I. J. Computer Network and Information Security, 2016, 2, 1-14
Published Online February 2016 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2016.02.01

Performance Evaluation of IPv4/IPv6 Transition Mechanisms

Adira Quintero, Francisco Sans, and Eric Gamess
School of Computer Science, Central University of Venezuela, Caracas, 1040, Venezuela
E-mail: {adira.quintero, francisco.sans, eric.gamess}@ciens.ucv.ve

Abstract—The exhaustion of IPv4 addresses has forced the deployment of the new version of the Internet Protocol (IPv6). However, the migration to the new protocol is done gradually and with the due care for many reasons that include: cost, inclusion of support for IPv6 in existing applications, training of technical staff, lack of web content available over IPv6 from important providers, and obsolete devices not anymore supported by manufacturers. For those reasons, many transition mechanisms have been proposed, each one to fill distinct requirements, with different operational theory and availability according to the network environment. A performance evaluation of these mechanisms can help network administrators and researchers in their selection of the best transition technology for their environment. In this paper, we present a performance comparison of some transition mechanisms such as ISATAP, 6to4, and NAT64 in real testbeds with Debian, Windows 7, Windows 8, and Windows 10. For NAT64, two different tools were tested: TAYGA and Jool. We measure the OWD and the throughput for UDP and TCP for every mechanism, for both Ethernet and Fast Ethernet technologies. From this research, we can conclude that all the modern operating systems for PCs already have good support for IPv6, and a very similar network performance. Also, we can infer from our work that in controlled environments, native IPv4 has the best performance, closely followed by native IPv6. The difference is essentially due to the length of the IP header (20 bytes in IPv4 and 40 bytes in IPv6). The tunneling solutions chosen for this research (ISATAP and 6to4) have a similar performance, which is the lowest of the studied technologies, because of the additional IPv4 header in the tunnel.

Index Terms—Performance Evaluation, Benchmarking Tools, IPv6, IPv4, Transition Mechanisms, ISATAP, 6to4, NAT64.

I. Introduction

With its unexpected growth, Internet has been facing for several years the exhaustion of available IPv4 addresses. To tackle this problem, many solutions have been proposed. One of the most popular solutions is the implementation of Network Address Translation (NAT) [1], where internal hosts with private IPv4 addresses are "represented" by a NAT server in Internet. Nevertheless, this is not a practical solution, since NAT clients do not have true end-to-end connectivity.

Another solution to this problem is the use of Internet Protocol version 6 (IPv6) [2]. This new version of IP has 128-bit addresses, while IPv4 is limited to 32-bit addresses. Furthermore, IPv6 adds many improvements in areas such as routing, multicasting, security, mobility, and network auto configuration. Also, it is worth to mention that the majority of the current operating systems has IPv6 support.

It is not possible to carry out the transition from IPv4 to IPv6 in a short period of time. Hence, during the transition, IPv4 and IPv6 will coexist. Many reasons extend the time of this coexistence period, including cost, inclusion of support for IPv6 in existing applications, training of technical staff, lack of web content available over IPv6 from important providers, and obsolete devices not supported anymore by manufacturers where it is impossible to upgrade to a new firmware with IPv6 support. As a consequence, IPv6 must be deployed gradually. Therefore, in the early deployment of IPv6, there are small isolated islands of IPv6 in an IPv4 ocean, and some transition mechanisms were created to allow the coexistence of the two versions of the protocol.

There are three types of transition mechanisms: dual-stack, translators, and tunneling techniques. In dual-stack, devices have IPv4 and IPv6 simultaneously, allowing the communication with both versions of the protocol. The translation mechanisms translate IPv6 packets into IPv4 packets and vice versa, allowing the communication between hosts with different IP versions. In the tunneling techniques, the basic idea is to communicate IPv6 hosts that are separated by an IPv4-only network. In this case, IPv6 packets are encapsulated into IPv4 packets to traverse IPv4 networks. Similarly, the tunneling technologies also include the reverse case, that is, the communication of IPv4 hosts that are separated by an IPv6 network. However, this last case is seldom used since IPv4 is still by far the dominant protocol.

Since there are several different transition mechanisms, it is important to know which is more suitable and will perform better in a specific situation. In this paper, we propose testbeds and do a performance evaluation of three transition mechanisms: ISATAP, 6to4, and NAT64. Their performance is compared with the one of native IPv4 and native IPv6 traffic. We use benchmarking tools to measure One Way Delay (OWD) and throughput for

the different mechanisms and make an analytical comparison of the results.

The rest of this paper is organized as follows. Section II makes a survey of the related work. Section III briefly describes the transition mechanisms that we chose for our study. Information about our testbeds and traffic generation tools is presented in Section IV. In Section V, an analytical comparison of the obtained results is done. Finally, Section VI discusses the conclusions and future work.

II. Related Work

Many studies have been conducted to measure the progress of IPv6 deployment. For example, Colitti et al. [3] developed a method for measuring IPv6 adoption. Claffy [4] provided an excellent summary of the deployment of IPv6 and also identified areas that require further study.

In the field of performance evaluation, on one hand there are several studies that evaluate the performance of IPv4 and IPv6 in different operating systems. Zeadally and Raicu [5] compared the performance of IPv6 in Windows 2000 and Solaris 8. An upper bound model to compute TCP and UDP throughput for IPv4 and IPv6, in a full-duplex point-to-point connection, was presented by Gamess and Surós [6]. They compared the performance of various operating systems with this upper bound. Narayan, Shang, and Fan [7][8] studied the performance of IPv4 and IPv6 traffic on various distributions of Windows and Linux for TCP and UDP. A similar study was conducted by Kolahi et al. [9], where the TCP throughput of Windows Vista and Windows XP was compared using IPv4 and IPv6. A comparison of network performance evaluation between Windows XP, Windows Vista, and Windows 7 was conducted by Balen, Martinovic, and Hocenski [10]. Svec and Munk [11] considered the usage of TCP, UDP and SCTP traffic in IPv4 and IPv6 networks. Furthermore, Gamess and Velásquez [12] made an analysis of the forwarding engine performance of PC-based routers using Solaris, Windows, and Debian for both, IPv4 and IPv6. Narayan et al. [13] did a similar work but with Fedora, Ubuntu, and Windows Server as operating systems. Finally, the performance of the IP protocols has also been compared in wireless networks [14][15][16], and with the use of QoS [17].

On the other hand, many authors have evaluated the performance of various tunneling mechanisms. Huang, Wu, and Lin [18] did a complete evaluation of the performance of Teredo on a testbed. Both Teredo latency and throughput were compared to ISATAP by Aazam et al. [19]. Web-based applications were implemented by Zander et al. [20] to test latency and delay introduced by Teredo and to compare it with 6to4. Narayan and Tauch [21][22] compared 6to4 with configured tunnels, using Windows Server 2003, Windows Server 2008, Ubuntu 11.0, and Fedora 9.10. In a more recent studies, Hadiya, Save, and Geetu [23] also evaluated the performance of 6to4 and configured tunnels in a network infrastructure, while Amr and Abdelbaki [24] compared the routing convergence of 6to4 and configured tunnels versus the conventional IPv4 and IPv6 protocols. Yoon et al. [25] did a comparison of 6to4, 6rd, and ISATAP throughput and CPU utilization when using these communication techniques on testbeds for Ubuntu. An analytical performance evaluation for ISATAP, 6to4, 6rd, and Teredo was performed by Sans and Gamess [26].

At the level of translation techniques, many studies are specific to NAT64/DNS64. For example, Llanto and Yu [27] measured the performance of TAYGA and TOTD (a DNS proxy nameserver). Monte et al. [28], and Yu and Carpenter [29] evaluated the performance of an Ecdysis NAT64 and DNS64 implementation. Hozdic and Mrdovic [30] combined an Ecdysis NAT64 with a BIND DNS64 server for their evaluation. Performance of BIND DNS64 has also been evaluated with TAYGA by Lencse and Takacs [31]. BIND and TOTD DNS64 servers' performance has been compared by Lencse and Repas [32] in Linux, OpenBSD, and FreeBSD. These authors also studied the stability of two NAT64 implementations: TAYGA and PF, using ICMP [33]. Finally, Repas, Farnadi, and Lencse [34] did a performance evaluation of free NAT64 implementations at the level of UDP and TCP.

Simulation tools have also been used for performance evaluation of IPv4 and IPv6. For example, the authors of [35] worked with OPNET IT Guru to investigate areas of performance weakness in VoIP.

In our work, we consider native IPv4, native IPv6, and three transition mechanisms (ISATAP, 6to4, and NAT64), on conventional PCs with four different operating systems (Debian, Windows 7, Windows 8, and Windows 10) that are used as a router, and as communication end-points. For NAT64, we propose a comparison between TAYGA and Jool implementations (a comparison that has not been done to the best of our knowledge).

III. Transition Mechanisms

Tunneling mechanisms allow remote IPv6 nodes to communicate with each other through an IPv4-only network. These methods provide an IPv6 virtual-link on an actual IPv4 network to enable IPv6 communication. The basic idea of tunnels is to encapsulate IPv6 packets into IPv4 packets, by adding an IPv4 header with protocol number 41. In this way, IPv6 packets can traverse IPv4-only networks to reach the IPv6 destination. In one end-point of the virtual link, a device encapsulates (adds an IPv4 header to the IPv6 packet), while decapsulation (removing the IPv4 header from the IPv6 packet) is done in the second end-point.

In the other hand, translation mechanisms translate IPv6 packets into IPv4 packets and vice versa. When an IPv4 to IPv6 translator receives an IPv4 packet that goes to the IPv6 world, the IPv4 header is eliminated and used to construct an IPv6 header as its substitute. A similar process occurs when an IPv6 packet is received by an IPv6 to IPv4 translator.

For this study, we chose ISATAP and 6to4 for

tunneling mechanisms, and NAT64 for translation mechanisms. We selected these technologies due to their popularity, their different range of applications, and because many implementations are available from both the Internet community and manufacturers. It is worth remembering that Network Address Translation-Protocol Translation (NAT-PT) was a popular translation technique, which has been deprecated by [36], hence it is not covered in this research.

