International Journal of Research in Engineering, Science and Management
Volume-1, Issue-10, October-2018
www.ijresm.com | ISSN (Online): 2581-5792

IPv4/IPv6 Address Translation Using Loadable Kernel Module

Mandar Lokhande[1], Siddhi Pathak[2]
[1]*Professor, Department of Computer Science, MGM's Jawaharlal Nehru Engg. College, Aurangabad, India*
[2]*Student, Department of Computer Science, MGM's Jawaharlal Nehru Engg. College, Aurangabad, India*

***Abstract*—Internet Protocol version 4 (IPv4) addresses have been reported to be nearing exhaustion and the next generation Internet Protocol version 6 (IPv6) is gradually being deployed in the Internet. IPv6 provides a much larger address space, better address design and greater security, among other benefits. The migration from IPv4 to IPv6 cannot be achieved in a short period thus the two protocols will co-exist for some time. Unfortunately, these two protocols are incompatible; hence for them to co-exist, various IPv4-to-IPv6 transition mechanisms have been developed. We analyzed the different site-to-site tunneling mechanisms through a theoretical and experimental evaluation to study their appropriateness in IPv6 deployment. We implemented NAPT-PT concept in an existing IPv4-IPv6 translator. NAPT-PT allows a set of V6 hosts to share a single V4 address thereby allowing more V6 nodes, than supported by NAT-PT, to establish communication with V4 nodes.**

***Index Terms*—Linux, IPv4, IPv6, IPV4/IPv6 transition, Dual Stack, Tunneling, NAT-PT.**

I. Introduction

An Internet Protocol address or IP address is a numerical label tagged to devices that take part in a local or wide area computer network that actively uses IP for communicating data and information. An IP address is a 32-bit number and this system is the Internet Protocol Version 4 or the IPv4 and most networks still use this format today, no one expected that the number of internet users will increase in this way so when IPv4 was designed it takes that the maximum numbers of devices that will have IP is 2^32. That means an IP address can provide a maximum (4,294,967,296) possible address, but in 1992 , the Internet Engineering Task Force (IETF) realized that current IP address space was running out as a result of rapid growth of Internet size and applications, so the need for a new protocol that has larger address space and improvement features was needed, because of that a solution for this problem was solved by IPv6 (internet protocol version 6) which offers a huge address space which will be more than enough ,but unfortunately IPv4 and IPv6 are incompatibles protocols and it is impossible to migrate from IPv4 to IPv6 in one day. Many transition mechanisms from IPv4 to IPv6 and vice versa had been proposed, some researchers divided these methods according to the techniques used in the transition to three transition methods: Dual-Stack method , Tunneling method and translation method as shown in Fig. 1 [5].

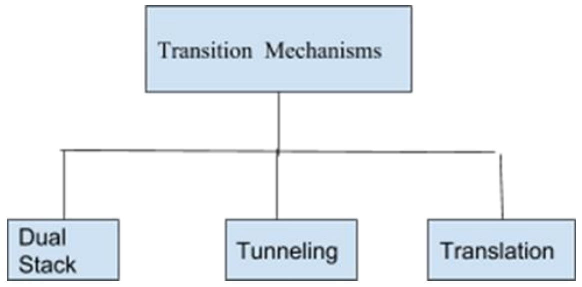

Fig. 1. IPV4/IPV6 Transition mechanisms

A. *Dual Stack*

It support both IPV6 and IPV4 on network device i.e. both IPv4 and IPv6 protocol stacks are deployed on the same node. As the word means, dual-stack mechanisms include two protocol stacks that operate in parallel and allow network nodes to communicate either via IPv4 or IPv6. They can be implemented in both end system and network node. In end systems, they enable both IPv4 and IPv6 applications to operate at the same time. The Dual-stack capabilities of network nodes support the transport of both IPv4 and IPv6 packets. In the dual-stack mechanism, specified in IETF RFC2893, a network node includes both IPv4 and IPv6 protocol stacks in parallel (Fig. 2). IPv4 applications use the IPv4 stack, and IPv6 applications use the IPv6 stack. Flow decisions are based on the version field of IP header for receiving, and on the destination address type for sending.

