Asian Journal of Information Technology 6 (5): 533-537, 2007

IPv6 with DNS Server

[1]G. Kavitha and [2]R.S.D. Wahidabanu
[1]Department of Computer Science and Engineering,
Mahendra Engineering College, Mahendrapuri, India
[2]Department of Computer Science and Engineering,
Government College of Engineering, Salem, India

Abstract: The research is aimed to develop a Domain Name System (DNS) server, which supports Internet Protocol version 6 (IPv6) name resolution. By configuring the server, which has provision to support both Internet Protocol version 4 (IPv4) and IPv6 in hosts and routers, the aim of the research is achieved. The method of encapsulation of IPv6 packets within IPv4 headers over IPv4 network is called as IPv6 over Ipv4. With an objective to setup own Linux IPv6 DNS server to allow IPv6 name resolution using the latest version of bindconf (which is used to add new domains and new records for each domain pretty easily) tool and to set a server that supports both IPv4 and IPv6 name resolution, this research has been carried out. By creating kernel with appropriate network parameters using shell scripts, the above objective is acomplished.

Key words: IPV6, IPV4, DNS, server, resolution, kernel

INTRODUCTION

The research mainly deals with creating a dual IP stack node with provision of complete support for both IPv4 and IPv6 in hosts and routers. This can be achieved by making a recursive copy of the kernel and configuring the necessary network properties to support both IPv4 and IPv6. The kernel is configured using scripts in shell scripts and C programming.

The main idea of the research is to configure a DNS server in Linux with IPv6 name resolution facility. The concept of IPv4 tunneling is used to encapsulate the Ipv6 packets within IPv4 headers to carry through an IPv4 network.

The need for a new internet protocol is well understood and accepted in the network industry. Requirements for more address space, simple address design and handling of addresses at the IP layer, better QoS support, greater security, increasing number of media types and internet-capable devices have all contributed to drive the development of Internet Protocol version 6 (IPv6). This study reviews the basics of IPv6, its deployment and strategies for managing the transition from IPv4 to IPv6.

PROBLEM DEFINITION

As described in the introduction section of the synopsis there exists an inherent problem in using the domain name address or IPv4 and IPv6 name resolution. This research resolves these issues of increasing the address space.

Thus the problem for the present research has emerged as creating a dual IP stack node with provision of complete support for both IPv4 and IPv6 in hosts and routers.

Works of various authors like Lee and Stewart, Paul Tipper and books like Binh (2004), Christopher (2003), Danny (2002), Matchtelt (2002-2005), Narthan (2000), Lee (1997), Richard (2001), Tigran Aivazian (2002), Yehuda and Tomer (1998).

PROPOSED SOLUTION

The proposed solution is aimed at removing the drawbacks of the existing system. Dual stack server which is employed can easily help a user to give Ipv6 support to the linux system configuration and tunneling were established by the researcher through codes written in JAVA and also transfer of packets from client for the server through same coding.

Solution for the problem is presented in this study. Implemented solution consists of following 5 modules.

- Kernel compilation
- Member adder
- Configuring IPv6 inside IPv4
- Configuring the kernel
- Resource access

Corresponding Author: G. Kavitha, Department of Computer Science and Engineering, Mahendra Engineering College, Mahendrapuri, India

About kernel: The kernel is the part of the operating system that handles the most basic functions and control interactions with the computer hardware. The Linux kernel is very modulating. Each driver, for a file system or a piece of hardware, needs to be compiled into the kernel, or inserted as a module. The file/proc/file systems contain a list of all the file systems that the kernel understands.

There are two ways to get IPv6 support. The simplest is to probe the IPv6 module to the kernel (as root). The second way is to recompile the kernel, you. This sounds like a lot of work, but isn't that hard.

Kernels, being the backbone of an operating system are of two types: Open kernel and closed kernel. The open kernel is a free source and easily modifiable by any one familiar with Linux. The closed kernel is a kernel of proprietary software that cannot be modified by normal user.

The open kernel is again of two kinds stable kernel (2.4.20.8), if the third octet is even it is stable kernel and development kernel (2.4.19.8) if the third octet is odd it is development kernel. The kernel has release, version and volume specifications each separated by a period. If the version of the kernel is even then the kernel is a stable kernel and can be used for implementation purposes. If the version is odd then the kernel is a development kernel and it is not advisable for implementation purposes.

The kernel versions prior to 2.4 did not have provisions for IPv6, so a module has to be added by recompiling the kernel. But the current versions from 2.4 have provisions to support IPv6 and are IPv6 ready kernels. Appropriate rpms have to be enabled in the kernel. Most users won't have to compile anything to enable Ipv6 support. Almost every Linux distribution comes with IPv6 support out of the box. For any RedHat 7.3 or RedHat 8.0 user, the user probably doesn't have to recompile the kernel. If there is an older version of RedHat, or other distribution that doesn't include IPV6 support, then the user has to recompile the kernel.

