International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Study for IPv4 and IPv6 Coexistence

Abhay Singh[1], Alabhya Maheshawari[2], Dushyant Shukla[3]

[1, 2, 3]Galgotias College of Engineering and Technology, Knowledge Park- II, Greater Noida, Uttar Pradesh (201306), India

Abstract: ***IPv6 developed by Internet Engineering Task Force (IETF) regarded as the next generation Internet Protocol asks for the need for replacing the current Internet Protocol (also known as IPv4). IETF being harbinger to this new technology proposes several transition mechanisms for the purpose of integration of IPv6 with the existing networks. This work examines and conjecturally evaluates various approaches aimed at the coexistence of IPv4 together with IPv6, namely dual stack, tunneling and translation mechanisms. This paper throws light on the basis of these different approaches along with the various issues associated with their practical implementations.***

Keywords: Dual stack, IPv4, IPv6, tunneling.

1. Introduction

Migrating instantly from IPv4 to IPv6 is impractical because of the large size of the Internet and the vast number of users that are already on the IPv4. Moreover, more and more organizations are becoming increasingly dependent on the Internet for their work and therefore, cannot afford to undergo the downtime required to switch from IPv4 to IPv6. Therefore, the situation does not call to decide a single stage at which all the IPv4 networks are turned off, and the IPv6 ones are turned on. Rather the two versions of the internet protocol can exist together simultaneously, and the transition from IPv4 to IPv6 can be carried out step by step, node by node. Meanwhile, the complete transition takes place the two versions can work together without any problems. It should be ensured that the migration from IPv4 to IPv6 is carried out node by node using auto-configuration procedures to eliminate the need to configure the IPv6 hosts manually. This way the newly implemented IPv6 networks can provide internet users with the various advantages that the IPv6 has to offer and at the same time maintaining the possibility of communicating with IPv4 users or peripherals. Therefore there seem to be no reason not to move to IPv6. In this paper, we are going to have a look at various mechanisms as devised by the IETF to ensure all that is being promised.

IETF has been working on the deployment of the next generation Internet Protocol that can take the place of the current version 4 of the Internet Protocol. As it is very impractical and expensive to switch already existing IPv4-based infrastructure with IPv6 and therefore to confirm swift integration of IPv6 into existing networks, the IETF IPng Transition working party has been working on many transition methods, tools, and mechanisms. In general, what is done in these transition mechanisms is that the IPv6 packets are encapsulated into IPv4 packets and are transported over an IPv4 network infrastructure. As the internet completes its transition from IPv4 to IPv6-based infrastructure, it is only valid to have faith in these conversion techniques.

The aim of this work is to examine and conjecturally evaluate various approaches aimed towards the coexistence of IPv4 together with IPv6, namely dual stack, tunneling and translation mechanisms. This paper throws light on the basis of these different approaches along with the various issues associated with their practical implementations.

2. Background

Internet Protocol first came into existence in the early 1980s. In the 1990s, the pace at which the Internet was growing made it only evident that the IPv4 address space would ultimately get exhausted. However, some solutions were devised to cope with the situation. Some of them are NAT (Network Address Translation) and CIDR (Classless Inter-Domain Routing). However, the work on next generation Internet Protocol, namely IPv6 had already started.

The chief reason for a new Internet Protocol was to be at par with the ever increasing rate of Internet users by increasing the address space; IPv6 was designed with 128 bit address scheme, enough to label every molecule on the surface of the earth with a unique address. Moreover, at the time when IPv4 came into existence Internet was loaded with elastic traffic, such as emails and file transfers which can mold itself according to the network conditions. Whereas inelastic traffic is not much flexible in terms of the condition of the network and can render any application useless if a certain quality of performance is not ensured. IPv6 supports both elastic and inelastic traffic.

[1]IPv6 is structured to support scalability, security, and multimedia transmissions:

- IPv6 has addressing space of 128 bits
- IPv6 header mandates IPsec support.
- The Flow Label field in the IPv6 packet header is now responsible for identifying payload for QoS handling by routers.
- In IPv6 hosts instead of routers are responsible for Fragmentation support.
- IPv6 brings in extension headers and scraps support for checksum and options included in the header.
- IPv6 is equipped with auto-configuration mechanisms which do not need manual configuration or DHCP (Dynamic Host Configuration Protocol).