A. ISATAP

Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) [37] is a tunneling mechanism that allows dual-stack hosts within an IPv4 underlying network to communicate with remote IPv6 devices.

The ISATAP address is configured with a modified EUI-64 format, by concatenating the 24-bit IEEE OUI (00-00-5E for private IPv4 address and 02-00-5E for public IPv4 address), the 8-bit hexadecimal value 0xFE, and the 32-bit IPv4 address of the node. In this way, the node configures an initial link-local address, which allows it to communicate with other ISATAP nodes that are in the same virtual local link.

ISATAP mechanism assumes that multicast is not available in the IPv4 network, so it provides a way to discover potential routers. ISATAP hosts keep a list of routers, configured manually or obtained by a DNS query to isatap or isatap.company.com (where the local domain is company.com). With the IPv4 address of the router, hosts can perform a router solicitation to acquire a global IPv6 address, with the same modified EUI-64 format described above. This allows an ISATAP host to communicate with the IPv6 Internet.

ISATAP is implemented in most platforms (Windows XP/Vista/7/8/10, Linux, etc) and it is easy to configure.

B. 6to4

6to4 [38] is a tunneling mechanism that allows routers with a global IPv4 address to create an IPv6 site. It uses the 2002::/16 prefix, designated by the Internet Assigned Numbers Authority (IANA) to create the address block. A 6to4 site can have up to 2^{80} IPv6 addresses of the type 2002:<IPv4Address>:/48, which are made by combining the prefix designated by the IANA with the 32 bits of the IPv4 address of the 6to4 router. It is possible to route any packet to these addresses by encapsulating and sending the packet to the IPv4 address of the 6to4 router embedded in the IPv6 address, allowing the communication between any 6to4 domains.

The communication between a 6to4 domain and the IPv6 Internet requires a 6to4 relay. This device advertises the 2002:/16 prefix to the IPv6 Internet, letting native IPv6 hosts to send packets to a 6to4 host. When a 6to4 router has to forward an IPv6 packet to a native IPv6 host, it encapsulates and sends the packet to one of the 6to4 relays, which then routes the packet in the IPv6 Internet in a normal way, after decapsulation. To avoid manual configuration of the IPv4 addresses of the relays, IANA has designated the IPv4 anycast address 192.88.99.1 [39]. With this anycast address, routing is improved, and third-party relays can be used for robustness, redundancy, and high availability.

C. NAT64

NAT64 [40] is a translation mechanism that allows IPv6-only clients to access IPv4-only servers, through translation of IP headers. Two different devices are needed in NAT64: a DNS64 server and a NAT64 router. Also, this technology requires two IP pools. The first pool is an /96 IPv6 prefix [41] (usually 64:ff9b::/96) used to embed the IPv4 address of the servers to be reached by IPv6-clients. It is also known as the designated IPv6 prefix. The other pool is a pool of IPv4 addresses (at least one IPv4 address) that will be temporarily employed to represent the IPv6-clients in the IPv4 Internet. It is also known as the dynamic IPv4 pool. When an IPv6 client want to establish a connection with a server, it consults its DNS server (in this case a DNS64 server). If the solicited server has an IPv6 address, the DNS64 server answers with the corresponding IPv6 address (AAAA record), and the IPv6 packet is submitted to regular IPv6 routing. Otherwise, the DNS64 server generates an IPv6 address that represents the IPv4-only server by concatenating the /96 well-known prefix with its IPv4 address (usually 64:ff9b:<IPv4Address>). Hence, the IPv6 client does not need support for NAT64/DNS64. The resulting IPv6 packet will travel in the IPv6 world (using the IPv6 address of the IPv6 client as source address, and the IPv6 address given by the DNS64 server as destination address) until the NAT64 router. The later translates the IPv6 header into an IPv4 header, by using one of the IPv4 address in its second pool as the source IPv4 address, and the IPv4 address embedded in the IPv6 destination address, as the destination IPv4 address. The NAT64 router also adds an entry in the session table, for the inverse translation (IPv4 to IPv6), for returning packets. This entry is established with the first packet of the flow, and maintained while there is communication activities between the IPv6 client and the IPv4 server.

IV. Benchmarking Tools and Testbeds

A. Test Method

In this section, we present the benchmarking tools selected for our study. We also propose the testbed for the different technologies: native IPv4, native IPv6, ISATAP, 6to4, and NAT64.

To calculate the OWD, we chose a benchmark developed previously by this research group [42]. As stated in [42], OWD or Round Trip Time (RTT) reported in many evaluation tools are not reliable since they are based on synchronized computers, which is difficult to achieve at the level of microseconds. The key is to take all the timestamps in the same computer. Our benchmark is based on the client/server model. Basically, a packet (IPv4 or IPv6) of a fixed length is exchanged between the client and the server a number of times (defined by the users). The benchmark takes a timestamp before and after the interchange. The difference of the timestamps is

divided by the number of time the packet was sent and received to obtain an average of the RTT over the path, and then again by 2 to get the average OWD.

For the throughput, we selected Iperf [43], also based on the client/server model. This benchmark sends traffic from the client to the server, which then calculates the number of bits per seconds received. A high bandwidth has to be specify to ensure that the tool saturates the network in order to obtain the maximum throughput. Even though Iperf can measure both TCP and UDP throughput, the tool only allows to specify the bandwidth in UDP. Therefore, we only perform throughput tests for UDP in our experiments.

The performance evaluation is done by sending IPv6 packets (or IPv4 packets, depending of the technology used) between two PCs with the benchmarking tools mentioned above. The main skeleton of the testbeds consists of two PCs (PC1 and PC2) connected with another PC (R1) with two NICs (Network Interface Cards) functioning as a router. This basic configuration is shown in Fig. 1. We used PCs with the same characteristics: HP xw4600 with an Intel Core 2 Duo E6750 CPU at 2.67 GHz and 8 GB of RAM. For the NICs, we installed Intel PCI Ethernet adapters (Intel PRO/100 S).

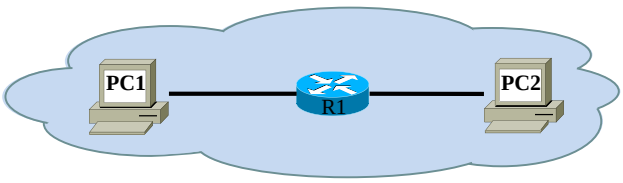

Fig.1. Skeleton of Testbed.

We created four partitions in the hard disk of the PCs (PC1, PC2, and R1) and installed the following operating systems: Debian 7.0.8 (Wheezy), Windows 7, Windows 8, and Windows 10, all in their 64-bit version. We chose those operating systems because of their popularity and robustness. For any experiment, we run the same operating system in the three PCs, with the goal of evaluating the performance of a transition technology for this particular operation system.

Every test was performed for Ethernet and Fast Ethernet technologies. All the experiments were repeated 10 to 20 times to get consistent measurements. The average obtained for each test is presented as the result. The packets were sent using UDP and/or TCP as the transport protocols. We varied the values for the UDP and TCP payload as specified in the following list: 50, 100, 250, 500, 750, 1000, 1250, 1500, 1750, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, and 10000 bytes.

B. Testbeds

1) *Native IPv4:* In the native IPv4 testbed, both router and PCs were configured only with IPv4 as shown in Fig. 2. The IPv4 addresses and the routes of all devices were statically configured.

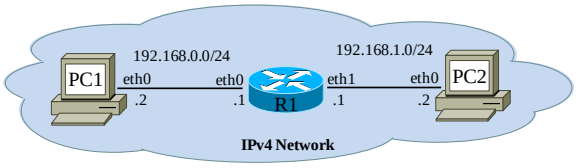

Fig.2. Native IPv4 Testbed.

Enabling IPv4 forwarding in Debian is done by setting the sysctl kernel parameter *net.ipv4.conf.all.forwarding* to 1. In the case of Windows, variable *IPEnableRouter* must be set to 1 in the following system registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

This router's configuration was used in every testbed in which an IP4 router was needed.

2) *Native IPv6:* Similarly to the native IPv4 testbed, the addresses and routes of both router and PCs were configured statically, as depicted in Fig. 3.

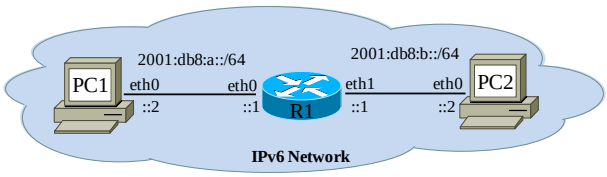

Fig.3. Native IPv6 Testbed.

Enabling IPv6 forwarding in Debian is straightforward. It is just a matter of setting the sysctl kernel parameter *net.ipv6.conf.all.forwarding* to 1. In the case of Windows, the command indicated in Fig. 4 must be typed into the console to activate the forwarding of packets. Since the router has two NICs, it is necessary to execute the command twice, once for every interface. This router's configuration was used in every testbed that needed an IPv6 router.

Console commands
01: netsh interface ipv6 set interface <NIC> forwarding=enabled

Fig.4. Enabling Forwarding in R1 for Native IPv6 (Windows).

3) ISATAP: Fig. 5 shows the ISATAP testbed. For the configuration of the ISATAP tunnel, PC1 (an ISATAP host) must have the IPv4 address of the ISATAP router. In our experiments, R1 acts as the ISATAP router.

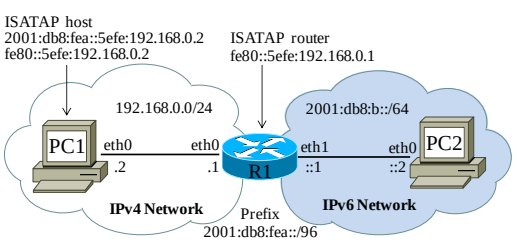

Fig.5. ISATAP Testbed.