The modules from the enhanced version cannot be loaded into the prior versions, because the facilities in the prior versions are restricted and the modules attached to the degraded version will not cope up with each other. So for a version that has no support for IPv6 a patch up rpm corresponding to the kernel version has to be downloaded from the net and run in the root using the command rpm-ivh<rpmname>. Thus, making the kernel to support IPv6.

IPv4: IPv4 is the Internet protocol version 4, the protocol in current usage. This can be simply called as IP. IP is the transmission mechanism used by the TCP/IP protocols. It is an unreliable and connectionless datagram protocol-a best effort delivery service. The term best effort means that IP provides no error checking or tracking. IP assumes the unreliability of the underlying layers and does its best to get a transmission through to its destination, but with no guarantees. The transmission along a physical network can be destroyed by a number of reasons like bit errors due to noise, discarding of a datagram because of congestion in routers etc.

If reliability is important, IP must be paired with a reliable protocol such as TCP. IP transports data in packets called datagram, each of which is transported separately. Datagram may travel along different routers and may arrive out of sequence or duplicated. IP does not keep track of the routers and may arrive out of sequence or duplicated. IP does not keep track of the routers and has no facility for reordering datagram once they arrive. Because it is a connectionless service, IP does not create virtual circuits for delivery. There is no call setup to alert the receiver to an incoming transmission.

The limited functionality of IP should not be considered a weakness; however IP provides bare-bone transmission functions that free the user to add only those facilities necessary for a given application and thereby allows for maximum efficiency.

IP addresses: The IP networking protocol understands addresses as 32-bit numbers. Each machine must be assigned a number unique to the networking environment. If you are running a local network that does not have TCP/IP traffic with other networks, you may assign these numbers according to your personal preferences. There are some IP address ranges that have been reserved for such private networks. These ranges are listed in Table 1 However, for sites on the Internet, numbers are assigned by a central authority, the Network Information Center (NIC). IP addresses are split up into four; eight-bit numbers called octets for readability. For example, quark.physics.groucho.edu has an IP address of 0×954C0C04, which is written as 149.76.12.4. This format is often referred to as dotted quad notation.

Another reason for this notation is that IP addresses are split into a network number, which is contained in the leading octets and a host number, which is the remainder. When applying to the NIC for IP addresses, you are not assigned an address for each single host you plan to use. Instead, you are given a network number and allowed to assign all valid IP addresses within this range to hosts on your network according to your preferences.

The size of the host part depends on the size of the network. To accommodate different needs, several

Table 1: IP address ranges reserved for private use

Class	Networks
A	10.0.0.0 through 0.255.255.255
B	172.16.0.0 through 172.31.0.0
C	192.168.0.0through192.168.255.0

classes of networks, defining different places to split IP addresses, have been defined. The class networks are described here:

Class A: Class A comprises networks 1.0.0.0 through 127.0.0.0. The network number is contained in the first octet. This class provides for a 24-bit host part, allowing roughly 1.6 million hosts per network.

Class B: Class B contains networks 128.0.0.0 through 191.255.0.0; the network number is in the first two octets. This class allows for 16,320 nets with 65,024 hosts each.

Class C: Class C networks range from 192.0.0.0 through 223.255.255.0, with the network number contained in the first three octets. This class allows for nearly 2 million networks limited 254 hosts.

Classes D, E and F: Addresses falling into the range of 224.0.0.0 through 254.0.0.0 are either experimental or are reserved for special purpose use and do not specify any network. IP Multicast, is a service that allows material to be transmitted to many points on an Internet at a time, to which been assigned addresses have been assigned from within this range.

For example we find that 149.76.12.4, the address of quark, refers to host 12.4 on the class B network 149.76.0.0.

You may have noticed that not all possible values in the previous list were allowed for each octet in the host part. This is because octets 0 and 255 are reserved for special purposes. An address where all host part bits are 0 refers to the network and an address where all bits of the host part are 1 is called a broadcast address.

This refers to all hosts on the specified network simultaneously. Thus, 149.76.255.255 is not a valid host address, but refers to all hosts on network 149.76.0.0.

A number of network addresses are reserved for special purposes. 0.0.0.0 and 127.0.0.0 are two such addresses. The first is called the default route and the latter is the loop back address. The default route has to do with the way the IP routes datagram.

