Implementation and Evaluation of Protocols Translating Methods for IPv4 to IPv6 Transition

Cristian Pérez Monte, María Inés Robles, Gustavo Mercado, Carlos Taffernaberry,
Marcela Orbiscay, Sebastián Tobar, Raúl Moralejo y Santiago Pérez
Grupo UTN Gridtics - Departamento de Electrónica,
Universidad Tecnológica Nacional – Facultad Regional Mendoza
Mendoza, 5500, Argentina
{cristian.perez, ines.robles, gustavo.mercado, carlos.taffernaberry,
marcela.orbiscay, sebastian.tobar, raul.moralejo}@gridtics.frm.utn.edu.ar
www.gridtics.frm.utn.edu.ar

ABSTRACT

Today millions of computers are interconnected using the Internet Protocol version 4 (IPv4) and can not switch to the new version, IPv6, simultaneously. For this reason the IETF has defined a number of mechanisms for transitioning to the new protocol in a progressively and controlled manner. On the other hand, Internet Service Providers (ISP) will not have new IPv4 global addresses to offer in the near future due to the fact that these addresses will be exhausted [1]. A very interesting alternative for ISPs is to use IPv6 global addresses and, by some transitional method, access the current IPv4 backbone. This study aims to compare two methods of transparent access to the IPv4 Internet backbone, from networks that are "IPv6 only". To make the comparison, a software was developed, implementing an Application Layer Gateway (ALG), and Ecdysis was used to implement NAT64. Both trials used a network IPv6 Test Bed. This paper details the design principles and fundamental aspects of the ALG implementation, as well as the implementation of NAT64. Finally, we present the tests performed and conclusions drawn on the test platform.

Keywords: Internet, IPv6 Protocol, Transition Methods, ALG, NAT64, ISP.

1. INTRODUCTION

After 25 years, IPv4 begins to show signs of weakness. It can no longer provide adequate answers, especially regarding to the gradual exhaustion of IP addresses available, as measured in our region, will succeed half of 2014 [1]. The necessity of environments, like “Internet of Things” [2], expands nowadays the requirements of addresses. In 1992 the Internet Engineering Task Force (IETF), called the research community to study alternatives for IPv4. The result arose in 1995 and was called Internet Protocol version 6 (IPv6) [3].

One of the most important steps, in the adoption of IPv6, is the "Transition" from IPv4 to IPv6. Jordi Palet said "Since IPv6 is a new protocol, it is not compatible with IPv4, and therefore IPv6 has been designed considering a long period of transition and co-existence between them" [4]. Although for a complete transition is necessary that the current backbone switch to IPv6, it is also true that end users and ISPs can begin to implement the protocol.

In this aspect the present work is developed, allowing a final network "IPv6 only" connects to Internet IPv4 and IPv6 using transition techniques. This will provide experience and training in the transition from IPv4 to IPv6.

The following section details some of the most used transition mechanisms. Section 3 presents the scenario and the problem to solve through these mechanisms. Section 4 discusses which one is best technique for this scenario. Throughout Section 5 is developed and tested an ALG. Section 6 shows details of the implementation of a NAT64, while Section 7 makes an evaluation and comparison of both methods. Finally, in section 8, valuable conclusions are obtained.

2. TRANSITION MECHANISM OVERVIEW

IPv6 is now widely available for most operating systems in hosts and routers, and not only in the ISP networks [5]. To communicate with other IPv6 systems, is essential to have access to the global IPv6 Internet. The practical facts show a co-existent between IPv4 and IPv6, in an intermediate transition state. Expanding IPv6 functionality from a small to a large network infrastructure can be a difficult and complex adventure. For a large site, the different requirements and conditions make it necessary to employ various mechanisms depending on the specific transition.

Two widely used methods are "mechanism of dual stack" and "tunnelling techniques", but in this work we will implement and evaluate methods of "translation". We will do a brief introduction in the following paragraphs about that.