To configure ISATAP in Debian, the daemon *isatapd* was installed in PC1. PC1 was set as illustrated in Fig. 6. The ISATAP daemon is run in line 06, and an IPv6 address is statically assigned in line 07. The ISATAP router was configured as shown in Fig. 7. The tunnel is setup in lines 01-03, indicating that the local end-point of the tunnel is associated to IP address 192.168.0.1. Note that line 07 will not have been required in PC1 (see Fig. 6), if an advertisement daemon (such as *radvd*) were installed and configured in R1. In this case, PC1 will have learnt its global unicast address from the advertisement daemon.

```
File /etc/network/interfaces
01: auto eth0
02: iface eth0 inet static
03: address 192.168.0.2
04: netmask 255.255.255.0
05: gateway 192.168.0.1
06: up isatapd –d –l eth0 –r 192.168.0.1
07: up ip -6 addr add 2001:db8:fea::5efe:192.168.0.2/64 dev is0
```

Fig.6. Configuration of PC1 for ISATAP (Debian).

```
Console commands
01: ip tunnel add is0 mode isatap local 192.168.0.1
02: ip link set is0 up
03: ip -6 addr add 2001:db8:fea:5efe:192.168.0.1/64 dev is0
```

Fig.7. Configuration of R1 for ISATAP (Debian).

In the case of Windows, PC1 must be configured by typing the commands indicated in Fig. 8. In line 01, the ISATAP tunnel is enabled, whereas line 02 indicates the IPv4 address of the ISATAP router.

```
Console commands
01: netsh interface ipv6 isatap set state enabled
02: netsh interface ipv6 isatap set router 192.168.0.1
```

Fig.8. Configuration of PC1 for ISATAP (Windows).

The Windows router, R1, is configured with the commands depicted in Fig. 9. Lines 01-02 enable ISATAP and indicate the IPv4 address of the router. Since this IPv4 address belongs to R1, it now knows that it is an ISATAP router. Then, packet forwarding is enable for IPv6 in lines 03-04, allowing the router to forward packets. Lines 05-06 indicate that R1 must advertise prefix 2001:db8: fea:/64 through Router Advertisement to ISATAP clients.

```
Console commands
01: netsh interface ipv6 isatap set state enabled
02: netsh interface ipv6 isatap set router 192.168.0.1
03: netsh interface ipv6 set interface 21 forwarding=enabled
04: netsh interface ipv6 set interface 22 forwarding=enabled
05: netsh interface ipv6 set interface 21 advertise=enabled
06: netsh interface ipv6 add route 2001:db8: fea:/64 21 publish=yes
07: netsh interface ipv6 add route:/0 22 2001:db8: b: 2 publish=yes
```

Fig.9. Configuration of R1 for ISATAP (Windows).

4) 6to4: This testbed was configured as shown in Fig.10. In this case, R1 is an IPv4-only router, and PC1 and PC2 are configured as 6to4 hosts/routers. That is, PC1 and PC2 have both stacks. For the IPv6 packets generated by PC1 and PC2, an IPv4 header is added by the generating host before sending them through the IPv4 network toward its destination.

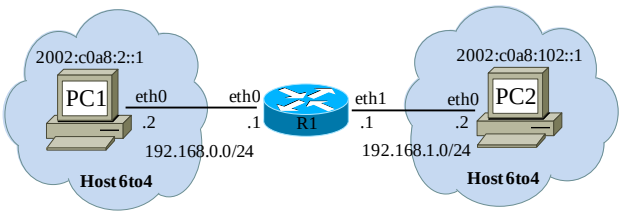

Fig.10. 6to4 Testbed.

For Debian, the configuration of PC1 as a 6to4 host/router is shown in Fig. 11. In this case, lines 01-03 create and setup the pseudo-interface (called 6to4). The configuration of PC2 is similar to PC1, with the exception of the local IPv4 address (192.168.1.2).

```
File /etc/network/interfaces
01: auto 6to4
02: iface 6to4 inet6 static
03: local 192.168.0.2
```

Fig.11. Configuration of PC1 for 6to4 (Debian).

In Windows, 6to4 is configured as presented in Fig. 12. PC1 and PC2 only require the command of line 01 to indicate that 6to4 must be enabled.

```
Console commands
01: netsh interface ipv6 6to4 set state enabled
```

Fig.12. Configuration of PC1 and PC2 for 6to4 (Windows).

5) NAT64: The testbed for NAT64 is depicted in Fig. 13. In this case PC1 is an IPv6-only host, and PC2 is an IPv4-only host. The router R1 is in charge of translating the headers to enable the communication between the two networks. We chose 192.168.255.0/24 for the dynamic IPv4 pool and 2001:db8: ffff: /96 for the designated IPv6 prefix. In this testbed, we only use Debian as operating system, because no suitable implementation of NAT64 was found for Windows. Two implementations of NAT64 were installed in R1: TAYGA [44] and Jool [45].

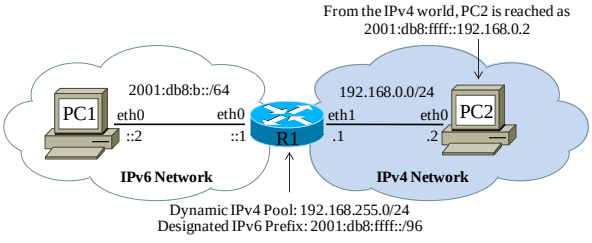

Fig.13. NAT64 Testbed.

TAYGA is an out-of-kernel stateless NAT64 implementation for Linux. We installed TAYGA v0.9.2 in R1 from its source code. Modified files and configuration commands in R1 can be observed in Fig. 14. Lines 01-02 create and setup the translation interface

(called *nat64*). The IPv4 and IPv6 addresses of the translation interface are indicated in line 03 and line 04, respectively. These addresses will be used by R1 when it requires to send an ICMP or ICMPv6 error message. Lines 05-06 route the dynamic IPv4 pool and the designated IPv6 pool into the NAT64 interface. Lines 07-08 activate the IPv4 and IPv6 forwarding. Finally, the TAYGA daemon is started in debugging mode in line 09.

```
File  /usr/local/etc/tayga.conf
tun-device nat64
ipv4-addr 192.168.0.1
ipv6-addr 2001:db8:b::1
prefix 2001:db8:ffff::/96
dynamic-pool 192.168.255.0/24
data-dir /var/db/tayga

Console commands
01: tayga --mktun
02: ip link set nat64 up
03: ip -4 addr add 192.168.0.1 dev nat64
04: ip -6 addr add 2001:db8: b: 1 dev nat64
05: ip -4 route add 192.168.255.0/24 dev nat64
06: ip -6 route add 2001:db8: ffff: /96 dev nat64
07: sysctl -w net.ipv4.conf.all.forwarding=1
08: sysctl -w net.ipv6.conf.all.forwarding=1
09: tayga -d
```

Fig.14. Configuration of R1 for TAYGA (Debian).

Jool is a Linux tool that allows users to work with SIIT (Stateless IP/ICMP Translation) or NAT64. To do so, Jool has two kernel modules (*jool_siit.ko* for SIIT and *jool.ko* for NAT64). Also, Jool provides two userspace applications to configure and query the state of the modules (*jool_siit* for SIIT and *jool* for NAT64). We compiled Jool v3.3.3 from the source code in R1. Fig. 15 gives the necessary commands for the compilation and configuration of Jool as a NAT64 server. Line 01 installs the dependency, in this case the kernel header files. Then, Jool is downloaded from its home page (line 02), before been unzipped (line 03). Compilation and installation is done in lines 04-05. Finally, the NAT64 module is loaded into memory in line 06, specifying the correct dynamic IPv4 pool and designated IPv6 prefix.

```
Console commands
01: apt-get install linux-headers-$(uname -r)
02: wget https://www.jool.mx/download/Jool-3.3.3.zip
03: unzip Jool-3.3.3.zip
04: cd Jool-3.3.3/mod
05: make && make modules_install && depmod
06: modprobe jool pool4=192.168.255.0/24 pool6=2001:db8: ffff: /96
```

Fig.15. Configuration of R1 for Jool (Debian).

V. Results and Analytical Comparison

In this section we show the results obtained by our OWD and throughput measurements in the testbeds presented in Section IV. First we introduce the results for OWD, followed by the results for throughput.

A. Performance Results for One Way Delay

For measuring OWD, the tool developed by Velásquez and Gamess [42] was used. Theoretically, the OWD of native IPv4 and native IPv6 represents the lower bound OWD for each IP version. For reasons of space, results are presented only for TCP in the case of Ethernet, and for UDP and TCP in the case of Fast Ethernet.

1) *Ethernet:* Fig. 16 shows the OWD for TCP segments over Ethernet using Debian. For almost all payload sizes, ISATAP and 6to4 have the highest OWD since these technologies add an additional IPv4 header to the IPv6 packets, resulting in more processing and serialization time than the other technologies. For NAT64 in Debian, Jool and TAYGA have similar measurements, where there is a small tendency of a lower OWD for Jool. It is worth to observe that NAT64 has a lower OWD than IPv6 in almost all cases despite of the translation. This is due to the high computational power of the PCs and the slow serialization of Ethernet (10 Mbps), hence the translation is done quickly, but the serialization of the segments is costly, resulting in a higher OWD for IPv6 where two serializations of a IPv6 packet are required, against NAT64 where one IPv6 and one IPv4 packet serializations are needed.

In Fig. 17, Fig. 18, and Fig. 19, the OWD for TCP segments over Ethernet is depicted for Windows 7, Windows 8, and Windows 10, respectively. The behavior of the measurements is similar to the one of Debian: OWD of IPv4 and IPv6 are lower than the OWD of ISATAP and 6to4. At the level of the operating systems, we can observe that Windows 8 is performing better than the other ones for almost all the payload sizes.