Network 127.0.0.0 is reserved for IP traffic local to your host. Usually, address 127.0.0.1 will be assigned to a special interface on your host, the loop back interface, which acts like a closed circuit. Any IP packet handed to this interface from TCP or UDP will be returned to them as if it had just arrived from some network. This allows you to develop and test networking software without ever using a real network. The loop back network also allows you to use networking software on a standalone host. This may not be as uncommon as it sounds; for instance, many UUCP sites don't have IP connectivity at all, but still want to run the INN news system. For proper operation on Linux, INN requires the loop back interface. Some address ranges from each of the network classes have been set aside and designated reserved or private address ranges. These addresses are reserved for use by private networks and are not routed on the Internet. They are commonly used by organizations building their own intranet, but even small networks often find them useful. The reserved network addresses appear in Table 1.

DNS OPERATION IN IPv4

The DNS configuration in Linux can be done through a graphical tool called Berkeley Internet Name Domain (BIND) software.
The basic components of BIND include the following:

DNS server daemon (usr/sbin/named): The named daemon listens on a support for DNS service request and then fulfills those request based on information in the configuration files that you create. Mostly, named receives requests to resolve the host names in your domain to IP addresses.

DNS configuration files (named.conf and /var/named/*): The/etc/named.conf file is where you add most of the general configuration information that you need to define the DNS services for your domain. Separate files in the /var/named directory contain specific zone information.

DNS lookup tools: You can use several tools to check that your DNS server is resolving host names properly. These include commands such as host, dig and nslookup (which are part of the build-utils software package).

To maintain your DNS server correctly, perform the following configuration tasks.

Logging: You can indicate what you want to log and where log files reside.

Remote server options: You can set options for specific DNS servers to perform such tasks as blocking information from a bad server, setting encryption keys to use with a server, or defining transfer methods.

IPv6: IP version 6 is new IP protocol designed to replace IP version 4, the Internet protocol that is predominantly deployed and extensively used throughout the world.

RATIONABLE FOR NEW VERSION OF IP

IPv4 has proven to be robust, easily implementable and interoperable and has stood the test of scaling an

Internet work to a goal utility the size of the Internet today. However, the initial design did not anticipate the following conditions:

- Exponential growth of the Internet and the impending exhaustion of the IPv4 address space.
- Growth of the Internet and the ability of Internet backbone routers, to maintain large routing Tables.
- Need for simpler auto configuration and renumbering.
- Security at the IP level.
- Need for better support for real-time delivery of data-also called as Quality of Service (QoS).

The Internet Engineering Task Force (IETF) first recognized the problem of eventual IPV4 address around 1990 and predicted that we had about 10 years solving this problem. Interestingly, this prediction was made before the explosive growth of the Internet World Wide Web. The current IP address space is unable to satisfy the potential huge increase in the number of users or the geographical needs of the Internet expansion. IPv6 is designed to meet these requirements and allow a return to a global environment applications.

The lifetime of IPv4 is extended use of techniques such as address reuse with Network Address Translation (NAT), Classless Inter Domain Routing (CIDR) and temporary-use allocations (Dynamic Host Configuration Protocol). Although, these techniques appear to increase the address space and satisfy the traditional server/client setup, they fail to meet the requirements of the peer = to = peer and server- client applications.

USAGE OF IPv6 FEATURES AND BENEFITS

In addition to meeting the anticipated future demand for globally unique IP addresses. IPv6 provides the following benefits to network and IT professionals:

- Larger address space, for global reachability and scalability
- Simplified header format for efficient packet handling
- Hierarchical network architecture for routing efficiency
- Support for widely deployed routing protocols
- Auto configuration and plug and play support
- Elimination of need for network address translation and application's layered gateway
- Embedded security with mandatory IPSec implementation
- Enhanced support for mobile IP and Mobile Computing Devices and
- Increased number of Multicast address

OPERATION OF IPv6

The IPv6 neighbour discovery protocol and the Internet Message Control Protocol (IMCP) are critical to operation of IPv6.
The following are the major operations of IPv6:

- Neighbour discovery
- Router discovery
- Stateless auto configuration
- Path Maximum Transfer Unit (MTU) DISCOVERY
- Dynamic Host Configuration Protocol version 6 (DHCPv6)
- Domain Name System (DNS) op

Let us see Domain Name Server (DNS) operation, which plays a major part in this research in detail.

IPv6 DOMAIN NAME SYSTEM OPERATION

IPv6 introduces new DNS record types for IPv6 addresses that are supported in the DNS name-to-address and address-to-name lookup processes. DNS query is possible over an IPv4 transport or an IPv6 transport. But, DNS root servers are not yet reachable through an IPv6 transport. The record types are as follows:

AAAA record: Also known as a quad A record, maps a host name to an IPv6 address. This record is equivalent to an A record in IPv4 and uses the format: WWW.abc.test AAAA 3FFE:B00:C18: 1:2. The Internet Engineering Task Force (IETE) has decided to use this record for host name-to-IF address resolution.

