Port Number Consumption of the NAT64 IPv6 Transition Technology

Sándor Répás, Tamás Hajas and Gábor Lencse

***Abstract*—Due to the depletion of the global IPv4 address pool, the internet service providers will be able to supply their new clients with IPv6 addresses only in the near future. The application of the DNS64 and NAT64 technologies can enable the IPv6 only clients to communicate with the still dominant IPv4 only servers. However, the clients of certain applications such as HTTP and FTP use multiple sessions and thus they consume multiple ports. This phenomenon may cause a lack of ports situation at the NAT64 gateway. Therefore the port consumption of the different applications is an important design parameter of the NAT64 gateways. In this paper, the port consumption of different NAT64 compatible applications was measured. It was also determined what factors can influence the port consumption of a web or an ftp client. The detailed description of our measurement method is given. Our results can give a valuable help for careful design and configuration of a NAT64 gateway.**

***Keywords*—IPv6 deployment, IPv6 transition solutions, NAT64, port numbers, lack of ports, Packet Filter, TAYGA.**

I. Introduction

It is well known since 1992 that the world is going to run out of IPv4 addresses. IPv6 addresses could have been used instead and they are still available since 1998 [1]. But the deployment of the IPv6 protocol has just been started and there are only few websites which use this protocol. According to Google statistics only the 2.8% of the network traffic used IPv6 protocol on January 1, 2014 [2]. The depletion of the global IPv4 address pool[1] will be a huge driving force for the deployment of IPv6 in the forthcoming years.

ISPs can hardly provide IPv4 addresses to new servers and will not able to provide them to new subscribers. The lack of IPv4 addresses may set back the spread of Internet and Internet of things.

Manuscript received February 14, 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.

The transition to IPv6 will take a long time, and it is a firm fact that IPv4 and IPv6 will be in use parallel. The authors are convinced that using NAT64 [5] and DNS64 [6] together is one of the best solutions to have proper communication between an IPv6 only client and an IPv4 only server. There is a need for further investigations in order to use NAT64 appropriately. (A brief description of its operation will be provided later.)

We have also tested how the most used application protocols could work through a NAT64 gateway [7]. In this paper, we publish our research results on the port consumption of those applications that were found NAT64 compatible in [7].

IP based applications need dedicated source and destination ports to identify each other in a session. The TCP and UDP headers each contain only (twice) two bytes for this purpose. It means that 2^{16}=65536 is the theoretical limit of usable number of ports. Some applications need just a few ports, but others (e.g.: opening a web page in a web browser) need more. It is not an important topic, if just a few users sit behind a NAT64 gateway, but in a provider environment it can be a serious problem.

The authors of [8], [9] and [10] have already made some research related to the port consumption of different applications, but not in a NAT64 environment. We believe that further research on NAT64 is needed to help network operators.

The remainder of this paper is organized as follows: first, the port numbers and their problems related to NAT64 solution is described, second, the research results related with port numbers are surveyed, third, the applications and NAT64 implementations are selected for testing, fourth, the description of the test network and the testing method of each applications are given together with their results, and finally, our conclusions are presented.

The volume of the IPv6 traffic of the Internet is expected to explode in the following years 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 and configuring the appropriate IPv6 transition solution for their networks.

II. The Port Number Consumption of NAT64

In a communication process, it is inevitable to identify the two parties. In a TCP/IP environment, the applications use the combination of the source address + source port and destination address + destination port for this purpose. Some applications use multiple sessions at the same time, and all of the sessions need different identifiers, consequently different ports. This is especially significant in the case of browsing a web page which contains many images or other embedded objects.

If an IPv6 only client connects to an IPv4 only server, it needs to use port numbers, too. To provide connectivity be-

tween an IPv6 only client and an IPv4 only server one can use a stateful NAT64 gateway combined with a DNS64 server. For an introduction to DNS64+NAT64, see our paper [7].

A stateful NAT64 gateway replaces the source port number of the incoming IPv6 packet to an unused one, and registers the original IPv6 source address, source port, the new IPv4 source address and the new IPv4 source port in its internal database. Then it sends out an IPv4 packet using its own IPv4 address as source address thus the IPv4 server addresses its reply to the NAT64 gateway. When the gateway receives the IPv4 answer it can look up the original source IP address and port number of the packet and by using these data it can create the IPv6 packet and send the IPv6 answer to the originating client.