2) *Fast Ethernet:* OWD measurements for UDP in Fast Ethernet can be observed in Fig. 20, Fig. 21, Fig. 22, and Fig. 23, for Debian, Windows 7, Windows 8, and Windows 10, respectively. Native IPv4 has the lowest OWD, closely followed by native IPv6. ISATAP and 6to4 have a similar OWD for almost all payload sizes, which is higher than the ones shown by native IPv4 and native IPv6. In the case of NAT64 in Debian, Jool shows a normal behavior. On the other hand, TAYGA has an unexpected high OWD for payload sizes greater than 1750 bytes. At the level of the operating systems, there are minor differences between the OWD for UDP.

OWD measurements for TCP in Fast Ethernet are shown in Fig. 24, Fig. 25, Fig. 26, and Fig. 27 for Debian, Windows 7, Windows 8, and Windows 10, respectively. The global behavior of the OWD for TCP for Fast Ethernet in the studied transition mechanisms is almost identical to the OWD of TCP for Ethernet (see Fig. 16, Fig. 17, Fig. 18, and Fig. 19), but ten times faster. As we can see, native IPv4 has the lowest OWD, followed by native IPv6. The measurements of ISATAP and 6to4 are similar to each other, and higher than native IPv4 and

native IPv6. In the case of NAT64 in Debian, Jool has a lower OWD than TAYGA, however no surprising behavior is shown by TAYGA in this case. At the level of the operating systems, the lowest OWD for TCP is obtained by Windows 7 for almost all payload sizes.

B. Performance Results for Throughput

As mention before, throughput was measure using Iperf [43]. Theoretically, the throughput of native IPv4 and native IPv6 represent the upper bound for each IP version. We measured the throughput using Ethernet and Fast Ethernet. For a limitation of Iperf already mentioned in Section IV, results are only presented for UDP.

1) *Ethernet:* The UDP throughput measurements for Ethernet are depicted in Fig. 28, Fig. 29, Fig. 30, and Fig. 31, for Debian, Windows 7, Windows 8, and Windows 10, respectively. As expected, native IPv4 throughput represents the upper bound, followed by native IPv6 throughput. Then, the throughput of ISATAP and 6to4 is always lower than the throughput of IPv6, and has a similar behavior. This is expected, as these tunneling techniques add an IPv4 header to the IPv6 packets to travel the IPv4 network, which causes an additional processing and serialization overhead. In the case of NAT64 in Debian, the throughput of Jool and TAYGA are similar for almost all payload sizes. At the level of the operating systems, we can see that Debian outperforms the other ones for almost all payload sizes.
2) *Fast Ethernet:* Fig. 32, Fig. 33, Fig. 34, and Fig. 35 show the UDP throughput measures for Fast Ethernet in Debian, Windows 7, Windows 8, and Windows 10, respectively. As we can see, the global behavior of the throughput of UDP for Fast Ethernet for the studied transition mechanisms is almost identical to the throughput of UDP for Ethernet (see Fig. 28, Fig. 29, Fig. 30, and Fig. 31), but ten times faster. As we can observe, native IPv4 has the higher throughput, followed by native IPv6. The measurements of the tunneling techniques (ISATAP and 6to4) are similar to each other, and smaller than native IPv4 and native IPv6. At the level of the operating systems, as in Ethernet, Debian outperforms the other ones for almost all payload sizes.

VI. Conclusions and Future Work

In this paper, several testbeds were proposed to do a performance evaluation of native IPv4, native IPv6, and different transition techniques. We chose ISATAP, 6to4, and NAT64 due to their popularity, their different range of applications, and because several implementations of these transition technologies are available. The tests were performed with different operating systems: Debian, Windows 7, Windows 8, and Windows 10. We measured the OWD and the throughput at the level of UDP and TCP.

Similarly to other studies [6][26], our research confirmed that native IPv4 has a better performance than native IPv6 in controlled testbeds. This is due to the length of the IP headers (20 bytes for IPv4, and 40 bytes for IPv6). Also, our experiments showed that generally, ISATAP and 6to4 have similar network performances, encouraging network administrators to choose the one that better suit their needs, or to mix both technologies if required. For translation mechanisms, NAT64 is now the de facto standard. Since it is still a recent and evolving technology, NAT64 is supported just by a few manufacturers, such as Cisco Systems in new routers. In this paper, we compared Jool and TAYGA, two open source implementations of NAT64 for Linux. In this case, our recommendation to network administrators is to use Jool because it did not show the performance problems depicted in Fig. 20, and it is an active project. The last version of TAYGA was released in June 2011, and it seems to be a dead project now.

The studies presented in this paper were done in small testbeds to compare the performance of different transition mechanisms in a controlled environment. For further studies, we plan to consider more complex and realistic networks. We are also interested in developing analytical models for the performance evaluation of some transition technologies.

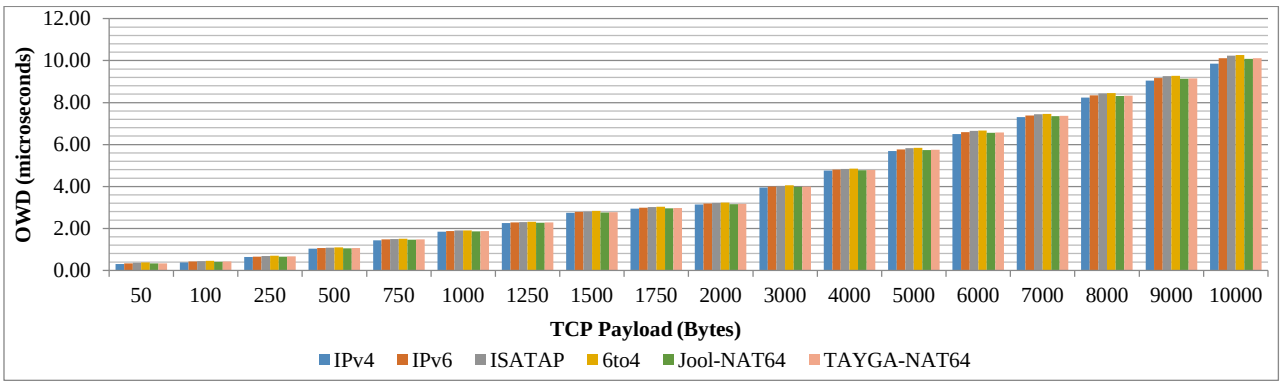

Fig.16. OWD for Ethernet with TCP – Debian.

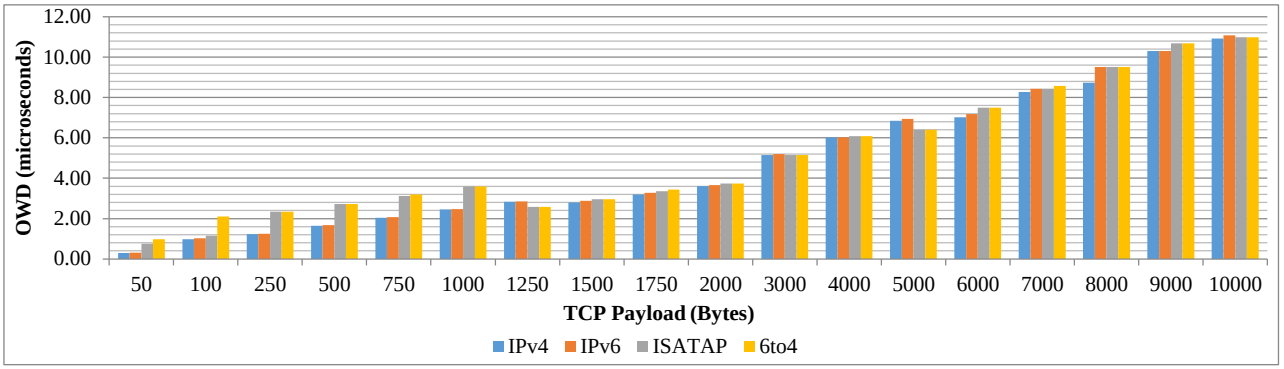

Fig.17. OWD for Ethernet with TCP – Windows 7.

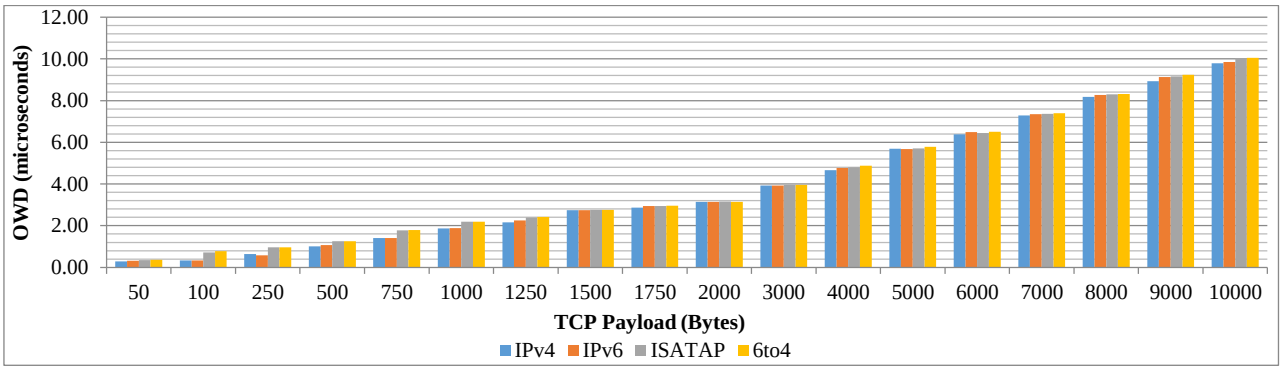

Fig.18. OWD for Ethernet with TCP – Windows 8.

