Application Compatibility of the NAT64 IPv6 Transition Technology

Sándor Répás, Tamás Hajas and Gábor Lencse

***Abstract*—The proliferation of smart phones and other Internet capable devices together with the depletion of the global IPv4 address pool will be a huge driving force for the deployment of IPv6 in the forthcoming years. The communication of an IPv6 only client with an IPv4 only server is a typical practical task to be solved among the many issues of the co-existence of IPv4 and IPv6. The usage of DNS64+NAT64 may be a good solution if our applications can flawlessly work with it. As for NAT64 implementations, TAYGA running under Linux and Packet Filter (PF) of OpenBSD were tested with the following application level protocols: HTTP, HTTPS, SMTP, POP3, IMAP4, Telnet, SSH, FTP, OpenVPN, RDP, Syslog, BitTorrent, Skype and SIP. The client-server application protocols could traverse through the NAT64 gateway flawlessly but the peer to peer ones failed. In contrast to the results of other researchers, OpenVPN worked perfectly with NAT64.**

***Keywords*—IPv6 deployment, IPv6 transition solutions, NAT64, application compatibility, Packet Filter, TAYGA.**

I. Introduction

The specification of the Internet Protocol version 6 (IPv6) exists since 1998 [1]. However, IPv4 is used by the vast majority of the current Internet sites today. E.g. only 2.8% of the Internet traffic reaching Google used IPv6 on January 1, 2014 [2]. The proliferation of smart phones and other Internet capable devices together with the depletion of the global IPv4 address pool [1] will be a huge driving force for the deployment of IPv6 in the forthcoming years as the internet service providers (ISPs) will not be able to supply their large number of new clients with IPv4 addresses. They can get IPv6 addresses only. However, the IPv6 only devices must reach the Internet sites that still use IPv4 only. Thus from the many issues of the co-existence of IPv4 and IPv6, the communication of an IPv6 only client with an IPv4 only server is the first practical task to solve in the upcoming years. The authors believe that the application of NAT64 [5] + DNS64 [6] is the best available solution that makes it possible for an IPv6 only client to communicate with an IPv4 only server. (A brief description of their operation will be provided later.)

Are the different Internet applications compatible with the NAT64 technology? The authors of [7] and [8] have already tested the NAT64 compatibility of few popular applications with a given NAT64 implementation (Ecdysis [9]), but we believe that further NAT64 implementations should also be tested to give alternatives to network operators.

The remainder of this paper is organized as follows: first, the operation of the DNS64+NAT64 solution is described, second, the application compatibility research results are surveyed, third, TAYGA and Packet Filer of OpenBSD are introduced, fourth, the most important applications are selected for testing, fifth, the description of the test network and the testing method of each applications are given together with their results, sixth, our results are summarized and discussed, and finally, our conclusions are given.

The volume of the IPv6 traffic of the Internet is low in many countries of the western world now and it will probably grow slowly for some years, but its volume may explode later on and the networks should be ready to cope with it. Thus our results are expected to give valuable information to many network administrators when selecting the appropriate IPv6 transition solution for their networks.

II. The Operation of DNS64 and NAT64

To provide connectivity between an IPv6 only client and an IPv4 only server one can use DNS64+NAT64 as it is shown in Fig. 1. The explanation of the process is the following:

1. The IPv6 only client sends a query to its nameserver about the IPv6 address of the destination server with the DNS name of the server (query AAAA record).
2. The DNS64 server tries to resolve the DNS name with recursive queries or just with a simple forwarding.
3. The DNS server resolves the given name to an IPv4 address, but not IPv6.
4. The DNS64 server generates an answer with an AAAA record and sends it back as an answer for the query of its client with a synthesized IPv6 address. This IPv6 address contains the given IPv4 address of the server at the last 32 bits, while the first 96 bits can be a network specific prefix or the NAT64 well-known prefix. This special IPv6 address is called IPv4-Embedded IPv6 Address [10].
5. The client sends an IPv6 packet with a TCP SYN segment through its (NAT64) gateway with this special IPv6 address as destination address.

Manuscript received February 4, 2014.

The work of Sándor Répás was supported in the framework of TÁMOP 4.2.4. A/2-11-1-2012-0001 "National Excellence Program – Elaborating and operating an inland student and researcher personal support system convergence program" The project was subsidized by the European Union and co-financed by the European Social Fund.

The work of Gábor Lencse was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0012: "Smarter Transport" – IT for co-operative transport system – The Project is supported by the Hungarian Government and co-financed by the European Social Fund.

This publication was supported by the Multidisciplinary Doctoral School of the Faculty of Engineering Sciences, Széchenyi István University.

