NAT-PT: Providing IPv4/IPv6 and IPv6/IPv4 Address Translation

By J. W. Atwood, Kedar C. Das, and Ibrahim Haddad

In this article, the authors demonstrate how to set up a NAT-PT environment to allow IPv6-only hosts residing on IPv6 networks to communicate with IPv4-only hosts residing on IPv4 networks through a NAT-PT Linux server.

Introduction

This article is a continuation of the series that focuses on IPv6. In previous articles, we covered the features of the new protocol, demonstrated how to enable IPv6 support on a Linux server and provide basic IPv6 functionalities, presented on connecting Linux servers to the IPv6 Internet, and how to provide streaming services over IPv6. In this article we will explain and demonstrate how to connect IPv4-only hosts sitting in IPv4 networks with IPv6-only hosts sitting in IPv6 networks through a Linux machine acting as a NAT-PT (Network Address Translation and Protocol Translation) server.

Scope

IPv6 is the next generation protocol designed by the IETF (Internet Engineering Task Force) to replace the current version of the Internet Protocol, IPv4. Most of today's Internet uses IPv4, which has been remarkably resilient in spite of its age, but it is beginning to have problems. Most importantly, there is a growing shortage of IPv4 addresses, which are needed by all new devices connecting to the Internet. As a result, IETF defined IPv6 to fix the problems in IPv4 and to add many improvements to cater for the future Internet. These improvements come in different areas such as routing, autoconfiguration, security, quality of service (QoS), and mobility. In a previous article in the Linux User and Developer (LUD) in May 2003, we addressed the problems in IPv4 that lead to the design of IPv6 and presented the capabilities of IPv6 that have been developed in direct response to critical business requirements for scalable network architectures. Please refer to this article for detailed information on IPv6.

Since the migration to IPv6 will not happen over night, we expect to go through a transition period that might last several years during which IPv6 and IPv4 networks will coexist, and communication should be possible across the boundary of the coexisting networks. V6ops, IPv6 Operation working group of IETF, which took the responsibility of the former working group NGTrans (Next Generation Transition), has been working to resolve the problems that may arise during the transition period. The address format and IP header format are different in IPv4 and IPv6. Therefore, to make communication possible between two types of networks there must be a mechanism in between them. The working group already proposed different transition mechanisms. The main transition mechanisms are Dual Stack, Tunnelling and NAT-PT.

However, these mechanisms are not alternative to each other; each one is suitable in a different environment. By implementing Dual Stack, a node will be able to understand both protocols. Tunnelling will allow making a link between two IPv6 networks via IPv4 networks and vice-versa. To establish communication directly between IPv4 and the IPv6 network, we have to use either Dual Stack or NAT-PT. NAT-PT is used when we have IPv6-only and IPv4-only networks that must communicate with each other.

The NAT-PT Concept

The term NAT-PT stands for Network Address Translation and Protocol Translation. NAT refers to translation of an IPv4 address into an IPv6 address and vice-versa and PT stands for the translation of the IPv4 packet into a semantically equivalent IPv6 packet and vice-versa. NAT-PT allows native IPv6 hosts and applications to communicate with native IPv4 hosts and applications, and vice-versa. A NAT-PT device resides at the boundary between an IPv6 and IPv4 network. It uses a pool of IPv4 addresses for assigning to IPv6 nodes dynamically, and this assignment is finished when sessions are initiated across IPv4-IPv6 boundaries.

NAT-PT allows establishing communication between IPv4 and IPv6 networks. Figure 1 illustrates a simple scenario of the use of NAT-PT. Such a typical environment includes an IPv6 network, an IPv4 network and a border router, which is essentially a dual stack NAT-PT server. The environment also requires a DNS server for each network. In the IPv6 network, all hosts must have IPv6 stack active only (No IPv4 addresses are provided for these hosts). In the IPv4 network, all hosts must have the IPv4 stack active only.

Figure 1: A typical NAP-PT environment

The steps required to create a NAT-PT environment are the following:

a. Setting up and configuring the IPv4 and IPv6 networks

b. Setting up and configuring the IPv4-only and IPv6-only DNS servers

c. Setting up and configuring the Border Router (NAT-PT server)

For the purpose of this article, we assume that steps (a) and (b) are already complete as you already have an IPv4 network with its own IPv4 DNS server and an IPv6 network with its own IPv6 DNS server. We will be focusing on step (c): building a NAT-PT Linux server. Nevertheless, we will explain very briefly and give our readers some pointers into building an IPv6 network and setting up an IPv6 DNS server.