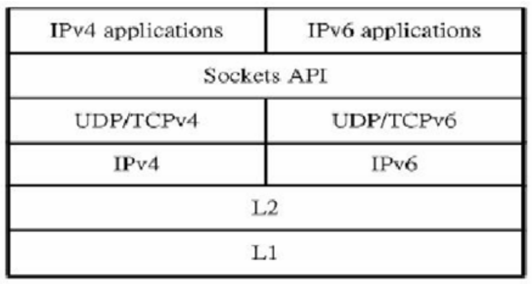

Fig. 2. Dual-stacks transition mechanism

The types of addresses are usually derived from DNS lookups; the appropriate stack is selected in response to the types of DNS records returned. Many off-the-shelf commercial operating systems already have dual IP protocol stacks [6]. Hence, the dual-stack mechanism is the most extensively employed transition solution. However, dual stack mechanisms enable only similar network nodes to communicate with each other (IPv6-IPv6 and IPv4-IPv4). Much more works are required to create a complete solution that supports IPv6-IPv4 and IPv4-IPv6 communications.

B. Tunneling

Tunneling, from the perspective of transitioning, enables incompatible networks to be bridged, and is usually applied in a point-to-point or sequential manner. Three mechanisms of tunneling are presented: IPv6 over IPv4 [9], IPv6 to IPv4 automatic tunneling, and Tunnel Broker.

1) IPV6 over IPv4 mechanism

The IPv6 over IPv4 mechanism embeds an IPv4 address in an IPv6 address link layer identifier part, as shown in Fig. 3 and defines Neighbor Discovery (ND) over IPv4 using organization-local multicast. An IPv4 domain is a fully interconnected set of IPv4 subnets, within the scope of a single local multicast, in which at least two IPv6 nodes are present. The IPv6 over IPv4 tunneling setup provides a solution for IPv6 nodes that are scattered throughout the base IPv4 domain without direct IPv6 connectivity. The mechanism allows nodes, on physical links, which are directly connected IPv6 routers to become fully functional IPv6 nodes.

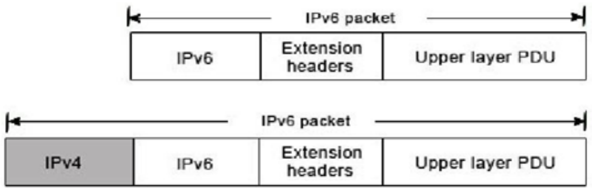

Fig. 3. IPV6 over IPV4 address link layer identifier

2) IPv6 to IPv4 automatic tunneling mechanism

Automatic tunneling refers to a tunnel configuration that does not need direct management. An automatic IPv6 to IPv4 tunnel enables an isolated IPv6 domain to be connected over an IPv4 network and then to a remote IPv6 networks. Such a tunnel treats the IPv4 infrastructure as a virtual non-broadcast link, so the IPv4 address embedded in the IPv6 address is used to find the other end of the tunnel. The embedded IPv4 address can easily be extracted and the whole IPv6 packet delivered over the IPv4 network, encapsulated in an IPv4 packet. No configured tunnels are required to send packets among 6to4 capable IPv6 sites anywhere in IPv4 Internet. Figure 4 shows the structure of the 6to4 address format. The value of the prefix field (FP) is 0x001, which the identifies global unicast address. The Top-Level Aggregation identifier field (TLA) is assigned by the IANA for the IPv6 to IPv4 mechanism. Hence, the IPv6 address prefix is 2002::/16 and the 32 bits after 2002::/16 represent the IPv4 address of the gateway machine of the network in question. The packets thus know the way to any other network. The 6to4 mechanism is the most widely extensively used automatic tunneling technique. It includes a mechanism for assigning an IPv6 address prefix to a network node with a global IPv4 address.