Sándor Répás, Tamás Hajas and Gábor Lencse are with the Department of Telecommunications, Széchenyi István University, 1 Egyetem tér, Győr, H-9026, Hungary (e-mail: repas.sandor@sze.hu, hajas@tilb.sze.hu, lencse@sze.hu)

[1] IANA delegated the last five "/8" IPv4 address blocks to the five Regional Internet Registries in 2011 [3], of which APNIC has already depleted its IPv4 address pool in 2011 and RIPE NCC did so in 2012 [4]. It means that RIPE NCC also uses a more strict allocation policy for its very last /8 block.

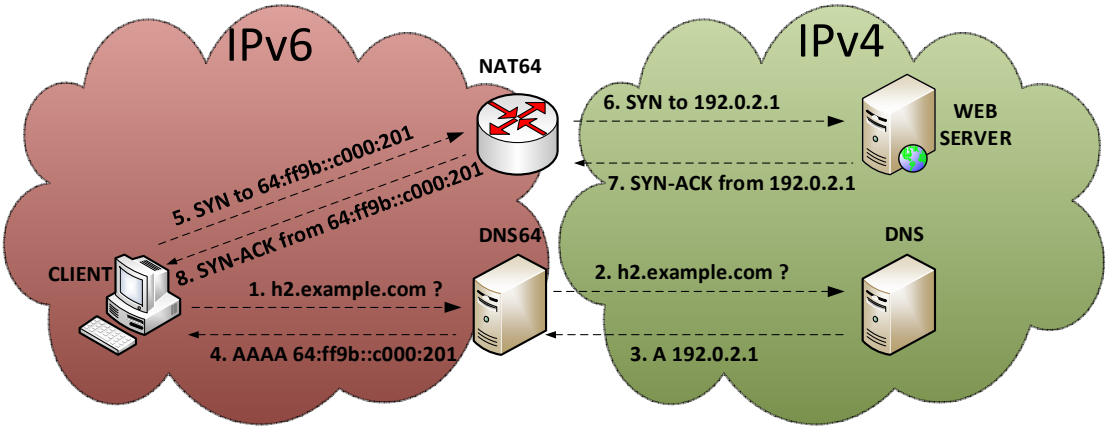

Figure 1. The operation of the DNS64+NAT64 IPv6 transition solution.

6. The gateway knows from the prefix of the destination address that the IPv6 packet is destined to an IPv4 server. The NAT64 gateway makes a stateful network address translation between IPv6 and IPv4 and sends the IPv4 packet containing the TCP SYN segment to the IPv4 only server with its own IPv4 address as source address. The NAT64 gateway needs to have IPv4 and IPv6 addresses too, to make this process happen.
7. The IPv4 only web server sends its answer (containing a TCP SYN-ACK segment to the IPv4 address of the NAT64 gateway.
8. The NAT64 gateway assembles the IPv6 version of the received packet using its own state table and the payload of the IPv4 packet and then sends the IPv6 packet to the originating client.

Thus the IPv6 only client can communicate with the IPv4 only server as if it was an IPv6 capable server and also the IPv4 only server can see the IPv6 only client as if it was an IPv4 client. However, the server will log the IPv4 address of the NAT64 gateway instead of the IPv6 address of the client and this is true for all the clients of the NAT64 gateway. This can be a drawback of the usage of NAT64 when collecting exact statistics is important. However the same drawback exists when Application Level Gateway (ALG) or Carrier Grade NAT (CGN) solutions are used.

For a more detailed but still easy to follow introduction to DNS64+NAT64, see [11] and for the most accurate and detailed information, see the relating RFCs [5] and [6].

III. A Short Survey of the Current Research Results

Eighteen widely used application-layer network protocols were tested theoretically and empirically by Škoberne and Ciglarič in a virtualized environment in 2011 [7] to find out how well they traverse Ecdysis, an open source stateful NAT64 implementation. They were differentiated the following three levels of the translation performance: poorly, conditionally and well translated. They concluded that NAT64 and DNS64 translation might not provide a good experience, because the VoIP and Instant Messaging were poorly translated. In a research about the flow-based identification of failures caused by Dual-Stack Lite and NAT64 transition techniques in 2012 [8], the researchers did not find any problem with DS Lite, but they successfully identified such problems with Skype, OpenVPN, BitTorrent and SIP. They did not mention finding any problem with FTP. Finally they did not find any usable Flow-based method to automatically detect the problem in an ISP system.

It is inevitable to do more compatibility tests of different NAT64 implementations as application compatibility is a serious aspect of selecting an appropriate NAT64 gateway.