Creating an IPv6 Network

An IPv6 network in its simplest form is a number of hosts that are connected together and use the IPv6 protocol to communicate between each other. As part of the typical IPv6 network, we need to provide a DNS server that supports IPv6 and provides IPv6 routing services. To learn how to setup an IPv6 DNS server, please follow the technical tutorial presented at the following link: http://www.linux.ericsson.ca/ipv6/v6dns.html.

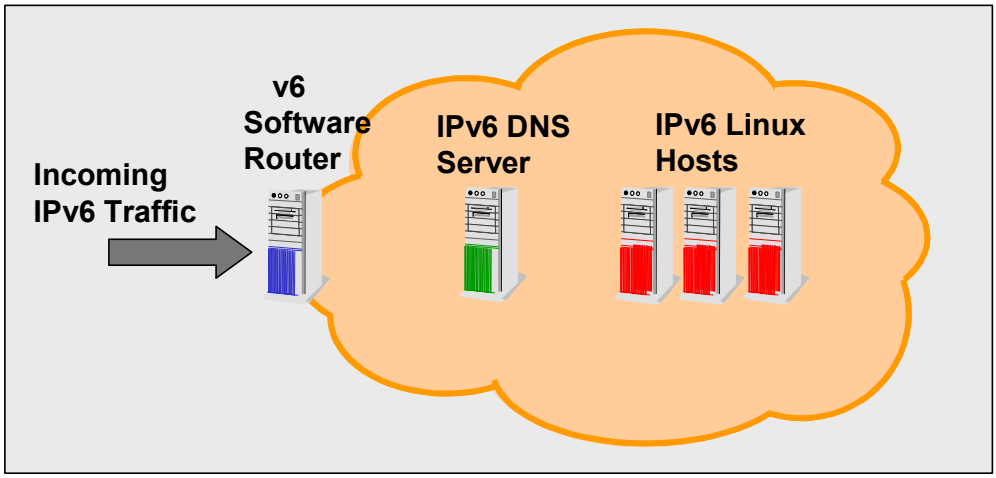

Figure 2: A simple IPv6 network that consists of an IPv6 router, IPv6 DNS server and IPv6 hosts.

As far as configuring the hosts, we need to recompile the kernel on these machines to support IPv6. The best approach is to rebuild the latest kernel available from the official site of the Linux kernel with IPv6 support. This was the subject of a previous article in LUD titled *"IPv6 on Linux: A Tutorial Approach"*. The article was published in the June 2003 issue and covers how to enable IPv6 on Linux.

The NAT-PT Mechanism

In this section, we present the basic working on the NAT-PT mechanism. It is important to know that IPv4 support will always be present on the nodes that will support IPv6. However, we can switch it off by setting up a special address on the Ethernet device as such:

```
% ifconfig eth0 0.0.0.0
```

After that, IPv4 connections on that interface will no longer be possible and only IPv6 will be activated.

Figure 3 illustrates the essential concept on NAT-PT:

- Node IPv6-A is located on the IPv6 network with an IPv6 address of `FEDC:BA98::7654:3210`.
- Node IPv6-B is located on the IPv6 network with an IPv6 address of `FEDC:BA98::7654:3211`.

- Node IPv4-C is located on the IPv4 network with an IPv4 address of `132.146.243.30.`

The NAT-PT Linux server has two Ethernet ports, each connected to a different network. This server has a pool of IPv4 addresses including the IPv4 subnet `120.130.26/24.`

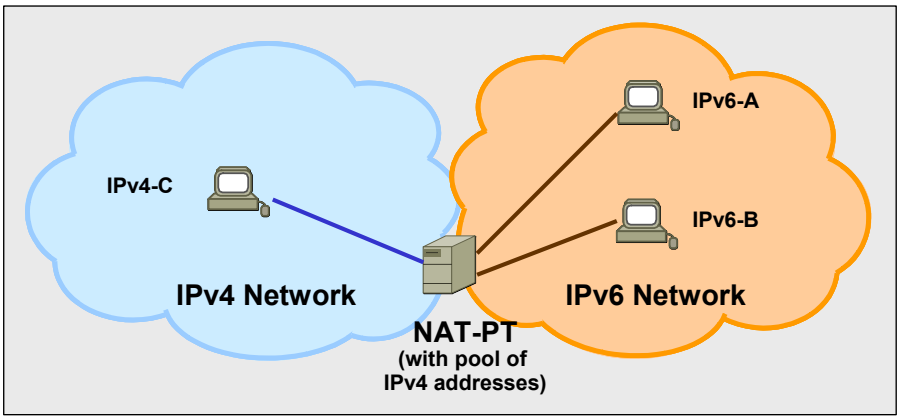

Figure 3: IPv4/IPv6 and IPv6/IPv6 communication

Below we will illustrate an example of a session starting from the IPv6 side. We will demonstrate a sample example of the IPv6-A node that wants to communicate with the IPv4-C Node. Node IPv6-A creates a packet with the Source Address, SA=FEDC::BA98:7654:3210 and destination address, DA=PREFIX:132.146.243.30 The PREFIX is static and any packet originating from an IPv6 node destined to the IPv4 network will contain that PREFIX as a part of the IPv6 destination address.

