Proceedings of the Asia-Pacific Advanced Network **2011** *v. 31, p. 12-22.*
http://dx.doi.org/10.7125/APAN.31.2
ISSN 2227-3026

Internet Protocol Version 6 (IPv6) - Internet Protocol Version 4 (IPv4) Network Address, Port & Protocol Translation And Multithreaded DNS-Application Gateway

Abhinav Singh[1], Navpreet Singh[1,*] and Vinay Bajpai[2]

1. Computer Center, IIT Kanpur, U.P. India.
2. Indian Institute of Science, Education and Research, Bhopal - 462 023, India

E-Mails: abhinavs@iitk.ac.in; navi@iitk.ac.in; vinayb@iiserbhopal.ac.in

* Author to whom correspondence should be addressed; Tel.: +91-512-2597371; Fax: +91-512-2590413

Abstract: Due to emerging growth in Internet, deployment of IPv6 has become mandatory. Transition of IPv4 to IPv6 is not a single day task. Due to compatibility issues, both the protocols will coexist for years during the transition period. In this paper we will study the design and implementation of an IPv6 to IPv4 translator which provides the communication mechanism from IPv6 only to IPv4 only networks and vice versa, by doing alteration in IP header of packets. The multithreaded DNS application gateway is also implemented to simultaneously serve multiple DNS queries and convert DNS replies from IPv4 to IPv6 or vice versa.

Keywords: IPv6-IPv4; NAT; DNS; TCP; UDP; HTTP; Sk_Buff.

---

1. Introduction

The unexpected infusion of network equipments and growth of Internet is creating serious problem with the available IPv4 address space. The solution of this problem is to move on to next generation Internet Protocol version, IPv6 (Internet Protocol Version 6) [2].

But the transition from IPv4 to IPv6 is a slow process and the major drawback is that there is no compatibility between IPv4 and IPv6. In this paper we have presented an implementation of

an IPv6-IPv4 translator by using Network Address, Port and Protocol translation and Multithreaded DNS Gateway to handle simultaneous DNS requests.

Using this translator, an IPv6 only host can communicate to any IPv4 only host as the translator converts IPv6 packets received from the IPv6 host into IPv4 packets and vice versa.

DNS Gateway is used to convert A address into AAAA address of requested name. Since this conversion consumes some time so multithreading is used to handle multiple requests simultaneously.

2. Approach

We use the methodology as defined in RFC 2766 [1]. The basic architecture of translator is shown in Fig 1.

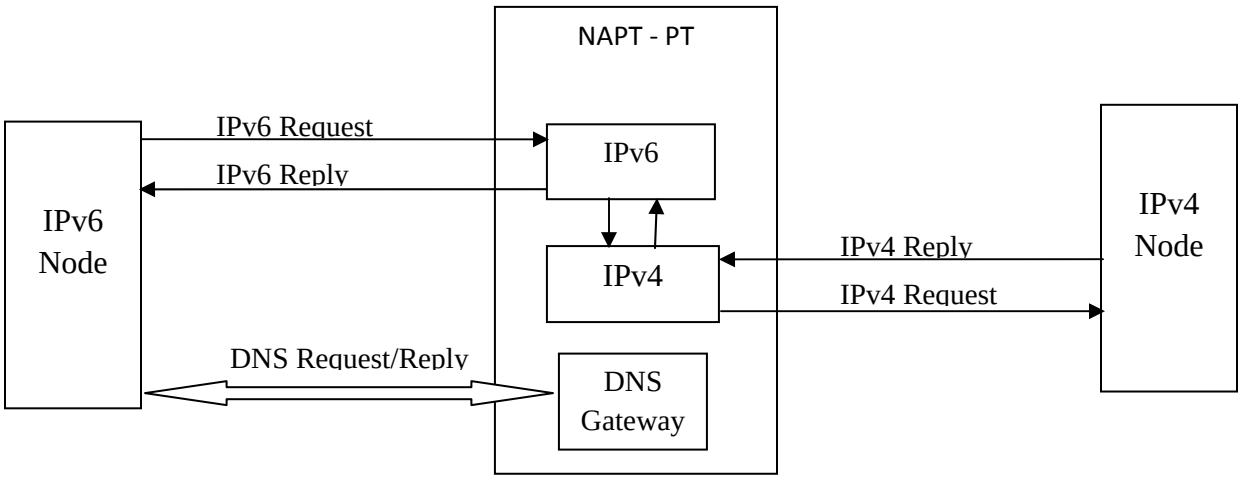

Figure 1. Architecture of Translator

The translation of packet from IPv6 to IPv4 takes place in following steps:

1. An IPv6 only host wants to access IPv4 only host using its DNS name. IPv6 host sends query to DNS gateway to resolve the address of target host. IPv6 hosts sends type 'AAAA' query to DNS gateway.