Internet Assigned Numbers Authority (IANA) is the responsible organization, who coordinates the usage of the port numbers [11]. The full 0-65535 port range is divided into three segments:

- System ports (earlier: Well Known Ports): 0-1023
- User ports: (earlier: Registered ports): 1024-49151
- Dynamic ports (known as Private ports, too): 49152-65535

Earlier the Dynamic port range was used for the NAT process. Nowadays the NAT44 and NAT64 gateways use the 1024-65535 range. All of the clients behind the NAT gateway have to share this range. If this range is not enough to serve all of the clients of the NAT64 gateway because it has many clients then it needs more public IPv4 addresses to multiply the numbers of usable ports.

Even though we focus on NAT64, the port number consumption problem exists with NAT44, too.

III. A Short Survey of the Current Research Results

The port number usage of different applications was tested in 2010 by Fourcot and Grelot [8]. First, the authors collected statistics on a Linux gateway from the conntrack table by a script in a 150 user environment. Then they made an experiment with limited number of ports. They found that web browsing and P2P file sharing are the most port eager applications.

The consequence of lack of ports was tested in 2011 by Kraemer and Perrin in a DS Lite environment [9]. They concluded that the lack of ports resulted in a very poor user experience.

Miyakawa presented some spectacular experiment by a web browser with limited number of session usage [10]. He also made a survey about how many sessions a web site uses. He concluded that, some web sites use few hundreds of TCP sessions concurrently and we have to carefully define how many sessions will be allowed per user using any address sharing scheme. For this reason, further study is needed to satisfy the customers.

It is inevitable to do investigations in a NAT64 environment, because the port consumption of the various applications is a serious aspect of selecting and configuring an appropriate NAT64 gateway. We believe that our results could be utilized with many IP address sharing solutions.

IV. The Scope of our Measurements

A. Application Selection for Testing

Only NAT64 compatible applications were chosen for the tests. These applications and their detailed testing process can be found in [7].

Protocols and their attributes are listed in table I.

TABLE I. Application Protocols, Protocols and Ports

Application protocol	Transport Protocol	Standard port
HTTP	TCP	80
SMTP	TCP	25
POP3	TCP	110
IMAP4	TCP	143
Telnet	TCP	23
SSH, SCP	TCP	22
FTP	TCP	20,21
OpenVPN	UDP/TCP	1194
RDP	TCP	3389
Syslog	UDP	514

B. NAT64 Implementation Selection

Our considerations of the selection of the NAT64 implementations can be found in [7]. The selected implementations were: TAYGA [12] under Linux and the Packet Filter [13] of OpenBSD.

While TAYGA is a so called one to one NAT64 solution and needs an external stateful NAT44 solution (iptables under Linux), Packet Filter is a stateful NAT64 solution. Their most important characteristics can be found in [7] and their performance were analyzed and compared in [14].

C. Our Measurements

The aim of our tests was to examine the port number consumption of the selected applications at the NAT64 gateway. All of the tests were executed both with TAYGA and Packet Filter using as a NAT64 gateway, but the results were absolutely the same.

Port consumption of the HTTP protocol was tested with real web pages, while the other protocols were tested with dedicated test servers in the laboratory.

The measurement methods and the results are detailed in the rest of the paper.

V. HTTP Measurements and Results

Ten web sites without IPv6 addresses were selected from [15] to be used for the measurements. The domain names of the selected web sites are shown in table II.

TABLE II. Tested web sites

Web pages
amazon.com
ask.com
baidu.com
ebay.com
linkedin.com
live.com
sohu.com
taobao.com
twitter.com
yandex.ru

Different combinations of operating systems and web browsers were tested to explore what parameters may influence the port number consumption. The combinations were the following:

Windows 7 Enterprise:

- Mozilla Firefox 21.0
- Internet Explorer 10.0.9200.16576
- Google Chrome 27.0.1453.110 m
- Opera 12.15

Debian Wheezy 7.1 Linux + KDE 4.8.4:

- Iceweasel 17.0.6
- Konqueror 4.8.4
- Google Chrome 28.0.1500.45

Ubuntu 12.04 LTS Linux:

- Mozilla Firefox 21.0

A. *HTTP measurements*

The topology of the network is shown in Fig. 1. The different web browsers were always executed on the IPv6 client computer and the port usage measurements were taken on the NAT64 gateway. The measurements were automated by scripts including the remote starting and stopping of the appropriate sub-scripts with appropriate timing. The timing of a measurement is shown in Fig. 2.