NAT-PT will translate the IP header including the source and destination address. After translation, the source address will be one pool address (say 120.130.26.1) and the destination address is 132.146.243.30. The NAT-PT will retain the mapping between 120.130.26.1 and FEDC:BA98::7654:3210 until the end of the session. In case of a reverse trip, the source address will be 132.146.243.30 and the destination address will be 120.130.26.1 and NAT-PT will change the source address to PREFIX:132.146.243.30 and destination address to FEDC:BA98:7654:3210, and the communication will continue.

IPv6 network needs to be pre-configured so that all packets containing that PREFIX in the destination address must be directed to the NAT-PT gateway, where it is translated to the IPv4 address. We can do it in the following way:

```
% route -A inet6 add <ipv6network>/<prefixlength> gw  $addressgw [dev $devicegw]
```

The `$addressgw` directive is the address of the gateway. You also need to specify a device if the IPv6 address of the gateway is a link local address.

Following our setup, we used the following command:

```
% route -A inet6 add 2000::/3 gw 3ffe:ffff:1234:5678::1 dev eth0
```

Alternatively, we can make the border router to be the default router of IPv6 network.

NAT-PT Border Router Setup

This router machine is dual stack, supports both IPv4 and IPv6, and must have two Ethernet interfaces. An IPv4 address will be assigned to one interface in the IPv4 network and an IPv6 address will be assigned to the second interface connected to the IPv6 network.

To configure the machine, we need to activate support for the required IPtables and netfilter, IPv4 and IPv6 packet forwarding in the Linux kernel. While configuring the kernel options on this machine, you need to enable support for the IPtables and netfilter. You do so, from the kernel configuration's main menu by following the `Networking Options` menu, then activating the `Network packet filtering (replaces ipchains)`. You can select this option as a loadable module.

To complete the kernel configuration, from the main kernel configuration menu, go to the `Networking Options` menu and follow the `IP: Netfilter Configuration` menu. Then you need to enable the IPtables support and the IPtables subsystems. For routing support from the kernel you need to enable the `IP: Advanced router` from the `TCP/IP networking in the Networking Options` menu. You must enable all these options as built-in kernel features.

Once you finish configuring the kernel, you can compile it and reboot the machine with the new kernel that has all the enabled features needed to support the NAT-PT functionalities.

Installing IPtables

Based on our experience with the NAT-PT implementation from the Korean Electronics and Telecommunications Research Institute, we found one drawback that is important to mention. Each packet directed to the NAT-PT is also grabbed by the kernel IP stack and that packet is processed by both the NAT-PT and kernel IP stack simultaneously. As the IPv4 pool addresses and the IPv6 address created with static PREFIX are not the addresses from any network, the kernel IP stack cannot forward to the destination address of the packet. As a result, kernel IP stack produces an ICMP error message, and we do not see the expected results. However, we can avoid this phenomenon by adding the IPtables rules.

To install IPtables on the machine that will be acting as the NAT-PT server, we recommend following the installation instructions outlined in the Netfilter/IPtables' home page at http://www.netfilter.org/.

Turning the NAT-PT Server into an IPv6 Router

Now we are ready to support IPv6 routing on the NAT-PT machine. The following steps describe how to configure it as a router for IPv6 packets:

1) We need to update `/etc/sysconfig/network` to enable IPv6 support. The file should look as follows:

```
NETWORKING=yes
HOSTNAME=<The name of the node or FQDN>
FORWARD_IPV4=yes
NETWORKING_IPV6=yes
IPV6FORWARDING=yes
IPV6AUTOCONF=no
```

2) `/etc/sysconfig/network-networkscripts/ifcfg-eth0` file should contain the following entries:

```
DEVICE=eth0
ONBOOT=yes
IPV6INIT = yes
IPV6ADDR=<IPv6 Address>
GATEWAY=<IP Address of the gateway>
```

With IPv6, at boot time, every node in an IPv6 network will get a prefix from the router advertisement daemon. So if `IPV6FORWARDING=yes` is set on a node then that node itself will act as a router, and in that case it will discard the prefix advertised by any router. This means, a router must have a static IPv6 address. We can assign a static IPv6 address to an interface using the following command:

```
% ifconfig <interface> inet6 add <ipv6address>/<PrefixLength>
```

In our setup we used:

```
% ifconfig  eth0 inet6 add fec0::1/64
```

Alternatively, you can add an entry such as `IPV6ADDR = <IPv6_Address>` (for example `IPV6ADDR is 3ffe:0302:0011:0002:250:b7ff:fe14:35d0`) in `/etc/sysconfig/network-scripts/ifcfg-eth0` to assign a static IPv6 address to the eth0 interface.