Translating Protocols

Translation methods were developed to achieve communication between IPv4-only and IPv6-only;

such as:

- Stateless IP/ICMP Translator (SIIT) [6] and Network Address Translation - Protocol Translation (NAT-PT) [7] are mechanisms, unlike the tunnels, which translate IPv4 headers to IPv6 and vice versa. These techniques share the same problems of NAT and must deal with the semantics of converting the fields successfully. In some cases, during the conversion process, header information is lost. For this reason, the IETF recommends these methods only as a last resort.

- Bump In the Stack (BIS) [8] is an approach similar to the previous SIIT, but implemented directly in the operating system on each host (between the TCP/IP and network driver). It is only available for IPv4 applications and IPv6 networks. It is a complex implementation and rarely used.

- Bump in the API (BIA) [9] adds an API translation between the Socket API and TCP/IP stack, allowing an upgrade to BIS method in terms of the dependence of the network driver, but has the same limitations as BIS.

-Transport Relay Translator (TRT) [10] is a protocol conversion at the transport layer level based on a DNS proxy. It receives queries from IPv6 hosts and if the required name is associated with IPv4 address, it returns an IPv6 address composed with a prefix IPv6 format (64 bits) + zeros (32 bits) + "IPv4address" (32 bits). This method was replaced by NAT64.

- NAT 64 [11] consists of a server with at least one public IPv4 address and an IPv6 segment with a /96 prefix (eg 64: ff9b :: / 96). In the case of connecting to an IPv6 address, the client builds the IPv6 destination address using the previous range of 96 bits plus 32 bits of the IPv4 address wich want communicate to, sending packets to the resulting address. The NAT 64 server then creates a NAT mapping between IPv6 and IPv4 addresses, enabling communication. It is also necessary to use DNS 64.

- DNS 64: [12] When a DNS server, with DNS64 functionality, receives a request for domain AAAA record, but only has A records, create a AAAA records from these A records. The first portion of the IPv6 address created points to a IPv6/IPv4 translator, and the second includes the IPv4 address of the A register. The translator usually is a NAT64 server.

- ALG is a translation made in the application layer. There is no specific RFC for that, therefore its implementation depends on the application layer protocol that will be supported.

3. TEST SCENARIO

Figure 1 shows the scenario implemented to evaluate the transition methods.

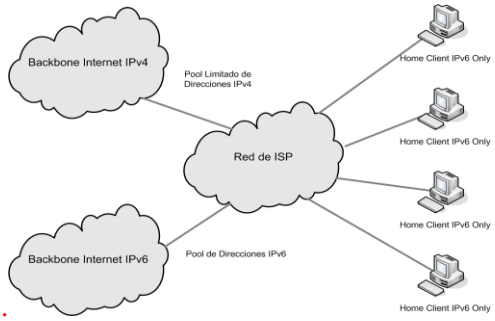

Figure 1. Common scenario for ISPs today

The proposed topology consists of several "home clients" hosts that constitute a network of customers of an ISP, configured using only native IPv6. The ISP has both IPv4 and IPv6 connectivity. The aim is to enable "home clients" to access servers and services available in Internet v4 without requiring changes in their hosts; either by installing dual stack, tunneling or by configuring protocol translation. Notice that the ISP has no more new IPv4 addresses, so only IPv6 addresses can be delivered on the customers.

4. TRANSITION METHODS EVALUATION

First, to achieve the objective should be to implement some of the techniques listed in paragraph 2.3, since communication is exclusively between IPv4 only hosts and IPv6 hosts only, ruling out dual-stack techniques or tunneling.

The following alternatives were analyzed:

- The application of SIIT and NAT-PT is discarded due to the normal problems of NAT and the possible loss of header information [13].

- To use BIS or BIA is necessary to modify the client's operating systems. Problems will be found for operating systems that do not have the source code available.

- The alternative of a TRT is feasible, but is obsolete.

- NAT64 is heavily used, even was find a free implementation available for testing. One drawback is that requires a DNS Proxy (DNS64) specially configured to work properly.

- The implementation of ALG is also viable, if it is not taken into account the decline in performance, by doing all the conversion in the application layer.

Taking into account the considered aspects, we chose NAT64 and ALG for evaluation of functionality and performance [14].