A single measurement took 65 seconds. First the metering was started with one second delay, and two more seconds later the web browser was started. The web browser was closed after 60 seconds, and finally the metering was stopped three more seconds later. These guard intervals were used to avoid synchronization problems as (remote) starting and stopping of processes took nonzero time.

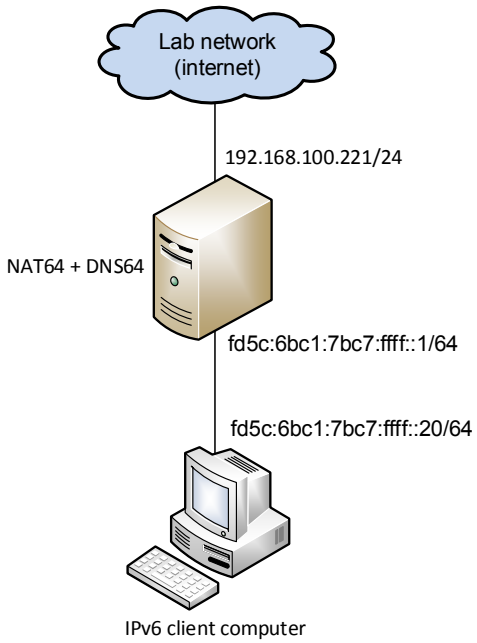

Figure 1. Topology of the HTTP test network.

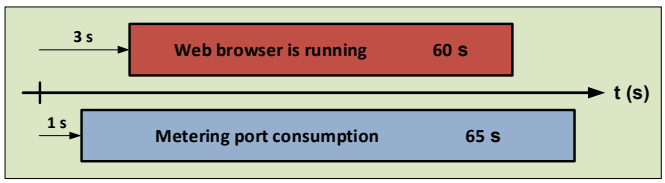

Figure 2. Timing of a measurement.

All of the measurements were performed eleven times by running a script on the client computer. Now, we start with the scripts of the Linux clients as they are easier to follow. Windows clients did the same task but they were more complicated due to the less convenient remote execution of batch files.

1) *Linux Clients*

The shell script below (**starter.sh**) was started and then this script started all the necessary processes:

```
#!/bin/bash
a=[Actual URL]
b=[Actual folder]
for c in {1..11}
do
  ssh root@fd5c:6bc1:7bc7:ffff::1 \
    ./http_metering/$b/$a/http_meter.sh $c &
  sleep 3
  [actual browser] www.$a > /dev/null 2>&1 &
  sleep 60
  kill $(ps aux | grep [actual browser] | \
    awk '{print $2}') > /dev/null 2>&1
  sleep 5
done
```

Notes:

- The values of **[Actual URL]** and **[Actual folder]** were set manually.
- This script is simple as parameters could be conveniently given to the remotely executed script.

The cache of Google Chrome was emptied from the script, while caching was simply switched off in the other browsers (as they allowed it).

2) *Windows Clients*

The script below (**starter.bat**) was started and then this script started all the necessary processes:

```
set A=[Actual URL]
set B=[Actual folder]
FOR /L %%C IN (1,1,11) DO (
putty.exe -ssh root@fd5c:6bc1:7bc7:ffff::1 -pw lab –m
C:\Users\Administrator\Desktop\http_metering\%B%\%A%\rem
ote%%C.bat | start
C:\Users\Administrator\Desktop\http_metering\http_get.ba
t
)
```

Notes:

- The core of the **FOR** cycle is a single command line (within parenthesis). It was broken up into several lines by the word processor only.
- The single command line contains two commands (**putty.exe** and **start**) issued simultaneously with the help of the pipe ("|") sign.
- Different batch scripts were used for every single remote execution as we could not figure out how to give parameters to the script executed remotely by using **putty.exe**.