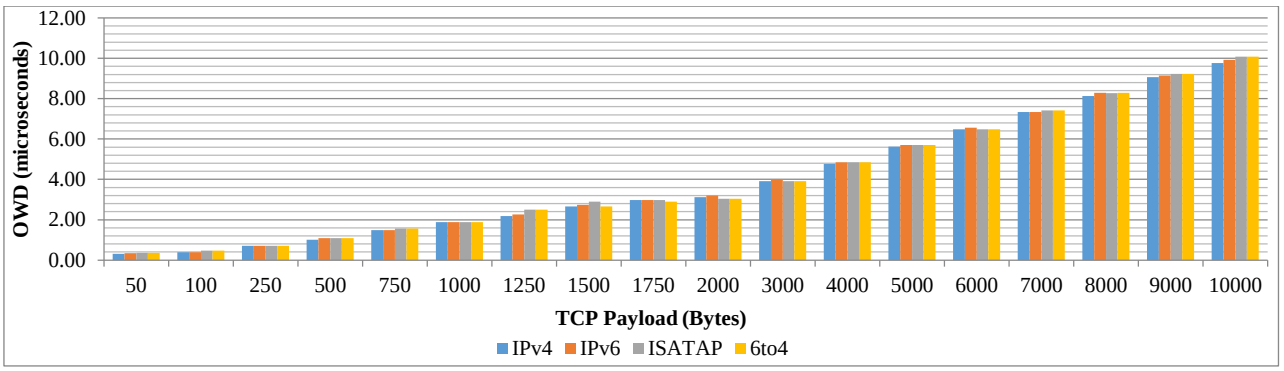

Fig.19. OWD for Ethernet with TCP – Windows 10.

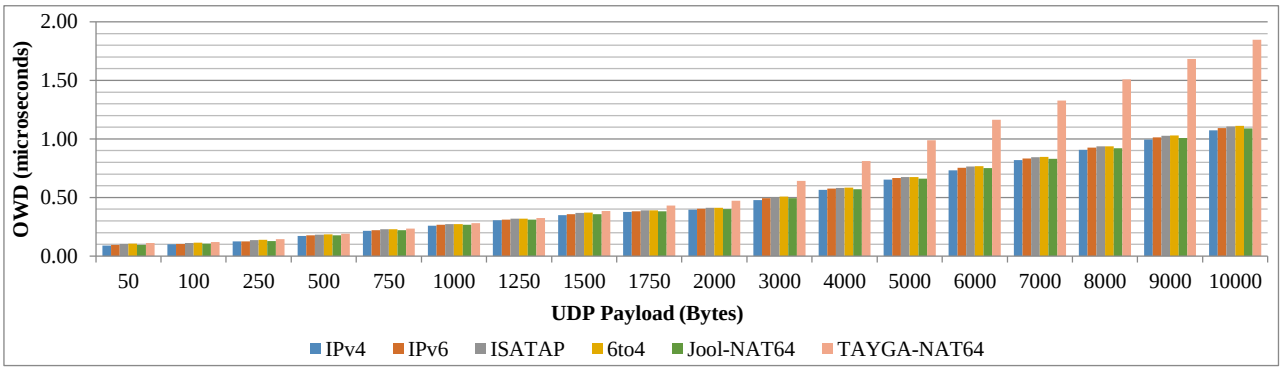

Fig.20. OWD for Fast Ethernet with UDP – Debian.

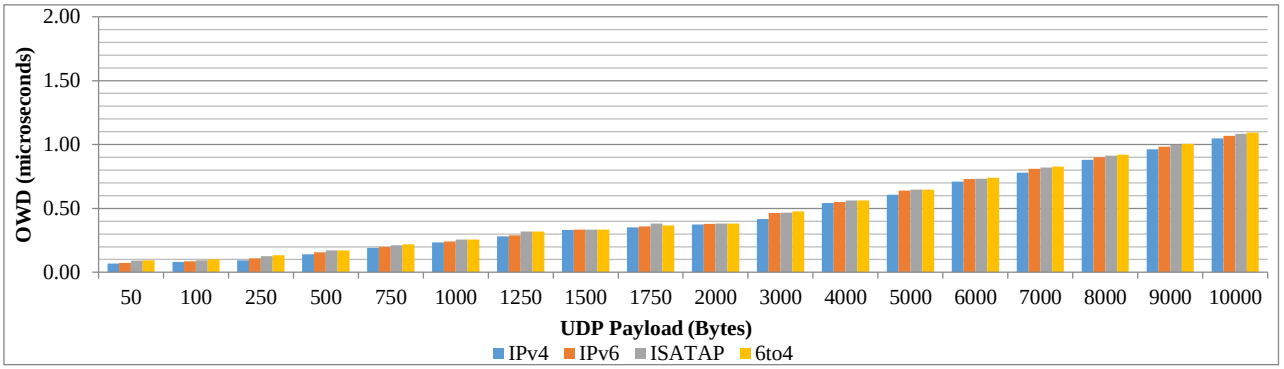

Fig.21. OWD for Fast Ethernet with UDP – Windows 7.

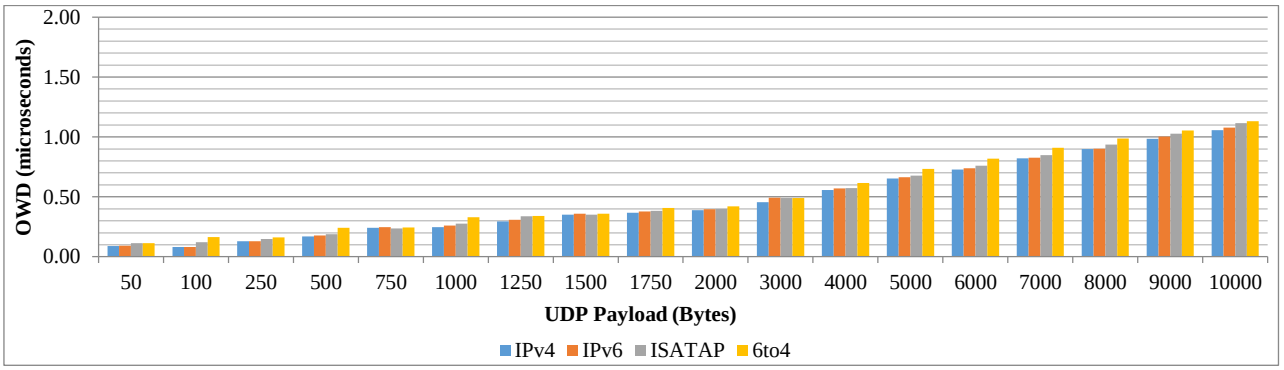

Fig.22. OWD for Fast Ethernet with UDP – Windows 8.

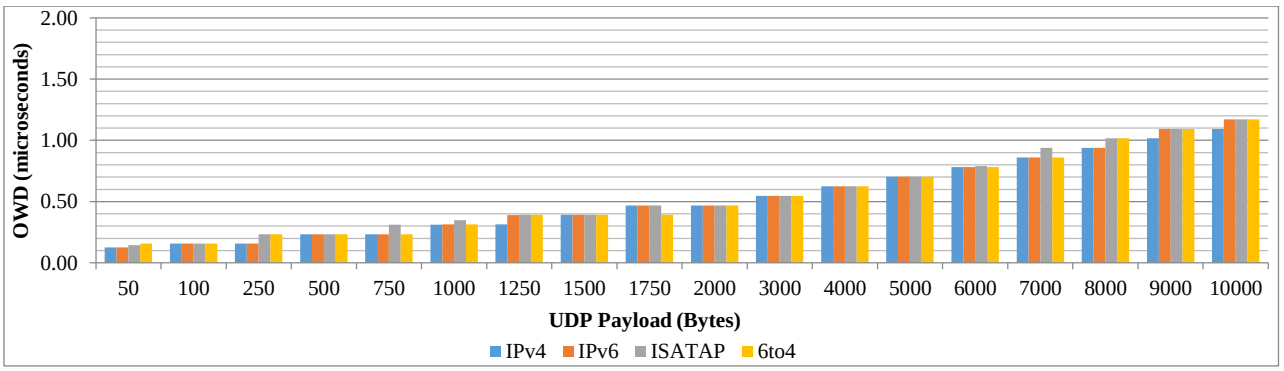

Fig.23. OWD for Fast Ethernet with UDP – Windows 10.

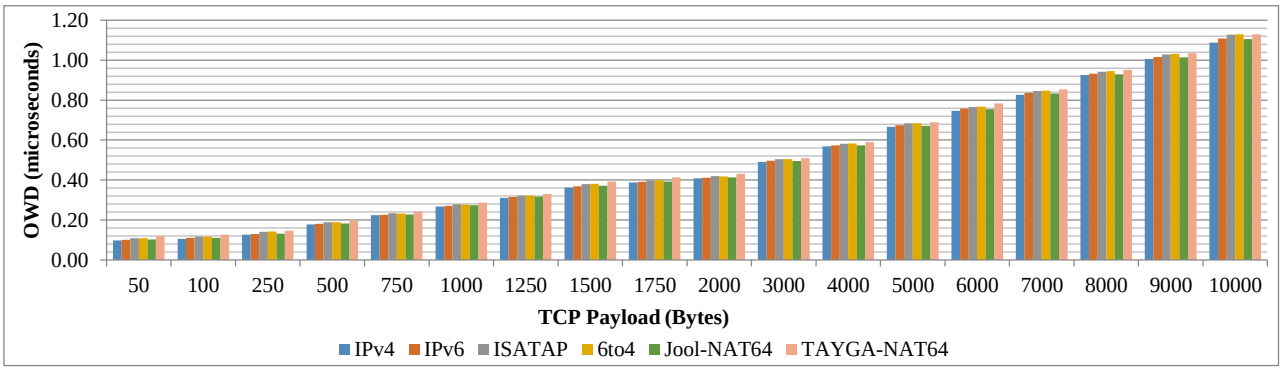

Fig.24. OWD for Fast Ethernet with TCP – Debian.

