IPv6 migration mechanism. Many techniques are available to establish a tunnel [11][16]. We probe the internal mechanisms from the examples of IPv6 over IPv4 tunnels, sharing the common concepts with IPv4 over IPv6 tunnels.

IPv6 over IPv4 tunneling is the encapsulation of IPv6 packets with an IPv4 header so that IPv6 packets can be sent over an IPv4 infrastructure. Within the IPv4 header:

- The IPv4 Protocol field is set to 41 to indicate an encapsulated IPv6 packet.
- The Source and Destination fields are set to IPv4 addresses of the tunnel endpoints. The tunnel endpoints are either manually configured as part of the tunnel interface or are automatically derived from the sending interface, the next-hop address of the matching route, or the source and destination IPv6 addresses in the IPv6 header.

Figure 2.4 shows address transformation in IPv6 over IPv4 tunnel

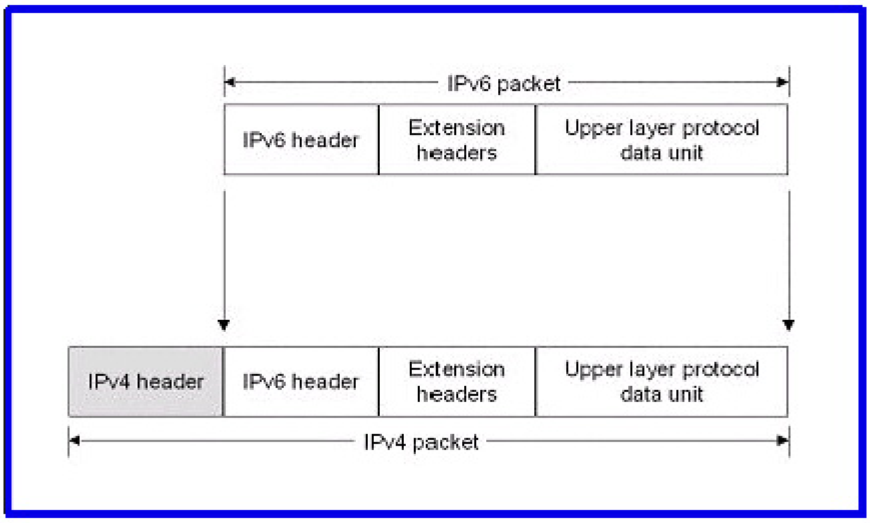

Figure 2.4. IPv6 over IPv4 Tunnel

2.3.1 Implementing Scenarios

IPv6/IPv4 hosts and routers can tunnel IPv6 datagrams over regions of IPv4 routing topology by encapsulating them within IPv4 packets. Tunneling can be used in a variety of ways:

- Router-to-Router.

IPv6/IPv4 routers interconnected by an IPv4 infrastructure can set up a tunnel for IPv6 packets between themselves. In this case, two IPv4 or IPv6 infrastructures are connected by two dual stack routers over an IPv4 infrastructure. The tunnel endpoints span a logical link in the path between the source and destination. The IPv6 over IPv4 tunnel between the two routers acts as a single hop. Routers within each IPv4 or IPv6 infrastructure point to the dual stack routers on the edge. For each dual stack router, there is a tunnel interface representing the IPv6 over IPv4 tunnel and routers that use the tunnel interface. (Figure 2.5.)

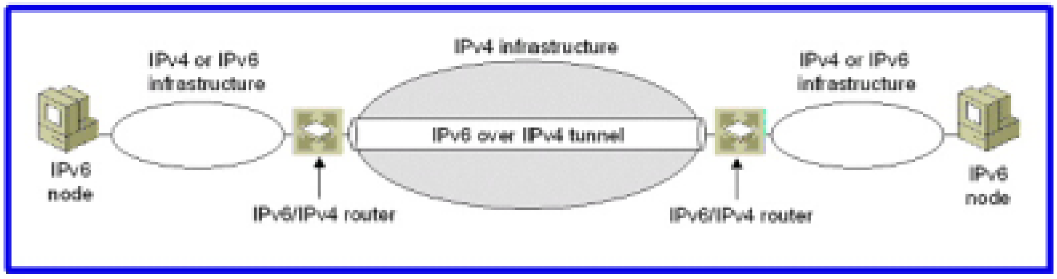

Figure 2.5. Router-to-Router Tunneling

- Host-to-Router or Router-to-Host.

In the host-to-router tunneling configuration, an IPv6/IPv4 node that resides within an IPv4 infrastructure creates an IPv6 over IPv4 tunnel to reach an IPv6/IPv4 router. The tunnel endpoints span the first segment of the path between the source and destination nodes. The IPv6 over IPv4 tunnel between the IPv6/IPv4 node and the IPv6/IPv4 router acts as a single hop.

On the IPv6/IPv4 node, a tunnel interface representing the IPv6 over IPv4 tunnel is created and a route (typically a default route) is added using the tunnel interface. The IPv6/IPv4 node tunnels the IPv6 packet based on the matching route, the tunnel interface, and the next-hop address of the IPv6/IPv4 router.

In the router-to-host tunneling configuration, an IPv6/IPv4 router creates an IPv6 over IPv4 tunnel across an IPv4 infrastructure to reach an IPv6/IPv4 node. The tunnel endpoints span the last segment of the path between the source node and destination node. The IPv6 over IPv4 tunnel between the IPv6/IPv4 router and the IPv6/IPv4 node acts as a single hop.

On the IPv6/IPv4 router, a tunnel interface representing the IPv6 over IPv4 tunnel is created and a route (typically a subnet route) is added using the tunnel interface. The IPv6/IPv4 router tunnels the IPv6 packet based on the matching subnet route, the tunnel interface, and the destination address of the IPv6/IPv4 node. Figure 2.6 shows host-to-router (for traffic traveling from Node A to Node B) and router-to-host (for traffic traveling from Node B to Node A) tunneling.

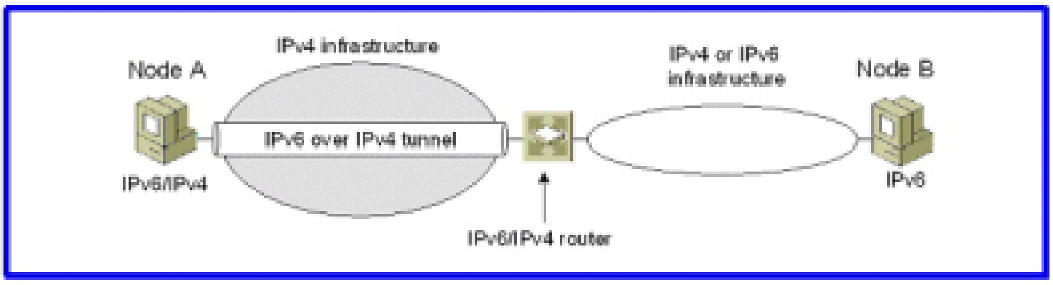

Figure 2.6. Host-to-Router and Router-to-Host Tunnel

- Host-to-Host.

IPv6/IPv4 hosts that are interconnected by an IPv4 infrastructure can tunnel IPv6 packets between themselves. In this case, the tunnel spans the entire end-to-end path that the packet takes. In the host-to-host tunneling configuration, an IPv6/IPv4 node that resides within an IPv4 infrastructure creates an IPv6 over IPv4 tunnel to reach another IPv6/IPv4 node that resides within the same IPv4 infrastructure. The tunnel endpoints span the entire path between the source and destination nodes. The IPv6 over IPv4 tunnel between the IPv6/IPv4 nodes acts as a single hop.

On each IPv6/IPv4 node, an interface representing the IPv6 over IPv4 tunnel is created. Routes might be present to indicate that the destination node is on the same logical subnet defined by the IPv4 infrastructure. Based on the sending interface, the optional route, and the destination address, the sending host tunnels the IPv6 traffic to the destination. Figure 2.7 shows host-to-host tunneling.

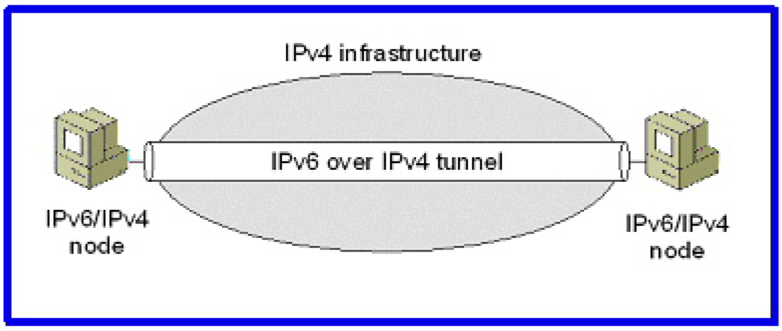

Figure 2.7. Host-to-Host Tunnel

2.3.2 Two Types of Tunnels

Tunneling techniques are usually classified according to the mechanism by which the encapsulating node determines the address of the node at the end of the tunnel. In the first two tunneling methods listed above, router-to-router and host-to-router, the IPv6 packet is being tunneled to a router. The endpoint of this type of tunnel is an intermediary router which must decapsulate the IPv6 packet and forward it on to its final destination. When tunneling to a router, the endpoint of the tunnel is different from the destination of the packet being tunneled. So the addresses in the IPv6 packet being tunneled cannot provide the IPv4 address of the tunnel endpoint. Instead, the tunnel endpoint address must be determined from configuration information on the node performing the tunneling. We use the term "configured tunneling" to describe the type of tunneling where the endpoint is explicitly configured.

In the last two tunneling methods, host-to-host and router-to-host, the IPv6 packet is tunneled all the way to its final destination. In this case, the destination address of both the IPv6 packet and the encapsulating IPv4 header identify the same node. This fact can be exploited by encoding information in the IPv6 destination address that will allow the encapsulating node to determine tunnel endpoint IPv4 address automatically. Automatic tunneling employs this technique, using a special IPv6 address format with an embedded IPv4 address to allow tunneling nodes to automatically derive the tunnel endpoint IPv4 address. This eliminates the need to explicitly configure the tunnel endpoint address, simplifying configuration. We make a further explanation on these two types of tunnel techniques.

2.3.2.1 Automatic Tunnel

An automatic tunnel is a tunnel that does not require manual configuration. Tunnel endpoints are determined by the use of logical tunnel interfaces, routers, and source and destination IPv6 addresses. Automatic tunnels use “IPv4-compatible” addresses, which are hybrid IPv4/ IPv6 addresses. A compatible address is created by adding leading zeros to a 32-bit IPv4 address to pad it to 128 bits. Automatic tunnels are not associated to any distant end point.

For example, in a host-to-host automatic tunnel, when Host1 (with the public IPv4 addresses of 157.60.91.123 and corresponding IPv4-compatible address of ::157.60.91.123) sends traffic to Host2 (with the public IPv4 addresses of 131.107.210.49 and corresponding IPv4-compatible address of :: 131.107.210.49), the addresses in the IPv4 and IPv6 headers are as listed in Table 2.1.

Field	**Value**
IPv6 Source Address	::157.60.91.123
IPv6 Destination Address	:: 131.107.210.49
IPv4 Source Address	157.60.91.123
IPv4 Destination Address	131.107.210.49

Table 2.1. Example IPv6 Automatic Tunnel Addresses

Because IPv4-compatible addresses are only defined for public IPv4 addresses, they are not widely used. Although IPv4-compatible addresses are easy way to auto-tunnel, this mechanism may be deprecated soon. Furthermore, current IPv6 stacks assume a static IPv4 address at both ends of the tunnel to be established, a concept which does not apply to those machines who connect and disconnect through Internet Service Providers (ISP) and, in most cases, are assigned an IP address dynamically. So automatic tunnel has its applicability limitation.

2.3.2.2 Configured Tunnel

A configured tunnel requires manual configuration of tunnel endpoints. In a configured tunnel, the IPv4 addresses of tunnel endpoints are not derived from addresses that are encoded in the IPv6 source or destination addresses or the next-hop address of the matching route, but the encapsulating or decapsulating node.

Typically, router-to-router tunneling configurations are manually configured. The tunnel interface configuration, consisting of the IPv4 addresses of the tunnel endpoints, must be manually specified along with static routes that use the tunnel interface. When encapsulating an IPv6 packet in an IPv4 datagram, the added IPv4 header's source address and destination address fields are set as the IPv4 address of outgoing interface of the encapsulating node and the IPv4 address of tunnel endpoint respectively. The protocol field is set as 41. When the other end point receive an IPv4 datagram that is addressed to one of its own IPv4 address, and the value of the protocol field is 41, it reassembles if necessary, and then it remove the IPv4 header and submits the IPv6 datagram to its IPv6 layer code.

Although the two tunneling techniques – automatic and configured – differ primarily in how they determine the tunnel endpoint address, most of the underlying mechanisms are the same:

- The entry node of the tunnel (the encapsulating node) creates an encapsulating IPv4 header and transmits the encapsulated packet.

- The exit node of the tunnel (the decapsulating node) receives the encapsulated packet, reassembles the packet if needed, removes the IPv4 header, updates the IPv6 header, and processes the received IPv6 packet.

- The encapsulating node may need to maintain soft state information for each tunnel recording such parameters as the MTU of the tunnel in order to process IPv6 packets forwarded into the tunnel. Since the number of tunnels that any one host or router may be using may grow to be quite large, this state information can be cached and discarded when not in use.

2.4 Translator

NAT-PT [12] stands for Network Address Translation - Protocol Translation, using a pool of IPv4 addresses for assignment to IPv6 nodes on a dynamic basis as sessions are initiated across IPv4-IPv6 boundaries. NAT-PT binds addresses in IPv6 network with addresses in IPv4 network and vice versa to provide transparent routing for the datagrams traversing between two address realms. This is achieved using a combination of Network Address Translation (NAT) [13] and Protocol Translation (PT) [14]. This technique requires no changes to end nodes and IP packet routing and is completely transparent to end nodes. It does, however, require NAT-PT to track the sessions it supports and mandates that inbound and outbound datagrams pertaining to a session traverse the same NAT-PT router.