While the script used the SSH protocol to remotely start the port number measurement script on the NAT64 gateway, it started the web browsing with another script simultaneously. The script below (**http_get.bat**) was responsible for opening the given web page and closing the browser 60 sec-

onds later. The caching function of Firefox and Opera web browsers were switched off, and the cache of Internet Explorer and Google Chrome were erased with commands in the previous script.

```
timeout 3
start www.[Actual URL]
timeout 60
taskkill /f /im [browser].exe
exit
```

The eleven **remote[1-11].bat** scripts, which were used in the **starter.bat** script, were generated with the following script:

```
set A=[Actual URL]
set B=[Actual folder]
cd http_metering
cd "%B%"
mkdir "%A%"
cd "%A%"
FOR /L %%C IN (1,1,11) DO (
echo ./http_metering/%B%/%A%/http_meter.sh %%C > re-
mote%%C.bat
)
```

3) Measurements with TAYGA

Two shell scripts were used for the measurements on the NAT64 gateway. The following script (**mkall.sh**) was started before every series of measurements:

```
#!/bin/bash
a=[Actual URL]
b=[Actual folder]
cd /root/http_metering/$b
mkdir $a
cd $a
for i in {1..11}
do
  mkdir $i
done

cat > http_meter.sh <<EOF
#!/bin/bash
conntrack -F
sleep 1
for j in {1..65}
do
  cat /proc/net/ip_conntrack > \
    /root/http_metering/$b/$a/\$1/writeout\$j
  sleep 1
done
EOF
chmod 700 http_meter.sh
```

This script was responsible to prepare the necessary directory structure and the measurement script.

The generated scripts emptied the conntrack table first, and then they recorded the contents of the conntrack table in every seconds.

The processing of the generated data files was done with the following script:

```
#!/bin/bash
a=[Actual URL]
b=[Actual folder]
cd /root/http_metering/$b/$a
for h in {1..11}
do
  for i in {1..65}
  do
    c=$(cat /root/http_metering/$b/$a/$h/writeout$i \
      | grep src=10. | wc -l)
    printf "%s\t" "$c" >> \
      /root/http_metering/$b/$a/results
  done
  printf "\n" >> /root/http_metering/$b/$a/results
done
```

This script first filtered the private IPv4 addresses which were generated by TAYGA, and then it counted the number of them. These numbers were represented the used ports in every seconds. The numbers were finally stored in the **results** file.

4) Measurements with Packet Filter

Two similar scripts were used for the measurements. The following script (**mkall.sh**) was started before every series of measurements:

```
#!/bin/ksh
a=[Actual URL]
b=[Actual folder]
cd /root/http_metering/$b
mkdir $a
cd $a
for i in $(jot 11 1)
do
  mkdir $i
done

cat > http_meter.sh <<EOF
#!/bin/ksh
pfctl -Fs #empty state table
sleep 1
for j in \$(jot 65 1)
do
  pfctl -ss > /root/http_metering/$b/$a/\$1/writeout\$j
  #print out state table
  sleep 1
done
EOF
chmod 700 http_meter.sh
```

The processing script was the following:

```
#!/bin/ksh
a=[Actual URL]
b=[Actual folder]
cd /root/http_metering/$b/$a
for h in $(jot 11 1)
do
  for i in $(jot 65 1)
  do
    c=$(cat /root/http_metering/$b/$a/$h/writeout$i \
    | grep tcp | grep 192.168.100.221 | wc -l)
    printf "%s\t" "$c" \
    >> /root/http_metering/$b/$a/results
  done
  printf "\n" >> /root/http_metering/$b/$a/results
done
```

B. HTTP results

1) Preleminary overview

A first overview of the results can be found in Table III. The figures give the maximum number of ports consumed by the given browser opening the given URL. Our very first observations are:

- There is a huge difference between the least port consuming twitter.com (9-16 ports, depending on the browser) and the most port consuming sohu.com web page (122-198 ports).
- While the port consumption of some sites highly depends on the browser (e.g. for baidu.com, it is 8 with Windows/Explorer and 27 with Ubuntu/Firefox) the

TABLE III. MAXIMUM PORT USAGE OF DIFFERENT HTTP CLIENTS

	amazon.com	ask.com	baidu.com	ebay.com	linkedin.com	live.com	sohu.com	taobao.com	twitter.com	yandex.ru
Windows/Firefox	74	28	25	42	18	26	198	135	11	26
Windows/Opera	58	24	16	32	14	37	140	73	13	20
Windows/Explorer	67	24	8	37	15	21	123	77	14	19
Windows/Chrome	48	25	10	30	14	23	122	67	11	18
Debian/Chrome	52	32	13	48	23	39	122	72	16	23
Debian/Iceweasel	55	31	25	43	19	25	186	131	12	20
Debian/Konqueror	35	26	11	39	12	21	122	135	9	17
Ubuntu/Firefox	52	33	27	40	20	25	166	117	11	21

port consumption varies much less for others (e.g. for ask.com, it is always between 24 and 33).