5. APPLICATION LAYER GATEWAY

An ALG for HTTP/HTTPS protocols was implemented. It justified by the ease of implementation of ALG and they are not necessary additional elements, such as a DNS Proxy or the source code of the OS. Due there is almost no difference between a proxy and an ALG application, initially was tried to use the known HTTP/HTTPS proxy called Squid. But at that moment it didn't have support IPv6, so finally we decided to perform our own application to meet the target.

Design Proposal

A proposal of the ALG method is shown in Figure 2.

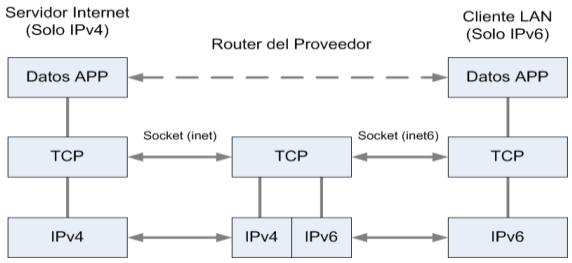

Figure 2. Architecture Diagram of ALG

The basic idea of an ALG is to allow the ISP router be the responsible for exchanging information between the two extremes. It needs to have dual stack and run the application ALG.

The home clients initiate communication using an INET6 socket, and make a HTTP solicitation to the ALG, which will be stored in a buffer. The ALG, using an INET socket, starts a new connection as a client, to the requested site, forwarding the original HTTP request previously stored. The response of the service requested will be forwarded by the ALG to the home client. The application should resolve the domain name applied for, before sending the request to Internet v4.

Implementation

It was performed a prototype to evaluate the proper functionality of this mechanism. The programming was done using Python. Below, the most relevant portions of code are shown:

```
   #Main
  def listen (self):
      escucha = socket(AF_INET6,SOCK_STREAM)#IPv6
Only
      escucha.bind(self.ADDR6,self.PORT)
      escucha.listen(10)        #hasta 10 a la
espera
      while True:
          interno,cliente = escucha.accept()
          pid = os.fork()
          if pid != 0 :         #proceso hijo
              self.servicio()
          else:                 #proceso padre
              interno.close()

  def servicio (self):
      Pedido = interno.recv(self.buffer)
      externo = socket(AF_INET,SOCK_STREAM)# a
InternetV4
      externo.connect(res[0][4][:2])
      externo.send(Pedido) #reenvio requerimiento
      RespInternet = ''
      while RespInternet <> ''  #lee IPv4 ->
escribe IPv6
          RespInternet =
externo.recv(self.buffer)
          interno.send(RespInternet)
      interno.close()           #Termino el envio
de IPV4
      externo.close()
      sys.exit()
```

The "listen" function, creates an AF_INET6 (IPv6) socket and waits for a home client to connect, using the escucha.accept() method. Once connected, by calling os.fork(), a children is created, serving each home client, using the self.servicio() method. The "service" function stores in a local variable "Pedido" the client's original request. Then, using the socket API creates an AF_INET(IPv4) socket and connects as a client with the server which the request was for, by calling externo.send (Pedido).

Once the response arrives, using the externo.recv(self.buffer) method, is forwarded to the IPv6 socket used by the original home client. The router on which the method was tested was a GNU / Linux distribution Ubuntu 9.04. Windows XP was chosen as home client with IPv6 support only, being the most widespread operating system. However it can be used other operating systems like GNU / Linux, Solaris, Mac OS or Win Vista. Successful tests were also done with a cell phone Nokia N95 with Symbian OS. In all cases, the IPv4 stack was disabled. It was set the proxy in the HTTP client application (browser), the IPv6 address of local router and the port where the ALG was listening the home clients.

Because this method only allows access to transition IPv4 Internet servers, an improvement was made to give access to Internet servers also IPv6, transparently to the end user. In this work it was only evaluated the IPv6/IPv4 translation, so this feature is not used, even though the Figure 3 shows it.