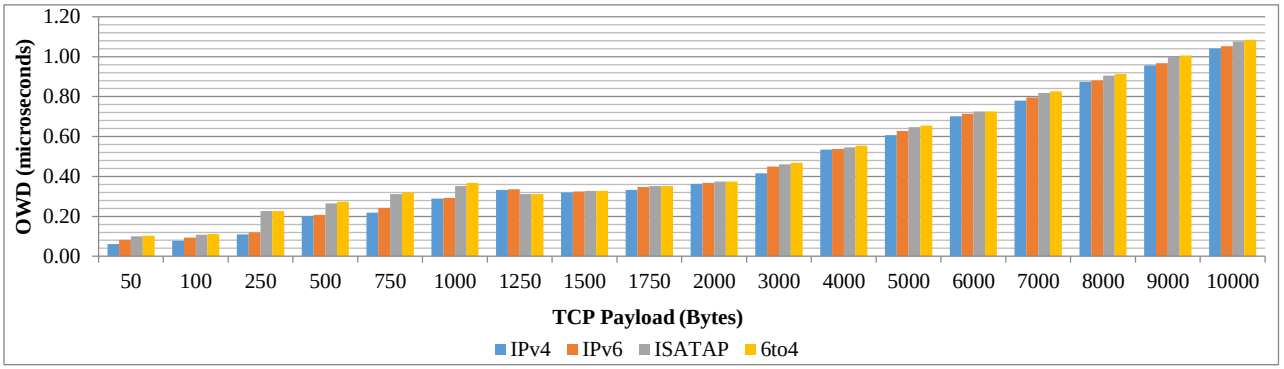

Fig.25. OWD for Fast Ethernet with TCP – Windows 7.

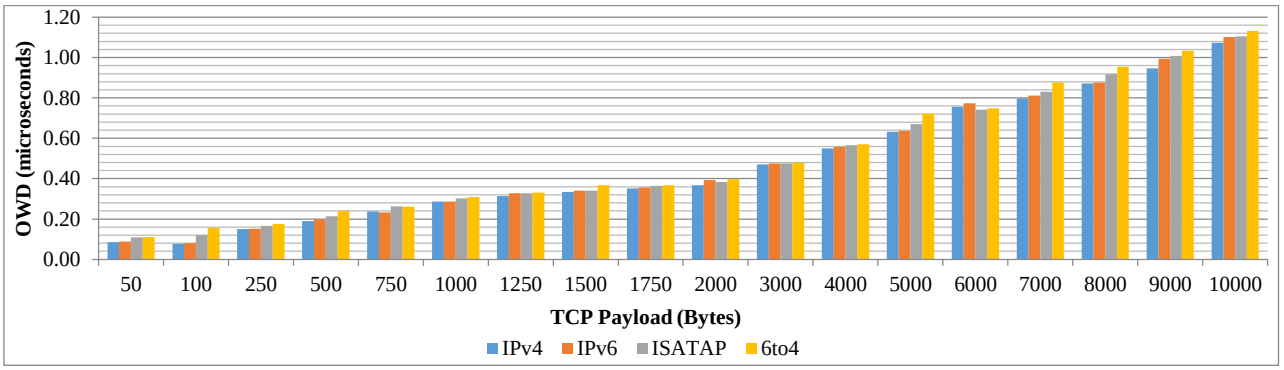

Fig.26. OWD for Fast Ethernet with TCP – Windows 8.

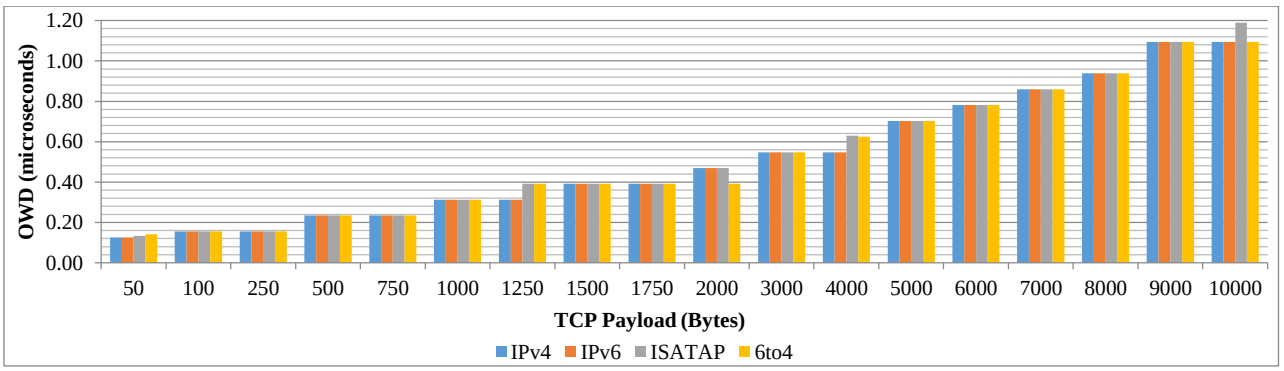

Fig.27. OWD for Fast Ethernet with TCP – Windows 10.

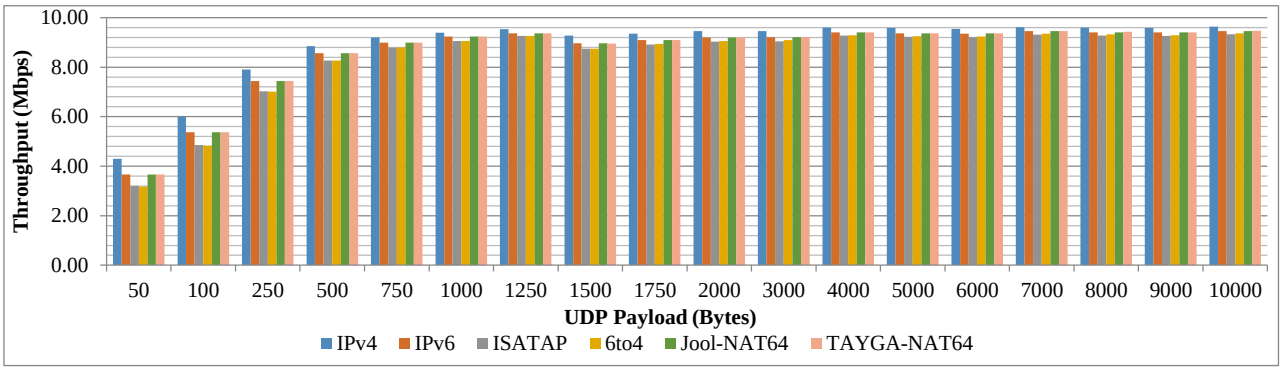

Fig.28. Throughput for Ethernet with UDP – Debian.

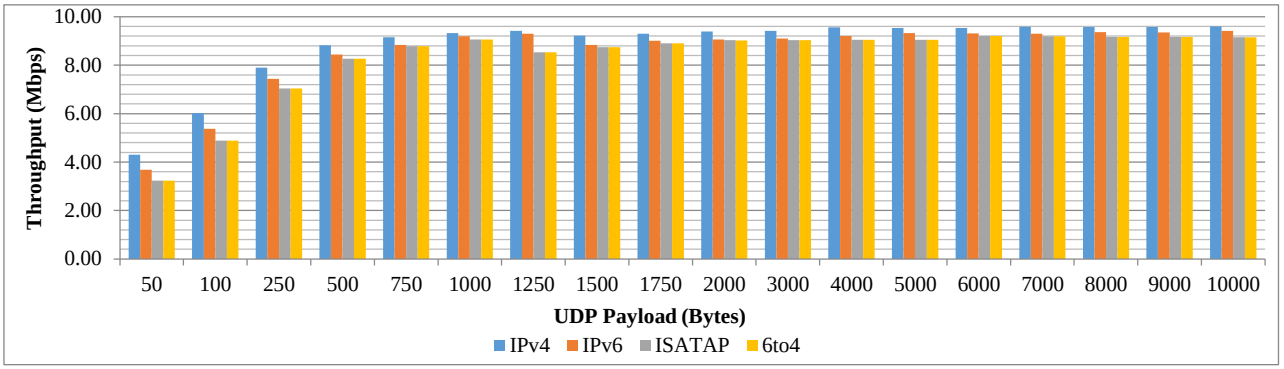

Fig.29. Throughput for Ethernet with UDP – Windows 7.

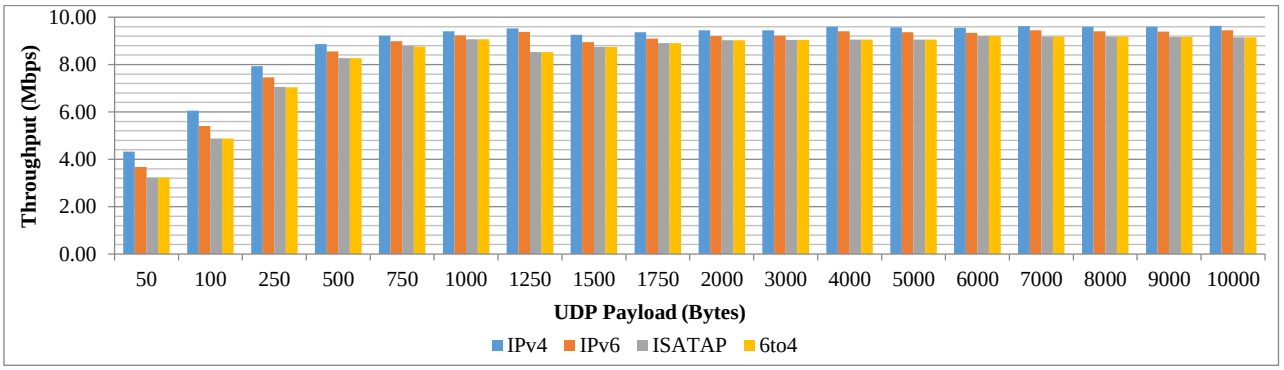

Fig.30. Throughput for Ethernet with UDP – Windows 8.