- Depending on the URL, the port consumption of the same browser may differ significantly under different operating systems (e.g. for ebay.com, Windows/Chrome used 30 ports and Debian/Chrome used 48 ports).

Besides to the observations above, we have to lay down that these figures are maximum values and if they would be used instead of the real distribution of the port number consumption, it may lead to too conservative estimations (depending on the nature of the distribution). Thus, on the one hand, we need to dive deeper. However, on the other hand, there would be very desirable to replace the distributions by a single number, because there are so many parameters that consideration of the complete distributions may prevent us from understating the main point.

2) Can the average values reflect the distribution?

Two significantly different cases were chosen for the graphical representation of the minimum, maximum, average values. Figures 3 and 4 show the [average - standard deviation, average + standard deviation] intervals with the error bars, too. Fig. 3 suggests that the value of the standard deviation (typical value: about 8) is quite small compared to the value of the average (typical value: about 139) thus the average can represent the distribution well enough. Fig. 4 shows a counter example: here the value of the standard deviation is comparable with that of the average (especially between 40 and 60 seconds).

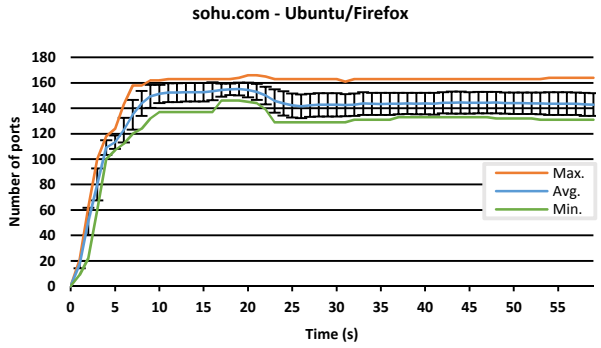

Figure 3. Port usage of sohu.com with Ubuntu/Firefox.

However, if we would like to compare the port usage behavior of the different browsers in the function of time, the eight graphs showing the average port usage of the eight tested operating system and browser combinations are more than enough in a single figure (further lines would make the figure totally indigestible).

3) How does the browser influence the port number consumption?

The average values of port consumption of all the tested operating system and browser combinations with sohu.com web page can be seen on Fig. 5.

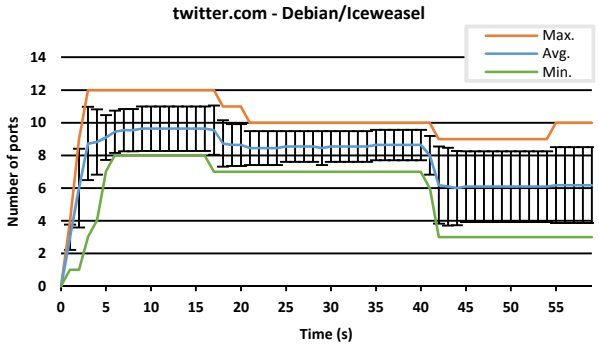

Figure 4. Port usage of twitter.com with Debian/Iceweasel.

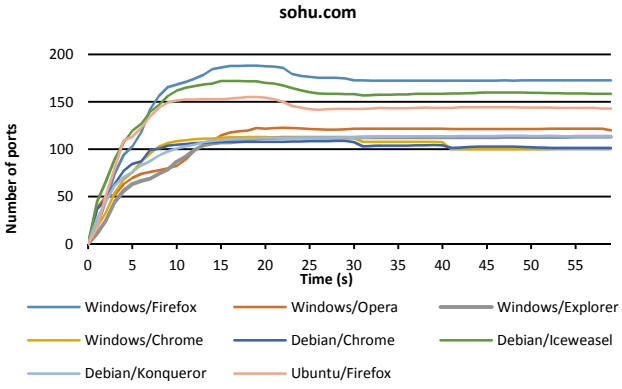

Figure 5. Average port usage of sohu.com with different browsers.

The differences of the combinations are clearly visible on the graph. For example, the maximum value of the used port numbers is 188.18 (Windows/Firefox), when the minimum value is 107.82 (Debian/Chrome), which is a significant difference.