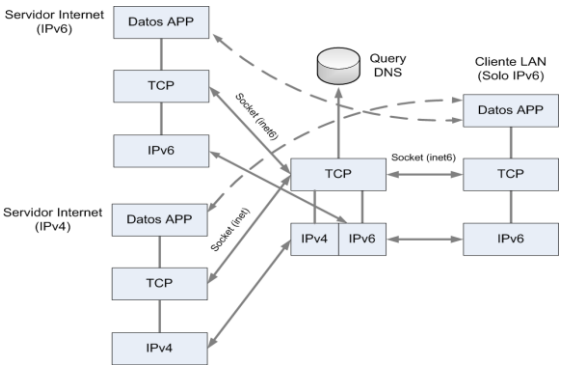

Figure 3. Final diagram of ALG

6. IMPLEMENTATION OF NAT64

The NAT64 was installed on the ISP dual stack router with access to Internet v6 and Internet v4. In addition, there were configured two LAN links. The first, with both IPv4 and IPv6 connectivity, was used by the server that performs the task NAT64 + DNS64. The second LAN link, with IPv6 only addresses, was located in the home clients. It was set a default route to the server NAT64 + DNS64 for the /96 network assigned to the NAT64. It was elected a public range for the /96, not acording with RFC 6146, for their use as public NAT64 in remote networks.

The DNS64 + NAT64 server was implemented on Fedora 14 Linux operating system. The Ecdysis-nf-NAT64[15] was installed to work as a NAT64 server and Ecdysis-unbound, to implement the DNS64 server. Static IPv4 and IPv6 addresses were assigned to the interfaces. Also the default routes and the default route to the NAT64 interface for the NAT64's network. Finally, the client's addresses a default route was assigned by autoconfiguration. Through DHCPv6, the DNS server corresponding to the IPv6 server DNS64 + NAT64 was assigned. The Figure 4 shows the topology of the implementation performed.

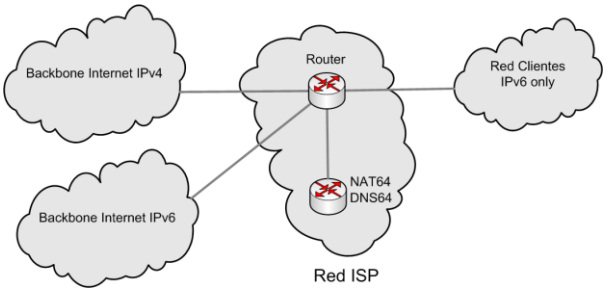

Figure 4. NAT64 topology

7. EVALUATION OF TRANSLATION METHODS

The important thing for home clients is the ability to access Internet services, in a transparent and secure way. The "IPv6 only" home clients must have at least the same functionality as IPv4 clients. The evaluation of the mechanisms examined in this paper, aims to verify if the majority of services have a right functionality and performance.