2.4.1 Network Address Translation

Figure 2.8 illustrates Network Address Translation mechanism of NAT-PT technique. We explain each step in the following details:

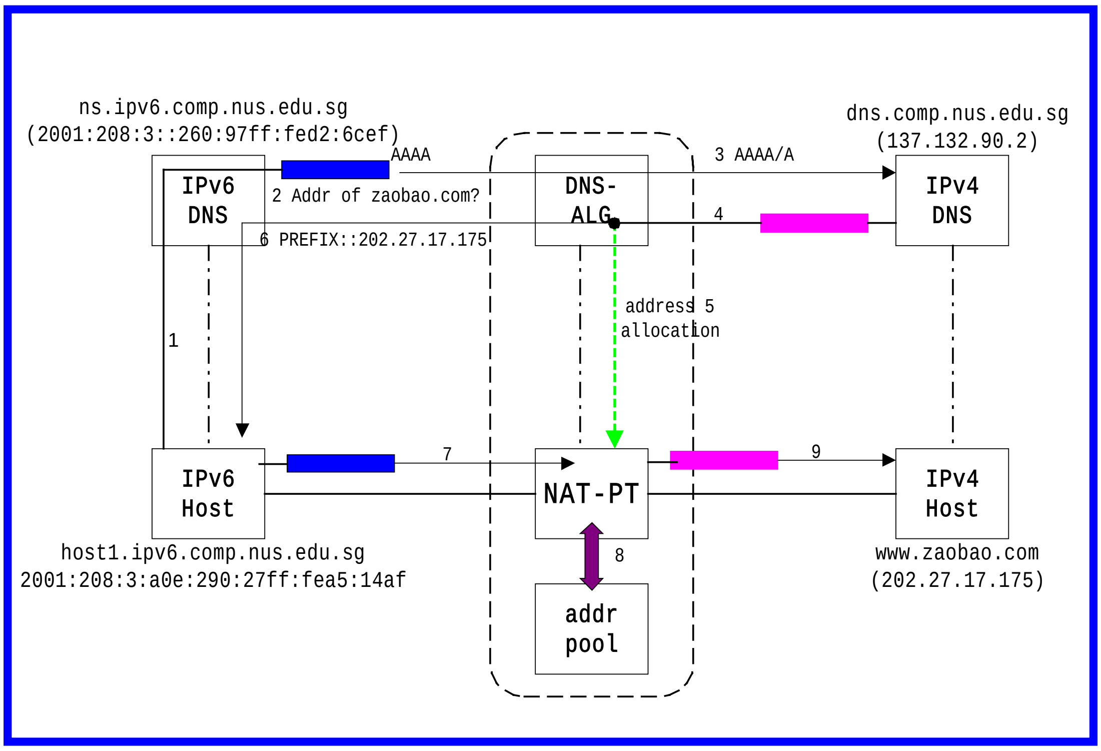

Figure 2.8. NAT-PT Mechanism

1. IPv6 host sends out IPv6 query to native IPv6 DNS server
2. IPv6 DNS server directs this AAAA query to DNS-ALG on the NAT-PT device
3. DNS-ALG modify DNS query from “AAAA” to “AAAA” or “A”
4. IPv4 DNS server replies this query with “A” address
5. DNS-ALG adds a prefix to this “A” address to form “AAAA” address, and records this mapping to NAT-PT mapping table

   202.27.17.175 = PREFIX :: 202.27.17.175
6. Native IPv6 DNS server returns this reply to IPv6 host
7. IPv6 host sends out IPv6 connection request with

   <SA>fec0::260:97ff:fed2:6cef <DA> PREFIX::202.27.17.175

8. NAT-PT allocates an IPv4 address from its address pool to IPv6 host, and records this mapping to its mapping table
   fec0::260:97ff:fed2:6cef = 137.132.80.109
9. NAT-PT modifies this “AAAA” request into
   <SA>137.132.80.109 <DA>202.27.17.175

Note: SA: Source Address
DA: Destination Address

2.4.2 Protocol Translation

Besides Network Address Translation function, successful IPv4 and IPv6 translation technique needs to conduct Protocol Translation, or PT, which translate an IPv4 packet into a semantically equivalent IPv6 packet and vice versa. Stateless IP/ICMP Translation Algorithm [14], or SIIT, translates between IPv4 and IPv6 packet headers (including ICMP headers) in separate translator “boxes” in the network without requiring any per-connection state in those “boxes”. This new algorithm can be used as part of a solution that allows IPv6 hosts, which do not have a permanently assigned IPv4 address, to communicate with IPv4-only hosts.

The IPv6 protocol has been designed so that the TCP and UDP pseudo-header checksums are not affected by the translations specified in this document, thus the translator does not need to modify normal TCP and UDP headers. However, there are some exceptional conditions.

- Unfragmented IPv4 UDP packets need to have a UDP checksum computed since a pseudo-header checksum is required for UDP in IPv6. Also, ICMPv6 include a pseudo-header checksum but it is not present in ICMPv4 thus the checksum in ICMP messages need to be modified by the translator.

- In addition, ICMP error messages contain an IP header as part of the payload thus the translator need to rewrite those parts of the packets to make the receiver be able to understand the included IP header.

However, all of the translator's protocol-relative operations, including path MTU discovery, are stateless in the sense that the translator operates independently on each packet and does not retain any state from one packet to another. This allows redundant translator boxes without any coordination and a given TCP connection can have the two directions of packets go through different translator boxes. We explain two direction of protocol translation in the follows.

- Translating from IPv4 to IPv6

When an IPv4-to-IPv6 translator receives an IPv4 datagram addressed to a destination that lies outside of the attached IPv4 island, it translates the IPv4 header of that packet into an IPv6 header. It then forwards the packet based on the IPv6 destination address. The original IPv4 header on the packet is removed and replaced by an IPV6 header. Except for ICMP packets the transport layer header and data portion of the packet are left unchanged.

One of the differences between IPv4 and IPv6 is that in IPv6 path MTU discovery is mandatory but it is optional in IPv4. This implies that IPv6 routers will never fragment a packet – only the sender can do fragmentation. A few rules ensure that when packets are fragmented either by the IPv4 sender or by IPv4 routers that the low-order 16 bits of the fragment identification is carried end-end to ensure that packets are correctly reassembles. In addition, the rules use the presence of an IPv6 fragment header to indicate that the sender might not be using path MTU discovery i.e. the packet should not have the DF flag set should it later be translated back to IPv4. Other than these special rules for handling fragments and path MTU discovery the actual translation of the packet header consists of a simple regular mapping.

- Translating from IPv6 to IPv4

When an IPv6-to-IPv4 translator receives an IPv6 datagram addressed to an IPv4 address, it translates the IPv6 header of that packet into an IPv4 header. It then forwards the packet based on the IPv4 destination address. The original IPv6 header on the packet is removed and replaced by an IPv4 header. Except for ICMP packets the transport layer header and data portion of the packet are left unchanged.

There are some differences between IPv6 and IPv4 in the area of fragmentation and the minimum link MTU that effect the translation. An IPv6 link has to have an MTU of 1280 bytes or greater. The corresponding limit for IPv4 is 68 bytes. Thus, unless there were special measures, it would not be possible to do end-to-end path MTU discovery when the path includes an IPv6-to-IPv4 translator since the IPv6 node might receive ICMP "packet too big" messages originated by an IPv4 router that reports an MTU less than 1280. However, IPv6 [6] requires that IPv6 nodes handle such an ICMP "packet too big" message by reducing the path MTU to 1280 and including an IPv6 fragment header with each packet. This allows end-to-end path MTU discovery across the translator as long as the path MTU is 1280 bytes or greater. When the path MTU drops below the 1280 limit the IPv6 sender will originate 1280 byte packets that will be fragmented by IPv4 routers along the path after being translated to IPv4.

Other than the special rules for handling fragments and path MTU discovery the actual translation of the packet header consists of a simple regular mapping. In addition, ICMP packets require special handling gin order to translate the content of ICMP error message and also to add the ICMP pseudo-header checksum.

2.4.3 Application Level Gateway

In the above figure and explanation, we notice an important component – Application Level Gateway, or ALG, – while NAT-PT implements translation process. Some applications carry network addresses in payloads. NAT-PT is an application that does not snoop the payload and thus is unaware of appropriate address change. ALG is an application specific agent that allows an IPv6 node to communicate with an IPv4 node and vice versa. DNS_LAG [15] is such an example. ALG could work in conjunction with NAT-PT to provide support for many such applications.

By combining SIIT protocol translation with the dynamic address translation capabilities of NAT and appropriate ALGs, NAT-PT provides a complete solution that would allow a large number of commonly used applications to interoperate between IPv6-only nodes and IPv4-only nodes.

So far, we have elaborated internal mechanisms of three transition techniques respectively; each of them has its particular applicable scenarios. These proposals and corresponding implementations facilitate communication during IPv4 to IPv6 migration. As this period is unlikely a short term, current widely-deployed applications based on IPv4 protocol expect such transition not influence their service quality quite a lot, although theoretically speaking all kinds of transition techniques will bring on side effect on network performance. We simulate these transition techniques in our testbed and test network performance under these mechanisms respectively.

Chapter 3 Distributed NAT-PT

As mentioned in last chapter, most of current NAT-PT solutions are centralized systems consisting of a single CPU, its memory, peripherals and some terminals, which reside in certain border routers between IPv4 and IPv6 realms. This kind of centralized mechanism has its advantages, such as easy data sharing and convenient communication between different users and different sessions. However, it also has many disadvantages, which in some cases induce serious problems. Starting in the mid-1980s, two advances in technology began to change that situation. One was the development of powerful microprocessors. The other development was the invention of high-speed computer networks. The result of these technologies is that it is now not only feasible, but also easy, to put together computing systems composed of large numbers of CPUs connected by a high-speed network. They are usually called **distributed systems** [17], in contrast to the previous **centralized systems**. We first take a look at the definition of distributed system and then list some common features of distributed system.

3.1 What is A Distributed System?

Various definitions of distributed systems have been given in the literature, but there is no one-size-fit-all definition. For our purpose it is sufficient to give a loose characterization:

A distributed system is a collection of independent computers that appear to the users of the system as a single computer.

This definition has two aspects. The first one deals with hardware: the machines are autonomous. The second one deals with software: the users think of the system as a single computer. Both are essential. We can find lots of distributed system applications in the real world. For example, think about a large bank with hundreds of branch offices all over the world. Each office has a master computer to store information of local accounts and handle local transactions. In addition, each computer has the ability to talk to other branch computers. If transactions can be done without regard to where a customer or account is, and the users do not notice any difference between this system and the old centralized mainframe that is replaced, it would be considered a distributed system.

Just because it is possible to build distributed system does not necessarily mean that it is a good idea. After all, with current technology it is possible to put four floppy disk drivers on a personal computer. It is just that doing so would be pointless. In the next section, we will discuss the motivation and goals of typical distributed systems and look at their advantages compared with traditional centralized systems.

3.2 General Advantages of Distributed Systems

3.2.1 Economical Investment

The real driving force behind the trend toward decentralization is economics. According to Grosch’s law: The computing power of a CPU is proportional to the square of its price. It fits the mainframe technology quite well, and led most organizations to buy the largest single machine they could afford. With microprocessor technology, however, Grosch’s law no longer holds. For a few hundred dollars you can get a CPU chip that can execute more instructions per second than one of the largest 1980s mainframes. If you are willing to pay twice as much, you get the same CPU, but running at a somewhat higher clock speed. As a result, the most cost-effective solution is to harness a large number of cheap CPUs together in one system. Thus the leading reason for the trend toward distributed systems is that these systems potentially have a much better price/performance ratio than a single large centralized system would have.

3.2.2 Higher Reliability

Another potential advantage of a distributed system over a centralized system is higher reliability. By distributed the workload over many machines, a single failure will bring down at most one machine, leaving the rest intact. Ideally, if 5 percent of the machines are down at any moment, the system should be able to continue to work with at most 5 percent loss in performance. For critical applications, such as control of nuclear or aircraft, using a distributed system to achieve high reliability may be the dominant consideration.

3.2.3 Convenient Augment

Incremental growth is also potentially a big plus. Often, a company will buy a mainframe with the intention of doing all its work on it. If the company prospers and thus the workload grows, at a certain point the mainframe will no longer be adequate. The only solutions are either to replace the mainframe with a larger one (if it exists) or to add a second mainframe. Both of these can wreak major havoc on the company’s operation. In contrast, with a distributed system, it may be possible simply to add more processors to the system, thus allowing it to expand gradually as the need arises.

3.2.4 More Flexibility

Finally, a distributed system is potentially more flexible than giving each user an isolated personal computer. Although one model is to give each person a personal computer and connect them all with a LAN, this is not the only possibility. Another one is to have a mixture of personal and shared computers, perhaps of different sizes, and let jobs run on the most appropriate one, rather than always on the owner's machine. In this way, the workload can be spread over the computers more effectively, and the loss of a few machines may be compensated for by letting people run their jobs elsewhere.

3.3 Related works

NAT-PT (Network Address Translation - Protocol Translation) is an IETF RFC specification for an IPv4 to IPv6 protocol translator. This version [26] is a userland application, which is developed for a boundary router running Linux 2.4.0-test9 operating system. This referred to other NAT-PT, developed by British Telecom (BT), which also resides within an IP router. But BT's NAT-PT is running on FreeBSD and using the KAME IPv6 stack.

3.3.1 Test Environment

A typical experimental environment is suggested as shown in Figure 3.1.

This test has three hosts (like A, C and D) and one NAT-PT enabled Router (like B). The native DNS (like E) system is located in native IPv6 network.