IV. The Selection of NAT64 Implementations

Only free software [12] (also called open source [13]) implementations were considered for testing. We had multiple reasons for this decision:

- The licenses of certain vendors (e.g. [14] and [15]) do not allow the publication of benchmarking or other results.
- Free software can be used by anyone for any purposes thus our results can be helpful for anyone.
- Free software is free of charge for us, too.

Ecdysis [9] could have been a choice, as it was the first NAT64 implementation, and it also contains DNS64 support. However, test of Ecdsys had already been done in both [7] and [8]. For these reasons, two other NAT64 implementations were selected for application compatibility analysis: TAYGA and PF. We have selected them for NAT64 performance testing for our earlier paper [16] and their description below is taken from there. (For the result of our performance testing of some free DNS64 implementations, see [17].)

A. TAYGA

TAYGA [18] is free software under GPLv2 license and according to its developers it was intended to provide *production quality NAT64 service*. TAYGA is a *stateless* NAT64 solution for Linux. It means that by itself it can create only a one-to-one mapping between IPv6 and IPv4 addresses. For this reason TAYGA is used together with a stateful NAT44 packet filter (`iptables` under Linux): TAYGA maps the source IPv6 addresses to different IPv4 addresses from a suitable size of private IPv4 address range, and from the private IPv4 addresses the stateful NAT44 packet filter performs an SNAT (Source Network Address Translation) to the public IPv4 address of the NAT64 gateway (sometimes also called Port Address Translation, PAT). In the reverse direction, the stateful NAT44 packet filter "knows" which private IPv4

address belongs to the reply packet arriving to the IPv4 interface of the NAT64 gateway. After the NAT44 translation TAYGA can determine the appropriate IPv6 address using its one-to-one address mapping and then it rewrites the packet to IPv6.

When configuring TAYGA, a suitably large private IPv4 address range should be provided.

B. Packet Filter

The Packet Filter (PF) of OpenBSD 5.1 [19] includes NAT64 support that is based on the Ecdysis program code [20]. PF is free software under the BSD license.

PF [21] was first released in OpenBSD v3.0 in 2001. PF is a very powerful tool to filter and manipulate IP packets in modern BSD systems. It can be configured by editing **`/etc/pf.conf`**, and packets can be manipulated by many attributes. PF supports NAT, load balancing, logging and it can also operate as stateless and stateful packet filter at the same time.

PF supports IPv4 and IPv6 stateful NAT for many years, and now it supports NAT64, too. This feature of PF is called address family translation. PF in stateful mode can translate many IPv6 client addresses to one outgoing IPv4 address via address family translation. Because of stateful translation, PF needs to build a "states table" to find the correct IPv6 destination address of the incoming IPv4 packets. There is no need to use stateless and stateful NAT one after the other. In this case, a single state table is enough for perfect system operation against TAYGA's two-table solution. Therefore PF generates lower load for the NAT64 gateway [16].

We have made our tests on personal computers because of their proliferation, but TAYGA exists on OpenWRT, too [22].

V. Application Selection for Testing

First, we had to collect frequently used applications in order to accomplish compatibility tests. HTTP, HTTPS, SMTP, POP3, IMAP4 protocols are essential ones.

In addition, Telnet, SSH, FTP, OpenVPN, RDP, Syslog, P2P, SIP are also very common. Protocols and their attributes are listed in table I.

VI. Tests and Results

The aim of our tests was to examine and compare the compatibility of the selected NAT64 implementations.

A. The Structure of the Test Network

As applications operate in different ways, we had to use different network structures during our tests. Detailed information of these structures is available at each protocol demonstration below.

It was necessary to separate the testbed from the network of the laboratory. However, we had to connect to the network of the laboratory for a few times. Our solution was a VLAN using a 3Com Baseline 2948-SFP Plus switch.

We used the fd5c:6bc1:7bc7:ffff::/64 Unique Local IPv6 Unicast Address (ULA) and worked with 5 computers with different roles. Symbols of two different switches in Figures 2, 3, 4 and 5 mean two different VLANs of one physical switch.

B. NAT64 Gateway Settings

1) Preparation of the TAYGA system

The network interfaces of the freshly installed Debian Squeeze 6.0.6 Linux operating system on the Pentium III computer were set according to Fig. 2.