Evaluation of ALG

First of all, it was checked the validity of the transition method, capturing network traffic. Figure 5 shows the traces captured both the IPv6 LAN and the IPv4 WAN interfaces.

```
 1  fe80::16fc:eeff:fe7f:a2ff -> ff02::1      ICMPv6 Router advertisement
 2  2001:1938:110:23:213:d3ff:fe78:c33d -> 2001:1938:110:23:21b:9eff:fe2d:668
TCP 1093 > 8080 [SYN] Seq=0 Win=16384 Len=0 MSS=1440
 3  2001:1938:110:23:21b:9eff:fe2d:668 -> ff02::1:ff78:c33d ICMPv6 Neighbor
solicitation
 4  2001:1938:110:23:213:d3ff:fe78:c33d -> 2001:1938:110:23:21b:9eff:fe2d:668
ICMPv6 Neighbor advertisement
 5  2001:1938:110:23:21b:9eff:fe2d:668 -> 2001:1938:110:23:213:d3ff:fe78:c33d
TCP 8080 > 1093 [SYN, ACK] Seq=0 Ack=1 Win=5760 Len=0
 6  2001:1938:110:23:213:d3ff:fe78:c33d -> 2001:1938:110:23:21b:9eff:fe2d:668
TCP 1093 > 8080 [ACK] Seq=1 Ack=1 Win=17280 Len=0
 7  2001:1938:110:23:213:d3ff:fe78:c33d -> 2001:1938:110:23:21b:9eff:fe2d:668
HTTP GET
http://sitecheck2.opera.com/?host=www.altavista.com&hdn=nubrKnkzLB7qxAS86ab
tMw== HTTP/1.0
 8  2001:1938:110:23:21b:9eff:fe2d:668 -> 2001:1938:110:23:213:d3ff:fe78:c33d
TCP 8080 > 1093 [ACK] Seq=1 Ack=498 Win=6432 Len=0
 9  2001:1938:110:23:213:d3ff:fe78:c33d -> 2001:1938:110:23:21b:9eff:fe2d:668
HTTP GET http://www.altavista.com/ HTTP/1.0
 10  2001:1938:110:23:21b:9eff:fe2d:668 -> 2001:1938:110:23:213:d3ff:fe78:c33d
TCP 8080 > 1094 [ACK] Seq=1 Ack=539 Win=6456 Len=0
 11  192.168.1.223 -> 192.168.1.1  DNS Standard query AAAA www.altavista.com
 12  192.168.1.223 -> 192.168.1.1  DNS Standard query AAAA sitecheck2.opera.com
 13  192.168.1.1 -> 192.168.1.223 DNS Standard query response CNAME
avatw.search.a00.yahoodns.net
 14  192.168.1.223 -> 192.168.1.1  DNS Standard query A www.altavista.com
 15  192.168.1.1 -> 192.168.1.223 DNS Standard query response CNAME
avatw.search.a00.yahoodns.net A 72.30.186.25
 16  192.168.1.223 -> 72.30.186.25 TCP 43019 > 80 [SYN] Seq=0 Win=5840 Len=0
MSS=1460 TSV=2203659 TSER=0 WS=6
 17  1.731010 72.30.186.25 -> 192.168.1.223 TCP 80 > 43019 [SYN, ACK] Seq=0
Ack=1 Win=8712 Len=0 MSS=1452 WS=0 TSER=2203659
 18  1.731031 192.168.1.223 -> 72.30.186.25 TCP 43019 > 80 [ACK] Seq=1 Ack=1
Win=5888 Len=0 TSV=2203661 TSER=3240479415
 19  1.731092 192.168.1.223 -> 72.30.186.25 HTTP GET
http://www.altavista.com/ HTTP/1.0
 20  1.749107 72.30.186.25 -> 192.168.1.223 TCP 80 > 43019 [ACK] Seq=1
Ack=539 Win=15846 Len=0 TSV=3240479417 TSER=2203661
 21  2.188310 72.30.186.25 -> 192.168.1.223 HTTP HTTP/1.0 200 OK  (text/html)
 22  2.188401 192.168.1.223 -> 72.30.186.25 TCP 43019 > 80 [ACK] Seq=539
Ack=1441 Win=8768 Len=0 TSV=2203775 TSER=3240479460
 23  2.188462 2001:1938:110:23:21b:9eff:fe2d:668 ->
2001:1938:110:23:213:d3ff:fe78:c33d HTTP HTTP/1.0 200 OK  (text/html)
 24  2.198668 2001:1938:110:23:213:d3ff:fe78:c33d ->
2001:1938:110:23:21b:9eff:fe2d:668 TCP 1096 > 8080 [SYN] Seq=0 Win=16384
Len=0
 25  2.198693 2001:1938:110:23:21b:9eff:fe2d:668 ->
2001:1938:110:23:213:d3ff:fe78:c33d TCP 8080 > 1096 [SYN, ACK] Seq=0 Ack=1
Win=5760
```

Figure 5. Capturing traffic using ALG

Afterwards, measures were made for connection time and full access time to various sites via IPv4, using Apache Benchmark [16]. Finally, functional assessments were made, bearing in mind that this method only allows translation of HTTP/HTTPS. It was possible to successfully use this method even in relatively old operating systems like Windows XP. It was necessary to set the name of the router as a proxy in the HTTP client (browser) and was added in Win XP hosts's file the IPv6 address of the router. Also, ALG worked correctly in the mobile operating system Symbian and all operating systems that prefer IPv4 to IPv6 for navigation.