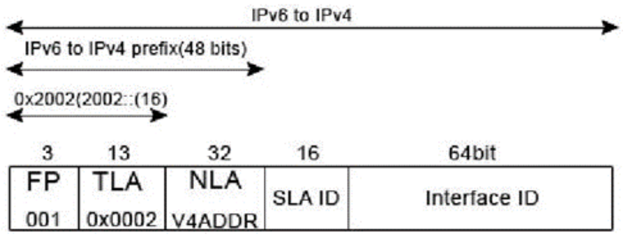

Fig. 4. IPv6 to IPv4 address format

3) IPv6 tunnel broker

The IPv6 Tunnel Broker provides an automatic configuration service for IPv6 over IPv4 tunnels to users connected to the IPv4 Internet. IPv4 connectivity between the user and the service provider is required. The service operates as follows (Fig. 5).

1) The user contacts Tunnel Broker and performs the registration procedure.
2) The user contacts Tunnel Broker again for authentication and providing configuration information (IP address, operating system, IPv6 support software, etc.).
3) Tunnel Broker configures the network side end-point, the DNS server and the user terminal.
4) The tunnel is active and the user is connected to IPv6 networks.

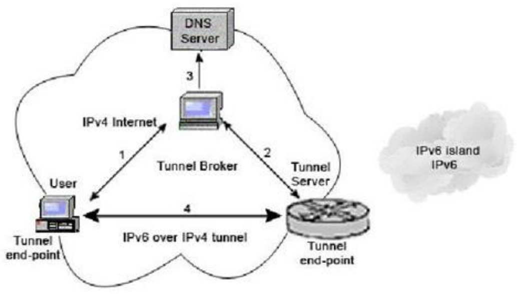

Fig. 5. IPV6 tunnel broker

C. Translation

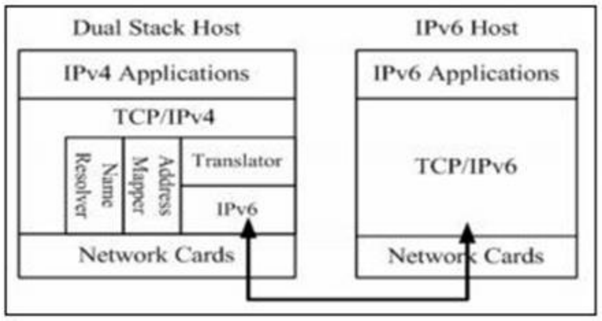

Fig. 6. BIS architecture

The basic function of translation in IPv4/IPv6 transition is to translate IP packets. Several translation mechanisms are based on the SIIT (Stateless IP/ICMP Translation algorithm) algorithm. The SIIT algorithm is used as a basis of the BIS (Bump in the Stack) and NAT-PT (Network Address Translation-Protocol Translation) Mechanism [6].

1) Bump-In-the-stack mechanism

Bump-In-the-Stack (BIS) mechanism (RFC 2767) includes a TCP/IPv4 protocol module and a translator module, which consists of three bump components and is layered above an IPv6 module (Fig. 6). Packets from IPv4 applications flow into the TCP/IPv4 protocol module. The identified packets are translated into IPv6 packets and then forwarded to the IPv6 protocol module. The three bump components are the extension name resolver, which examines DNS lookups to determine whether the peer node is IPv6-only; the address mapper, which allocates a temporary IPv4 address to the IPv6 peer and caches the address mapping; and the translator, which translates packets between IPv4 and IPv6 protocol [6].

2) Network address translation-protocol translation

The NAT-PT mechanism is a stateful IPv4/IPv6 translator. NAT-PT nodes are at the boundary between IPv6 and IPv4 networks. Each node maintains a pool of globally routable IPv4 addresses, which are dynamically assigned to IPv6 nodes when sessions are initiated across the IPv6/IPv4 boundary. This mechanism allows native IPv6 nodes and applications to communicate with native IPv4 nodes and applications, and vice versa. The NAT-PT translation architecture, depicted in Fig. 7, also include one or more ALGs (Application Level Gateways).