The settings (in the **`/etc/network/interfaces`** file) were the following:

```
#The loopback network interface
auto lo
iface lo inet loopback
#Internal interface
auto eth1
iface eth1 inet6 static
address fd5c:6bc1:7bc7:ffff::1
netmask 64
post-up /root/nat64-config.sh
#External
auto eth0
iface eth0 inet static
address 192.168.100.221
netmask 255.255.255.0
gateway 192.168.100.1
```

TABLE I. Application Protocols, Protocols and Ports

Application protocol	Transport Protocol	Standard port
HTTP	TCP	80
HTTPS	TCP	443
SMTP	TCP	25
POP3	TCP	110
IMAP4	TCP	143
Telnet	TCP	23
SSH	TCP	22
FTP	TCP	20,21
OpenVPN	UDP/TCP	1194
RDP	TCP	3389
Syslog	UDP	514
P2P (BitTorrent)	TCP/UDP	6881-6999
SIP	TCP/UDP	5060
RTP/RTCP for SIP	UDP	1024-65535

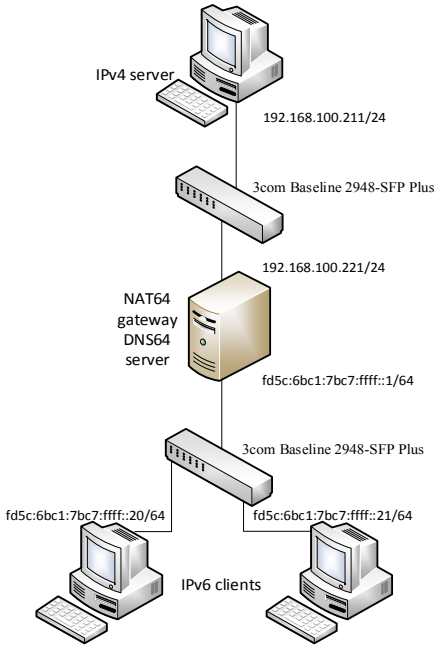

Figure 2. Topology of the HTTP, HTTPS, Telnet, SSH, FTP, Syslog test network.

TAYGA 0.9.2 was installed on the test computer and the content of **tayga.conf** was set as follows:

```
tun-device nat64
ipv4-addr 10.0.0.1
prefix fd5c:6bc1:7bc7:ffff:ffff:ffff::/96
dynamic-pool 10.0.0.0/8
data-dir /var/db/tayga
```

TAYGA was started with the following script:

```
#!/bin/bash
tayga --mktun
ip link set nat64 up
#Private IPv4 address space for NAT
ip addr add 10.0.0.1 dev nat64
ip route add 10.0.0.0/8 dev nat64
#NAT64 network specific IPv6 address space
ip addr add fd5c:6bc1:7bc7:ffff::2 dev nat64
ip route add fd5c:6bc1:7bc7:ffff:ffff:ffff::/96 \
  dev nat64
#Start TAYGA
tayga
#Enable routing
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
#Enable NAT44
iptables -t nat -A POSTROUTING -s 10.0.0.0/8 -o eth0 \
  -j MASQUERADE
```

Some applications required name resolution, thus DNS64 was used. BIND 9.9.2-P1 was chosen. The settings in the file **/etc/bind/named.conf** were the following ones:

```
options {
auth-nxdomain no;
listen-on-v6 { any; };
allow-query { any; };
dns64 fd5c:6bc1:7bc7:ffff:ffff:ffff::/96 { };
};
```

1) Preparation of the OpenBSD system

The settings (in the **/etc/hostname.fxp0** file) were the following:

```
inet 192.168.100.221 255.255.255.0
!route add -inet default 192.168.100.1
```

The settings (in the **/etc/hostname.rlp0** file) were the following:

```
inet6 alias fd5c:6bc1:7bc7:ffff::1 64
```

Settings of **/etc/pf.conf** in order to use NAT64:

```
#NAT64
pass in on rl0 inet6 from any to \
  fd5c:6bc1:7bc7:ffff:ffff:ffff::/96 af-to inet \
  from 192.168.100.221
```

Settings of **/etc/sysctl.conf** were used to permit packet forwarding:

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
```

DNS64 settings were the same as with the TAYGA system.

C. Preparation of the IPv6 client computers

Debian Wheezy 7.1 operating system was installed on all clients and servers. Different settings of tests are detailed at each test. Here are the network settings only.

The settings (in the **/etc/network/interfaces** file) were the following:

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet6 static
address fd5c:6bc1:7bc7:ffff::[20-21] # on the first \
  client: 20, on the other client: 21
netmask 64
gateway fd5c:6bc1:7bc7:ffff::1
up sleep 1
post-up /etc/network/routeadd.sh
```

The fd5c:6bc1:7bc7:ffff:ffff:ffff::/96 network specific NAT64 address range was routed to the NAT64 gateway in the **routeadd.sh** script:

```
route add -A inet6 fd5c:6bc1:7bc7:ffff:ffff:ffff::/96 \
  gw fd5c:6bc1:7bc7:ffff::1
```

DNS settings (in the **/etc/resolv.conf** file) were the following:

```
nameserver fd5c:6bc1:7bc7:ffff::1
```

The same settings were used at the Windows client.