Evaluation of NAT64

NAT64's performance was satisfactory as a solution for network connectivity to IPv4 from "IPv6 only" networks provided that the NAT64 device is located close to the network service networks. It can be observed almost complete compatibility with all application layer protocols based on TCP, UDP or ICMP. To evaluate the performance were measured for connection times and different places full access to IPv4. However, regarding the functional assessment, to analyze the NAT64 within a range of public address translation and use it remotely over the Internet IPv6 many problems could be seen problems when accessing certain HTTP IPv4, which were solved by making changes to the MTU of the interface end nodes. In the

specific case of access to other services such as SSH was not observed any problems. It could be some inconsistencies in the network hosts "IPv6 only" that prevented their use when they had relatively old operating systems, as the case of Win XP. The problem occurs because Windows XP's inability to perform DNS queries over IPv6. Could be observed in new operating systems like Windows 7 and later versions of Linux complete compatibility and configuration automatically and transparently to the end user. For updated versions of Linux unless required only DNS64 server settings in the configuration file (/etc/resolv.conf), besides making sure that network manager does not modify the change.

Referred to the communication with IPv6 sites, it is direct without the intervention of any intermediate device which imposes an advantage to other methods. Figure 6 shows the captured traffic traces NAT64 with access to IPv4.

```
  1   0.000000 2001:1938:110:23:32::4 ->
2001:1291:217:23:250:56ff:feae:27 DNS Standard query AAAA
www.yahoo.com
  2   0.999437 2001:1938:110:23:32::4 ->
2001:1291:217:23:250:56ff:feae:27 DNS Standard query AAAA
www.yahoo.com
  3   1.532224 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c TCP 54826 > 80 [SYN] Seq=0
Win=8192 Len=0 MSS=1440 WS=2
  4   2.346770 2001:1291:217:64:9b:0:43c3:a04c ->
2001:1938:110:23:32::4 TCP 80 > 54826 [SYN, ACK] Seq=0
Ack=1 Win=5840 Len=0 MSS=1440 WS=8
  5   2.346898 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c TCP 54826 > 80 [ACK] Seq=1
Ack=1 Win=17280 Len=0
  6   2.347022 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c HTTP GET / HTTP/1.1
  7   3.151339 2001:1291:217:64:9b:0:43c3:a04c ->
2001:1938:110:23:32::4 TCP 80 > 54826 [ACK] Seq=1 Ack=602
Win=7168 Len=0
  8   3.159450 2001:1291:217:64:9b:0:43c3:a04c ->
2001:1938:110:23:32::4 HTTP HTTP/1.1 302 Found
(text/html)
  9   3.162587 2001:1938:110:23:32::4 ->
2001:1291:217:23:250:56ff:feae:27 DNS Standard query AAAA
ar.yahoo.com
 10   3.359405 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c TCP 54826 > 80 [ACK]
Seq=602 Ack=666 Win=16612 Len=0
 11   3.993892 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c TCP 54827 > 80 [SYN] Seq=0
Win=8192 Len=0 MSS=1440 WS=2
 12   4.958166 2001:1291:217:64:9b:0:43c3:a04c ->
2001:1938:110:23:32::4 TCP 80 > 54827 [SYN, ACK] Seq=0
Ack=1 Win=5840 Len=0 MSS=1440 WS=8
 13   4.958303 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c TCP 54827 > 80 [ACK] Seq=1
Ack=1 Win=17280 Len=0
 14   4.958430 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c HTTP GET /?p=us HTTP/1.1
 15   5.654285 2001:1291:217:64:9b:0:43c3:a04c ->
2001:1938:110:23:32::4 TCP 80 > 54827 [ACK] Seq=1 Ack=774
Win=7424 Len=0
 16   6.286348 2001:1291:217:64:9b:0:43c3:a04c ->
2001:1938:110:23:32::4 TCP [TCP segment of a reassembled
PDU]
 17   6.485549 2001:1938:110:23:32::4 ->
2001:1291:217:64:9b:0:43c3:a04c TCP 54827 > 80 [ACK]
Seq=774 Ack=537 Win=16744 Len=0
 18   6.528961 2001:1938:110:23:32::4 ->
2001:1938:110:23::1 ICMPv6 Neighboradvertisement
```