Linkedin.com uses much less number of ports as seen on Fig. 6. In this case the difference is clearly visible, too. The maximum value is 18.36 (Debian/Icewiesel), when the minimum one is 10.82 (Debian/Konqueror), which is also a significant difference.

4) How does the operating system influence the port number consumption?

There were two different web browsers (Firefox and Chrome) that were tested on two platforms. While the results were usually the same with the majority of web pages, significant differences can be discovered with some other pages. For example, the results of Firefox with amazon.com can be seen on Fig. 7. The maximum value of the average port counts is 55 (Windows), when the minimum is 40.18 (Ubuntu).

The results of Chrome on two different operating system can be seen on Fig. 8.

The maximum value of the number of used ports is 29.81 on Debian system, while at the same time the value on Windows system is 21.27, which is a significant difference.

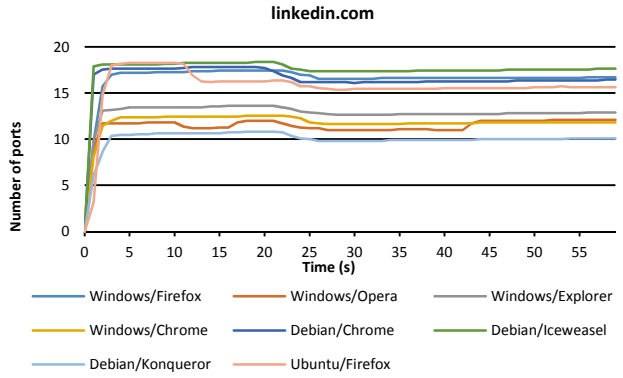

Figure 6. Average port usage of linkedin.com with different browsers.

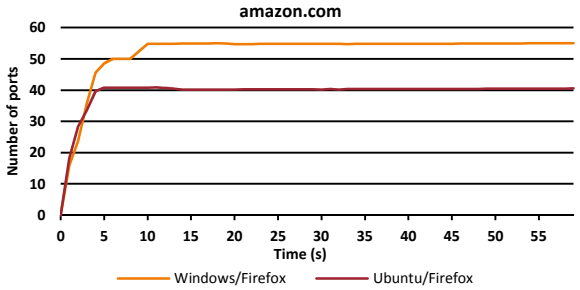

Figure 7. Average port usage of amazon.com with the same browsers on different operating systems.

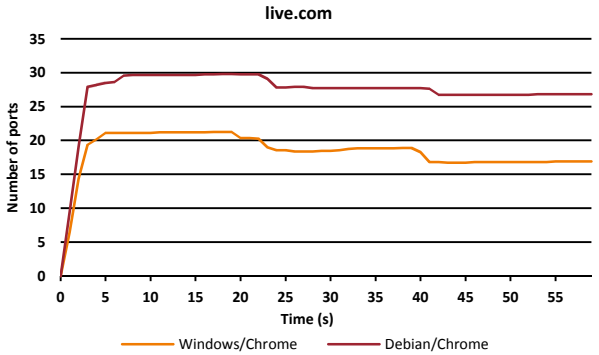

Figure 8. Average port usage of live.com with same browsers on different operating systems.

VI. FTP Measurements and Results

The topology of the network is shown in Fig. 9. The IPv4 only server at the top of the figure was played the FTP server role.

The scripts of the HTTP measurements were used with the necessary modifications. Different combinations of operating systems and FTP clients were tested like with the HTTP protocol. The combinations were the following:

Windows 7 Enterprise:

- FileZilla 3.7.3 (with 1 and 10 parallel downloadable files settings)
- Total Commander 8.01

Debian Wheezy 7.1 Linux:

- command line ftp 0.17-27
- Midnight Commander 4.8.3-10

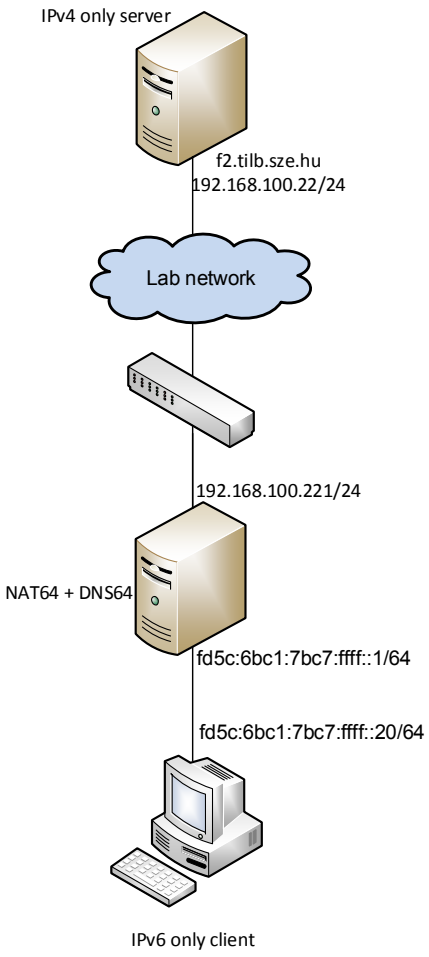

Figure 9. Topology of the FTP test network

The following file sizes and numbers were used for the measurements:

- 100 MB
- 30 x 1 MB
- 100 x 1 MB
- 500 MB

A. FTP results

The results can be found in Table IV. Evaluation of the results:

- The number of used ports is nearly proportional with the number of transferred files.
- The number of ports is always somewhat greater, then the number of transferred files.
- Midnight Commander transfers different directory listings, which is the explanation of the high value in the case of the transfer of just one file.
- Filezilla with the setting of 10 parallel transfers allowed uses 9 extra ports, in contrast with the 1 transfer case.

TABLE IV. MAXIMUM NEEDS OF PORTS OF DIFFERENT FTP CLIENTS

File size (MB)	100	30 x 1	100 x 1	500
Debian/FTP	2	32	102	2
Debian/MC	10	39	109	10
Windows/TC	3	32	102	3
Windows/Filezilla 1 par.	4	33	103	4
Windows/Filezilla 10 par.	-	42	112	-

VII. TELNET, SSH, SCP, OPENVPN, SMTP, POP3, IMAP4, RDP, SYSLOG

All of these protocols were tested mainly manually, because of the simplicity. The SCP protocol was tested with the same files as the FTP one. All of them used just one source port in the communication process.

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. The application of NAT64+DNS64 could be a good solution in this case, but the network architects have to consider the number of users and their typical network usage. For a good user experience, it is indispensable to allocate enough public IPv4 addresses for the NAT64 gateway to satisfy all users. The number of ports needed by a user mostly depends on what applications he/she uses. We have shown that HTTP and FTP applications can consume a large number of ports while others use just one per session.

The port consumption of web browsing varies from several tens to several hundreds and highly depends on the web site, but also depends on the browser and it may even depend on the client operating system, too.

The port consumption of the file transfers with FTP depends on the number of the transferred files, on the type of the FTP client and also on the permitted number of parallel transfers.

We conclude that the port number consumption of especially of the web applications is a crucial parameter in the capacity planning of NAT64 gateways and therefore it deserves further analysis. We plan to do so and hereby we invite other researchers to take part in it.

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] S. Répás, T. Hajas and G. Lencse, "Application Compatibility of the NAT64 IPv6 Transition Technology" Proc. 37h International Conference on Telecommunications and Signal Processing (TSP-2014, July 1-3, 2014) Berlin, Germany

[8] F. Fourcot, B. Grelot, “Migrating to IPv6 with Address+Port translation”, SLR Project Report, March 2010, https://svn.fperrin.net/v6fication/documentation/rapport-Fourcot-Grelot.pdf

[9] I. Kraemer, F. Perrin, “Impact of the lack of ports in a DS-Lite architecture”, https://svn.fperrin.net/v6fication/article/simulation.pdf

[10] S. Miyakawa, “IPv4 to IPv6 Transformation Schemes”, IEICE Transactions on Communications, vol. E93-B, no 5, May 2010, pp. 1078-1084

[11] “Service Name and Transport Protocol Port Number Registry” http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

[12] “TAYGA: Simple, no-fuss NAT64 for Linux” http://www.litech.org/tayga/

[13] 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

[14] G. Lencse and S. Répás, "Performance Analysis and Comparison of the TAYGA and of the PF NAT64 Implementations", Proceedings of the 36th International Conference on Telecommunications and Signal Processing (TSP 2013), (Rome, Italy, 2013. July, 2-4.), 71-76.

[15] Alexa, "The top 500 sites on the web", http://www.alexa.com/topsites