Turning the NAT-PT Server into an IPv4 Router

The steps to enable routing for IPv4 packets on the NAT-PT server are the following:

1) Enable forwarding for IPv4 packets:

```
% echo "1" >/proc/sys/net/ipv4/ip_forward
% /etc/sysconfig/network-scripts/ifup-routes <device_name>
```

Where `<device_name>` is either `eth0` or `eth1` depending on your own setup. Alternatively, you can add the following lines into `/etc/sysctl.conf`:

```
# Enable packet forwarding
net.ipv4.ip_forward = 1
```

2) After that, we need to restart the network services:

```
% service network restart
```

All the packets originating in an IPv4 network for any host in an IPv6 network must pass through the border router. This border router will act as a NAT-PT box, where the address and protocol will be translated.

The NAT-PT server requires having a pool of IPv4 addresses so that it can create a mapping with the IPv6 addresses. In a previous section we mentioned that one interface of the border must be connected to IPv4 network and have an IPv4 address assigned to it.

The pool address may be from the same subnet of the IPv4 side of the NAT-PT box or from a different network. However based on our experience in our test environment, we found that if the pool of addresses is from a different network, it is less problematic than if the pool of addresses is allocated from the same network. Nevertheless, both cases are possible.

Any packet originating in IPv4 network and destined to an IPv6 network will contain one of the IPv4 pool addresses as a destination address. The IPv4 host will assume that it is actually sending the packet to an IPv4 network, which is really an IPv4 network of the border router; therefore, the packet must pass through the border router. If we do not have any IPv4 router other than the border router, we can send the packet directly to the border router by changing the routing table of IPv4 host as follows:

```
%route add –net <network_address> netmask <netmask_address> gw
<border_router>
```

Installing the NAT-PT Software

You can download the NAT-PT source code package from the Korean Electronics and Telecommunications Research Institute web site. Please note that the implementation is still experimental and not yet ready for deployment in large commercial environments.

To install the NAT-PT software, please follow these instructions:

1) Create a directory in /usr/src to download the source package into.

```
% mkdir /usr/src/natpt
```

2) Decompress the source package:

```
% cd /usr/src/natpt

% tar –xzf linux-usermode-natpt-src.tar.gz
```

Once done, the directory will contain the following files:

- ***Active_ALG.list:*** It contains the list of application and corresponding port numbers for which application we need application level gateway (ALG). This source only supports dns_alg and ftp_alg. To add more application level gateways, we need to add the name of the application with the proper port number.

- ***address_pool.c:*** This module provides the IPv6/IPv4 address mapping facility for the NAT-PT software. IPv4 addresses are allocated from a pool address. Address mappings are stored for each session, or until the time_to_live (TTL) parameter expires.

- ***alg_manager.c:*** This module determines whether there is any ALG to handle a packet received by NAT-PT or not, if so, it invokes the appropriate ALG for translating address information embedded within a packet payload. The ALG manager also includes several functions, which are common to all ALG's.

- ***dns_alg.c:*** This module is the DNS Application Level Gateway, and is responsible for the translation of payload data within the packets that comprise of DNS messages.

- ***ftp_alg.c:*** This module is the ftp Application Level Gateway, and is responsible for the translation of payload data within packets that comprise of ftp messages.

- ***IPv4_Addresses.list:*** contains the pool address, and static mapping between a pool address and IP address of the IPv6 DNS server. However, if we like to use the IP address directly instead of the FQDN, then instead of the IPv6 DNS address, we can use the IP address IPv6 http/ftp/telnet (or any) server. Therefore, we have to edit this file to write our pool addresses and mapping in this file. The IPv4_Address.list file contains an example describing the format of the file.

- ***nat-pt.c:*** This is the main program for implementing NAT-PT. It uses two pointers to the interface names of the NAT-PT box. These are as *pIFnameIP4 and *pIFnameIP6. We needed to edit this file so that the first one points to the actual IPv4 interface and the second one points to the actual IPv6 interface. It uses the Berkeley Packet Filter raw interface to the data link layers, which exists as character special devices, /dev/bpf0, /dev/bpf1, etc.