Figure 6. Capture traffic using the NAT64

Comparison of ALG-NAT64 Accessing IPv4 Servers

Characteristics Comparison: The following table shows the results of the evaluated parameters applied to ALG and NAT64/DNS64 on a scale of four levels:

- Nonexistent - Low - Medium - High

Table 1. Parameters comparison

Parameter	ALG	NAT64/DNS64
Complexity in Service Setup	Low	Medium
Maintainability	Medium	Medium
Response time performance	Medium (IPv6/IPv4)	High (IPv6) Medium (IPv4)
Access issues	Nonexistent (only HTTP/S)	Low
Supported protocols	Low	High
Scalability	Medium	High
Security Integration	No tested	No tested
Latency	Low	Low (IPv6) Medium(IPv4)
Complexity in Host Setup	Medium	Low or Nonexistent
Compatibility of operating systems of the client nodes	High	Medium
Installation Cost	Medium	Medium to High

Performance comparison: For performance tests, were measured connections time and full access connection time to various sites via IPv4, using Apache Benchmark (http://ipv4.google.com and http://www.mit.edu). The arguments supplied to AB were -c100, indicating the number of requests to perform for the benchmarking session and -c10, indicating the number of multiple requests to perform at a time. For the ALG tests also was needed to supply another argument -X proxy:port , indicating the need to use a Proxy Server, in this case the ALG application.

The ALG and the NAT64+DNS64 applications where running in the same router, so no hardware differences affected the comparison.

An additional configuration was needed in all the cases, to complete successfully the tests, set the MTU in the home clients to 1280. It was due the use of tunnel mechanisms in the router (NAT64 or ALG).

The performance tests of both methods are shown in Figure 7 and 8. The first shows minimum, average and maximum time to connect (ALG conn and NAT64 conn) and the minimum, average and maximum time to complete the requirement (ALG total and NAT64 total) to http://ipv4.google.com. The second displays the same values to access http://www.mit.edu. It should be noted that were compared only the HTTP/HTTPS protocols accessing only IPv4 servers. Testing performance time to access IPv6 servers is beyond the scope of this work, as NAT64 does not intervene in it. ALG does, so the performance would be slightly lower in the second case due to the addition of middleware.

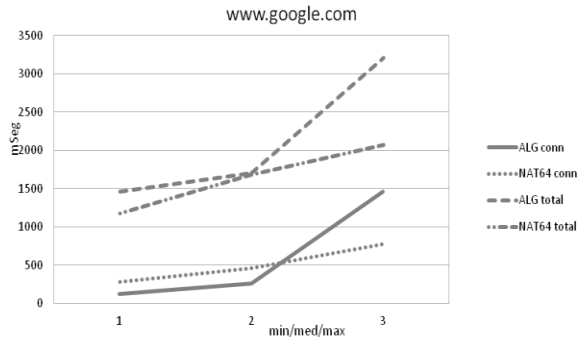

Figure 7. Performance test from google.com

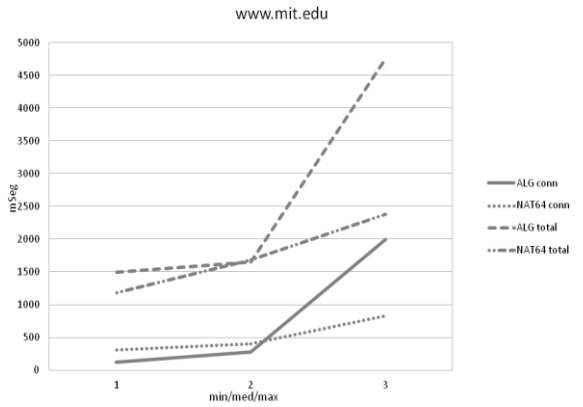

Figure 8. Performance test from mit.com

8. CONCLUSIONS

This paper is intended as an additional tool for ISPs to evaluate alternatives when making the transition.

