Volume 7, Issue 5, May 2017 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Software and Hardware Enhancements to Support IPV6

Pavithra R*
M.Tech, CNE, BMSCE,
Bangalore, India

M. Dakshayini
Professor, ISE, BMSCE,
Bangalore, India

Abstract— Internet Protocol Version 4 (IPV4) has been the standard Internet Protocol which provides the addressing scheme for the existing Internet. Internet has grown beyond reaches and has exceeded the scope which it was initially built for. With the advent of Modern technologies and Research in areas such as IoT, cloud computing, Big Data etc., Internet is not restricted to usage for email communication. Users of Internet have grown across the globe and the purpose of Internet has had a paradigm shift. End users are identified with the aid of IP addresses and the vast growth in the end user has led to exhaustion of these addresses. The problem of address exhaustion was identified in the early 1995 and a solution to this problem was provided by the invention of new addressing scheme called the Internet Protocol Version 6 (IPV6) formerly called as the IP next generation. The applications and tools which were developed to use and have in built dependency on the earlier IPV4 addressing scheme will have to be now scaled to include the new addressing scheme. This paper proposes solutions for making the applications compatible with IPV6 addressing scheme realizing the software dependencies of applications on IPV4.

Keywords— Internet Protocol Version 6, Internet Protocol Version 4, Sockets, Software Applications, Socket Programming.

I. INTRODUCTION

Internet has become the primary means of establishing communication between any two end points. In the recent days it is no longer being considered for communication alone and its usage has grown beyond bounds. The huge growth of internet has accounted for the IP address depletion. The IANA provided address space got exhausted as early as 2001 and the RIR's address space depleted by 2011 [1]. It was therefore not possible for IPV4 to meet the demands of the existing internet. One of the other disadvantages of IPV4 was that the network management with IPV4 was complex [2]. This problem of IP address exhaustion was answered with the Internet Protocol Address 6 or the IP next generation addressing scheme. The IPV6 has a 128 bit long address, thereby increasing the address space to several million. The address space in IPV6 has also been divided into various categories thereby addressing the complex network problems.

When the existing internet has to shift to the new addressing format there exists basically two fronts that needs to be addressed. One end would be the network, where the network components including the routers, switches, gateways need to accommodate the new addressing scheme. And the other front would be the application software's. Applications which were written earlier and show dependency on the IPV4 addressing scheme needs to be changed now in order to aid the IPV6 transition process.

At an Organization, since it owns a network and applications both, changes will have to be done at both the ends.

In this paper we are giving out the various options available to implement the IPV6 transition at both the problem ends. The hardware changes as well as the software changes that need to be implemented are proposed thereby easing the job of an application developer. The rest of the paper is put in the following format. The second section consists of the background. The third section gives the hardware implementation and the various approaches. The fourth section gives the software related changes and the fifth section gives the results and conclusion.

II. LITERATURE SURVEY

IPV6 will be effectively implemented only after the current applications and new applications make use of this technology along with the hardware support for them. The transitions of the two technologies are hindered by few things, namely the following. The two protocols are not directly compatible with one another [3]. A new infrastructure would be needed to support the new protocol. However since it is not ideal or realizable to change the infrastructure overnight both of these protocols needs to exists with one another. Also IPV4 is being used at present by using the concept of NAT. with the aid of this private IP addresses can connect to the internet with a single public IP address. There are considerable differences between IPV4 and IPV6.

- Address Bits: IPV4 address consists of 32 bits whereas the IPV6 address consists of 128 bits. This leads to an increase in the address space to a larger extent thereby fulfilling the intent of the protocol need.
- Routing efficiency: On IPV4 the data gets fragmented before it reaches the destination depending on the bandwidth and other constraints. However with the advent of IPV6 this would not be required.
- In-built QoS Support and Security: IPV6 has been designed to provide support for QoS. Security in IPV4 is not provided by the network and applications that use them had to implement it. However IPV6 has in built security.