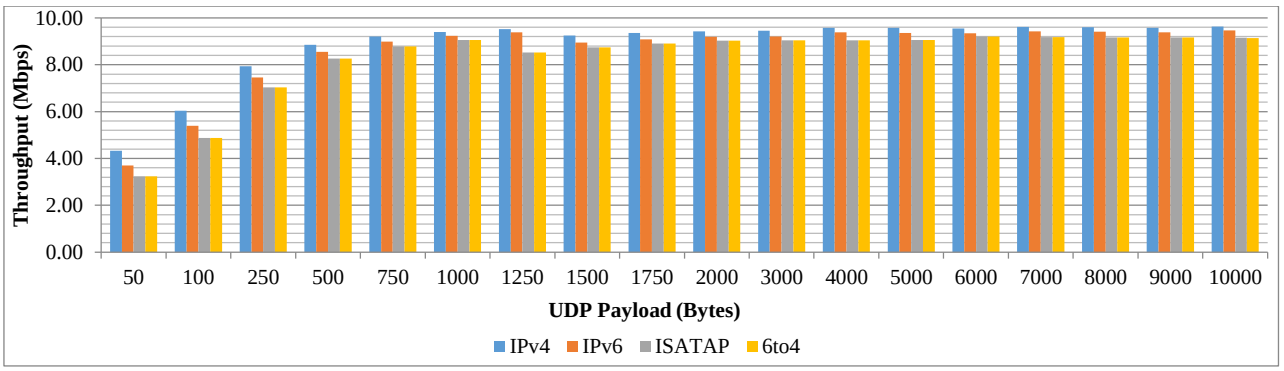

Fig.31. Throughput for Ethernet with UDP – Windows 10.

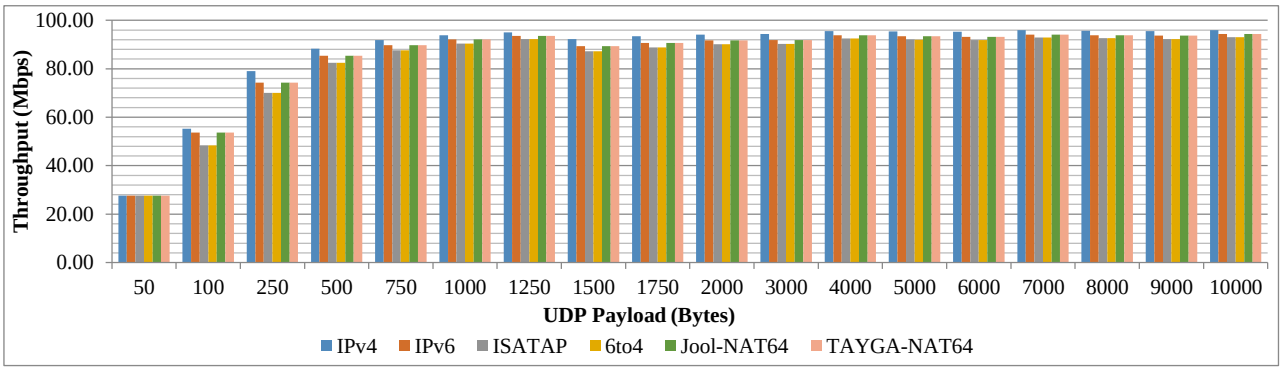

Fig.32. Throughput for Fast Ethernet with UDP – Debian.

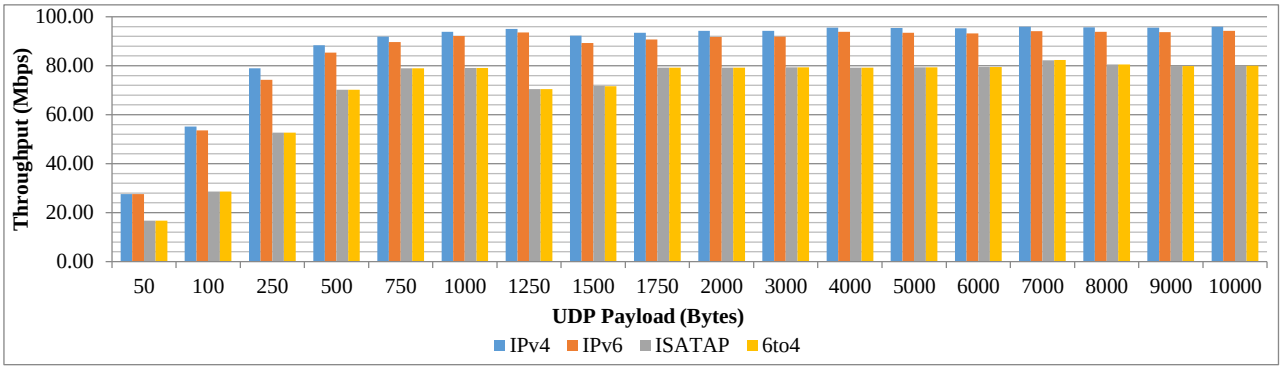

Fig.33. Throughput for Fast Ethernet with UDP – Windows 7.

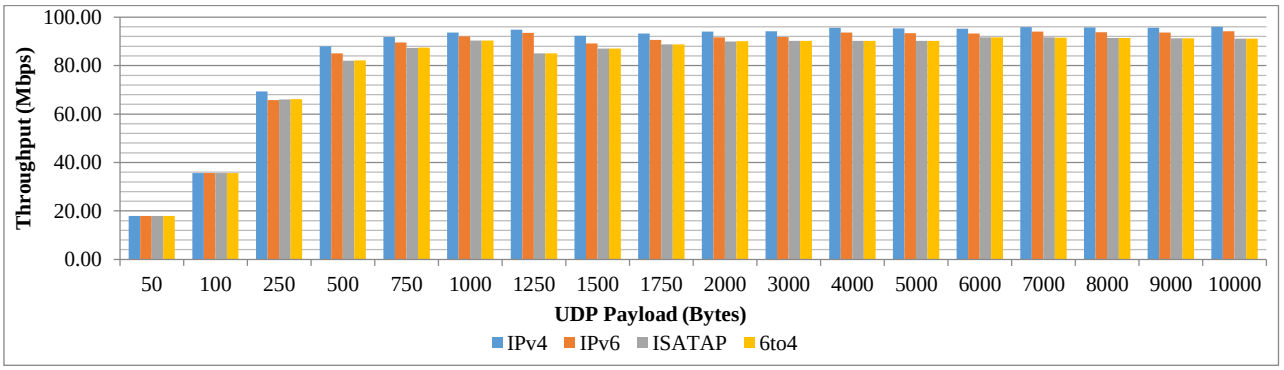

Fig.34. Throughput for Fast Ethernet with UDP – Windows 8.

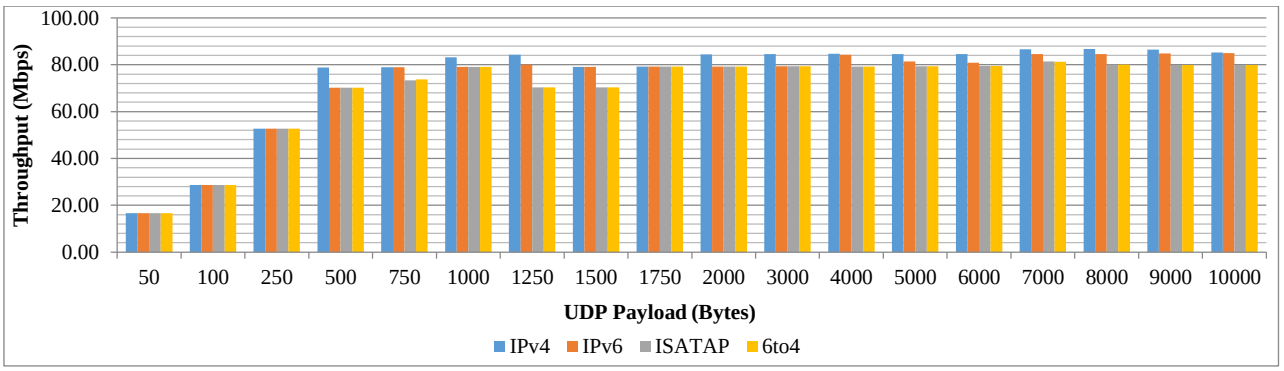

Fig.35. Throughput for Fast Ethernet with UDP – Windows 10.

ACKNOWLEDGMENT

We want to thank the CDCH-UCV (Consejo de Desarrollo Cient fico y Human stico) which partially supported this research under grant number: PG 03-8066-2011/1.

REFERENCES

[1] P. Srisuresh and K. Egevang, "Tradicional IP Network Address Translator (Traditional NAT)," IETF, RFC 3022, January 2001.

[2] J. Davies, *Understanding IPv6*, 3rd ed., Microsoft Press, June 2012.

[3] L. Colitti, S. H. Gunderson, E. Kline, and T. Refice, "Evaluating IPv6 Adoption in the Internet," in *Proceedings of the 11th International Conference on Passive and Active Measurement (PAM'10)*, Zurich, Switzerland, April 2010, pp. 141–150.

[4] kc Claffy, "Tracking IPv6: Data We Have and Data We Need," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 3, pp. 43–48, July 2011.

[5] S. Zeadally and I. Raicu, "Evaluating IPv6 on Windows and Solaris," *IEEE Internet Computing*, vol. 7, no. 3, pp. 51–57, May 2003.

[6] E. Gamess and R. Sur ós, "An Upper Bound Model for TCP and UDP Throughput in IPv4 and IPv6," *Journal of Network and Computer Applications*, vol. 31, no. 4, pp. 585–602, November 2008.

[7] S. Narayan, P. Shang, and N. Fan, "Performance Evaluation of IPv4 and IPv6 on Windows Vista and Linux Ubuntu," in *Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC'09)*, vol. 1, Wuhan, China, April 2009, pp. 653–656.

[8] S. Narayan, P. Shang, and N. Fan, "Network Performance Evaluation of Internet Protocols IPv4 and IPv6 on Operating Systems," in *Proceedings of the 6th International Conference on Wireless and Optical*

Communications Networks (WOCN'09), Cairo, Egypt, April 2009, pp. 1–5.

[9] S. S. Kolahi, B. K. Soorty, Z. Qu, and N. Chand, "Performance Analysis of IPv4 and IPv6 on Windows Vista and Windows XP over Fast Ethernet in Peer-peer LAN," in *Proceedings of the 3rd International Conference on New Technologies, Mobility and Security (NTMS'09)*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 450–453.