As long as the ISP does not obtain new IPv4 addresses from RIR, both techniques can gradually be implemented in a small group of home clients. This will be transparent to the rest of their customers and it will allow to make the necessary adjustments for proper deployment.

From the comparison made, it is determined that the ALG method is suggested when the "home clients" only access the Internet using HTTP / HTTPS. NAT64 + DNS64 excels it in terms of the amount of supported protocols. On the other hand, we observed that ALG is a perfect complement NAT64 + DNS64, due the hosts having operating systems like Windows XP or Symbian prefer for DNS queries, the A record over the AAAA. By using ALG to IPv4 Servers navigation from Networks "IPv6 only" is solved the inability to resolve names using IPv6.

The main disadvantage of using ALG over DNS64 + NAT64 is the lower performance in the HTTP requests, as seen in Figure 8 and 9. Additionally client applications (browsers) must be manually configured to setup a proxy. This inconvenience can be solved by setting the ALG as a transparent proxy [17], leaving this task for future research.

We can say that with the use of any of these mechanisms, end users will have a public address (Global IPv6). The advantage is that it returns to the initial strategy of an end to end Internet communication, allowing the installation of servers and services, as well as embedded devices with visibility from all over the Internet.

We can highlight that both, the NAT64 and ALG, where implemented in a way that can be used by ISPs. In the case of NAT64 using a public IPv6 range (not the range set by default in the RFC 6146, which no is routeable through Internet V6) so it can be used by an ISP within their Autonomous System (AS) or even outside their AS as a public service.

Future work is planned for implementation and comparison of IPv4/IPv6 transition models.

Finally we believe that the present work and the work performed GridTICS group contribute to the promotion, dissemination and training of human resources for the impending shift to Internet Protocol version 6 in the region.

9. REFERENCES

[1] Informe LACNIC, “Distribuciones / Asignaciones IPv4, espacio disponible y pronósticos (Report March 2011 – updated April 2011)”, http://www.lacnic.net/sp/registro/espacio-disponible-ipv4.html, [last visit: 14/07/2011]

[2] D. Giusto, A. Iera, G. Morabito, L. Atzori (Eds.), The Internet of Things, Springer (2010) ISBN: 978-1-4419-1673-0

[3] S. Deering y R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, December 1998

[4] J. Palet, “Cómo es la transición?”, http://portalipv6.lacnic.net/es/ipv6/novedades/ c-mo-es-la-transición [last visit: 14/07/2011]

[5] ¿Quienes están implementando IPv6 en la Región?, http://portalipv6.lacnic.net/es/quienes-est-n-implementando-ipv6-en-la-regi-n [last visit: 12/11/2011]

[6] E. Nordmark Stateless IP/ICMP Translation Algorithm (SIIT) RFC 2765, February 2000

[7] G. Tsirtsis, P. Srisuresh, “Network Address Translation - Protocol Translation (NAT-PT)”,RFC 2766, February 2000

[8] K. Tsuchiya, H. Higuchi, Y. Atarashi, “Dual Stack Hosts using the "Bump-In-the-Stack" Technique (BIS)”, RFC 2767, February 2000

[9] S. Lee, M-K. Shin, Y-J. Kim, E. Nordmark, A. Duran, “Dual Stack Hosts Using "Bump-in-the-

API" (BIA)", RFC 3338, October 2002

[10] J. Hagino, K. Yamamoto, "An IPv6-to-IPv4 Transport Relay Translator", RFC 3142, June 2001

[11] M. Bagnulo, P. Matthews, I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011

[12] M. Bagnulo, A. Sullivan, P. Matthews, I. van Beijnum," DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011

[13] C. Aoun, E. Davies, "Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status", RFC 4966, July 2007

[14] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010

[15] Ecdysis: open-source implementation of a NAT64 gateway - http://ecdysis.viagenie.ca/[last visit: 14/07/2011]

[16] Apache HTTP server benchmarking tool - http://httpd.apache.org/docs/2.0/ programs/ab.html [last visit: 14/07/2011]

[17] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC: 2616, June 1999