2. DNS gateway changes the type 'AAAA' query into type 'A' query and requests IPv4 DNS (in the IPv4 network) to resolve the name to IPv4 address.

3. DNS gateway receives DNS reply and converts the type 'A' reply into type 'AAAA' reply and sends back to IPv6 host.

4. Now IPv6 host has an IPv6 address of target IPv4 host.

5. IPv6 host generates IPv6 packet and the packet is routed to the translator. The translator receives the packet and translate IPv6 packet into IPv4 packet by changing the header and forwards the packet to IPv4 target host [6].

6. Target IPv4 host generates the IPv4 reply and packet is routed to the translator. Now the translator translates back the IPv4 packet into IPv6 packet and forwards the packet to IPv6 host.

There are two parts of the translator:

1. Network Address, Port and Protocol Translation.
2. DNS Application level gateway.

2.1 Network Address, Port and Protocol Translation

Socket buffers (sk_buff) are the buffers in which the Linux kernel handles network packets. The packet is received by the network card, put into a sk_buff and then passed to the network stack, which uses the sk_buff all the time. We have designed a translator module and inserted it into the kernel module system. This module uses the sk_buff and fetches those packets which need to be translated. Packet will be accepted by kernel network stack and no ICMP error for unable to find route is generated.

When packet is received by translator module, the header of the packet is removed and module adds the new header to the payload. Header translation contains only network address translation. Port translation depends on what protocol is used by packet i.e. TCP, UDP or ICMP.

If the protocol is UDP and TCP then port is translated and a new port is assigned to the packet. New checksum is calculated and module sets the new transport header for the packet. Now packet is ready for transmition into the new network.

Now translator module checks the new route of the packet and sends the translated packed to the desired network card by using *dst_output* command and network card transmits the packet in to the network.

2.1.1 Network Address Translation

Since IPv6 has 128 bit address space and IPv4 has 32 bit address network address translation is required to make communication between IPv4 and IPv6 networks.

When IPv4 address is translated into IPv6 address a 96 bit prefix is added to 32 bit IPv4 address to make it 128 bit IPv6 address.

Linux system is used as a translator. Two NIC cards are installed, one is connected to IPv4 network and other is connected to IPv6 network. A module [8] is installed in kernel which captures the packets from both the NIC cards and translates the IPv4/IPv6 addresses and forwards the new generated packet to desired network. If the packet is not recognized by the module, packet is dropped or rejected. An available IPv4 address and port number from pool is assigned to the new connection.

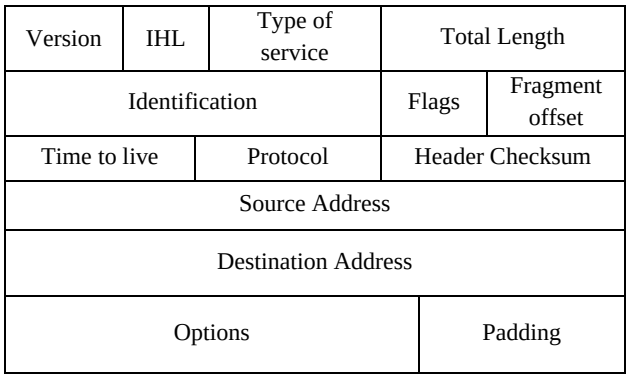

Figure 2.1. IPv4 Header Format

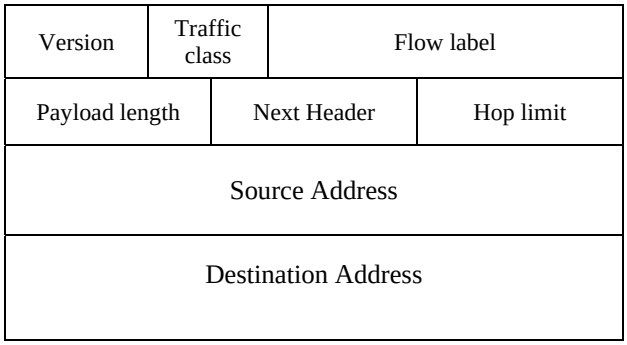

Figure 2.2. IPv6 Header Format

Fig 2 shows the IPv6 and IPv4 header format.

The fields to be translated are:

IPv4 Header

- Version :- 4
- Total length :- Payload length of IPv6 Header plus header length of IPv4 header.
- Time To Live:- Hop Limit value from IPv6 header decreased by one
- Protocol:- Next Header copied from IPv6 header.
- Header Checksum:- Computed when the IPv4 header has been created
- Source Address :- In 32 bit format.
- Destination Address :- In 32 bit format.