- Easier Network Administration: IPV6 allows for automatic configuration there by easing the network administrators jobs.
- No need to NAT: NAT which was used for increasing the address space is no longer required in IPV6.
- Better Header Structure: the header structure of IPV6 is optimized to include only the necessary fields.
- VPN support: IPV6 provides authenticated header features and extended headers which aid in the creation of a virtual private network in an easy and efficient manner [4].
- Real-time Application support: IPV6 provides "labeled-flow", thereby providing feature similar to the MLPS. This aids in supporting real time applications.

The header formats also differs in many respects. The IP addresses are assigned based on DHCPV6 where the IP addresses are automatically assigned to the hosts [5]. Though IPV6 provides an advantage over the existing IPV4 addressing scheme it is not void of the security issues. Various loops in the addressing scheme need to be identified and addressed. We will look into the various issues identified and proposed solutions.

III. HARDWARE IMPLEMENTATION

In order for the existing network to support IPV6 various migration strategies have been proposed.

MIGRATION STRATEGIES

When the infrastructure is ready for the migration, both of the IPV4 and IPV6 should exist together to provide support for the transition. The various methodologies provided for the same include:

A. Translation Technique

The figure 1 depicts the translation technique and in this technique a translation of the two protocol headers is done. A translator mediates the translation between IPV4/IPV6 to IPV6/IPV4, which is very similar to the NAT technique. This would require very good, sustainable translators [6].

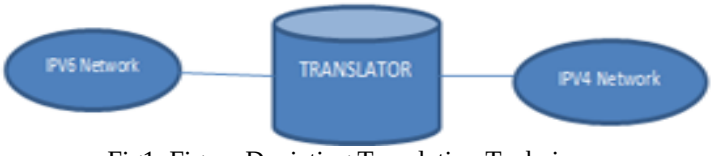

Fig1. Figure Depicting Translation Technique

B. Tunnelling

Figure 2 below depicts the tunneling technique. In this approach two IPV6 nodes get connected over an existing IPV4 network only. When packets get originate from the IPV6 network, they get encapsulated into an IPV4 header and gets transmitted across the network. At the receiver end they get removed from the IPV4 encapsulation and get delivered to the IPV6 host [6].

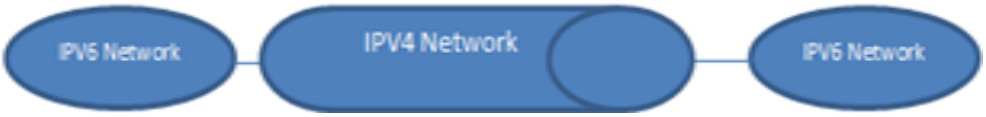

Fig2. Figure Depicting Tunnelling Technique

C. Dual Stack Technology

Figure 3 depicts the dual stack technology and because of its easy implementation, this method is one of the most widely used methodologies for supporting IPV6. In this approach both the protocols are made to co-exist in the same device and an application will invoke the appropriate protocol which it requires at the time of data transmission [8].

Fig3. Figure Depicting the Dual Stack Technique

In order to support IPV6 transition the existing hardware will have to be compatible to handle both the IP protocols. The computing environment we have used, uses the HPE 5500(1G) and 5700(10G) switches provided by the HP vendors. These switches are built with Gigabit Ethernet switches. They support large campuses and cities network and provide high resilience and security. These switches provide six 100GB uplinks of which two are fixed 10 GB ports for extension modules. The characteristic feature of the switch is:

- It supports tracertv6, telnetv6, pingv6, ARPv6, DNSv6 thereby abstracting the lower network protocol.
- It provides static routing support for IPv6 and also support IPV6 routing protocols such as the OSPFV3, BGP4+.

The other series of switch namely the HPE 5700 is a switch which provides low latency and high performance. It enhances the network performance and support virtual networks. It can support up to 40GB uplink speeds and has a fully featured dual stack support which provides the following:

- Different and separate stacks for each of the protocol (IPV4 and IPV6) are maintained.
- Provides static routing support for IPV6, which is simple and manually configured.
- The routing speeds provided are at media speeds and also supports RIPng.

Figure 4 given below shows the dual sack implementation with the HPE5500 and the HPE5700 switches.