D. Tests

1) HTTP

The topology of the network is shown in Fig. 2.

The server used Apache 2.2.22-13 of Debian Wheezy distribution, which was installed by the **apt-get install apache2** command. On the client computers, the Konqueror web browser was successfully used to open **index.html** using the following URL which contains the IPv4 embedded IPv6 address:

```
http://[fd5c:6bc1:7bc7:ffff:ffff:ffff:192.168.100.211]
```

2) HTTPS

After the HTTP test, the HTTPS protocol was enabled by the following commands:

```
a2ensite default-ssl
a2enmod ssl
```

Operation had been checked by the same browser again but using the following URL:

```
https://[fd5c:6bc1:7bc7:ffff:ffff:ffff:192.168.100.211]
```

Trouble free operation was observed.

3) Telnet

The server used the telnet daemon 0.17-36 of the Debian Wheezy distribution, which one was installed by the **apt-get install telnetd** command. The client could successfully connect to the server by the following command:

```
telnet fd5c:6bc1:7bc7:ffff:ffff:ffff:192.168.100.211
```

4) SSH

OpenSSH 6.0p1-4 server was installed using the command **apt-get install openssh-server**. We applied key-based authentication from both clients which worked correctly.

The clients used the following command for connecting:

```
ssh fd5c:6bc1:7bc7:ffff:ffff:ffff:192.168.100.211
```

5) FTP

Proftpd 1.3.4a-4+nmu1 was installed by the **apt-get install proftpd** command. An FTP client program was installed for each client workstations by the **apt-get install ftp** command.

FTP uses two TCP connections: the *control connection* is established in the beginning of an FTP session and it is alive until the end of the session; a data connection is established for each file transfer and also for each directory listing. FTP

can be used in *active mode* or in *passive mode*. When a data connection is needed in the active mode, the client sends its IP address and a selected port number to the server (through the control connection) and the server opens the data connection towards the client using the received IP address and port number. In passive mode, the opening of the data connection happens in the opposite way: the server sends its IP address and a selected port number to the client and the client opens the data connection. If the client resides behind NAT then only passive mode can be used as in active mode the server cannot open the data connection to the client (unless a so called FTP protocol helper is used). FTP was tested in both active and passive mode. The first one failed and the second one worked correctly, as we expected.

6) Syslog-ng

Version 3.3.5-4 of **syslog-ng** was installed on the server and on the two client computers. The modifications in the **syslog-ng.conf** configuration file of the clients were the following:

```
# Destinations
destination server {
tcp6("fd5c:6bc1:7bc7:ffff:ffff:ffff:c0a8:64d3"
port(514)); };
# Log paths
log { source(s_src); destination(server); };
```

On the server the following configuration was used:

```
# Sources
source ipv6client { tcp(port(514) keep-alive(yes)); };
# Destinations
destination ipv6client_file {
file("/var/log/ipv6client.log"); };
# Log paths
log { source(ipv6client); destination(ipv6client_file);
};
```

The logging over the NAT64 gateway were working correctly.

7) OpenVPN

The configuration of the OpenVPN was more complex. To perform the tests, we had to build a new network based on the original one, because we needed a connection to the Laboratory network. The topology of the test network of OpenVPN is show in Fig. 3.

We have used two IPv6 OpenVPN clients, one IPv4 OpenVPN client and one IPv4 only OpenVPN server. All of the computers were running Debian Linux. The server used OpenVPN version 2.2.1-8+deb7u2, which was installed by **apt-get istall openvpn**.

After the installation we generated the certificates on the server with the following steps:

The **openssl-1.0.0.cnf**, **wichopensslcnf** and **pkitool** files were copied from their original location (the **/usr/share/doc/openvpn/examples/easy-rsa/2.0/** directory) to the **/etc/openvpn** directory.

A **keys** subdirectory in the **/etc/openvpn** was created.

In the **/etc/openvpn** directory the following commands were issued:

```
. /usr/share/doc/openvpn/examples/easy-rsa/2.0/vars
. /usr/share/doc/openvpn/examples/easy-rsa/2.0/clean-all
. /usr/share/doc/openvpn/examples/easy-rsa/2.0/build-ca
. /usr/share/doc/openvpn/examples/easy-rsa/2.0/build-dh
```