IPv6 Header

- Version :- 6
- Payload length:- Total length of IPv4 header minus IPv4 header length.
- Next Header:- Protocol field copied from IPv4 header.
- Hop Limit:- Time To Live value from IPv4 header decreased by one
- Source Address :- 128 bit format
- Destination Address:- 128 bit format

Other fields are directly copied or discarded.

2.1.2 Port Translation

Since Main purpose of translator is to make communication between IPv4 and IPv6 network without dual-stack system. Our purpose will fail if we use one-to-one mapping of address i.e. IPv4 address are assigned to each communicating devices, so Port translation is required.

In IPv6-IPv4 translation the purpose of port translation is same as in IPv4 port translation. Only one public IPv4 address is assigned to the group of IPv6 hosts and they will communicate with IPv4 network by this IP only.

Port from available pool is assigned to every connection from the IPv6 hosts which try to communicate through translator. An entry is made in table for further communication.

2.1.3 Protocol Translation

Protocol translation [4] is for translation of UDP, TCP and ICMP. In TCP and UDP we need to change the port number and pseudo IP header and update checksum of the packet and forward to the desired network.

ICMP protocols are different in both networks. For IPv6 it is ICMPv6 and for IPv4 it is ICMPv4. So the two kinds of packets are translated into each other.

2.2 Multithreaded DNS – Gateway

DNS Gateway[7] synthesizes AAAA records from A records. DNS Gateway is used with an IPv6/IPv4 translator to enable communication between an IPv6-only host and an IPv4-only host, without requiring any changes to either the IPv6 or the IPv4 node, for the class of applications that work through NAT.

DNS gateway adds 96 bit prefix to A records to make it AAAA records. When IPv6 machine tries to communicate with IPv4 host, if queries DNS for AAAA records. Since there is no entry for such records, the DNS gateway returns AAAA record to IPv6 client.

2.3. Implementation

The translator system is implemented on Linux platform i.e. Linux machine will act as a IPv6/IPv4 translator gateway [5].

Kernel modules are created and installed in the system and translator machine is connected to the networks. It works as a Translator cum Router which receives the packets, translates them and forwards to desired networks.

2.3.1 Network Address, Port and Protocol Translation

Following steps are taken by the translator:-

A. From IPv6 to IPv4

- DNS resolves the IPv4 address of the destination host and adds a 96 bit prefix to the address of destination machine to make it 128 bit IPv6 address. This address is provided to the client host and it uses this address to send the packet.

- Packet is received by translator and added 96 bit is removed by translator and an available IPv4 address and port number from pool is assigned for header translation.

- Translated packet is forwarded to IPv4 network.

B. From IPv4 to IPv6

- IPv4 packet is received by translator and the table is searched for original destination of packet i.e. IPv6 address and port number.
- 96 bit prefix is added to source IPv4 address.
- Translated packet is forwarded to IPv6 network.

The Translator has been tested on following configuration:

Translator IPv6 address: 4001:4490::c0a8:103

Translator IPv4 address: 172.31.210.225

IPv6 machine address: 4001:4490::c0a8:105

Now IPv6 machine tries to communicate with multiple IPv4 hosts in LAN and Internet.

Packet from IPv6 Machine:-

Source:- 4001:4490::c0aa:103 Source port :- 15432
Destination:- 4001:4490::172.31.1.212 Destination port :- 80

After Translation :-

Source:- 172.29.0.1 Source port :-5000 (From Pool)
Destination:- 172.31.1.212 Destination port :- 80

Packet from IPv4 Machine:-

Source:- 172.31.1.212 Source port :- 80
Destination:-172.29.0.1 Destination port :- 5000

After Translation :-

Source:- 4001:4490::172.31.1.212 Source port :- 80 (From Table)
Destination:- 4001:4490::c0a8:103 Destination port :- 15432

Prefix :- 4001:4490::/96 (This prefix can be manually added for IPv4 destinations whose name is not known)

2.3.2 Multithreaded DNS – Gateway

Socket Programming is used to handle DNS requests. Socket is created and bound to DNS port i.e. port 53. When there is some request on port 53, it forwards the request to local DNS and when response is received from DNS it alters the DNS response and adds required prefix and sends it back to the requesting client.

Multiple threads are used to handle multiple clients. When one client request comes, it creates new thread to handle that request and again start waiting on port 53 for new requests.

The steps involved are:

- IPv6 client sends AAAA query to DNS gateway.
- DNS gateway changes AAAA query into A query and sends it to DNS.
- DNS gateway receives the response from DNS.
- DNS gateway adds 96 bit address prefix in 32 bit IPv4 address response from the server and makes the response AAAA type.
- Now sends the modified response back to IPv6 client.
- Client receives the AAAA response and it will think it is communicating with IPv6 machine.