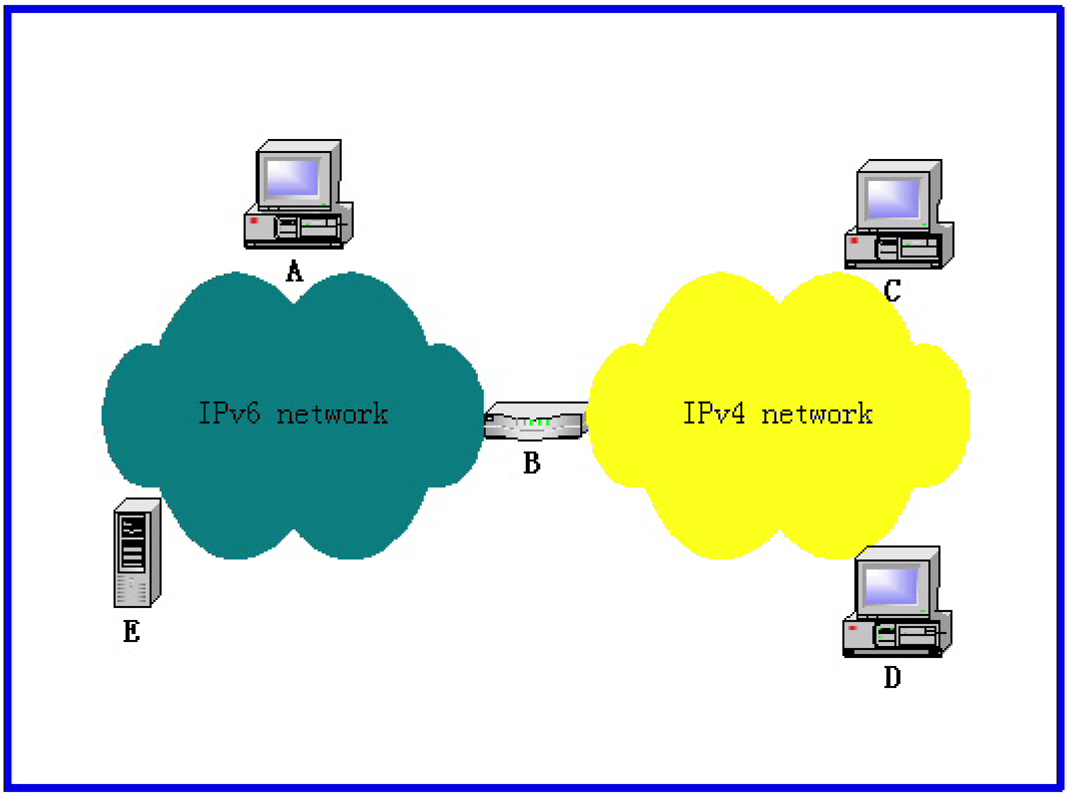

Figure 3.1. A typical experimental environment for NAT-PT

Host A is running IPv6 stack in Linux-2.4.0 kernel, and will be routed to Router B. That is, Host A and B belong to native IPv6 domain.

Router B is also running both IPv4 and IPv6 stack in Linux-2.4.0, and is default router for Host A and E. Therefore, Router B has two network interfaces, one connecting to IPv4 network (eth0), the other to the IPv6 network (like eth1). And Router B is running NAT-PT daemon for IPv4-to-IPv6 interconnection and vice verse.

Host C and D are running IPv4 only stack, and have Windows 2000 and Linux environments respectively.

3.3.2 System Requirements for Router B

NAT-PT has been written specifically to operate on a router using the Linux-2.4.0 kernel with IPv6 stack. That is, the NAT-PT device must be located on the default router for the IPv6 network.

NAT-PT has been tested on a border router with the following specifications:

- Pentium PC
- Linux Operating system
  . RedHat 6.2 package
  . Linux-2.4.0-test9 kernel (with IPv6 stack)
- Compiler: egcs-2.91.96
- Library: glibc-2.1

3.3.3 Application Requirements for Router B

The NAT-PT system has network tools to support IPv6 as follows:

- net-tools-1.57 (hostname, netstat, arp, ifconfig, rarp, route)
- xinetd-2.1.8.8p3 (eXtend INET Daemon)
- netkit-0.4.1 (finger[d], inetd, ping/ping6, telnet[d], etc)

3.4 Distributed NAT-PT Framework and Basic Features

So far, we see quite a few advantages of distributed systems brings on. Although there are some weaknesses in distributed systems, many people feel that advantages outweigh the disadvantages. Thus, we try to transform the single node NAT-PT into distributed NAT-PT. Figure 3.2 illustrates distributed NAT-PT logical structure.

As depicted in Figure 3.2, there are two nodes – NAT-PT A, and NAT-PT B – at the border of IPv4 and IPv6, performing bi-directions translation. Both routers link to router X, through which extend connections to the outside IPv4 network. One of these two border routers is the default gateway of IPv6 network, without losing generality let us suppose NAT-PT A is the default gateway. The other router NAT-PT B is the secondary gateway, acting as a standby translator. The translation processes like the following.

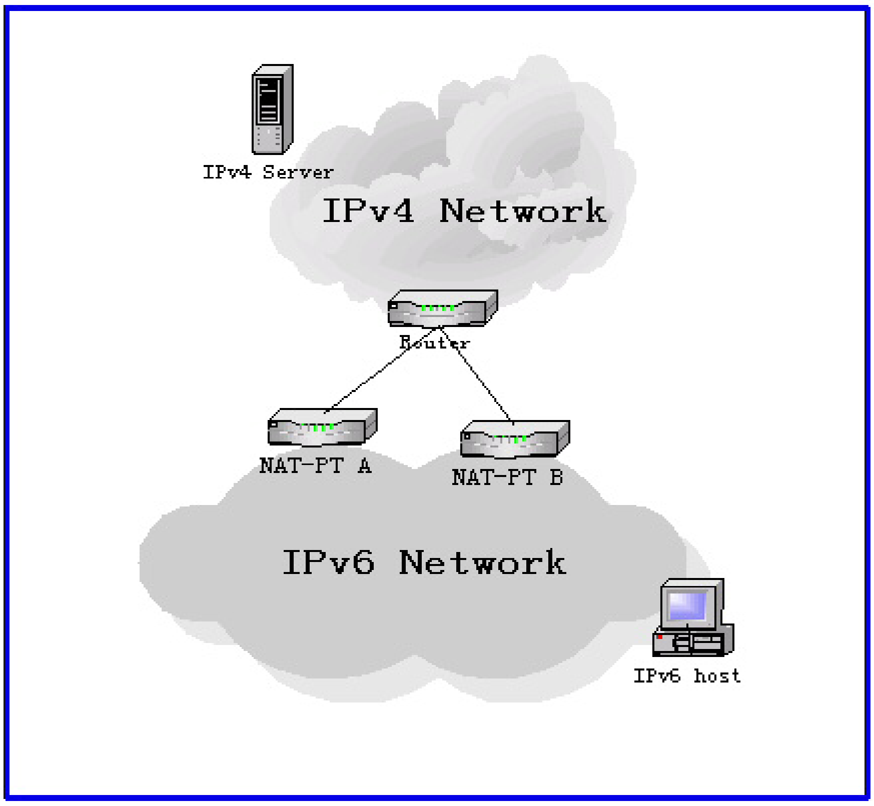

Figure 3.2. Logic Distributed NAT-PT Framework

Within IPv6 network, when certain IPv6 host initiates a connection with a remote IPv4 server, its IPv6 packets are forwarded to default gateway NAT-PT A. This translator allocates one available IPv4 address from its pre-set IPv4 address pool to this IPv6 address, writes corresponding item of address mapping into its local mapping table, commits address translation and protocol translation on the IPv6 packets and then forwards IPv4 packets to Router X. By now, all of these functions look the same as centralized NAT-PT; the key difference, however, is that NAT-PT A should inform its mapping table information to NAT-PT B. Through network communication, NAT-PT A makes NAT-PT B aware of its mapping table changes, including adding new mapping item, updating existing mapping items, or deleting outdated items. By keeping both mapping tables synchronized, NAT-PT B is able to take over translation function whenever NAT-PT A shuts down. This synchronization mechanism eliminates single point of failure of centralized NAT-PT, making our distributed NAT-PT more robust and reliable.

In addition, for the above distributed system it is possible to implement traffic distribution leading to further improvement of the whole system performance. While packets enter into IPv6 Network from IPv4 network, the edge Router X with the ability of load balancing between these two translators can distribute packets to the appropriate translator according to the current network status, instead of always forwarding the packet to certain fixed one, so that improve the whole system throughput.

3.5 Advantages of distributed NAT-PT over centralized NAT-PT

From the above discussion, we conclude some specific advantages of distributed NAT-PT over original centralized NAT-PT.

- **High reliability**

One of the most obvious advantages of distributed NAT-PT is high reliability. By inducing several routers around the realm border, a single failure on one NAT-PT will bring down the function of one router at most, leaving the rest intact. With the help of synchronization mechanism, each of NAT-PT contains address-mapping information not only of processes initialized by itself but also of those learned from other NAT-PTs through synchronization. In this way, all of these NAT-PTs have the ability to implement packet translation that travel cross the border. Therefore, the system has a higher reliability in case of partial failure.

- **Load balancing**

Redundancy is also useful to provide additional bandwidth for high traffic area so that workload balance could be implemented. In an OSPF (Open Shortest Path First) environment, for example, if there are a few equal-cost paths between nodes, routers automatically distribute workload over these paths equally. In real practice, it is possible that several equal-cost paths between nodes exist. For some more complex routing protocols such as IGRP (Interior Gateway Routing Protocol) or EIGRP (Enhanced Interior Gateway Routing Protocol), load balancing can even be implemented among several unequal-cost paths. Therefore, distributed NAT-PTs can improve the network performance by load balancing feature.

- **Incremental growth**

Incremental growth is also potentially a big plus. With the tremendous growth of Internet and intranet around the world, it is hard to foresee the future network scale. When a subnet is needed to add into the current network some time later, it is convenient to extend the present distributed NAT-PTs just adding another router running NAT-PT for this new subnet without changing the current configuration of other routers.

3.6 Distributed NAT-PT Implementation

As stated iteratively, for our distributed system, the most important thing is data sharing. In other words, two NAT-PTs should exchange information of their local mapping tables. We implement such synchronization with the help of socket communication. In this section, we first take a look at the most popular client-server mode with the tool of socket [22] communication.

3.6.1 Client- Server Socket Communication

A socket is an abstraction through which an application may send and receive data, in much the same way as an open file allows an application to read and write data to stable storage. A socket allows an application to “plug in” to the network and communicate with other applications that are also plugged in to the same network. Information written to the socket by an application on one machine can be read by an application on a different machine, and vice versa.

Sockets come in different flavors, corresponding to different underlying protocol families and different stacks of protocols within a family. A socket using the TCP/IP [23] protocol family is uniquely identified by an Internet address, an end-to-end protocol (TCP [24] or UDP [25]) and a port number. When a socket is first created, it has an associated protocol but no Internet address or port number. Until a socket is bound to a port number, it cannot receive messages from a remote application. There are several ways for a socket to become bound to an address.

Figure 3.3 depicts the logical relationships among applications, socket abstractions, protocols, and port numbers within a single host. Note that a single socket abstraction can be referenced by multiple application programs. Each program that has a reference (called a descriptor) to a particular socket can communicate through that socket. From Figure 3.3, we see that multiple programs on a host can access the same socket. In practice, separate programs that access the same socket would usually belong to the same application (e.g., multiple copies of a Web server program), although in principle they could belong to different applications.

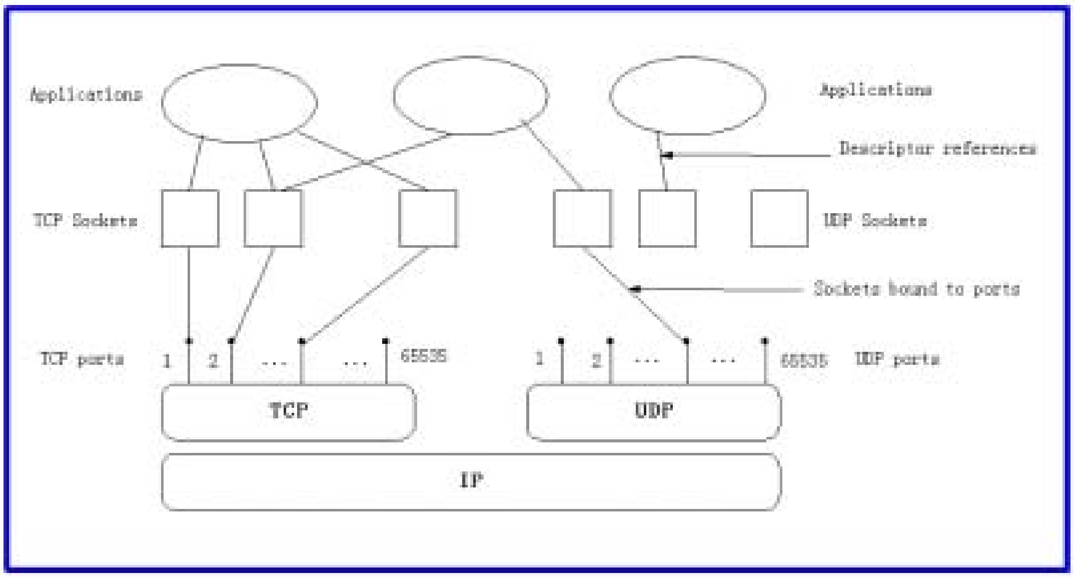

Figure 3.3 Sockets, protocols, and ports

In our system, we utilize stream socket in the TCP/IP family to conduct information communication. Stream sockets use TCP as the end-to-end protocol (with IP underneath) and thus provide a reliable two-way connected stream service. They are also error free, which is vital for information communication for distributed NAT-PT system. We implement socket communication with IPv4 addresses. We apply the most popular client-server mode for mapping information synchronization. One of these two NAT-PT, performing as a client, writes the latest change of its local mapping table into the socket, while the other peer, much like a server, receives the changes from the socket and updates its mapping table correspondingly. We illustrate the basic flow of client-server socket application in Figure 3.4.

The distinction between client and servers is important because each uses the sockets interface differently at certain steps in the communication. We first explain the client. Its job is to initiate communication with a server that is passively waiting to be contacted.

