Stability Analysis and Performance Comparison of Five 6to4 Relay Implementations

Sándor Répás, *Member, IEEE*, Viktor Horváth, and Gábor Lencse, *Member, IEEE*

***Abstract*—Even though the present form of IPv6 has been existing since 1998, the adoption of the new protocol has been very slow until recently. To help the adoption of the IPv6 protocol, several transition technologies were introduced. The 6to4 protocol is one of them, and it can be used when an IPv6 enabled host resides in an IPv4 only environment and needs to communicate with other hosts in such circumstances or with native IPv6 hosts. Five open source 6to4 relay implementations were investigated: Debian Linux – sit, Debian Linux – v4tunnel, OpenWrt – sit, FreeBSD – stf, NetBSD – stf. The measurement method is fully described including our measurement scripts and the results of the measurements are disclosed in detail. The measurements have shown that there are major differences between the different types of implementations.**

***Index Terms*—6to4 relay, IPv6 transition, network communication, performance evaluation, stability analysis**

I. INTRODUCTION

FOR more than two decades it is a known fact, that the size of the IPv4 address space is insufficient [1-2]. The lack of the IP addresses withholds the spread of the Internet and causes social and economic damage.

To prevent the IP address exhaustion, a new version of the Internet Protocol, IPv6 has been developed. IPv6 was standardized in 1998 and published in RFC 2460 [3], but it has not been widespread adopted. According to the statistics, less than 8% of the total amount of the traffic reached the Google servers used IPv6 protocol in December 10, 2015 [4]. Several tools and solutions have been developed to slow down the process of the address exhaustion. The Dynamic IPv4 allocation [5], the Classless Inter-Domain Routing (CIDR) [6], the Network Address Translation (NAT) [7], the Carrier-grade NAT (also called NAT444) [8], different type of proxies or Application Level Gateways (ALG), new policies of the IPv4 address transfers [9] successfully delayed the problems generated by the IP address exhaustion, but all of them generated other problems [5].

Three of the five Regional Internet Registries (RIR) already run out of their IPv4 address spaces [10]. The five RIRs have only 5.2 /8 ranges in total, whereas the IANA does not have more address space to assign to the five RIRs since 3 February 2011 [11]. The RIRs work according to strict policies and for a service provider, it is a harder task than ever to get IPv4 address spaces. The speed up of the transition to the new protocol is inevitable. Several IPv6 transition techniques have been developed, which can help the process in different phases of the adoption of the new protocol on the Internet.

There are different situations to solve during the coexistence of the two versions of the IP protocol in the different phases of the transition process:

In theory, the best solution is the Dual Stack (DS) transition method [12], but with the requirements that the two communicating hosts and the network between them have to support a common version of the IP protocol, and because of the IPv4 exhaustion, there is not enough IPv4 address to use this solution. The communicating hosts need both version of the IP addresses and it is almost impossible to provide enough public IPv4 addresses for the clients. Thus, even though it could have been the best solution, now it is too late for using DS as an IPv6 transition method.

In a situation where an IPv6 only client computer needs to communicate with an IPv4 only server, the DNS64 [13] and NAT64 [14] combination is a good solution. The performance, the stability and the application compatibility of some open source implementations of DNS64/NAT64 are examined and proved in [15-17].

If two IPv6 enabled hosts need to communicate with each other over an IPv4 network, they can use different tunneling methods. The 6in4 (also called manual tunnel) [18] with tunnel brokers [19-20], 6rd [21], Teredo [22] ISATAP [23] and 6to4 [24] have different requirements, benefits and drawbacks.

The above list is not exhaustive and a good survey of the different transition techniques can be found in [25].

In this paper, we deal with the 6to4 IPv6 transition solution. The remainder of this paper is organized as follows: first, some properties of the 6to4 transition technique are introduced, second, a short survey of the results of the most current publications is given, third, the selected 6to4 relay implementations are introduced, fourth, our test environment is described, fifth, the performance measurement method of the different implementations is detailed, sixth, the results are presented and discussed, seventh, the comparison of our results is presented, finally, our conclusions are given.

Manuscript received December 21, 2015, revised May 18, 2016.

S. Répás is with the Széchenyi István University, Győr, 9026 Hungary (phone: 36-30-459-9292; e-mail: repas.sandor@sze.hu).