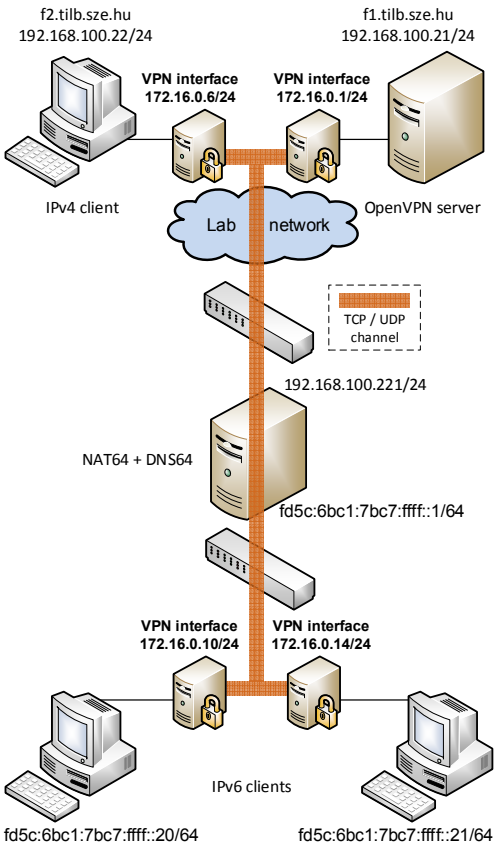

Figure 3. Topology of the OpenVPN test network.

The following commands were issued in the **/usr/share/doc/openvpn/examples/easy-rsa/2.0** directory:

```
./build-key-server server
./build-key client
```

The **ca.crt**, **dh1024.pem**, **server.crt**, **server.key** files were copied from the **/etc/openvpn/keys** directory into the **/etc/openvpn** directory.

The sample server configuration file of the OpenVPN distribution were used with the following modifications:

```
proto udp #First UDP then TCP communication
dev tun #TUN enabling interface
server 172.16.0.0 255.255.255.0 #The useable private \
  address space
duplicate-cn #Multiple clients can use the same key
group nogroup
client-to-client #VPN clients can communicate each other
```

The **ca.crt**, **client.crt**, **client.key** files were copied from the **/etc/openvpn/keys** directory of the server into the **/etc/openvpn/keys** directory of the clients. The sample client configuration file of the OpenVPN distribution was used with the following modifications on the IPv4 client:

```
proto udp #First UDP then TCP
remote 192.168.100.22 1194 #OpenVPN server
```

And on the IPv6 clients:

```
proto udp6 #UDP TCP over IPv6
remote fd5c:6bc1:7bc7:ffff:ffff:ffff:c0a8:6416 1194 \
  #OpenVPN server
```

After OpenVPN processes were started on the server and on all the client computers, ICMP communication was possi-

ble through the VPN channel between all of the clients using both UDP and TCP as VPN transport protocols.

Starting from version 2.3.0, OpenVPN fully supports IPv6 and in addition to that OpenVPN in Debian Wheezy system contains some unofficial developer patches for IPv6 support. This was the cause of the flawless communication through the NAT64 gateway [23].

8) E-mail (SMTP, POP3, IMAP4)

The topology of the network is shown in Fig. 4.

Two mail servers were used to provide a lifelike environment. The DNS names of the mail servers were provided by the DNS server of the laboratory network.

Windows 7 Enterprise with Mozilla Thunderbird 24.0 and Debian Wheezy 7.1 with KMail 1.13.7 were used on the client computers.

Postfix 2.9.6-2 for SMTP, courier-pop 0.68.2-1 for POP3 and courier-imap 4.10.0 for IMAP4 were installed on the servers. In the **`main.cf`** postfix configuration file, the 192.168.100.0/24 network was added to the **`mynetworks`** statement. The mailboxes were created for the tests.

All of the tests were successfully executed with SMTP, POP3 and IMAP4.

9) RDP

The topology of the SMTP test network was used, but with just one server. On the server, xrdp 0.5.0-2 was installed. Both of the clients were able to create RDP connection using krdc 4.8.4-1+b1 as client software on KDE.

10) P2P

The external interface of the NAT64 gateway was connected to the network of the laboratory to provide Internet connection.

Version 6.9.0.106 of Skype and version 3.3.2 of µTorrent were used on a Windows 7 client computer for the test.

Neither the BitTorrent nor the Skype were working.

Skype was unable to authenticate successfully, while BitTorrent was unable to download any files.

11) SIP

The topology of the network is shown in Fig. 5.

Trixbox 2.8.0.4 was used on the SIP proxy computer. Linphone 3.5.2 and Linphone 3.6.1 were used on the Windows and on the Linux client computers, respectively.

It was possible making calls from any clients, and the phones were ringing that is the setup of the calls was correct, but there was silence in the phones that is they could not communicate over RTP through the NAT64 gateway.

VII. Summary and Discussion of the Results

Our results with those of [7] and [8] can be found in Table II. The test results of the different FTP protocols in [8] were not completely clear, so we had to write N/A into the corresponding lines of the table.