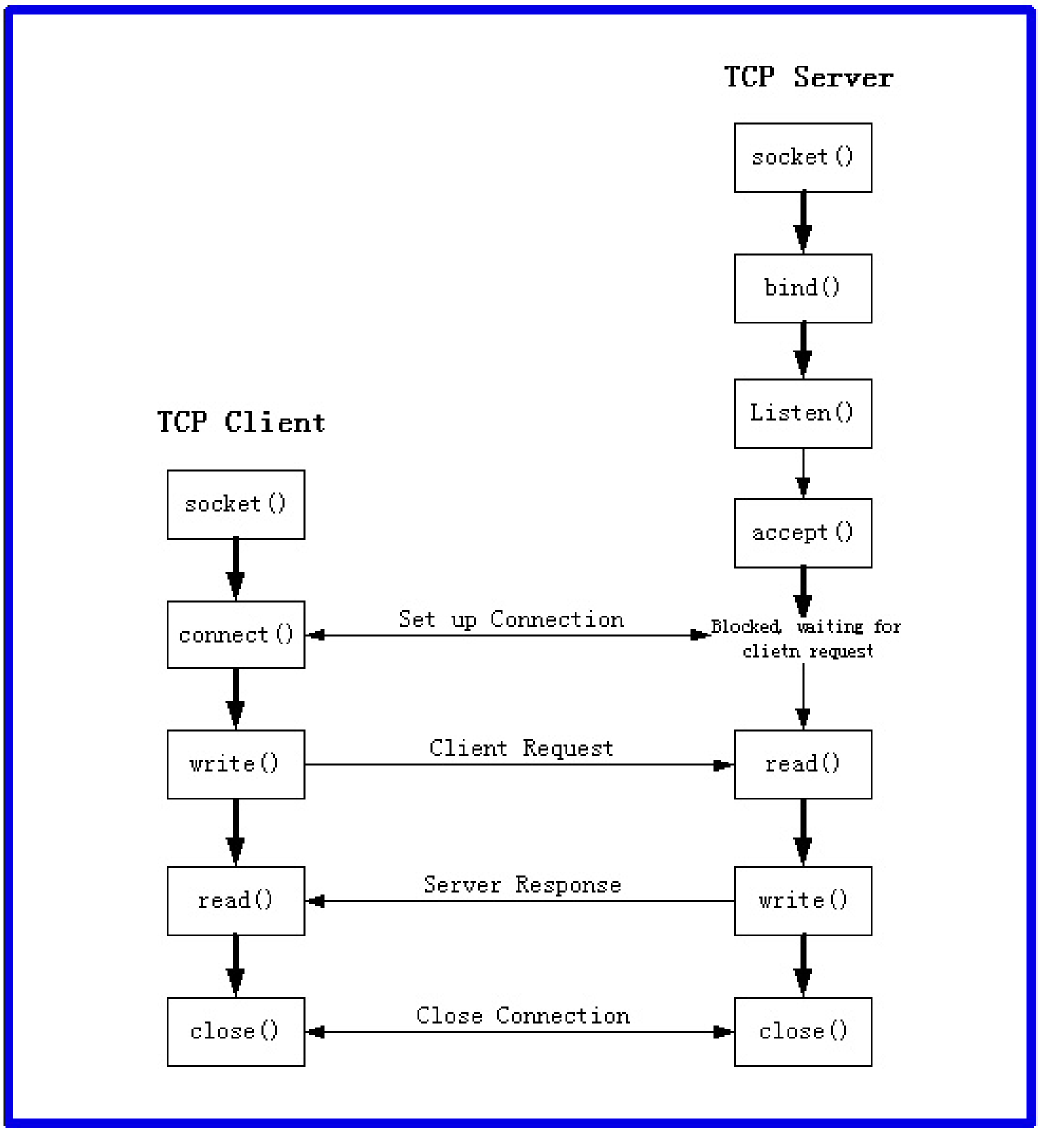

Figure 3.4 Client-Server Socket application frame

The typical TCP client goes through four basic steps:

1. Create a TCP socket using socket ( ).
2. Establish a connection to the server using connect ( ).
3. Communicate using send ( ) and recv ( ).
4. Close the connection with close ( ).

We now turn our focus on constructing a server. The server's job is to set up a communication endpoint and passively wait for a connection from the client. Also there are four steps for TCP server communication:

1. Create a TCP socket using socket ( ).
2. Assign a port number to the socket with bind ( ).
3. Tell the system to allow connections to be made to that port, using listen ( ).
4. Repeatedly do the following:
   - Call accept ( ) to get a new socket for each client connection.
   - Communicate with the client via that new socket using send ( ) and recv ( ).
   - Close the client connection using close ( ).

3.6.2 Synchronization Issues

In the last section, we have explained the basic flow of client-server mode. Now we focus on the particular synchronization contents that are exchanged between two NAT-PT nodes. In the original NAT-PT, a specific data structure, Mapping_List, is created containing corresponding mapping information between IPv4 addresses and IPv6 addresses. Its definition is:

```
Struct Mapping_List
{
        struct in_addr          IPv4_Address;
        unsigned short          IPv4_Port;
        struct in6_addr         IPv6_Address;
        unsigned short          IPv6_Port;
        unsigned int            Number_of_Sessions;
        unsigned int            TCP_Timer_Value;
        unsigned int            UDP_Timer_Value;
        struct Mapping_List     *pNext;
}
```

Mapping_List is a vital structure for NAT-PT function. During system initialization, such Mapping_List link is empty. When two hosts, one from IPv4 network and another from IPv6 network, begin to communicate with each other, a pair of address mapping created. NAT-PT allocates one available IPv4 address from its address pool to this IPv6 host, adding the corresponding communication information as well. These seven items, as shown above, form a tuple that will exist as long as communications to or from this IPv6 host keeps on going. During the mapping information survival period, some items (e.g. Number_of_Sessions, TCP_Timer_Value) may be updated according to current status. When all of the communications of this IPv6 host end, this mapping tuple will be deleted from this Mapping_List link. These are the main points of Mapping_List management.

In our distributed NAT-PT system, to keep both NAT-PT Mapping_List links uniform, we should implement synchronization on contents of this Mapping_List link. As we set before, we choose NAT-PT A, the default gateway of IPv6 network, as socket communication client, and standby NAT-PT B as socket communication server. NAT-PT A passes its latest changes to NAT-PT B passively waiting for the coming information. Thus, two NAT-PTs keep the same mapping table information all along, which means that whenever one of them fails, the other one can take over the translation function without losing any packets. Synchronization implies adding new items, updating existing items and deleting expiating items. We specify corresponding processes in the following three paragraphs respectively.

When a new mapping tuple is added in NAT-PT A, it writes this new Mapping_List tuple to the established socket, which is read by the passively waiting server, NAT-PT

B. Upon receiving this information, NAT-PT B picks up the IPv4 address and IPv4 port from the received information. These two items serves as a combined key for matching this mapping information with the existing mapping tuples in NAT-PT B. As it is a newly created tuple, NAT-PT B has no such tuple. This implies that NAT-PT B should insert this new mapping tuple to its local Mapping_List link.

For every existing tuple, when one of three items, Number_of_Sessions, TCP_Timer_Value, or DP_Timer_Value, is updated representing current network traffic status, such changes should also be informed to the other NAT-PT. In this case, the same story happens in NAT-PT A, which writes the newly updated Mapping_List tuple to socket. At the server side, NAT-PT B should match this received tuple with one of its local mapping tuples successfully. It then updates this Mapping_List information appropriately, keeping consistent with NAT-PT A.

With regard to deleting issues, there is a special mechanism in each NAT-PT, which automatically deletes the mapping tuples that has not been accessed for a specific time.

Besides the above basic implementation, we also add some abnormal situation judgment and corresponding disposals, thus increasing system fault-tolerance.

To present an overall formal representation of the synchronization of the distributed system, we illustrate the interactions between the two NAT-PTs in the following two state transition diagrams. Figure 3.5 shows server side, and Figure 3.6 shows client side.

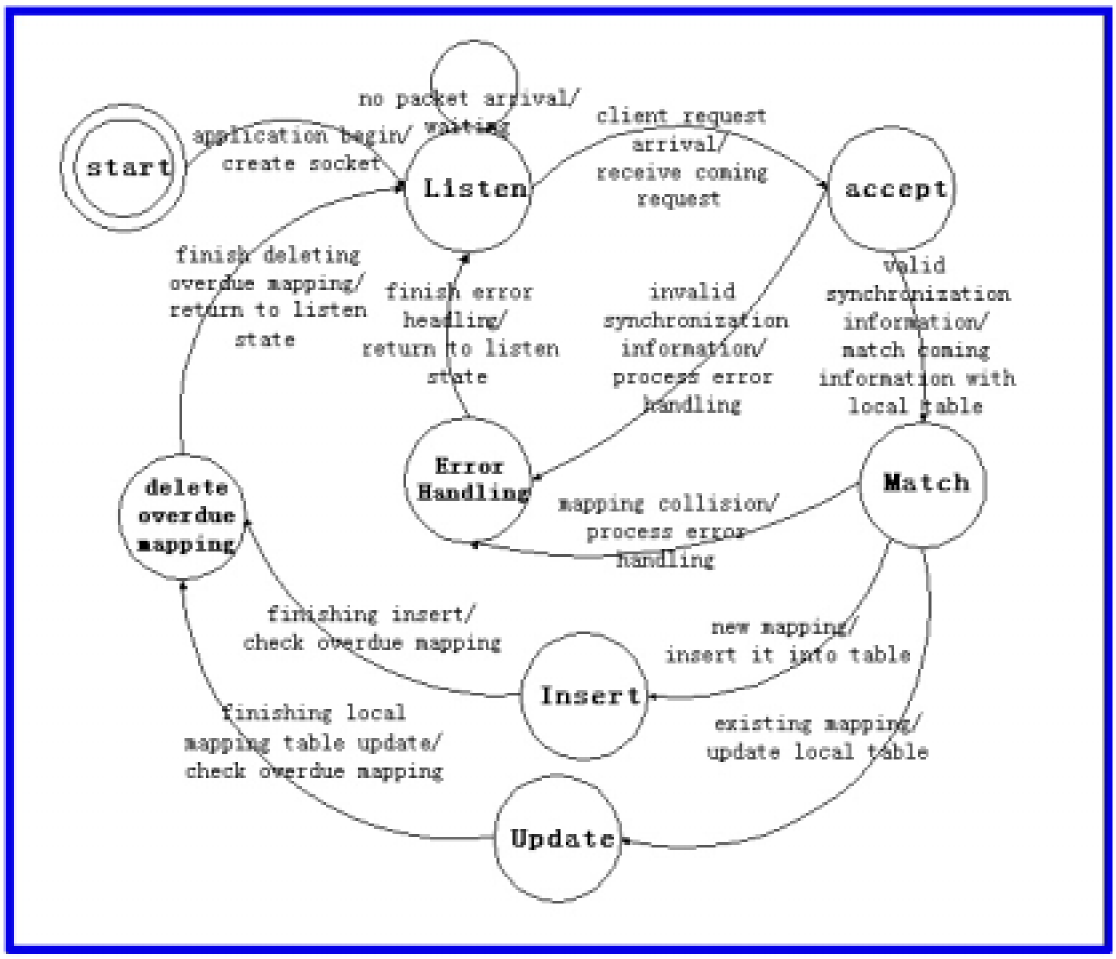
start
application begin/ create socket
no packet arrival/ waiting
Listen
client request arrival/ receive coming request
accept
finish deleting overdue mapping/ return to listen state
finish error headling/ return to listen state
invalid synchronization information/ process error handling
valid synchronization information/ match coming information with local table
delete overdue mapping
Error Handling
mapping collision/ process error handling
Match
finishing insert/ check overdue mapping
new mapping/ insert it into table
Insert
finishing local mapping table update/ check overdue mapping
existing mapping/ update local table
Update

Figure 3.5. State transition diagram of server

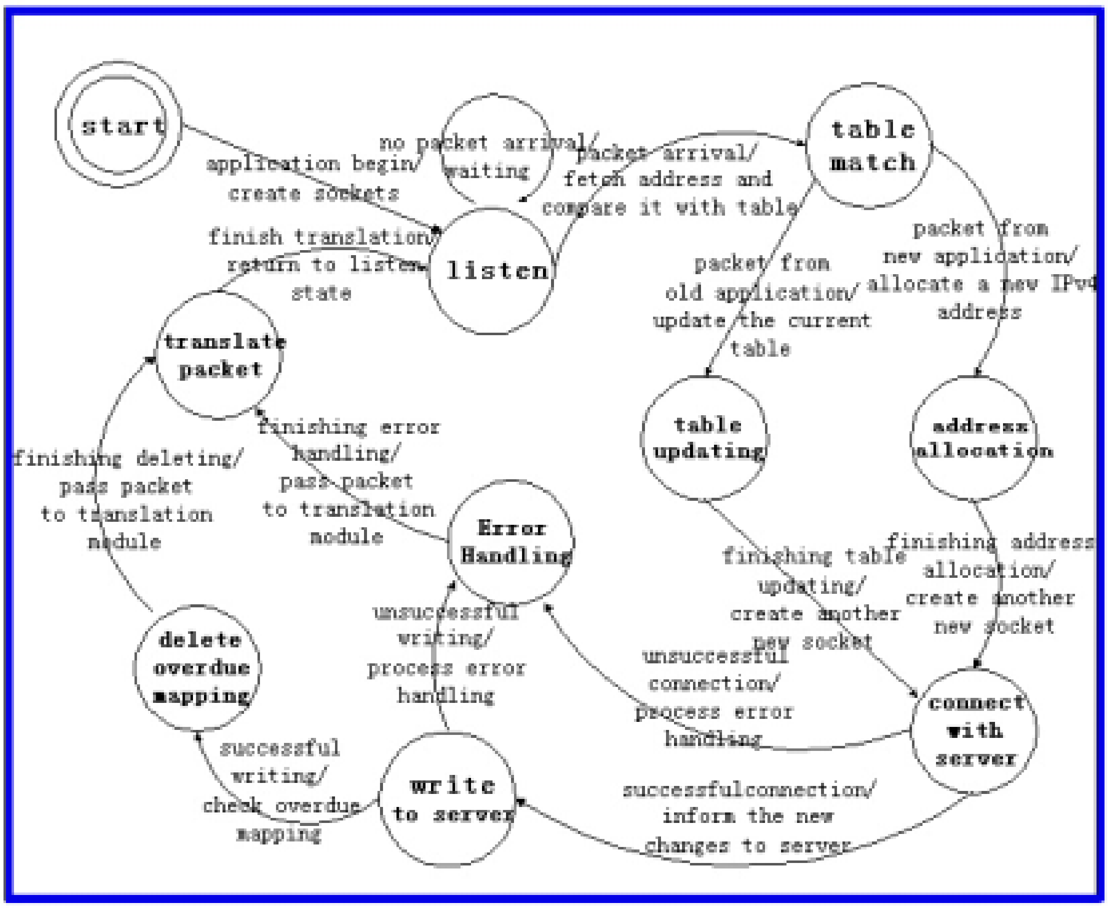

Figure 3.6. State transition diagram of client

Chapter 4 Experimental Results

To test performance of the above transition solutions, we build up a comprehensive testbed for conducting a variety of experiments. Before explaining the test item details, we first illustrate the testbed components and structure shown in Figure 4.1.