Overall, IPv6 is carefully designed and developed keeping in mind the needs of future applications.

The major change has been brought in the packet layouts for IPv4 and IPv6. The header length of IPv4 and IPv6 are 20 bytes and 40 bytes respectively. IPv6 has a lesser number of required fields by the virtue of the extension header that makes them optional in spite of the fact that IPv6 has an address space four times as large as that of IPv4. Now the question is does this increased 20 bytes cause any performance overhead. Statistics suggest that the performance overhead caused is highly insignificant in theory.

3. Migration Technologies

There are various transition techniques available such as:

- Dual-stack,
- DTI and Bump-in-dual-stack
- NAT Protocol Translator
- Stateless IP/ICMP Translator (SIIT)
- Assignment of IPv4 Global Addresses to IPv6 Hosts (AIIH)
- Tunnel Broker,
- 6-to-4 Mechanism
- 6-over-4 Mechanism
- IPv6 in IPv4 tunneling.

While dual stack mechanisms are easiest to implement, still complexity is increased at the hosts as the infrastructure cost is higher due to a more complex technology stack. NAT Protocol not only has scaling and DNS issues, it also suffers from the single point of failure disadvantage. The Tunnel Broker although dynamically gains access to tunnel servers but has authentication and scaling issues. [1] 6-to-4 technique brings about tunnels which are dynamic as well as stateless in nature over IPv4 infrastructure in order to connect to 6-to-4 domains. Whereas isolated IPv6 hosts are connected over the IPv4 infrastructure without the need of any IPv6 enabled routers or tunnels through 6-over-4 technique. Also with some help of manually configured tunnels the existing infrastructures can be used via IPv6 in IPv4 tunneling.

In this paper, we have chosen to restrict our study to migration techniques that fall under the following three categories:

- **Dual stack –** support both IPv4 and IPv6 on network devices.
- **Tunneling –** encapsulation of an IPv6 packet within an IPv4 packet for transmission over an IPv4 network.
- **Transition –** address or port transition of addresses such as via a gateway device or the host's or router's TCP/IP code that provides with the transition code.

4. Dual-Stack Methodology

This technique makes available the devices that can process IPv4 as well as IPv6 network together at the same time. However whenever the traffic is received by any such node IPv6 is given a preference over IPv4. If the received traffic only consists of IPv4, nodes can process it as well.

During the conversion process there will be devices, such as routers, other infrastructure devices and end-user devices etc. which require access to both network-layer technologies and therefore call for the need of implementing both IPv4 and IPv6 protocol stacks for on these devices. These devices can be configured with both IPv4 as well as IPv6 addresses, that too using the methods defined for the respective protocols as enabled by administrators. [2] For instance, an IPv4 address may be obtained via DHCPv4 while the IPv6 address may be auto-configured.

[2] The extent of dual-stack implementation may vary. The two IP versions can have a separate protocol stack with their unique qualities while sharing the part of it that is common to both of them. Usually, only the network layer would be dualized, using a common application, transition and data link layer. Another approach may utilize separate protocol stacks for the two protocol versions. While this may violate the benefits of layered protocol model, it is desirable especially in the case of network servers having multiple applications or services, some of which support only one version or the other.

4.1 Dual-Stack Implementation

As stated earlier the devices that share a common interface have common physical links serving both versions of the protocol. Moreover, the dual-stack devices require dual stacked routers that can support such links much like IPv4 or IPv6 support provided by the Ethernet and other layer 2 technologies. This is a very common approach during the transition.

4.2 DNS Issues

[2] DNS plays an important role in networking as it provides the linkage between end-user and the destination IP address. End-user will access a dual-stacked host by typing in the host name, and their application will query DNS. If the application can be configured by the administrators to support both an IPv4 and IPv6 address query, it may receive the destination's IPv4 and IPv6 addresses. Two different API namely, "bump in the stack" and "bump in the API" translation techniques support this feature if such a dual-query lookup is not natively supported by applications.

Any node with dual-stack implementation must support for the reception of IPv4 as well as IPv6 type of records as it preforms its DNS resolution and must ensure communication with destination intended through the use of address and protocol corresponding to the returned record. The definition of the network protocol preferred must be enabled by the resolved configuration in cases where the query returns both IPv4 as well as IPv6 records. It must also take care of the protocol to use when issuing DNS queries themselves.