[10] J. Balen, G. Martinovic, and Z. Hocenski, "Network Performance Evaluation of Latest Windows Operating Systems," in *Proceedings of the 20th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, vol. 7, Split, Croatia, September 2012, pp. 1–6.

[11] P. Svec and M. Munk, "IPv4/IPv6 Performance Analysis: Transport Layer Protocol Impact to Transmission Time," in *Proceedings of International Conference on Internet Technology and Applications (iTAP 2011)*, Wuhan, China, August 2011, pp.1–4.

[12] E. Gamess and K. Velásquez, "IPv4 and IPv6 Forwarding Performance Evaluation on Different Operating Systems," in *Proceedings of the XXXIV Latin American Computing Conference (CLEI'08)*, Santa Fe, Argentina, September 2008.

[13] S. Narayan, S. S. Sodhi, P. R. Lutui, and K. J. Vijayakumar, "Network Performance Evaluation of Routers in IPv4/IPv6 Environment," in *Proceedings of the 2010 IEEE International Conference on Wireless Communications, Networking, and Information Security (WCNIS)*, Beijing, China, June 2010.

[14] S. S. Kolahi, Z. Qu, B. K. Soorty, and N. Chand, "The Impact of Security on the Performance of IPv4 and IPv6 using 802.11n Wireless LAN," in *Proceedings of the 3rd International Conference on New Technologies, Mobility and Security (NTMS'09)*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 454–457.

[15] S. S. Kolahi, Z. Qu, B. K. Soorty, and N. Chand, "The Performance of IPv4 and IPv6 Using DP on IEEE 802.11n WLANs with WPA2 security," in *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human (ICIS'09)*. New York, NY, USA: ACM, 2009, pp. 873–876.

[16] H. Fahmy and S. Ghoneim, "Performance Comparison of Wireless Networks over IPv6 and IPv4 under Several Operating Systems," in *Proceedings of the IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS'13)*, Abu Dhabi, UAE, December 2013, pp. 670–673.

[17] M. Ahmed, M. A. Suhaimi, Q. S. Md. Faisal, and S. Haseeb, "Evaluating QoS Performance of Streaming Video on Both IPv4 and IPv6 Protocols," in *Proceedings of the 2007 Spring Simulaiton Multiconference - Volume 1 (SpringSim'07)*. San Diego, CA, USA: Society for Computer Simulation International, 2007, pp. 109–116.

[18] S.-M. Huang, Q. Wu, and Y.-B. Lin, "Tunneling IPv6 through NAT with Teredo Mechanism," in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications*, vol. 2, Taipei, Taiwan, March 2005, pp. 813–818.

[19] M. Aazam, S. A. H. Shah, I. Khan, and A. Qayyum, "Deployment and Performance Evaluation of Teredo and ISATAP over Real Test-bed Setup," in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems (MEDES'10)*, Bangkok, Thailand, October 2010, pp. 229–233.

[20] S. Zander, L. L. H. Andrew, G. Armitage, G. Huston, and G. Michaelson, "Investigating the IPv6 Teredo Tunnelling Capability and Performance of Internet Clients," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 5, pp. 13–20, October 2012.

[21] S. Narayan and S. Tauch, "Network Performance Evaluation of IPv4-v6 Configured Tunnel and 6to4 Transition Mechanisms on Windows Server Operating Systems," in *Proceedings of the 2010 International Conference on Computer Design and Applications (ICCDA)*, vol. 5, Qinhuangdao, China, June 2010.

[22] S. Narayan and S. Tauch, "IPv4-v6 Configured Tunnel and 6to4 Transition Mechanisms Network Performance Evaluation on Linux Operating Systems," in *Proceedings of the 2nd International Conference on Signal Processing Systems (ICSPS)*, vol. 2, Dalian, China, October 2010.

[23] D. Hadiya, R. Save, and G. Geetu, "Network Performance Evaluation of 6to4 and Configured Tunnel Transition Mechanisms: An Empirical Test-bed Analysis," in *Proceedings of the 6th International Conference on Emerging Trends in Engineering and Technology (ICETET'13),* Nagpur, India, December 2013, pp. 56–60.

[24] P. Amr and N. Abdelbaki, "Convergence Study of IPv6 Tunneling Techniques," in *Proceedings of the 10th International Conference on Communications (COMM'14),* Bucharest, Romania, May 2014, pp. 1–6.

[25] S.-J. Yoon, J.-T. Park, D.-I. Choi, and H. K. Kahng, "Performance Comparison of 6to4, 6rd, and ISATAP Tunnelling Methods on Real Testbeds," *International Journal on Internet and Distributed Computing Systems*, vol. 2, no. 2, pp. 149–156, July 2012.

[26] F. Sans and E. Gamess, "Analytical Performance Evaluation of Native IPv6 and Several Tunneling Technics Using Benchmarking Tools," in *Proceedings of the XXXIX Latin American Computing Conference (CLEI'13),* Naiguata, Venezuela, October 2013, pp 1-9.

[27] K. J. O. Llanto and W. E. S. Yu, "Performance of NAT64 versus NAT44 in the Context of IPv6 Migration," in *Proceedings of the International MultiConference of Engineers and Compuer Scientists (IMECS' 12)*, Hong Kong, China, March 2012, pp. 638–645.

[28] C. P. Monte, M. I. Robles, G. Mercado, C. Taffernaberry, M. Orbiscay, S. Tobar, R. Moralejo, and S. Pérez, "Implementation and Evaluation of Protocols Translating Methods for IPv4 to IPv6 Transition," *Journal of Computer Science & Technology*, vol. 12, no. 2, pp. 64–70, 2010.

[29] S.-Y. Yu and B. E. Carpenter, "Measuring IPv4 IPv6 Translation Techniques," Department of Computer Science, The University of Auckland, Tech. Rep. 2012-001, January 2012.

[30] E. Hodzic and S. Mrdovic, "IPv4/IPv6 Transition using DNS64/NAT64: Deployment Issues," in *Proceedings of the IX International Symposium on Telecommunications (BIHTEL'12)*, Sarajevo, Bosnia and Herzegovina, October 2012, pp. 1–6.

[31] G. Lencse and G. Takacs, "Performance Analysis of DNS64 and NAT64 Solutions," *Infocommunications Journal*, vol. 4, no. 2, pp. 29–36, 2012.

[32] G. Lencse and S. Repas, "Performance Analysis and Comparison of Different DNS64 Implementations for Linux, OpenBSD and FreeBSD," in *Proceedings of the IEEE 27th International Conference on Advanced Information Networking and Applications (AINA'13)*, Barcelona, Spain, March 2013, pp. 877–884.

[33] G. Lencse and S. Repas, "Performance Analysis and Comparison of the TAYGA and of the PF NAT64 Implementations," in *Proceedings of the 36th International Conference on Telecommunications and*

Signal Processing (TSP'13), Rome, Italy, July 2013, pp. 71–76.
[34] S. Repas, P. Farnadi, and G. Lencse, "Performance and Stability Analysis of Free NAT64 Implementations with Different Protocols," *Acta Technica Jaurinensis*, vol. 7, no. 4, pp. 404–427, 2014.
[35] M. Ahmed, A. Litchfield, S. Ahmed, A. Mahmood, Md. E. Hossain, "VoIP Performance Analysis over IPv4 and IPv6," *International Journal of Computer Network and Information Security*, vol.6, no.11, pp.43-48, October 2014.
[36] C. Aoun and E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status," IETF, RFC 4966, July 2007.
[37] F. Templin, T. Gleeson, and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)," IETF, RFC 5214, March 2008.
[38] B. Carpenter and K. Moore, "Connection of IPv6 Domains via IPv4 Clouds," IETF, RFC 3056, February 2001.
[39] C. Huitema, "An Anycast Prefix for 6to4 Relay Routers," IETF, RFC 3068, June 2001.
[40] M. Bagnulo, P. Matthews, and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," IETF, RFC 6146, April 2011.
[41] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators," IETF, RFC 6052, October 2010.
[42] K. Velásquez and E. Gamess, "A Survey of Network Benchmark Tools," *Machine Learning and System Engineering*, vol. 68, pp. 465–480, October 2010.
[43] NLANR/DAST. Iperf Homepage. http://iperf.fr.
[44] Nathan Lutchansky. TAYGA Homepage. http://www.litech.org/tayga.
[45] Tecnológico de Monterrey. Jool Homepage. http://www.jool.mx.

Authors' Profiles

Adira Quintero received a B.S. in Computer Science from the Central University of Venezuela, Venezuela, in 2015. Her research interests include Network Performance Evaluation, Data Security, and Voice over IP.

Francisco Sans received a B.S. in Computer Science from the Central University of Venezuela, Venezuela, in 2012 with magna cum laude. He is now a master student in Computer Science and will soon defend its thesis. He is currently working as a professor at Central University of Venezuela, Venezuela. His research interest includes Computer Networks, Computer Graphics, and Video Games.

Eric Gamess received a M.S. in Industrial Computation from the National Institute of Applied Sciences of Toulouse (INSA de Toulouse), France, in 1989, and a Ph.D. in Computer Science from the Central University of Venezuela, Venezuela, in 2010. He is currently working as a professor at Central University of Venezuela, Venezuela. Previously, he worked as a professor at University of Puerto Rico, Puerto Rico, and "Universidad del Valle", Colombia. His research interests include Vehicular Adhoc Networks, Network Performance Evaluation, IPv6, and Network Protocol Specifications. He is a member of the Venezuelan Society of Computing and has been in the organization committee of several national and international conferences.

How to cite this paper: Adira Quintero, Francisco Sans, Eric Gamess,"Performance Evaluation of IPv4/IPv6 Transition Mechanisms", International Journal of Computer Network and Information Security(IJCNIS), Vol.8, No.2, pp.1-14, 2016.DOI: 10.5815/ijcnis.2016.02.01