4.1 Testbed Construction

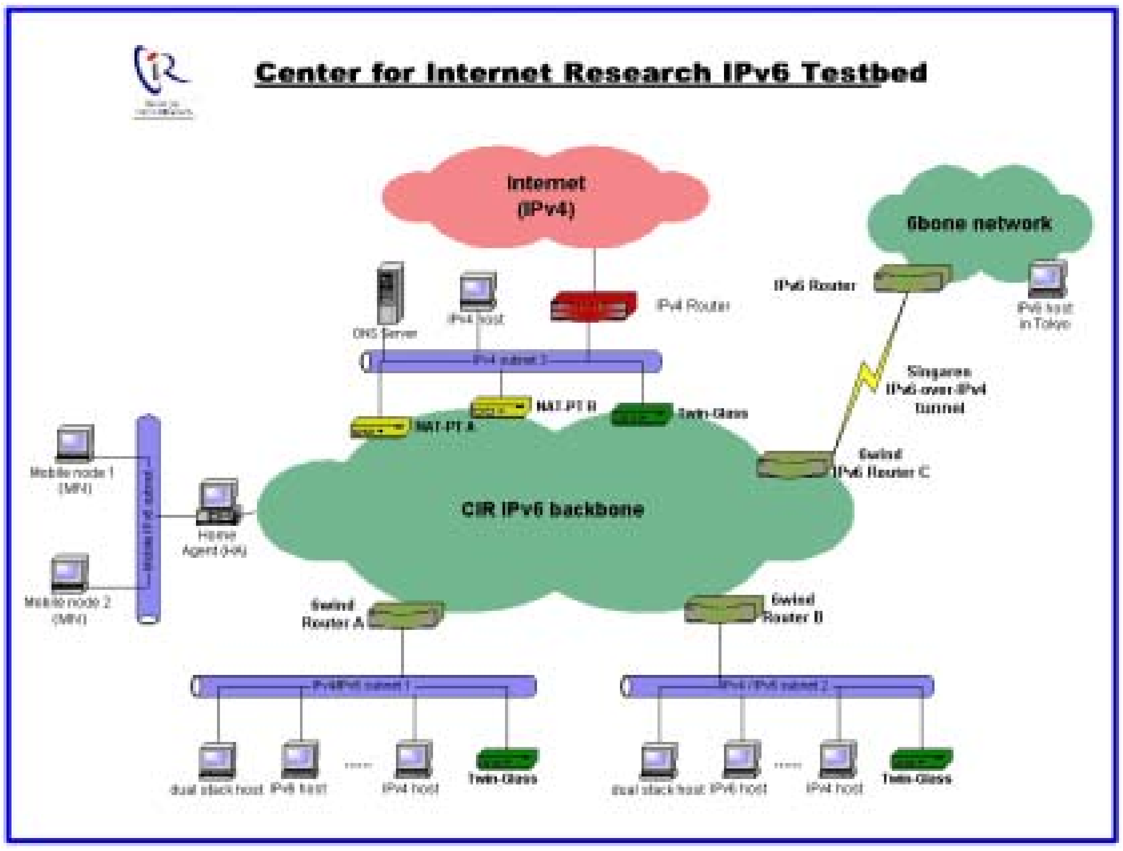

Fig 4.1. Framework of CIR IPv6 testbed

Displayed in Figure 4.1, “IPv6 backbone” is the main spot we process our test. Though labeled as “IPv6” backbone, it could also support old IPv4 protocol given appropriate network parameters. However, exactly one protocol is supported at a time, thus making the traffic within backbone entirely pure. Because this testbed is relatively independent of exterior public world, network performance of this community has almost no interference caused by outside traffic. Therefore, the test results acquired from the testbed exactly present internal network function, from which we can evaluate the technique performance accurately. Router C is the gateway of the backbone IPv6 traffic. It plays one of the two endpoints of a router-to-router IPv6 over IPv4 tunnel, establishing IPv6 connection between CIR testbed and the 6bone community over public IPv4 Internet.

Besides the backbone, there are three subnets employed in our experiments. “Subnet1” and “subnet2” act as two Intranets, which are hybrid networks containing not only IPv6 hosts but also IPv4 hosts and dual stack hosts. The community of miscellaneous protocol hosts just applies to network conditions during IPv4 to IPv6 transition time. Router A and Router B are two border routers, linking the Intranets with backbone. The two routers could be configured as IPv4 router that transmits only IPv4 packets or IPv6 only router that transmit only IPv6 packets to simulate various application scenarios corresponding to specific experiment requirements. “Subnet3” is a pure IPv4 domain that only supports pure IPv4 traffic and further extends testbed IPv4 connection with public IPv4 Internet.

In each of the above three subnets lies a Twin-Glass [27] node, which implements IPv4 over IPv6 tunneling endpoint function, encapsulating outgoing IPv4 packets with an new IPv6 header or decapsulating IPv6 header from incoming IPv6 packets to restore the original IPv4 packet, to offer a transparent transition for end users. When backbone simulates pure IPv6 community, Twin-Glass node serves as the default gateway of IPv4 traffic of subnet1 and subnet2. As to subnet3 and public IPv4 Internet, IPv4 packets with destination address belonging to subnet1 and subnet2 are all directed to the Twin-Glass node of subnet3. A pair of any two of these three gateways can build up a configured tunneling.

As shown in Figure 4.1, two nodes across the border between IPv6 backbone and IPv4 subnet3 perform NAT-PT function. The translation works as follows. When IPv6 only host of either from subnet1 or subnet2 want to set up connection with IPv4 only host of subnet3 or even remote hosts of public Internet, NAT-PT implements packet translation function, making traffic smoothly passing through different realms. In order for NAT-PT to create address mapping dynamically, DNS requests and responses must cross border between IPv4 and IPv6 network and be processed by the DNS ALG. This leads to requirement of a local DNS server. We establish a DNS server of our testbed that provides both IPv4 and IPv6 name resolving service for the nodes within our domain ".ipv6.comp.nus.edu.sg".

4.2 Testing Metric and Tools

Experimental data obtained from tests are presented in this chapter. End-to-end TCP performance metric **Throughput** [31] is measured at the receiver side in each test item. The time interval of each testing is 30 seconds. We repeat every item five times and keep the mean value as the final result eliminating random error as much as possible. In TCP/IP protocol suite, the size of window offered by the receiver may affect TCP performance. We thus test the end-to-end network performance at various window sizes by tuning the corresponding parameter in sender and receiver.

There are lots of network performance test tools. We choose three most popular tools for testing. Iperf [28] is used to generate raw TCP traffic, which could produce IPv4 packets and IPv6 packets as well. TCPdump [29] observes the TCP packets at the destination end and tcptrace [30] analyses the output files dumped from tcpdump.

Besides conducting experiments with testing tools, we also evaluate real application performance under these transition mechanisms and compare the results with the ones obtained from raw traffic testing. We conduct a set of tests on FTP connection, a typical TCP application. Similar to testing tool experiments, we use the end-to-end throughput as our performance metric, which are obtained by transferring a fixed-size file while at the various window sizes.

For clear specification and further comparison and analysis, we classify these experiments into three groups, including pure IPv4 performance and IPv4 over IPv6 Tunnel performance, pure IPv6 performance and IPv6 over IPv4 tunnel performance, and NAT-PT related performance. The following sections unfold our test results.

4.3 Pure IPv4 Performance and IPv4 over IPv6 Tunnel Performance

To evaluate certain tunnel performance, we conduct two sets of experiments. One is the pure connection, whether IPv4 or IPv6, between two nodes; the other is also conducted between the same nodes, but through certain kind of tunnel. Comparing these two performances lead us to more accurate evaluation on this tunnel technique.

Twin-Glass [27] employs the dual stack approach coupled with dynamic tunneling technique to offer a transparent and scalable IPv4 to IPv6 tunnel. We conduct IPv4 over IPv6 configured tunneling on a pair of Twin-Glass boxes, one of which encapsulates received IPv4 packets with the IPv6 tunnel header and transmits them to the IPv6 network. At the other end of tunnel, another Twin-Glass box decapsulates the IPv6 header and forwards IPv4 traffic to IPv4 network. Actually, a pair of Twin-Glass boxes conducts a router-to-router IPv4 over IPv6 configured tunnel function.

4.3.1 Raw TCP traffic testing

In this section, we present test results of raw traffic on such tunnel and corresponding pure IPv4 performance. Table 4.1 presents test results of pure IPv4 connection between host A of subnet1 and host B of subnet2. Table 4.2 presents test results of connection between host A of subnet1 and host B of subnet2 through IPv4 over IPv6 tunnel.

Pure IPv4 TCP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	3.784	6.498	6.75	7.778	12.8	13.0	13.36	13.36

Table 4.1. Pure IPv4 connection performance

IPv4 over IPv6 tunnel (Twin-Glass) TCP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	2.926	4.272	4.584	5.786	9.428	10.1	10.1	10.22

Table 4.2. IPv4 over IPv6 tunnel connection performance

4.3.2 FTP testing

In this section, we present test results of FTP testing on such tunnel and corresponding pure IPv4 performance. Table 4.3 presents test results of pure IPv4 connection between host A of subnet1 and host B of subnet2. Table 4.4 presents test results of connection between host A of subnet1 and host B of subnet2 through IPv4 over IPv6 tunnel.

Pure IPv4 FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	3.59	6.17	6.41	7.39	12.17	12.35	12.69	12.69

Table 4.3. Pure IPv4 FTP performance

IPv4 over IPv6 tunnel (Twin-Glass) FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	2.45	4.31	4.49	5.66	9.23	9.89	9.92	10.01

Table 4.4. IPv4 over IPv6 tunnel FTP performance

4.4 Pure IPv6 Performance and IPv6 over IPv4 Tunnel Performance

As stated, we test another type of transition technique, IPv6 over IPv4 tunnel. Like IPv4 over IPv6 tunnel, we tested transition performance with testing tools and with real application. For each kind of testing, two sets of experiments are conducted for performance comparison. We chose Router A and Router B as two endpoints of this tunnel (referring to Figure 4.1). These two dual stack routers interconnected by an IPv4 infrastructure implement encapsulating and decapuslating function respectively, setting up a router-to-route configured IPv6 over IPv4 tunnel. We present the test results in the following two sections.

4.4.1 Raw TCP traffic testing

This section shows the testing results of raw traffic TCP performance. Table 4.5 presents test results of pure IPv6 connection between host A of subnet1 and host B of subnet2, while Table 4.6 presents test results of connection between host A of subnet1 and host B of subnet2 through IPv6 over IPv4 tunnel.

Pure IPv6 FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	2.998	5.356	5.308	6.314	9.89	10.2	10.3	10.5

Table 4.5. Pure IPv6 connection performance

IPv6 over IPv4 Tunnel TCP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	2.758	4.34	4.468	5.76	7.918	8.41	8.406	8.45

Table 4.6. IPv6 over IPv4 tunnel connection performance

4.4.2 FTP testing

In this section, we present test results of FTP testing on IPv6 over IPv4 tunnel and corresponding pure IPv6 performance. Table 4.7 presents test results of pure IPv6 connection between host A of subnet1 and host B of subnet2. Table 4.8 presents test results of connection between host A of subnet1 and host B of subnet2 through IPv6 over IPv4 tunnel.

Pure IPv6 FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	2.698	4.83	4.81	5.682	8.91	9.18	9.19	9.21

Table 4.7. Pure IPv6 FTP performance

IPv6 over IPv4 Tunnel FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	2.482	3.91	4.02	5.191	7.13	7.569	7.57	7.59

Table 4.8. IPv6 over IPv4 tunnel FTP performance

4.5 NAT-PT Related Experimental Results

Translation is another kind of transition technique, which enables connections between IPv4 and IPv6 realms. In this section, we present NAT-PT related test results. We first focus our attention on the original centralized NAT-PT performance, comparing it with pure IPv4 performance and pure IPv6 performance between the same nodes. Then we turn to the evaluation of distributed NAT-PT performance, testing synchronization effect on system performance and also comparing overall network performance of distributed system with centralized system.

4.5.1 Pure IPv4 and Pure IPv6 versus NAT-PT

We first test NAT-PT performance on a new pair of hosts – host A of subnet1 and host C of subnet3. Three tables present the related test results: Table 4.9 and Table 4.10 present pure IPv4 and pure IPv6 performance between this pair of hosts respectively.

Table 4.11 presents performance between this same pair of hosts through NAT-PT transition technique.

Pure IPv4 TCP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	11.15	15.38	16.74	18.11	40.33	51.89	52.47	53.1

Table 4.9. Pure IPv4 connection performance

Pure IPv6 TCP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	10.99	15.32	16.68	17.82	39.96	51.4	52.05	52.76

Table 4.10. Pure IPv6 connection performance

NAT-PT TCP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	3.85	5.30	5.76	7.46	10.14	10.7	10.96	11.08

Table 4.11. NAT-PT performance

4.5.2 FTP testing

In this section, we present FTP testing results relating to NAT-PT mechanism. Table 4.12 and table 4.13 present FTP performance of pure IPv4 and pure IPv6 connection. Table 4.14 presents FTP performance through NAT-PT.

Pure IPv4 FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	10.93	15.07	16.41	17.75	38.52	50.05	51.40	52.03

Table 4.12. Pure IPv4 FTP performance

Pure IPv6 FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	10.77	15.01	16.35	17.46	39.16	50.37	51.01	51.70

Table 4.13. Pure IPv6 FTP performance

NAT-PT FTP performance								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	3.08	4.24	4.61	5.97	8.11	8.56	8.77	8.86

Table 4.14. NAT-PT FTP performance

4.5.3 Centralized NAT-PT versus Distributed NAT-PT

After the tests on single NAT-PT box, we turn to another facet, the distributed NAT-PT performance. Here, we conduct two groups of experiments. The first one is to evaluate the synchronization effect on system original performance. The second one is to compare performance between distributed system and centralized system. We first present the results of synchronization effect experiments.