4.3 DHCP Issues

[2] Each stack in dual-stack mechanism has its own version of DHCP. That is, DHCP and DHCPv6 provide IPv4 and IPv6 addresses or prefixes. Both forms of DHCP however, provide additional configuration information such as which DNS and NTP server is to be used. This information obtained may lead to incorrect behavior on the client depending on how the

information from both servers is merged together. Currently DHCP and DHCPv6 servers reside on a common physical server and are used for their respective versions of IP addresses.

5. Tunneling Methodologies

There are a number of technologies that have be developed to support IPv4 over IPv6 as well as IPv6 over IPv4 tunneling. These techniques are either manually configured or are automatically implemented. Configured tunnels have to be predefined, on the other hand automatic tunnels are created just in time.

Commonly, IPv6 packets are tunneled through an IPv4 network by prefixing each IPv6 packet with an IPv4 header (Figure 1). As a result the tunneled packet can now be routed over an IPv4 routing infrastructure. Encapsulation is performed by the entry node (a router or a host) of the tunnel. IPv4 address of this node and that of the tunnel endpoint constitutes the source IPv4 address and the destination IPv4 address in IPv4 header respectively. The protocol field of the IPv4 header is set to 41 (decimal) indicating an encapsulated IPv6 packet. Decapsulation of the tunneled packet is performed by the exit node of the tunnel which strips off the IPv4 header and appropriately routes the packet as to the ultimate destination using IPv6.

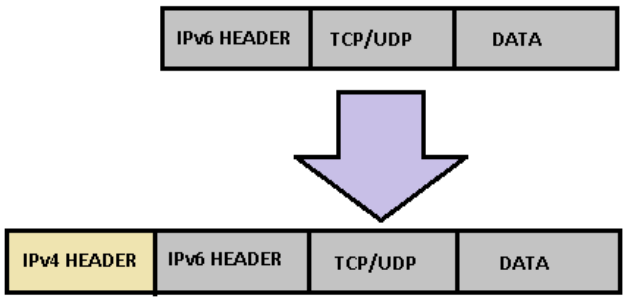

Figure 1: IPv6 over IPv4Tunnel

5.1 Tunnel Types

Based on the tunnel endpoints, tunnels can be grouped into various categories albeit the basic process of tunneling is the same for all kinds of tunneling. A typical approach for tunnel configuration is router-to-router tunnel.

5.1.1 Router-to-Router Tunnel

In figure 2, a packet is sent from the host on the left with IPv6 address of W across the network to the host on the right with the IPv6 address Z. A router with an IPv4 address of B and IPv6 address of X receives the packet. This router has been configured to tunnel packets over to the network on which host Z resides. The router encapsulates the IPv6 address packet within an IPv4 header. This router uses its IPv4 address as the source IPv4 address. The packet travels the tunnel over to the router with IPv4 address B and IPv6 address Y. Its IPv4 address is used as the destination IPv4 address. This router then decapsulates the packet, stripping off the IPv4 header and routes the original IPv6 packet to its intended destination (Z).

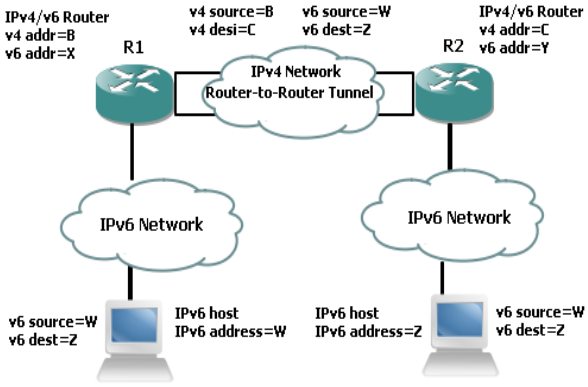

Figure 2: Router-to-Router Tunnel

5.1.2 Host-to-Router Tunnel

[2]Host-to-Router tunneling scenario features an IPv6/IPv4 host capable of supporting both IPv4 and IPv6 protocols. A packet is encapsulated and tunneled to a router, which is again decapsulated there and from there it is routed natively via IPv6. Figure 3 displays the flow as well as the packet header addresses. Apart from the tunnel endpoints, this tunneling technique is the same as the router-to-router tunnel configuration.