Evaluation of the results:

- PF and TAYGA produced exactly the same results.
- The majority of the most used protocols showed flawless operation.
- FTP in passive mode was translated without any issue.

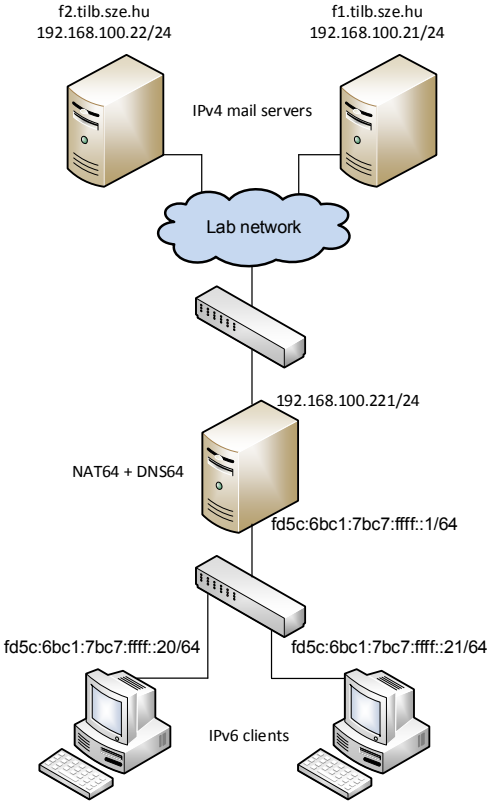

Figure 4. Topology of the E-mail, RDP test network.

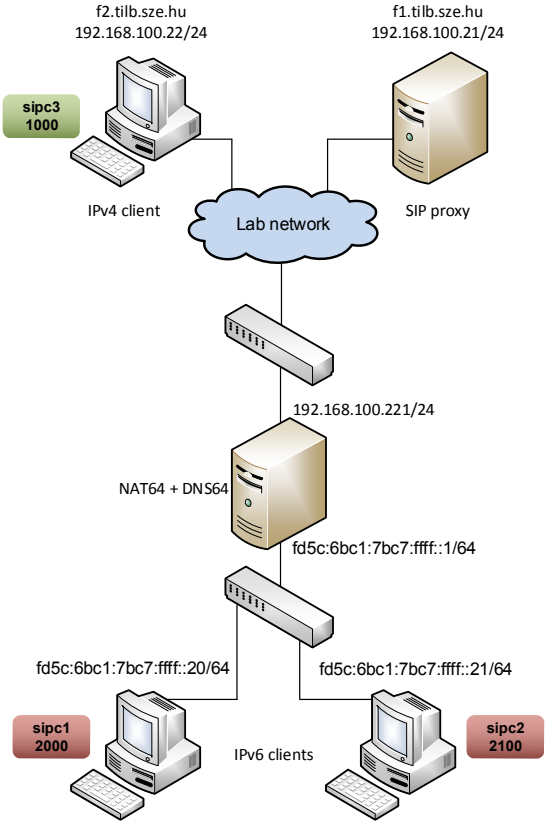

Figure 5. Topology of the SIP test network.

TABLE II. APPLICATION PROTOCOLS COMPATIBILITY

Application protocol	PF	Tayga	Ecdsys	
			[7]	[8]
HTTP	Yes	Yes	Yes	Yes
HTTPS	Yes	Yes	Yes	Yes
SMTP	Yes	Yes	Yes	Yes
POP3	Yes	Yes	Yes	Yes
IMAP4	Yes	Yes	Yes	Yes
Telnet	Yes	Yes	Yes	N/A
SSH	Yes	Yes	Yes	Yes
FTP passive mode	Yes	Yes	Cond.	N/A
FTP active mode	No	No	Cond.	N/A
OpenVPN	Yes	Yes	No	No
RDP	Yes	Yes	Yes	N/A
Syslog	Yes	Yes	N/A	N/A
Skype	No	No	No	No
BitTorrent	No	No	Cond.	No
SIP	No	No	No	No

- The most important difference between the previous research results and our results is the proper operation of the OpenVPN.
- The very common P2P and VoIP applications were not working at all.
- As we expected, Skype was not working, because it does not support IPv6 protocol actually [24].
- Our results reflect the current status, which could be improved in the future if protocol helpers will be available such as they exist for NAT44.

VIII. CONSLUSIONS

Due to the exhaustion of the IPv4 address pool, the internet service providers will not be able to provide IPv4 addresses to an increasing number of clients. Our research results proved that the usage of the DNS64/NAT64 is a viable solution to resolve this problem. The most important application protocols can traverse through the NAT64 gateway flawlessly. The very important drawback of the usage of the NAT64 transition technique is the missing support of P2P and VoIP applications. The development of an Application Level Gateway (ALG) or NAT helper for these applications is a must to satisfy the customers' needs. Another disadvantage of NAT64 is that the servers can log the IPv4 address of the NAT64 gateway only and not the IPv6 addresses of the clients and this is true for the ALG and CGN solutions, too.