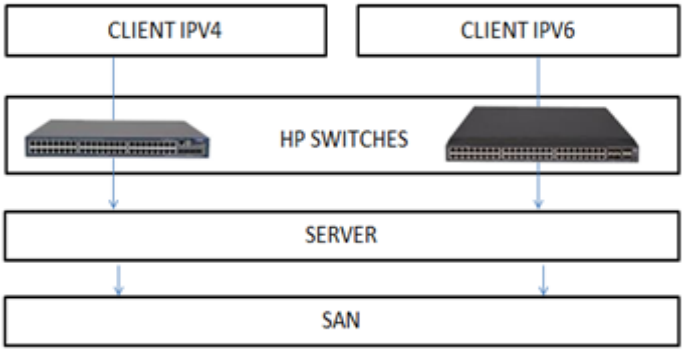

Fig4. Diagram depicting the switch implementation for hardware level enhancements.

The switches images used above are taken from the following source:
"https://www.google.co.in/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwib5IHsqrzTAhUJMY8KHSRvDCMQjRwIBw&url=https%3A%2F%2Fwww.hpe.com%2Fus%2Fen%2Fproduct-catalog%2Fnetworking%2Fnetworking-switches%2Fpip.fixed-port-l3-managed-ethernet-switches.4174795.html&psig=AFQjCNEHvml9OwAxyzidnw4LvhqIuWbtkA&ust=1493097251080719 "

IV. SOFTWARE IMPLEMENTATIONS

In TCP/IP the two communication ends are called as client and server and each of them have a distinct role to play. Client is a machine which initiates the connection and server waits for any incoming connection requests and accepts them once they arrive. In order for the client and server to communicate the client initiates a connection request on any port of the system and the server responds to this request. The Client server programming for C# includes the following commands and steps:

At the server end a TCP Listener is started on a local port specified. This is present in the System.Net.Sockets.TcpListener extension.

- TCPListener tcpListener= new TCPListener (port_num);
- tcpListener.Start ( );

The listener waits for incoming request and accepts incoming connection, in the form of System.Net.Sockets.Socket object.

- Socket s = tcpListener.AcceptSocket ();

A network streams needs to be created for the socket in the form of System.Net.Sockets.NetworkStreams.

- Stream newStream = new NetworkStream (s);

Now the communication with the client will proceed with the well-defined rules.

Once the communication completes, the stream gets closed.

- newStream.Close ();

And the socket gets closed.

- s.Close ();

In order for the server to serve multiple request at the same time, multiple threads needs to be invoked and each of the thread can then execute independently thereby allowing for multithreading.

A client program on the other hand would include the following steps:

Using the server host name and port number a TcpClient is created as shown below:

- TcpClient newClient = new TcpClient (hostname, port_num);

A network stream is obtained for the new client.

- Stream newStream = newClient.GetStream ();

The communication happens with the server using well-defined protocols ad rules.

Once the communication ends the stream is closed.

- newStream.Close ();

And the client ceases existence.

- newClient.Cose ();

A. Pseudocode for code changes to Support IPV6

In order for the application to support IPV6, the applications which use socket programming to connect the client to the Server will have to use the new address structure called the "sockaddr_in6" [10].


```
struct sockaddr_in6 {
        sa_family_t sin6_family;    /* AF_INET6 */
        in_port_t sin6_port;        /* Transport layer port # */
        uint32_t sin6_flowinfo;     /* IPv6 flow information */
        struct in6_addr sin6_addr;  /* IPv6 address */
        uint32_t sin6_scope_id;     /* IPv6 scope-id */
        };
```

A new family has been created to support for IPV6 called the "PF_INET6". For initializing a socket the following code will have to be used.

- socket(PF_INET6, SOCK_STREAM, 0); /* TCP socket */
- socket(PF_INET6, SOCK_DGRAM, 0); /* UDP socket */

A new connection will be accepted using the below function:

- struct sockaddr_in6 addr;
- socklen_t addrlen = sizeof(addr);
- accept(sockfd,(struct sockaddr *)&addr, &addrlen)

B. Inclusions at the Server

Our applications are built using the C# language and hence we are providing the changes with regarding to the C# language.