In order to test the synchronization effect on system performance, we conduct two sets of tests: two hosts connected by single centralized NAT-PT and the same pair of hosts connected by single NAT-PT with mapping table synchronization function. Table 4.15 presents TCP performance through original centralized NAT-PT. Table 4.16 presents TCP performance through single NAT-PT which implements mapping table synchronization.

NAT-PT TCP performance without synchronization								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	3.85	5.30	5.76	7.46	10.14	10.7	10.96	11.08

Table 4.15 Single Centralized NAT-PT performance

NAT-PT TCP performance with synchronization								
Window size (KB)	2	4	6	8	16	32	64	128
throughput (Mbit/sec)	3.35	4.67	5.10	6.51	10.03	10.64	10.88	11.06

Table 4.16 Single NAT-PT performance with synchronization

In the second group of testing, we implement performance comparison between distributed system and centralized system. We conduct two simultaneous FTP connections that both traverse through one single centralized NAT-PT and test end-to-end throughput of one of these two connections. Then we test network performance of the same two simultaneous FTP connections in the distributed NAT-PT environment. Each of these two connections traverses through two different distributed NAT-PT respectively. Like centralized system, end-to-end throughput of one connection is tested.

In our experiments, we test throughput at different file size varying from 1Mb to 500Mb. TCP window size is fixed at 16Kb for every experiment. From our test result, we could see obvious performance improvement with distributed NAT-PT. The following two tables present the test results. Table 4.17 presents FTP performance of original centralized NAT-PT performance; Table 4.18 presents FTP performance of distributed NAT-PT.

FTP performance					
File size (MB)	1	10	50	100	500
throughput (Mbit/sec)	5.57	7.54	8.38	7.92	7.44

Table 4.17 Centralized NAT-PT Performance

FTP performance					
File size (MB)	1	10	50	100	500
throughput (Mbit/sec)	8.56	10.71	11.57	10.98	10.43

Table 4.18 Distributed NAT-PT Performance

Chapter 5 Discussion and Analysis

In this chapter, we go through our experimental results and conduct comparison among these transition techniques. We then analyze possible factors that may lead to such performance difference. In the next section, we first explain a new criterion for performance comparison.

5.1 A New Criterion

We introduce a new criterion, transition efficiency, to evaluate transition effect on end-to-end performance of each transition technique. This metric helps us process precise quantitative analysis on these techniques. Its definition is:

$$\text{transition efficiency} = \frac{\text{throughput after transition}}{\text{throughput of pure IPv4 or IPv6}}$$

The numerator in above definition is the end-to-end throughput tested under various transition techniques, including IPv6 over IPv4 tunnel, IPv4 over IPv6 tunnel and NAT-PT translator. The denominator of the above definition, which should be pure IPv4 connection performance or pure IPv6 connection performance, takes the appropriate value according to various application scenarios. For example when we evaluate IPv6 over IPv4 tunnel transition efficiency, we calculate the above formula with pure IPv6 connection performance as denominator that is the end-to-end performance without transition technique under the uniform testing condition. The concept applies to other types of techniques. Therefore, we can judge and compare these techniques performance by the indicator. From its definition, the transition efficiency of these two types of transition techniques should be less than one. The higher value of transition efficiency some technique presents, the less effect the technique results in. For a specific transition technique, we hope its transition efficiency is as high as possible, which means that it brings less effect on the end-to-end performance.

5.2 Transition Efficiency of Distinct Transition Techniques

Following the above definition, we compute and explicate transition efficiency of each kind of transition technique in this section.

5.2.1 IPv6 over IPv4 configure tunnel

Our first set of experiments is implemented on IPv6 over IPv4 configured tunnelling, which is used in a router-to-router mode. Dual stack routers interconnected by an IPv4 infrastructure tunnel IPv6 packets between themselves. In this case, the tunnel spans one segment of the end-to-end path that the IPv6 packet takes. The motivation of this method is to allow isolated IPv6 sites or hosts, attached to a wide area network which has no native IPv6 support, to communicated with such other IPv6 domains with minimal manual configuration. Two hosts – host A of subnet1 and host B of subnet2 – act as server and client respectively. Both of them are configured to only send and receive IPv6 packets. We conduct tunnelling on two 6wind border routers and test TCP throughput between two hosts over this tunnel.

Figure 5.1 shows the end-to-end TCP throughput tuning in the increased host window sizes for pure IPv6 path and configured tunnel. In both cases, when the window size is increased, the throughput increases accordingly. We can see that TCP throughput through tunnelling transition is less than the throughput of pure IPv6 path of the same window size. Figure 5.2 presents transition efficiency of the IPv6 over IPv4 tunnelling.

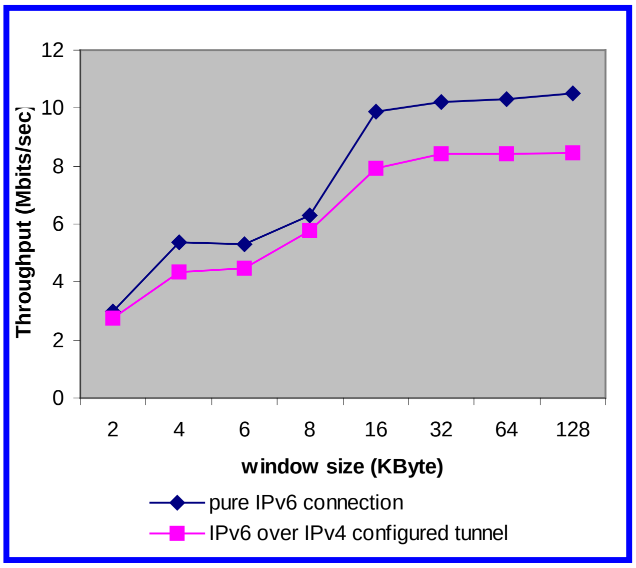

Figure 5.1. Pure IPv6 and IPv6 over IPv4 tunnel performance

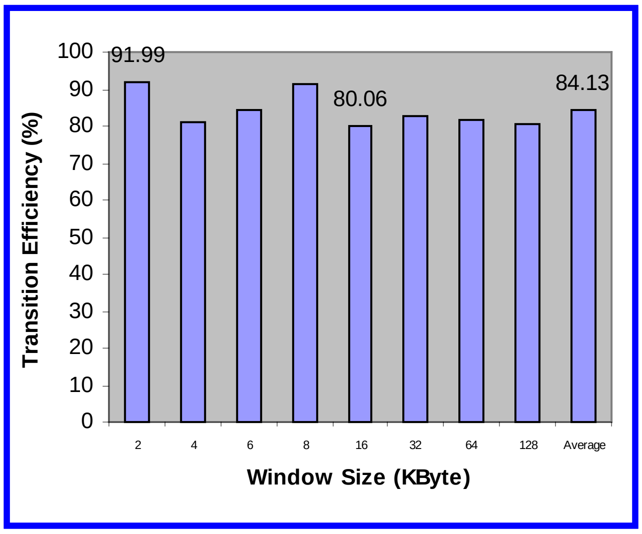

Figure 5.2. Transition Efficiency of IPv6 over IPv4 tunnel

As above figures show, the data range of this set is from 80% to 92%, with average 84%. That means under IPv6 over IPv4 tunnel transition mechanism, end-to-end throughput between two IPv6 hosts generally loses about 16%. Because we implement these tests on our independent testbed, which does not connect outside network, no extrinsic factors, such as traffic condition under different intervals etc., result in performance decline. Meanwhile, two groups of experiments are conducted in uniform physical situation, including capacity of two hosts and routers, packets-forwarding route. Based on these preconditions, we can attribute such performance degradation to encapsulation and decapsulation delay brought by two tunnel points.

Regarding the real application, we also figure out the tunnel effect on FTP performance. Figure 5.3 presents the throughput of "GET" at various TCP window size the same as raw traffic. The transition efficiencies of the IPv6 over IPv4 tunnel are calculated in Figure 5.4.

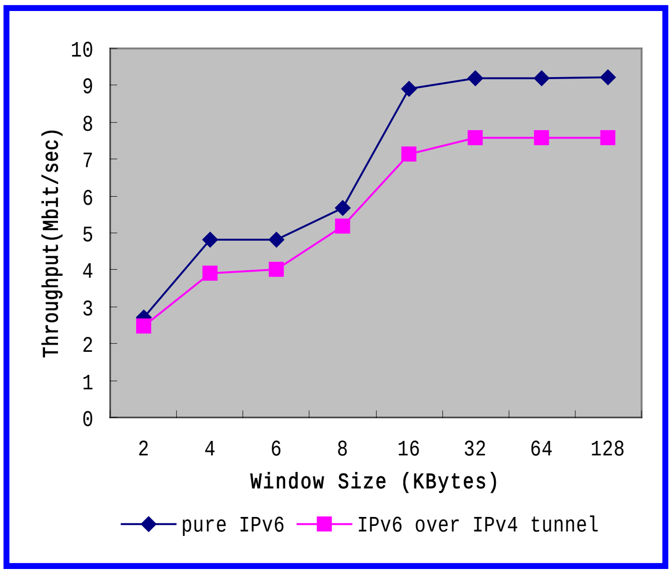

Figure 5.3 Pure IPv6 and IPv6 over IPv4 tunnel FTP performance

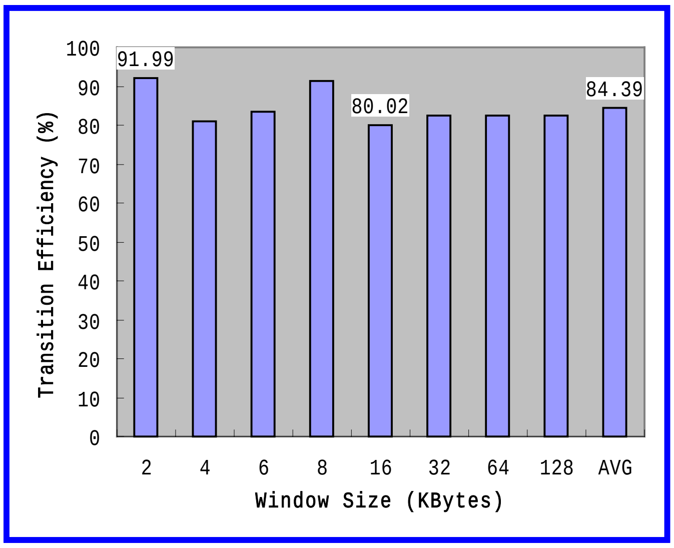

Figure 5.4 Transition Efficiency of IPv6 over IPv4 tunnel of FTP application

From Figure 5.4, we can see that the transition efficiency of IPv6 over IPv4 tunnel for FTP application varies from 80%-92%, which is consistent with the raw traffic test result.

5.2.2 IPv4 over IPv6 tunneling

Twin-Glass employs the dual stack approach coupled with dynamic tunneling technique to offer a transparent and scalable IPv4 to IPv6 transition. In our second set of experiments, we conduct IPv4 over IPv6 configured tunneling on a pair of Twin-Glass boxes, one of which encapsulates received IPv4 packets with the IPv6 tuneling header and transmits them to the IPv6 network. At the other end of tunnelling, another Twin-Glass box decapsulates the IPv6 header and forwards IPv4 traffic to desired IPv4 hosts. Actually, a pair of Twin-Glass boxes conduct a router-to-router IPv4 over IPv6 configured tunnelling function, which spans one segment of the end-to-end path that IPv4 packets take. Due to lack of IPv6 applications and services on current IPv4 host and network, Twin-Glass provides a useful solution to let IPv4 application run in IPv6 network.

We implement this type of IPv4 over IPv6 configured tunnel on the same hosts as pure IPv4 test. Also, two groups of testing are conducted in this set of testing. One is pure IPv4 path between two hosts, and the other is through IPv4 over IPv6 tunnel. Figure 5.5 and Figure 5.6 show the test results and computed transition efficiency respectively. As any transition technique, Twin-Glass brings on performance degradation as well, whose transition efficiency varies from 66% to 78%, with the average value of 74%.

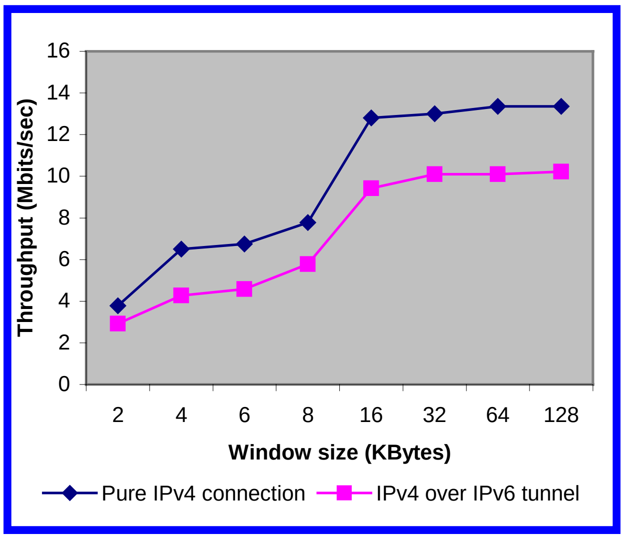

Figure 5.5. Pure IPv4 and Twin-Glass performance

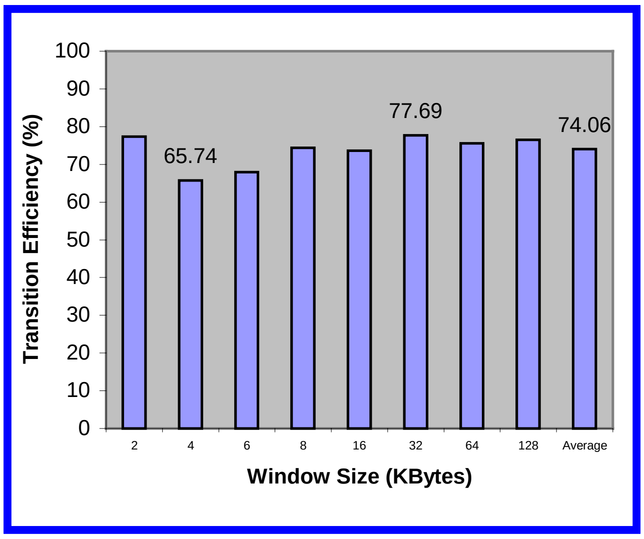

Figure 5.6. Twin-Glass transition efficiency

Regarding the real application, we calculate the tunnel effect on FTP performance. Figure 5.7 presents the throughput of "GET" at various TCP window size the same as raw traffic testing. The transition efficiencies of the IPv4 over IPv6 tunnel are presented in Figure 5.8.