REFERENCES

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

[2] Google, "IPv6 Adoption", http://www.google.com/ipv6/statistics.html

[3] The Number Resource Organization, "Free pool of IPv4 address space depleted" http://www.nro.net/news/ipv4-free-pool-depleted

[4] RIPE NCC, "RIPE NCC begins to allocate IPv4 address space from the last /8", http://www.ripe.net/internet-coordination/news/ripe-ncc-begins-to-allocate-ipv4-address-space-from-the-last-8

[5] M. Bagnulo, P. Matthews and I. Beijnum, "Stateful NAT64: Network address and protocol translation from IPv6 clients to IPv4 servers", IETF, April 2011. ISSN: 2070-1721 (RFC 6146)

[6] M. Bagnulo, A Sullivan, P. Matthews and I. Beijnum, "DNS64: DNS extensions for network address translation from IPv6 clients to IPv4 servers", IETF, April 2011. ISSN: 2070-1721 (RFC 6147)

[7] N. Škoberne and M. Ciglarič, "Practical Evaluation of Stateful NAT64/DNS64 Translation" Advances in Electrical and Computer Engineering, vol. 11, no. 3,August 2011, pp. 49-54. doi: 10.4316/AECE.2011.03008

[8] V. Bajpai, N. Melnikov, A. Sehgal and J. Schönwälder, "Flow-based Identification of Failures Caused by IPv6 Transition Mechanisms" *Proc. 6th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security* (AIMS 2012, June 4-8, 2012) Luxembourg, Luxembourg

[9] "Ecdysis: open-source implementation of a NAT64 gateway" http://ecdysis.viagenie.ca

[10] C. Bao, C. Huitema, M. Bagnulo, M Boucadair and X. Li, "IPv6 addressing of IPv4/IPv6 translators", IETF, October 2010. ISSN: 2070-1721 (RFC 6052)

[11] M. Bagnulo, A. Garcia-Martinez and I. Van Beijnum, "The NAT64/DNS64 tool suite for IPv6 transition", IEEE Communications Magazine, vol. 50, no. 7, July 2012, pp. 177-183. doi:10.1109/MCOM.2012.6231295

[12] Free Software Fundation, "The Free Software Definition", http://www.gnu.org/philosophy/free-sw.en.html

[13] Open Source Initiative, "The Open Source Definition", http://opensource.org/docs/osd

[14] Cisco, "End user licence agreement", http://www.cisco.com/en/US/docs/general/warranty/English/EU1KEN_.html

[15] Juniper Networks, "End user licence agreement", http://www.juniper.net/support/eula.html

[16] G. Lencse and S. Répás, "Performance analysis and comparison of the TAYGA and of the PF NAT64 implementations" *Proc. 36th International Conference on Telecommunications and Signal Processing* (TSP-2013, July 2-4, 2013) Rome, Italy, pp. 71-76.

[17] G. Lencse and S. Répás, "Performance analysis and comparison of different DNS64 implementations for Linux, OpenBSD and FreeBSD", *Proc. 27th IEEE International Conference on Advanced Information Networking and Applications* (AINA-2013, March 25-28, 2013) Barcelona, Spain, pp. 877-884.

[18] "TAYGA: Simple, no-fuss NAT64 for Linux" http://www.litech.org/tayga/

[19] Theo de Raadt, "The OpenBSD 5.1 Release", May 1, 2012, ISBN 978-0-9784475-9-5, http://www.openbsd.org/51.html

[20] Simon Perreault, "[Ecdysis-discuss] NAT64 in OpenBSD", http://www.viagenie.ca/pipermail/ecdysis-discuss/2011-October/000173.html

[21] P. N. M. Hansteen, *The Book of PF: A No-Nonsense Guide to the OpenBSD Firewall*, 2nd ed., San Francisco: No Starch Press, 2010. ISBN: 978-1593272746

[22] "IPv6 HowTo for Backfire and Attitude Adjustment until 12.09", http://wiki.openwrt.org/doc/howto/ipv6

[23] "OpenVPN Community Wiki and Tracker", https://community.openvpn.net/openvpn/wiki/IPv6

[24] S. J. Vaughan-Nichols, "VoIP and instant messaging problem looming: Skype doesn't support IPv6", http://www.zdnet.com/voip-and-instant-messaging-problem-looming-skype-doesnt-support-ipv6-7000007058/