A6 record: Set to Experiment by the IETF, the A6 record will not be used in production networks. Equivalent to an AAAA record, but enables the storing of IPv6 addresses in a hierarchical manner to simplify network renumbering. A6 record uses the format: www.abc.tes A6 0 3FFE:B00:C18: 1:2.

PTR record: Equivalent to a pointer (PTR) record that specifies address-to-host name mappings in IPv4. Inverse mapping used in IP address-to-host name look-up uses the PTR record. The top-level domain for the IPv6 addresses is ip6.arpa. Initially, for IPv6 DNS, the top-level domain was ip6.int. The PTR record uses the format: 2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.8.1.c.0.0.b.0.e.f.f.3.ip6.arpa PTR www.abc.test.

DNAME and binary labels records: These new records make renumbering easy for inverse mapping (IP address to host name).

Using AAAA records for DNS resolution: When a node needs the IPv6 address of another node (www.abc.test), it sends an AAAA request for www.abc.test to the DNS. The authoritative DNS for the abc.test domain responds with the IPv6 address requested, for example, 3FFE:0B00:0C! 8:0001:0290:27FF:FE17: FCID in this case. The AAAA record stores a single IPv6 address. A node with more than one address must have more than one record in the DNS database.

If DNS is used for renumbering a network, all nodes must change the prefix part of their IPv6 address. For nodes included in the DNS, all these new addresses must be changed in the DNS database.

INTEGRATION AND COEXISTENCE STRATEGIES

The IETF IPv6 working group has designed several strategies for the deployment of IPv6. The following transition strategies are covered in this chapter deploying IPV6 over:

- Dual stack backbones
- IPv4 tunnels
- Dedicated data links
- Multiprotocol Label Switching (MPLS) backbones
- Using protocol translation mechanisms

TRANSITION MECHANISMS

Network designers recommend deploying IPv6 at the edge first and then moving towards the network core to reduce the cost and operational impacts of the integration. The key strategies used in deploying IPv6 at the edge of a network involve carrying IPv6 traffic over the IPv4 network, allowing isolated IPv6 commands to communicate with each other before the full transition to a native IPv6 back bone. It is also possible to run IPv4 and IPv6 throughout the network, from all edges through the core, or to translate between IPv4 and IPv6 to allow hosts communicating in one protocol to communicate transparently with hosts running the other protocol. All techniques allow networks to be upgraded and IPv6 deployed incrementally with little to no disruption of IPv4 services. The 4 key strategies for deploying IPv6 are as follows:

Deploying IPv6 over dual-stack backbones: This technique allows IPv4 and IPv6 applications to coexist in a dual IP layer routing backbone. All routers or a portion of them (for example, access CPE routers and aggregation routers are dual stack, but core routers stay as they are) in the network need to be upgraded to be dual stack, with IPv4 communication using the IPv4 protocol stack and IPv6 communication using the IPv6 stack.

Deploying IPv6 over IPv4 tunnels: These tunnels encapsulate the IPv6 traffic within the IPv4 packets and are primarily for communication between isolated IPv6 sites or connection to remote IPv6 networks over an IPv4 backbone. The techniques include using manually configured tunnels, Generic Routing Encapsulation (GRE) tunnels, semiautomatic tunnel mechanisms such as tunnel broker services and fully automatic tunnel mechanisms such as 6 to 4 for the WAN and Intra-Site Automatic Tunnel Addressing Protocol (ISATAP).

Deploying IPv6 over dedicated data links: This technique enables IPv6 domains to communicate by using the same Layer2 infrastructure used for IPv4, but with IPv6 using separate Frame Relay or ATM Permanent Virtual Circuits (PVCs), separate optical links, or Dense Wave Division Multiplexing (DWDM).

Deploying IPv6 over MPLS backbones: This technique allows isolated IPv6 domains to communicate with each other, but over an MPLS IPv4 backbone without modifying the core infrastructure. Multiple techniques are available at different points in the network, but each requires little change to the backbone infrastructure or reconfiguration of the core routers because forwarding is based on labels rather than the IP header itself.

REFERENCES

Binh Ngayen, 2004. Linux Dictionary version 0.16.

Christopher Negus, 2003. RedHat Linux 8 Bible.

Danny Goodman, 2002. Java Script Bible, Wiley dreamtech India Pvt.Ltd (4th Edn).

Matchtelt Garrels, 2005. Introduction to Linux a hands on guide.

Narthan Meyers, 2000. JAVA Programming on Linux, Techmedia Publication.

Lee Purcell, 1997. Mary jene Mara, The ABCs of Java script, BPB Publications.

Richard Petersen, 2001. The complete Reference Linux.

Tigran Aivazian, 2002. Linux Kernel 2.4 internals.

Yehuda shiran and Tomer Shiran, 1998. Advanced Javascript programming, BPB Publications.