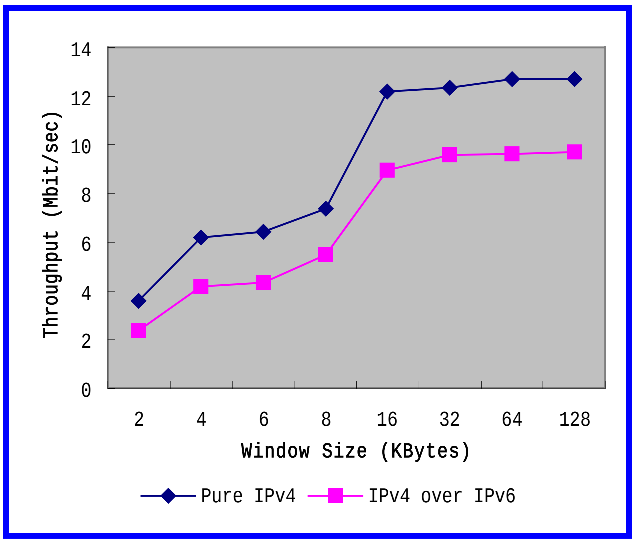

Figure 5.7. Pure IPv4 and IPv4 over IPv6 FTP performance

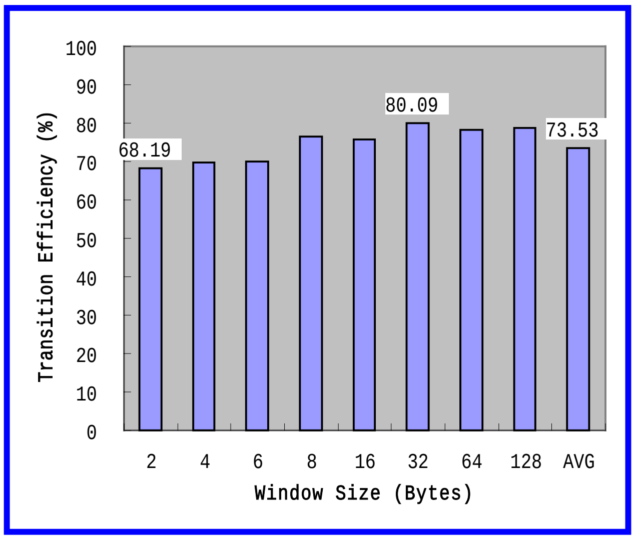

Figure 5.8 Transition Efficiency of IPv4 over IPv6 tunnel of FTP application

From Figure 5.8, we can see that the transition efficiency of IPv6 over IPv4 tunnel for FTP application varies from 68%-80%, which is consistent with the raw traffic test result.

5.2.3 Centralized NAT-PT

In the above two sections, we test two types of tunnel techniques that have similar function in some respects. The common point of these two solutions is that they provide transition mechanisms between hosts or networks running on the same IP. When two hybrid hosts or networks intend to communicate with each other, tunnel techniques do not apply in this scenario. Instead, we have to implement another type of transition mechanism – NAT-PT. In this set of experiments, we conduct a set of experiments on a new pair of nodes – host A of subnet1 as IPv6 only client and host C of subnet3 as IPv4 only server – a typical transition scenario that enables an IPv6 client to access service on an IPv4 server.

As previous tunnel mechanisms, we test TCP throughput on various window sizes in three ways, IPv6 host to IPv4 host while implementing NAT-PT, IPv4 host to IPv4 host and IPv6 host to IPv6 host. The latter two groups of tests are preceded as reference. When implementing pure IPv4 and IPv6 communications, we utilize the same hosts and border router node as well, ensuring the same test precondition.

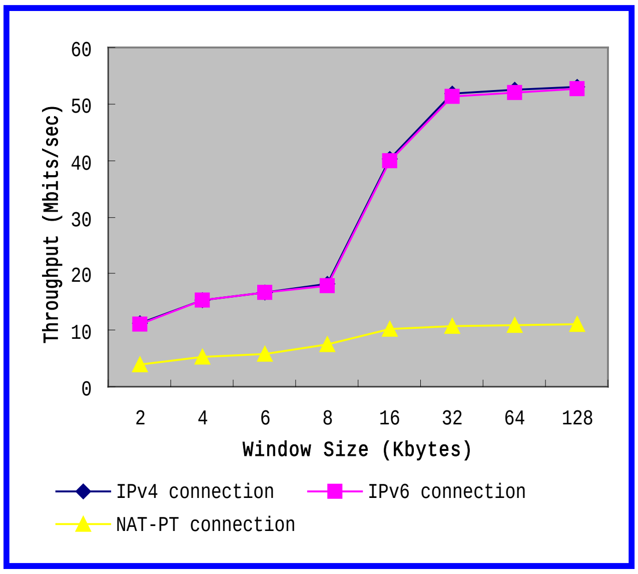

Figure 5.9. NAT-PT performance

Figure 5.9 presents the results of three groups tests. Due to a tiny diversity between pure IPv4 and pure IPv6 throughput, we set the median of these two as the denominator of NAT-PT performance transition efficiency. Transition efficiencies at various window sizes of NAT-PT are shown in Figure 5.10.

Similar to other two tunnel techniques, end-to-end throughput under NAT-PT improves as the window size increases, but it does not keep a uniform corresponding increase pace with pure IPv4 or IPv6 throughput. With window size less than 8Kbyte, NAT-PT presents transition efficiency ranging from 31.65 to 41.19. When window increases to 16Kbyte and above, NAT-PT transition efficiency however falls into a value around 20. The maximum end-to-end throughput is 41.53 at window of 8Kbyte and the minimum value is 20.72 at window of 32Kbyte. The average value of the set of data is 29, with a diversity of 20.

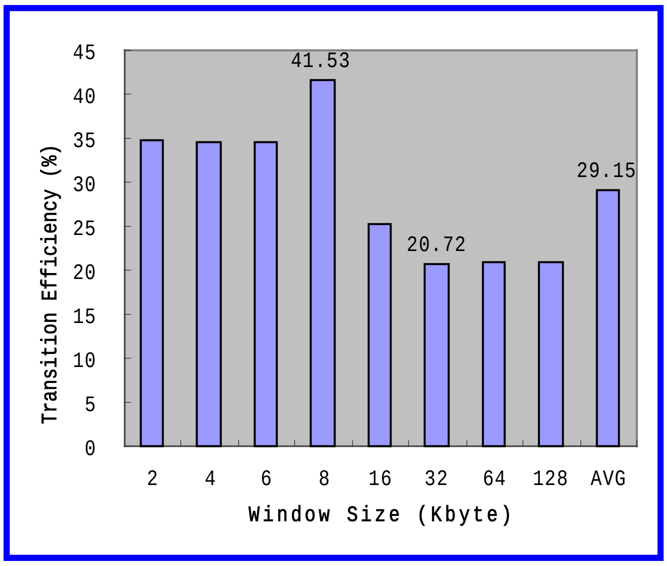

Figure 5.10. NAT-PT transition efficiency

Like the above two tunnel techniques, we also test effect on FTP performance resulted from transition technique. Figure 5.11 presents the throughput of “GET” at various TCP window size the same as raw traffic testing of pure IPv4 connection, pure IPv6 connection and connection through one NAT-PT box. The transition efficiencies of the translation mechanism are calculated in Figure 5.12.

Compared with raw traffic experiments, transition efficiency of FTP application is even less. The average transition efficiency of raw traffic is about 30%, while the average transition efficiency of FTP application is only 23%. The main reason of this difference is that FTP connection utilizes not only NAT-PT function but also FTP-ALG as well, which results in more process delay and further performance degradation.

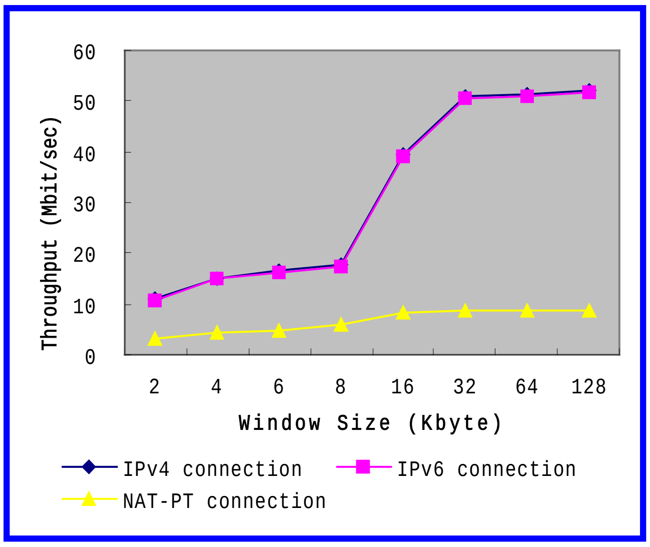

Figure 5.11 Pure IPv4, pure IPv6 and NAT-PT FTP performance

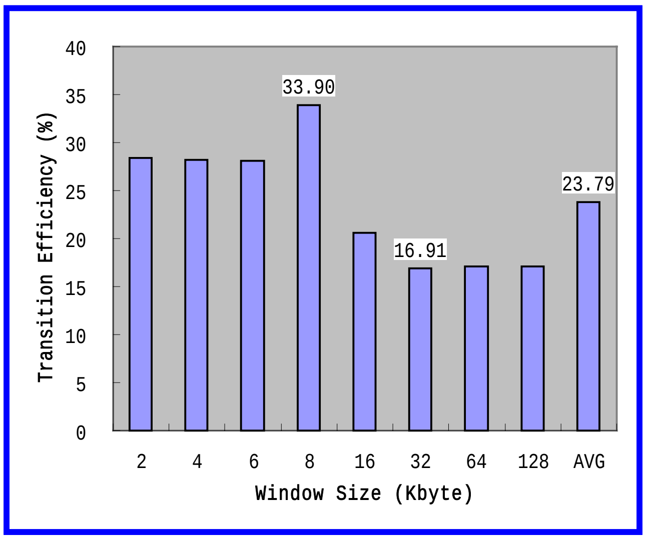

Figure 5.12 Transition efficiency of NAT-PT of FTP application

5.2.4 Distributed NAT-PT

For translation technique has to track the sessions it supports and mandates, inbound and outbound datagrams pertaining to a session have to traverse the same NAT-PT node, which thus turns to be a single point of failure for network communication. Therefore, we try to improve NAT-PT performance by transforming traditional centralized system into distributed system. Although our distributed NAT-PT inherits distributed system advantages over centralized NAT-PT, such as higher reliability, more flexibility, and convenient system augment, we should first test add-on synchronization effect on the original system.

We conduct two sets of testing to evaluate synchronization effect on performance. One is original centralized NAT-PT without any synchronization function. TCP throughput are tested between a pair of hosts, an IPv4 host C in subnet 3 and an IPv6 host A in subnet 1. The other set of testing are still implemented on the same pair, but through the distributed NAT-PT. The specific testing scenario is like following. NAT-PT A is the default gateway of IPv6 network so that IPv6 packets from host A are sent to it for translation. On the other side, IPv4 network set NAT-PT B as its default gateway. IPv4 packets returning from host C will be sent to NAT-PT B. As these two translators conduct mapping information synchronization, datagrams pertaining to the same session can pass through different NAT-PT nodes. We also test TCP throughput between this pair under such network condition. Figure 5.13 presents the comparison of two sets of results.

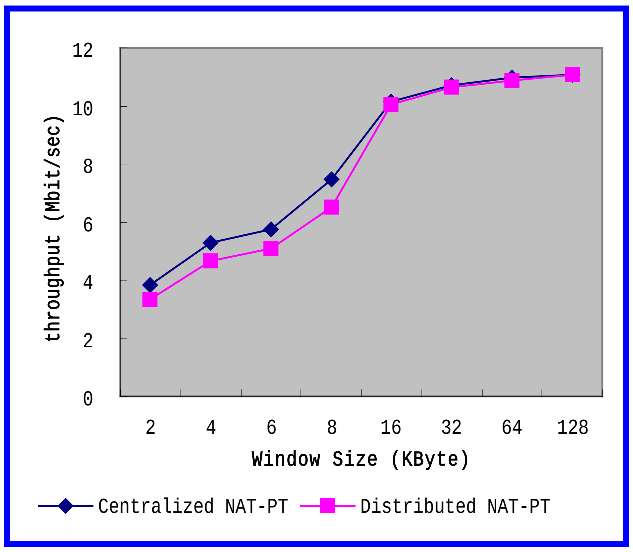

Figure 5.13. Centralized NAT-PT versus Distributed NAT-PT

From the above figure, we can easily see that synchronization brings on little effect on system performance. With the window size increasing, the overhead of synchronization has little effect on the throughput.

After evaluating synchronization effect, we conduct another group of experiments testing performance difference between distributed system and centralized system. As described before, we first establish two simultaneous FTP connections that both traverse through one single centralized NAT-PT and test end-to-end throughput of one of these two connections. Then we test network performance of the same two simultaneous FTP connections but through two distributed NAT-PT.

We test end-to-end throughput with different file size varying from 1Mb to 500Mb. TCP window size is fixed at 16KB for every experiment. We compare two sets of experiment results in Figure 5.14. It is obvious that network performance is improved in distributed system. Transmitting one 50 MB file, for example, end-to-end throughput of FTP connection through centralized NAT-PT is 8.37Mbit/sec. When the same experiment is conducted through two distributed NAT-PTs, the throughput increases to 11.57Mbit/sec, improved about 35%. These results confirm our initial assumption that distributed NAT-PT could improve system performance over centralize NAT-PT.