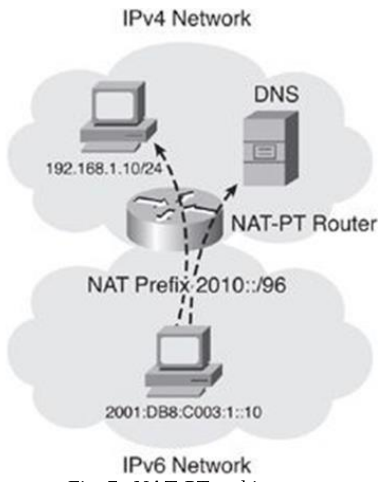

Fig. 7. NAT-PT architecture

The basic NAT-PT function does not snoop packet payloads, and the application may therefore be unaware of it. Hence, the NAT-PT mechanism depends on ALG agents that allow an IPv6 node to communicate with an IPv4 node and vice versa for specific applications. The NAT-PT mechanism is an interoperability solution that needs no modification or extra software, such as dual stacks, to be installed on any of the end user nodes, either the IPv4 or the IPv6 network. This mechanism implements the required interoperability functions within the core network, making interoperability between nodes easier to manage and faster to manifest [6].

II. Translation Mechanisms

The fundamentals of IPv4/IPv6 translation mechanisms are discussed in BIS. But the implementation method is not described because it is specific to the operating system. Additionally, SIP is usually used for multimedia communications, such as voice or video conference applications. However, as these SIP applications depend on service providers, it is difficult for the user to select optimum SIP applications that will support IPv6 communication. In this paper, we extend the BIS mechanisms to support SIP applications, clarify the design for implementation, and develop a special kernel module for Linux OS [4].

The Fig. 8 shows the system model for packet manipulation in the developed kernel module. The functions of this module are classified into address translation function, payload modification function, and DNS message handling function. The kernel module uses the Linux netfilter function to handle a socket buffer for each packet. Therefore, modification of the original Linux kernel is not required in order to use the developed kernel module [4].

A. *Packet Hook in Linux Netfilter*

The need for virtual interface creation can be detected by using netfilter hooks. Netfilter can be used by our implementation to identify many of the events that trigger the routing action. Netfilter consists of a number of hooks at various points inside the Linux protocol stack. It allows user-defined kernel modules to register callback functions to these hooks. When a packet traverses a hook, the packet flows through the user defined callback method inside the kernel module [2].

Netfilter provides a packet manipulation framework inside the Linux 2.4.x and 2.6.x kernel series, and it is also a set of hooks inside the Linux kernel. Therefore, kernel modules can register their callback function with the Linux network stack and the function is called when packets traverse the respective hook points. As netfilter also allows kernel modules to send the hooked packets back to the network stack, these modules can modify packet information without modification of the original Linux kernel [3].

There are five hooks defined in the net-filter architecture, as shown in Fig. 8. In the developed kernel module, outbound packets from both IPv4 and IPv6 applications are hooked at the point NF INET LOCAL OUT. In the Linux network stack, IPv4 and IPv6 are processed separately. Therefore, the developed kernel module receives both IPv4 and IPv6 [9] packets separately from the point NF INET LOCAL OUT. Whereas IPv6 packets from IPv6 applications are sent back to the Linux network stack immediately, at the point NF INET POST ROUTING, IPv4 packets from IPv4 oriented applications virtual IPv4 addresses to IPv4-oriented applications, enabling them to communicate with IPv6 hosts through IPv6 networks. Since the kernel module can be implemented without modification of the general Linux kernel, it can easily be used to support IPv4 applications undergo some manipulation, in respect of address translation and payload modification, before being sent back to the latter point. A similar differentiation is made for inbound packets, where the respective stack points are NF INET POST ROUTING and NF INET LOCAL IN.