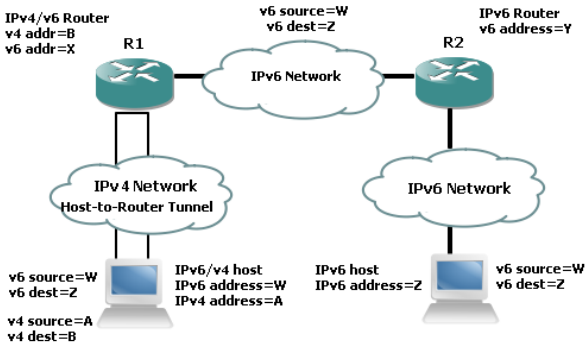

Figure 3: Host-to-Router Tunnel

5.1.3 Router-to-Host Tunnel

[2]The router-to-host configuration and router-to-router tunneling are very similar to each other. An IPv6 packet is sent by the IPv6 host on the left of the diagram to its local router, which in turn routes the packet towards a router nearest to the destination. The serving router is designed to tunnel IPv6 packets over IPv4 to the host, as shown within the figure.

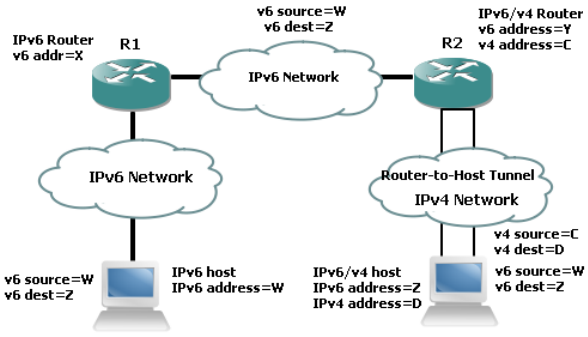

Figure 4: Router-to-Host Tunnel

5.1.4 Host-to-Host Tunnel

[2]The Host-to-Host tunneling configuration spans end-to-end, from host-to-host. If the routing infrastructure is yet to be upgraded to support IPv6, communication between two IPv6/IPv4 hosts via a tunnel over IPv4 network is enabled by this tunneling configuration. This has been shown in figure 5. The diagram illustrates an end-to-end IPv4 communication.

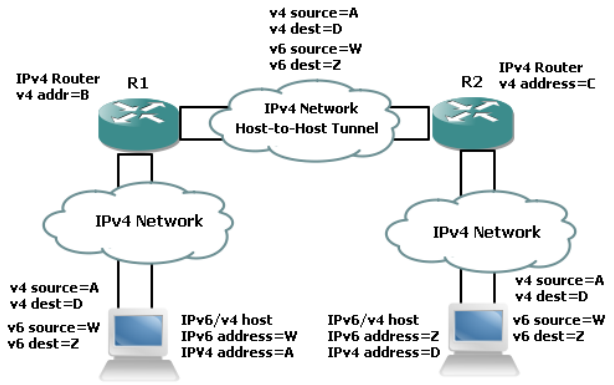

Figure 5: Host-to-Host Tunnel

6. Translation Methodologies

Translation techniques are used to perform IPv4-to-IPv6 translation (and vice versa) at a particular layer of the protocol stack, typically the network, transport or application layer. Translation techniques differ from tunneling in the sense that the later does not changes the tunneled data packet, whereas former modify or translate IP packets commutatively between IPv4 and IPv6. However, these techniques are only utilized in environments where IPv6-only nodes communicate with IPv4-only nodes. In dual-stack environments, native or tunneling mechanisms are used [1].

6.1 Stateless IP/ICMP Translation (SIIT) Algorithm

IP packet headers translation between IPv4 and IPv6 is done by SIIT. Once configured on aIPv6 enabled host, it converts the outgoing IPv6 packet headers into IPv4 headers, and incoming IPv4 headers into IPv6. The IPv6 enabled host must be provided with an IPv4 address as well so that the algorithm using DNS resolution to an IPv4 address would convert the IPv6 packet header into IPv4 header whenever the IPv6 host tries to communicate with an IPv4 host. The SIIT algorithm recognizes the situation when an IPv6 address is an IPv4-mapped address, formatted as shown in the figure 6. The bump-in-the-stack (BIS) or bump-in-the-API (BIA) techniques are responsible for conversion of resolved IPv4 address into an IPv4-mapped address.