- ***nat-pt_global.h:*** It contains the IPv6 PREFIX in the format of #define IPv6_PREFIX_HOST "aaaa:bbbb:cccc:dddd:eeee:ffff" . We need to change it to include our IPv6 PREFIX.

- ***protocol_translator.c:*** This module receives packets crossing the IPv4/IPv6 boundary, and converts IPv6 format packets into IPv4, and IPv4 into IPv6. Then it passes packets to the ALG Manager to check if further processing is required by any ALG.

- ***utils.c:*** This module contains some utility functions such as checksums, diagnostic print etc. that are generally available to NAT-PT modules. However, this is not a fully developed module and the diagnostic print function does not work.

To compile the source code, you need to execute the `make` command:

```
% make
```

Then, you can start the nat-pt daemon:

```
% ./nat-pt
```

Resulting Test Environment

After following the steps illustrated, you should be able to build a test environment similar to the one we achieved which is illustrated in Figure 4, and it shows the message flow per type of message. The IPv4 network consists of GAMMAY and the IPv4 DNS server is called MACON. The IPv6 network consists of MUSCADET and the IPv6 DNS server is called SAUTERNES. All the machines were configured with globally routable IP addresses. The nodes GAMMAY, MACON, and one interface of the NAT-PT server were connected to the IPv4 network; the remaining nodes including the second interface of the NAT-PT server were connected to the IPv6 network.

Figure 4: Test environment for NAT-PT

The IPv4 pool addresses should be from a different subnet than the IPv4 side of NAT-PT.

Case Scenario A: Packet Incoming from IPv4 Network

- Check TTL value. If it is zero, then drop the packet or else get the destination address of the packet.

- Check the mapping list. If the destination address is in the mapping list then get the corresponding IPv6 address from the list and translate the IP header of the packet.

- Check the source or destination port of the packet to find if there is any ALG to handle it or not. If there is, then call corresponding ALG to handle the packet.

- Write the packet to the destination network.

Case Scenario B: Packet Incoming from the IPv6 Network

- Check the TTL value. If it is zero, then drop the packet or else get the source address of the packet, and check the mapping list.

- If the source address is in the mapping list, get the corresponding IPv4 address from the list. Otherwise, create a new mapping between a pool address and the source address of the packet, and get the mapped IPv4 address.

- Translate the IP header of the packet.

- Check the source or destination port number of the packet to find if there is an ALG to handle the packet or not. If ALG exists for this packet, call the corresponding ALG to handle the packet.

- Write the packet to the destination network.

Conclusion

In this article, we presented a tutorial on how to install and configure a NAT-PT server to allow us to provide a seamless IPv4/IPv6 and IPv6/IPv4 address translation between different nodes residing on different networks.

NAT-PT is a very useful mechanism and it gives you the flexibility of maintaining your IPv4 environment untouched, while experimenting interoperability with your testing IPv6 environments. However, there are some limitations using the NAT-PT translation method. It is mandatory that all requests and responses pertaining to a session be routed via the same NAT-PT router. Since NAT-PT performs address translation, applications that carry the IP address in the higher layers will not work. In this case, the ALG needs to be incorporated in order to provide support for these applications. Another limitation with NAT-PT is the lack of end-to-end security. Transport and application layer security may not be possible for applications that carry IP addresses to the application layer. This is an inherent limitation of the Network Address Translation function. Other limitations are related to the DNS translation and DNSSEC. An IPv4 end-node that demands DNS replies to be signed will reject replies that have been tampered with by NAT-PT.

The current NAT-PT Linux implementation may not be ready for prime time deployment however, it is a serious step forward and we highly recommend testing it and contributing to the project.

References

IETF Home Page	http://www.ieft.org
IETF IPv6 RFCs	http://interop.ipv6.org.tw/IETFIPv6RFC.htm
IETF NGTrans Working Group	http://www.ietf.org/html.charters/ngtrans-charter.html
DNS 9.x How-To	http://langfeldt.net/DNS-HOWTO/BIND-9/
"IPv6 on Linux: A Tutorial Approach"	Linux User & Developer – June 2003
DNSSEC Information Home Page	http://www.dnssec.net/
NAT-PT Linux Implementation	http://www.ipv6.or.kr/english/download.htm
NAT-PT RFC 2766	http://www.ietf.org/rfc/rfc2766.txt
Linux IPv6 How-To	http://www.bieringer.de/linux/IPv6
NetFilter	http://www.netfilter.org
Linux Kernel	http://www.kernel.org
Open System Lab – IPv6 Page	http://www.linux.ericsson.ca/ipv6