- //Define the headers
  - Using System.Net.Sockets;
  - Using System.Net
- //Create a socket listener
  - Socket sck_listenr = new Socket(AddressFamily.InterNetworkV6,
    SocketType.Stream,
    ProtocolType.Tcp);
- //Bind the listener to a port using sck_listner.bind and allow it to listen
  - sck_listner.Listent(0);
- //Accept a connection using
  - sck_listner. Accept();
- //Communicate with the client on designated port and after finishing the conversation close the connection.

C. Inclusions at Client

- //Define the headers
  - Using System.Net.Sockets;
  - Using System.Net
- //create a socket connection
  - Socket sock_connection = new Socket(AddressFamily.InterNetworkV6,
    SocketType.Stream,
    ProtocolType.Tcp);
- //connect using the connect system call
  - sock_connection.Connect(IPV6 address, port);
- //Communicate with the server
- //Close connection

V. RESULTS AND CONCLUSIONS

With the help of the support provided by the hardware and the software which have been proposed above, we have been able to make the existing applications to support IPV6 addresses as well. The hardware changes were implemented with the help of HP provided 5500 and 5700 switches as shown above in the figure 4. These switches support dual stack technology. When an IPV4 client invokes an IPV4 application it will be taken care by the IPV4 stack and when the IPV6client invokes connection with an IPV6 Application, it is taken care by the IPV6 stack. In this way both of the protocols are made to co-exist. At the software end new features provided by socket were implemented in our software to support IPV6. Once the changes were made we have tested thoroughly with the Local IP Unicast address designed for the organization and testing has been done successfully. A static routing approach was used to connect the client to the distant server and the applications were able to communicate successfully using IPV6 addresses as well.

ACKNOWLEDGMENT

I am grateful to my Professor M. Dakshayini for allowing me to carry on this work and helping me in all regards. The department at my college have been very supportive in providing any help beyond reaches and have helped me in successfully completing my research work.

REFERENCES

[1] Yu Zhai, Congxiao Bao, Xing Li, "Transition from IPv4 to IPv6: A Translation Approach", 2011 Sixth IEEE International Conference on Networking, Architecture, and Storage

[2] Kuobin Dai, "IPv4 to IPv6 Transition Research Based on the Campus Network", 2011 International Symposium on Intelligence Information Processing and Trusted Computing

[3] Saadullah Kalwar, Nafeesa Bohra, Aftab A. Memon, "A Survey of Transition Mechanisms from IPv4 to IPv6 – Simulated Test Bed and Analysis", Department of Telecommunication Engineering, Mehran University of Engineering and Technology.

[4] Danial Minoli, "BUILDING THE INTERNET OF THINGS WITH IPv6 AND MIPv6", [TB]

[5] Venkata Krishna Kishore Terli, Swetha Prabha Chaganti, Navya Bharathi Alla, Shobhitha Sarab, Tarik El Taeib, "SOFTWARE IMPLEMENTATION OF IPv4 to IPv6 MIGRATION", Department of Computer Science and Engineering, University of Bridgeport, Bridgeport, USA.

[6] Muhammad Rizwan Sabir, "An Overview of IPv4 to IPv6 Transition and Security Issues", University of Engineering and Technology, Lahore, Pakistan

[7] Ch. Bouras1, 2 A. Gkamas K. Stamos,"From IPv4 to IPv6: The Case of OpenH323 Library", Research Academic Computer Technology Institute, Riga Feraiou 61, GR-26221 Patras, Greece

[8] Eun-Young Park, Jae-Hwoon Lee,"An IPv4-to-IPv6 Dual Stack Transition Mechanism Supporting Transparent Connections between IPv6 Hosts and IPv4 Hosts in Integrated IPv6/IPv4 Network", Dept. of Information and Communication Engineering Dongguk University Seoul, Korea

[9] Abidah Hj Mat Taib, UiTM Perlis and Rahmat Budiarto, USM,"Security Mechanisms for the IPv4 to IPv6 Transition", the 5th Student Conference on Research and Development –SCOReD 200711-12 December 2007, Malaysia

[10] Cheng Yuan-hang "Transition from IPv4 to IPv6 Based on Socket applications", The Library of Guizhou University Guizhou University GuiYang, China

[11] Yuan-yuan LU,"Transition from IPV4 to IPV6 Network Application", Educational Technology Center, Huanggang Normal University, Huangzhou, Hubei, China