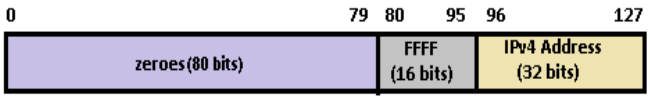

Figure 6: IPv4 Mapped Address Format

SIIT algorithm senses the presence of the IPv4-mapped address format as the destination IP address and performs header translation to yield an IPv4 packet for transmission via the data link and physical layers as shown in figure 7.

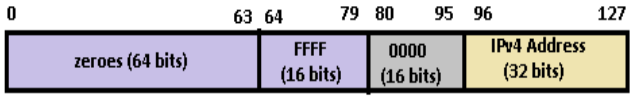

Figure 7: IPv4-Transalted Address Format used within SIIT

Figure 8 shows an example of the SIIT algorithm. SIIT stack is normally packed inside a bump-in-the-stack or bump-in-the API solution.

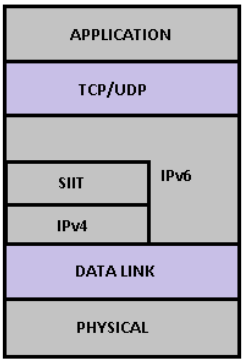

Figure 8: SIIT Stack

6.2 Bump-in-the-Stack (BIS)

[2]BIS is a technique through which IPv4 applications communicate over IPv6 networks. Data flowing between the link layer devices (e.g., network interface cards) and the TCP/IPv4 module is snooped as well as the IPv4 packet are translated into IPv6 by the BIS. Figure 9 illustrates the components of BIS.

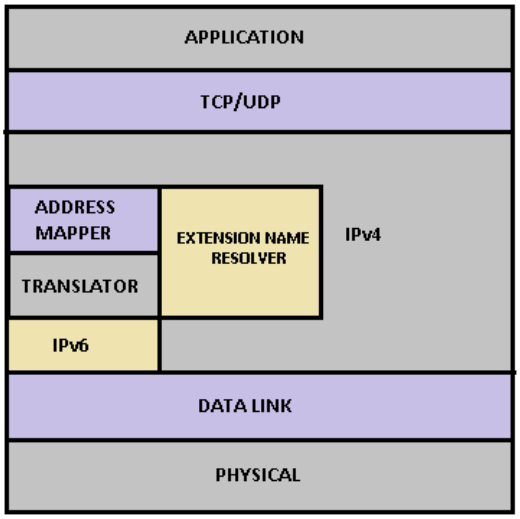

Figure 9: Bump-in-the-Stack Components

The translator performs the translation of the IPv4 header into an IPv6 header according to the SIIT algorithm. DNS queries for IPv4 record types are snooped by the Extension Name Resolver; an additional query for both IPv4 and IPv6 record types for the same host name is created by the Extension Name Resolver upon the receipt of such query. If no affirmative answer is acquired from the IPv6 query, the communication sticks to using IPv4; if the IPv6 query is resolved, the Extension Name Resolver instructs the Address Mapper component to associate the returned IPv4 address

(IPv4 record) with the returned IPv6 address (IPv6 record). If only IPv6 response is received, the Address Mapper assigns an IPv4 address from a configured pool of addresses.

For the application to be provided with the resolution to the IPv4 query an IPv4 address is required up the stack to the application. The real or self-assigned IPv4 addresses are mapped to the destination IPv6 addresses via the Address Mapper. The data packets using the IPv4 address are converted into IPv6 enabled packets for transmission via IPv6 enabled.

If an external source that has not already been mapped sends an IPv6 packet to the BIS host, an IPv4 address is assigned to the source by the Address Mapper from its pool and the IPv6 header is translated into IPv4 for communication up the stack.