Thus, IPv6 applications engage in real IPv6 communication in the normal way, while IPv4 oriented applications perform virtual IPv4 communication through IPv6 networks [3].

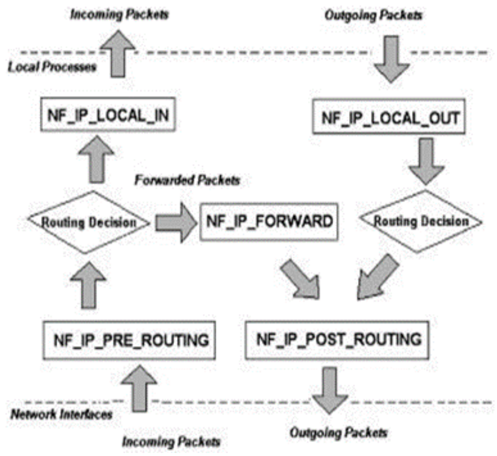

Fig. 8. Netfilter Hook

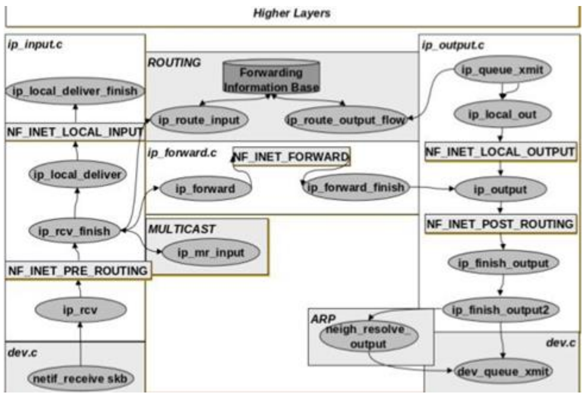

Fig. 9. IPV4/IPV6 translation implementation architecture

III. Conclusion

This paper presents a newly developed kernel module that performs IPv4/IPv6 address translation for IPv4-oriented Applications. This kernel module provides virtual IPv4 addresses to IPv4-oriented applications, enabling them to communicate with IPv6 hosts through IPv6 networks. Since the kernel module can be implemented without modification of the general Linux kernel, it can easily be used to support IPv4 applications in IPv6 networks.

References

[1] D.Shalini Punithavathani and K.Sankaranarayanan " IPv4/IPv6 Transition Mechanisms " European Journal of Scientific Research ISSN 1450-216X vol.34 No 1 (2009), pp.110-124
[2] M. Raste, D.B. Kulkarni "Design and implementation scheme for deploying IPv4 over IPv6 tunnel" Journal of Network and Computer Applications 32 (2008) 66 -77
[3] Katsuhiro Naito, Kazuo Mori, and Hideo Kobayashi " Kernel Module Implementation of IPv4/IPv6 Translation Mechanisms for Ipv4-oriented applications"
[4] Tomek Mrugalski " ip46nat Universal IPv4-IPv6 translator user's guide"
[5] Muzhir Shaban Al- Ani et al " IPv4/IPv6 Translation "International Journal of Engineering Science and Technology (IJEST)
[6] J. William Atwood, Kedar C. Das, Xing (Scott) Jiang "Allowing IPv4 hosts to communicate with IPv6 hosts without modifying the software on the IPv4 or IPv6 hosts ".
[7] Bhuwan Chhetri "Transition from IPv4 to IPv6" Turku University of Applied Science 2015, 52.
[8] Peng Wu, Yong Cui, Jianping Wu, Jiangchuan Liu, Chris Metz, "Transition from IPv4 to IPv6: A State-of-the-Art Survey"
[9] "IETF RFC 791", "I. S. I. at University of Southern California", Internet Protocol DARPA Internet Program Protocol Specification, 1981.
[10] Jun Bi, Jianping Wu, and Xiaoxiang Leng, " IPv4/IPv6 Transition Technologies and University Architecture ", IJCSNS ,VOL.7 No.1, January 2007.