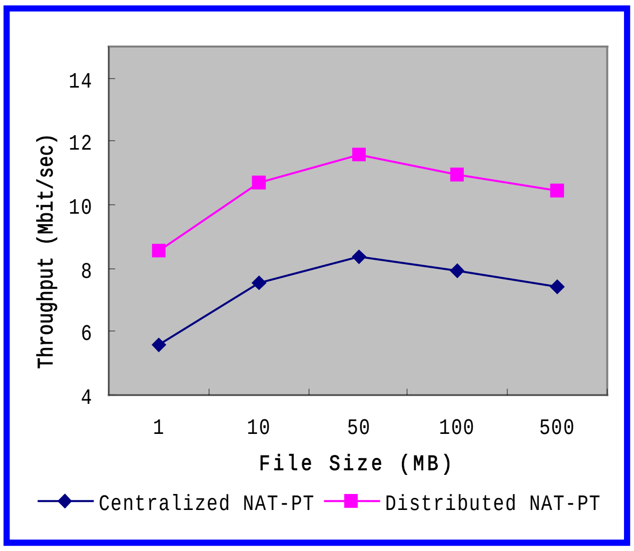

Figure 5.14. Distributed NAT-PT versus Centralized NAT-PT

5.3 Comparisons and Analysis

We elaborate comparison and analysis in the following two sections. In section 5.3.1, we compare performance between tunnel technique and translation technique. In section 5.3.2, we compare distributed NAT-PT with centralized NAT-PT.

5.3.1 Tunnel versus Translator

We summarize the transition efficiency of three resolutions in Figure 5.15. Three types of transition techniques are juxtaposed basing on window size. Obviously, tunnel techniques, whether IPv6-over-IPv4 or IPv4-over-IPv6 (Twin-Glass), present much higher transition efficiency than translation technique NAT-PT. Considering overall performance, the average transition efficiency of IPv6 over IPv4 tunnel techniques is 2.8 times than NAT-PT and Twin-Glass is 2.5 times than Nat-PT. There are two possible factors which may lead to such difference.

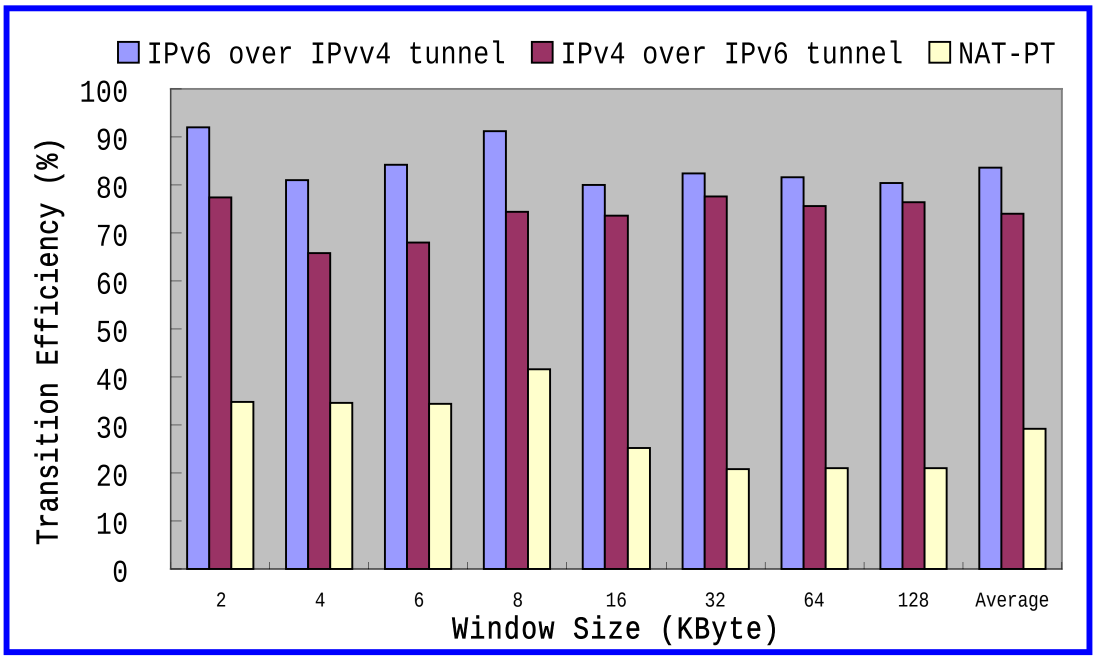

Figure 5.15. Transition efficiency of three kinds of transition mechanism

The primary reason is the longer delay that packets suffer from NAT-PT translation mechanism compared with tunnel technique. Regarding tunnel technique, encapsulating point just needs to modify outgoing packets, adding an IP header with the new source and destination address of two endpoints respectively. At the other end of tunnel, decapsulating point implements head removing and restores the original packet. Upon modifying, tunnel could leave the packet into the network community and do not have to track such modification.

When conducting transition, however, NAT-PT node has to implement a more complex course. Besides head modification on packets like tunnel techniques, it also has to track such modification. In other words, when IPv6 packets traverse the IPv4 and IPv6 border, NAT-PT node assigns an available IPv4 address from its IPv4 address pool and records this assignment into its address-mapping table. In the other direction, when the IPv4 packets with the destination address that has been assigned from the address pool return back to translator node, NAT-PT also needs to match the address from the corresponding mapping tuple to find the corresponding IPv6 address. Such tracking process is a more time-consuming process.

The other reason that may lead to longer delay of translator is that this NAT-PT version runs on user application level, while the other two tunnel techniques are both implemented in the kernel level of Operation System. In the system hierarchy, kernel functions processing less data transfer and thus are implemented more quickly. On the contrary, user application data needs to be transmitted from lower level up to the top level for modification. After being modified, it again returns back to the lower level. This transfer round also results in longer delay of NAT-PT.

5.3.2 Distributed NAT-PT versus Centralized NAT-PT

As shown in Figure 5.14, network performance of distributed NAT-PT is improved compared with centralized NAT-PT. This is one of the main purposes we implement this distributed system. Shown from our test results, NAT-PT is a time-consuming and comparatively less efficient transition solution. Since one NAT-PT runs slowly, we could improve the overall performance by conducting load balancing between two translation boxes. Our test results confirm our initial assumption. Besides the improved performance, this distributed system brings us more benefits as well, such as high reliability, incremental growth, better price/performance ratio etc.

Although distributed system has many advantages over centralized system, this needs a few additional prerequisites. One of the original goals of building distributed systems is to make them more reliable than single-processor systems. The idea comes from that if machine goes down, some other machine takes over the job. A highly reliable system must be highly available, but that is not enough. Data entrusted to the system must not be lost or garbled in any way, and if files are stored redundantly on multiple servers, all the copies must be kept consistent. In general, the more copies that are kept, the better the availability, but the greater the chance that they will be inconsistent, especially if updates are frequent.

To keep all the copies of data consistent, distributed system must implement particular mechanism, which unavoidably consumes part of system resource and effect system performance in a certain degree. We also conduct a group of tests to evaluate such effect. From Figure 5.13, we notice that synchronization brings on little effect on system performance. With the window size increasing, such effect lessons and even eliminates.

Chapter 6 Conclusion

The Internet, which plays more and more important role in our life, is growing at a striking speed. To meet the new emerging application requirements, it is destined to find a specific future direction for the replacement of the current IP version, and the feasible result seems to be IPv6. It not only solves the address shortage problem of current IPv4, but also brings on quite a few advantages, such as network security, auto-configuration, extensibility, mobility, and so on.

For IPv4 applications and services could not be replaced by IPv6 in a short term, integration and coexistence is a prerequisite to enable smooth transition. As the transition period will be lengthy, we can roughly divide it into three phases, which require corresponding transition techniques in each stage. Researching network performance under particular transition techniques leads us to a better understanding of the theoretical and empirical properties of these techniques.

Regarding one of specific transition techniques – NAT-PT, we implement some new features on top of existing transition function, transforming the original centralized system into a distributed system. We also conduct some experiments to evaluate distributed NAT-PT performance. Our test results show that overall performance of distributed NAT-PT is improved compared with original centralized system, thus confirming the usefulness of this distributed NAT-PT.

6.1 Summary of Work

In this thesis, we present comparison of three typical IPv6 and IPv4 transition techniques, which include IPv6 over IPv4 tunnel, IPv4 over IPv6 tunnel (Twin-Glass), and NAT-PT. End-to-end TCP performance metric Throughput is measured at the receiver side in each test item. In order to process precise quantitative analysis on these techniques, we form a new metric transition efficiency for quantificational comparison among various techniques.

Our results lead to the following conclusion that tunnel techniques, whether IPv6 over IPv4 tunnel or IPv4 over IPv6 tunnel (Twin-Glass), present more efficient performance than translation technique such as NAT-PT. They result in relatively less performance degradation on end-to-end network performance. This advantage turns even more obvious with the increasing window size. The primary reason is the different internal mechanisms of each specific transition technique. In addition, program implementation efficiency is another factor that may affect ultimate performance.

Besides conducting experiments with testing tools, we also evaluate FTP connection performance under these corresponding transition mechanisms and compare the results with the previous ones obtained from raw traffic testing. According to our experiment results, FTP performance of tunnel techniques presents consistent transition efficiency with raw traffic performance, while FTP performance under NAT-PT is even less efficient than raw traffic results. The main reason of such difference is that FTP-ALG has to be processed during a successful FTP connection, which results in further delay and transition efficiency decrease.

Shown in our test results, NAT-PT is a time-consuming and less efficient transition technique. How to improve its performance is significant for IPv4 to Ipv6 integration. We transform the original centralized NAT-PT to distributed NAT-PT, eliminating single point of failure and thus improving its performance. As a distributed system, this distributed NAT-PT inherits general advantages over centralized system, such as higher reliability, load balancing, and convenient system augment.

Upon implementing, we also conduct tests related to distributed NAT-PT. We first conduct a group of tests to evaluate the consequence of implementing data synchronizing between distributed NAT-PT systems. The test results present little side effect on original centralized system caused by synchronization. We then process another set of experiments, which conduct comparison between distributed NAT-PT system and centralized NAT-PT system. Our test results prove the usefulness of distributed NAT-PT, which improves overall network performance. Practically speaking, furthermore, this type of distributed system is convenient to combine with our existing dynamic IPv4 over IPv6 tunneling system – TwinGlass – and thus provides users an integrated and comprehensive transition solution for future IPv4 to IPv6 migration.

6.2 Future Works

During the period of IPv4 to IPv6 migration, many of transition techniques will be implemented in different phases and application scenarios. Among all these translation techniques, NAT-PT plays a vital role. As we see from our test results, it is a relatively slow solution. One of the possible reasons is that this version conducts translation issues in the userland, which results in longer delay. Thus, we could implement its function at the kernel level, which should result in shorter transition delay and present much better performance.

In addition, we could further test this distributed NAT-PT performance in a dynamic routing environment, which supports some advanced network features like load balancing. Theoretically, the whole network performance should be improved.

Bibliography

[1] Vinton G. Cerf and Robert E. Kahn, “A Protocol for Packet Network Intercommunication”, IEEE Trans on Comms, Vol Com-22 , No 5 May 1974

[2] General Packet Radio Service (GPRS),
http://www.gsmworld.com/technology/gprs/index.shtml

[3] 3GPP, http://www.3gpp.org/

[4] R. Droms. “Dynamic Host Configuration Protocol (DHCP)”, RFC 2131, March 1997.

[5] Egevang, K. and P. Francis, “The IP Network Address Translator (NAT)”, RFC 1631, May 1994.

[6] S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460, December 1998

[7] R. Hinden, S. Deering, “IP version 6 Addressing Architecture”, RFC 2373, July 1998

[8] S. Thomson, T. Narten, “IPv6 Stateless Address Autoconfiguration”, RFC 2462, December 1998

[9] T. Narten, E. Nordmark, W. Simpson, “Neighbor Discovery for IP Version (IPv6)”, RFC 2461, December 1998

[10] S. Kent, R. Atkinson, “IP Authentication Header”, RFC 2402, November 1998

[11] R. Gilligan, E. Nordmark, “Transition Mechanisms for IPv6 Hosts and Routers”, RFC 2893, August 2000

[12] G. Tsirtsis, P. Srisuresh, “Network Address Translation- Protocol Translation”, RFC 2766, February 2000

[13] P. Srisuresh, “IP Network Address Translator (NAT) Terminology and Considerations”, RFC 2663, August 1999

[14] Nordmark, E., “Stateless IP/ICMP Translator (SIIT)”, RFC 2765, February 2000

[15] Srisuresh, P., Tsirtsis, G., Akkiraju, P. and A. Heffernan, “DNA extensions to Network Address Translators (DNS_ALG)”, RFC 2694, September 1999

[16] B. Carpenter, K. Moore, “Connection of IPv6 Domains via IPv4 Clouds”, RFC 3056, February 2001

[17] Andrew S. Tanenbaum, “Distributed Operating Systems”, Prentice-Hall International, Inc.

[18] RIP, Routing Informaiton Protocol,
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/rip.htm

[19] OSPF, Open Shortest Path First,
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ospf.htm

[20] IGRP, Interior Gateway Routing Protocol
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/igrp.htm

[21] EIGRP, Enhanced Interior Gateway Routing Protocol
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/eigrp.htm

[22] Walton, Sean. “Linux Socket Programming”

[23] W. Richard Stevens, “TCP/IP Illustrated, Volume 1 The Protocol”, Addison-Wesley

[24] Postel, John. “Transmission Control Protocol”, RFC 793, September 1981

[25] Postel, John. “User Datagram Protocol” RFC 768, August 1980

[26] NAT-PT, http://www.ipv6.or.kr/english/natpt-overview.htm

[27] Kai Wang, Ann-Kian Yeo, A. L. Ananda. “Twin-Glass: A Transparent and scalable Solution for IPv4 to IPv6 Transition”, ICCCN2001, Scottsdale, Arizona, USA, 15-17 October 2001.

[28] IPERF, http://dast.nlanr.net/Projects/Iperf/

[29] TCPDUMP, http://www.tcpdump.org

[30] TCPTRACE, http://www.tcptrace.org

[31] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, “Framework for IP Performance Metrics”, RFC 2330, May 1998