6.3 Bump-in-the-API (BIA)

The BIA strategy makes it possible to use o the IPv4 applications while communicating over an IPv6 network. Unlike IP header modification provided by BIS, the BIA approach performs translations between IPv4 and IPv6 APIs. BIA is placed on the host in between the application and TCP/UDP layer of the stack. As figure 10 shows, an API Translator, Name Resolver, Function Mapper and an Address Mapper constitute BIA.

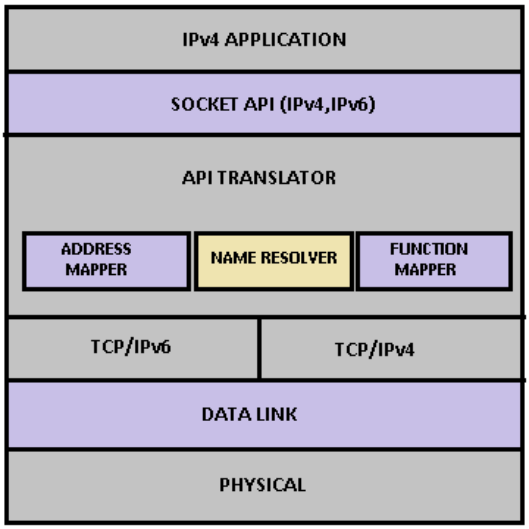

Figure 10: Bump-in-the-API

When the DNS query is sent by an IPv4 application in order to determine the IP address of a destination host, the query is intercepted by the Name Resolver and in turn creates a new query requesting both IPv4 and IPv6 records. A DNS reply with an IPv4 record will provide the answer with the given IPv4 address. Name Resolver is stimulated to request an IPv4 address from the Address Mapper to map the IPv6 address as returned from the DNS IPv4 reply. The mapped IPv4 address is utilized by the Name Resolver to return an IPv4 record response to the application. The Address Mapper maintains the mapping of IPv6 addresses with those assigned from internal address pool constituted by the unassigned IPv4 address space (0.0.0.0/24). The interception of API function calls and mapping of IPv4 API calls to IPv6 socket calls is performed by the Function Mapper.

6.4 SOCKS IPv6/IPv4 Gateway

[2]SOCKS, defined in RFC 1928, provides transport relay for applications traversing firewalls, effectively providing application proxy services. SOCKS protocol performs the translation of the IPv4 and IPv6 communications. And in a very similar manner to the other translation techniques we have seen, this technique encompasses t *DNS name resolving delegation*, a special DNS treatment, through which the resolved name is delegated to the SOCKS IPv6/IPv4 gateway from the resolver client. In order for an IPv4 or IPv6 application for communicating with the SOCKS gateway proxy, it has to be first "socksified" for eventual connection to a host enabled with the opposite protocol. As it has been shown in figure 11 an IPV6 host with a SOCKS client is connected to an IPv4 host, from left to right. An IPv4 host that has already been socksified can communicate to an IPv6 host, from right to left via the SOCKS gateway.

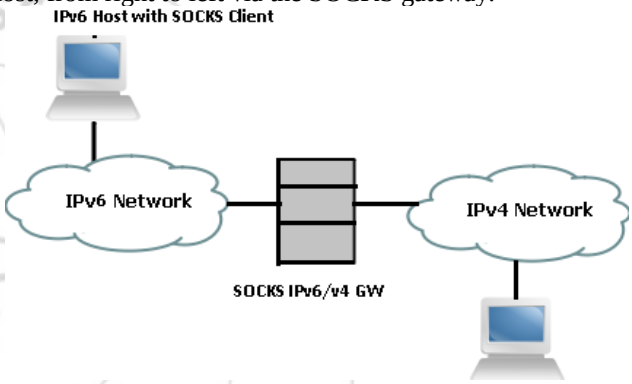

Figure 11: Basic SOCKS Gateway Configuration

6.5 Transport Relay Translator (TRT)

[2]Just like the SOCKS configuration, TRT has a stateful gateway device that connects two "independent" connections over different networks. The TCP/UDP connection from a host terminates on the TRT, which in turn creates a separate connection to the destination host and relays between the two connections. TRT needs a DNS Application Layer Gateway (DNSALG), which functions as a DNS proxy. TRT enables communication between IPv6 hosts and IPv4 terminals, e.g., web servers.