Example :-

IPv6 machine with IP address 4001:4490::c0a8:105 wants to know the AAAA address of www.iitk.ac.in.

- Gateway changes the query into A and sends it to DNS.
- Response from DNS is 172.31.1.212 type A.
- The address is changed to 4001:4490::172.31.1.212 type AAAA

Now IPv6 client understands the address and starts communicating with www.iitk.ac.in with the help of IPv6/IPv4 translator gateway.

Since the response of DNS is large and sometimes it contains many answers and it takes some time to search the CNAME, IP address from the response. So the gateway will be busy until it gives back response to IPv6 client. If there are multiple queries, then it decreases the efficiency

of DNS gateway. To overcome this drawback, the concept of Multithreading is used. A new thread is created for every query. By multithreaded DNS gateway, multiple queries will be resolved at same time.

3. Results

We were able to make successful communication between IPv6 only and IPv4 only networks and Multithreaded DNS gateway is able to handle multiple requests simultaneously which fasten the translation process.

This system has been tested on IIT-Kanpur IPv6 Network lab for following applications.

- HTTP/HTTPS
- SSH
- Telnet
- FTP (Passive mode)
- nslookup

We have also tested the performance of the translator in terms of per packet delay introduced in the translation process and also compared the DNS resolution delay as compared to IPv4 only resolution. It has been found that the per packet translation delay is ~ 40 usec. In the multithreaded DNS resolution, the corresponding mapping to a dummy IPv6 address is not adding any significant delay and the multithreaded DNS resolution time is comparable to simple IPv4 DNS resolution time (which is variable and depends on the remote site and Internet delays).

So under low load conditions, the delay introduced will not affect the end-to-end performance of the network. However the same could not be tested on high load conditions as the hardware used for implementing the translator is not a very high end system and if tested on the current hardware, the delay would significantly increase under high load conditions.

4. Conclusions

We have described the Implementation of an IPv6 - IPv4 Translator using Network Address, Port and Protocol translation between IPv6 and IPv4 and the design of multithreaded DNS Gateway to handle multiple simultaneous requests. There is no need of putting extra hardware,

just install the system into any Linux machine and assign an IPv4 pool of addresses. The Linux machine will server as the IPv6-IPv4 Translator Gateway.

By port translation only one IPv4 address is assigned to multiple IPv6 machines. It will consume less IPv4 address for translation because if we assign one IPv4 address to one IPv6 machine it will be like dual-stack system and will not solve our purpose.

Multithreaded DNS gateway is used to handle multiple requests which can speed up the translation process.

The current system is tested on low traffic volume. The work can be extended by testing the translator on high traffic load and the delay introduced on load can also be tested.

The translator is designed using stateful translation mechanism. The same can be extended to implement stateless translator [9] and the translator code can further be modified to work as stateless translator.

Many commercial products are also available which provide Gigabit or Carrier Grade performance. The current system can be upgraded to meet the performance of commercially available products.

Acknowledgments

We wish to acknowledge the support of BITCOE (BSNL IIT Kanpur Centre of Excellence), IIT Kanpur for providing the resources to design and implement the IPv6-IPv4 Translator.

References

1. G. Tsirtsis; P. Srisuresh. Network Working Group. Request for Comments 2766, February 2000.
2. S. Deering; R. Hinden. Internet Protocol Version 6 (IPv6) Specification. Request for Comments 2460, December 1998.
3. K. Egevang; P. Francis. The IP network address translator (NAT) Request for comments 1631, May 1994.
4. J. Hagino; K. Yamamoto. An IPv6-to-IPv4 Transport Relay Translator. Request for Comments 3142, June 2001.

5. Masaki Nakajima; Nobumasu Kobayas. IPv4/IPv6 Translation Technology.

6. Marc E. Fiuczynski; Vincent K. Lam; Brian N. Bershad; The Design and Implementation of an IPv6/IPv4 Network Address and Protocol Translator. Department of Computer Science and Engineering, University of Washington Seattle, Washington 98195.

7. P. Srisuresh; G. Tsirtsis; P. Akkiraju; A. Hefferan. DNS Extension to Network Address Translators. NAT Working Group INTERNET-DRAFT.

8. Tomasz Mrugalski. IP46nat. http://klub.com.pl/ip46nat/

9. C. Bao; C. Huitema; M. Bagnulo; M. Boucadair; X. Li. IPv6 Addressing of IPv4/IPv6 Translators. Request for Comments 6052.

Submitted March 12, 2011
Accepted May 3, 2011