[2] Whenever the IPv6 resolvers request an IPv6 source record query, the same is triggered by the DNS-ALG; the resolver is provided with a response and an IPv6 connection is ensured if and when am IPv6 record is returned. Otherwise, the DNS-ALG performs an IPv4 record query, and if an answer is acquired, the DNS-ALG formulates an IPv6 address using the IPv4 address contained in the acquired IPv4 record. The prefix C6:: / 64 is followed by 32 zeroes plus the 32-bit IPv4 address. However, the C6:: /64 prefix has not been allocated by IANA. Thus there exists a requirement of a locally configured prefix.

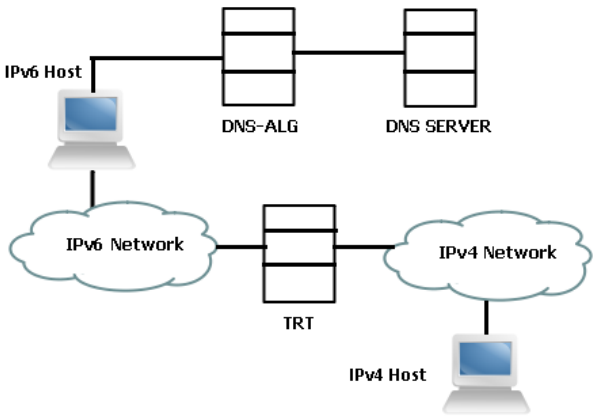

Figure 12: TRT Configuration with DNS-ALG

6.6 Application Layer Gateway (ALG)

Protocol transformations at the application layer is the responsibility of ALGs. They are also responsible for application proxy functions just like the HTTP proxies. For a proxy server, an application is needed to be configured with its IP address and it is only when the application is opened that the connection to the server is made just as a browser connects to the HTTP proxy server upon its launch. ALGs are useful for web or other application-specific access to the IPv4 Internet by hosts residing on an IPv6-only network.

7. Conclusion

Numerous IPv4-to-IPv6 transition mechanisms have been devised to readily enable the migration. Although the transition towards absolute IPv6 networks has begun but it will be carried out step-by-step. It is going to be a gradual process and until the time IPv6 is universally implemented, IPv4 and IPv6 devices will coexist.

References

[1] Ioan Raicu, Sherali Zeadally, "Evaluating IPv4 to IPv6 Transition Mechanisms"
[2] Tim Rooney, "IPv4-to-IPv6 Transition and Co-existence Strategies", Revised and Updated 2011 Edition Peng Wu, Yong Cui, Jiagchuan Liu, Chris Metz, "Transition from IPv4-to-IPv6: A State of the Arts Survey", IEEE COMMUNICATIONS SURVEYS and TUTORIALS, accepted for publication
[3] Microsoft, "IPv6/IPv4 Coexistence and Migration," White Paper, Washington, November 2001
[4] S. Tenebaum, Computer Networks, Third Edition, Prentice Hall Inc., 1996, pp. 686, 413-436, 437-449
[5] T. Dunn, "The IPv6 Transition," IEEE Internet Computing, Vol.6, Mo.3, May/June 2001, pp.11-13
[6] W. Richard Stevens, TCP/IP Illustrated, Volume 1: The Protocols, First Edition December 15, 1993
[7] IETF IPv6 Transition Working Group, http://www.6bone.net/ngtrans.
[8] IPv6 users' site: http://www.ipv6.org
[9] "Internet Usage Statistics," Miniwatts Marketing Group, Tech. Rep., Jun. 2011. [Online]. Available: http://www.internetworldstats.com
[10] W. Richard Stevens, TCP/IP Illustrated, Volume I: The Protocols, First Edition December 15, 1993
[11] "Internet Usage Statistics," Miniwatts Marketing Group, Tech. Rep., Jun. 2011. [Online]. Available: http://www.internetworldstats.com

Author Profile

Abhay Singh is currently pursuing B-Tech in Computer Science and Engineering from Galgotias College of Engineering and Technology, Greater Noida

Alabhya Maheshwari is currently pursuing B-Tech in Computer Science and Engineering from Galgotias College of Engineering and Technology, Greater Noida

Dushyant Shukla is currently pursuing B-Tech in Computer Science and Engineering from Galgotias College of Engineering and Technology, Greater Noida