V. Horváth was with the Széchenyi István University, Győr, 9026 Hungary (e-mail: vhorvath@biztributor.hu).

G. Lencse is with the Széchenyi István University, Győr, 9026 Hungary (E-mail: lencse@sze.hu).

II. The 6to4 Transition Technique

The 6to4 transition technique uses automatic tunnels, encapsulates the IPv6 packets into IPv4 packets (using protocol number 41, as the configured IPv6 over IPv4 tunnel [26]) [24]. The main advantage of the automatic tunneling is the unnecessity of the manual configuration of the endpoint address of the tunnel. Automatic IPv6-over-IPv4 tunneling determines the IPv4 tunnel endpoint address from the IPv4 address embedded in the destination address of the IPv6 packet being tunneled. 6to4 protocol uses the reserved 2002::/16 6to4 prefix to determine if a 6to4 tunnel creation is necessary [27]. A 6to4 address is an IPv6 address constructed using a 6to4 prefix. The first 16 bits of the 6to4 address contain the 2002 hexadecimal value, whereas the next 32 bits contain the IPv4 address of the 6to4 tunnel endpoint. The next 16 bits can be used to create subnets, and the final 64 bits of the 6to4 address contain the interface ID.

A 6to4 router is an IPv6 router supporting a 6to4 pseudo-interface. It is normally the border router between an IPv6 site and a wide-area IPv4 network, whereas the 6to4 pseudo-interface is the point of the encapsulation of IPv6 packets in IPv4 packets (with other words: the tunnel end-point) [24]. If a 6to4 host has to communicate with a non 6to4 host (for example: native IPv6, Teredo) it needs to use a 6to4 relay router.

Several operating systems can work as a 6to4 router or 6to4 relay router, but for the correct operation, the 6to4 routers and relay routers need public IPv4 addresses.

A 6to4 relay router can be private or public. Public 6to4 relays use the 192.88.99.1 anycast address [28] from the 192.88.99.0/24 6to4 Relay anycast address range [29]. An estimation of the 6to4 relay routers published in 2006 [30]. According to the publication, 8 autonomous systems (AS-es) advertised the 192.88.99.0/24, whereas 6 AS-es advertised the 2002::/16 networks. At the end of the year 2014 these values were 14 and 11, according to the RIPEstat database [31].

It is a good practice, if an Internet Service Provider (ISP) provides a 6to4 relay for its customers in addition to other transition solutions. In this case the relay does not have to be public, and it can use the well-known anycast address, or a network specific address.

Though some security weaknesses are known of the 6to4 transition technique [32], its advantage is that it helps the implementation of the IPv6 protocol without the cooperation of the ISP. This is the reason why we insist that 6to4 is still indispensable in several countries including Hungary. Although 6rd [33] eliminated some of the weaknesses of 6to4, the price of the improvements was that 6rd can only be implemented by the ISPs, and it cannot be used without the cooperation of the ISP of the user at all. We note that the second author of the RFC defining 6rd [33] recommended to move 6to4 to historic status in 2011 [34] and his efforts were only partially successful after several years because not 6to4 itself, but only the anycast prefix for 6to4 relay routers was deprecated in 2015 [35]. Whereas this seems to be a good decision considering the rapid deployment of IPv6 in certain countries (e.g. USA, China), we contend that it was done way too early considering the slow deployment of IPv6 in some other countries including Hungary, too. Despite the depletion of the public IPv4 address pool, the most ISPs in Hungary are rather reluctant to step forward towards IPv6. (What is even worse, it became a common practice that ISPs take away the public IPv4 address from their customers, and give private ones instead. The average user is OK with using CGN, and those who do not like it, will get back a public IPv4 address.) Thus an average countryside home user (one residing not in Budapest) is not able to get IPv6 Internet access. How can this user get access to the IPv6 Internet? We see the following possibilities:

- Use an explicit tunnel with a tunnel broker, however it requires registration and configuration.
- Use 6to4, which is a kind of automatic tunnel and is supported by several operating systems and SOHO routers and thus the user can access IPv6 only sites without any effort.
- Use Teredo as last resort. (But it is intended to be used as a last resort only.)

We agree that 6to4 is not a good solution, but as there is no real replacement, we consider it is still to be kept as working in those areas where the IPv6 deployment is still in its infancy and there is no other way for the clients to reach IPv6 internet without tunnel registration and explicit configuration. Therefore the performance analysis of 6to4 relays is still interesting for those network administrators who are willing to help these clients. We note that dimensioning a 6to4 relay is not an easy task because it is hard to predict where the return traffic will cross the border of the IPv6 Internet and IPv4 Internet. This is why it is crucial to have information about the performance and stability of different free software 6to4 relay implementations.

We also admit that many users of 6to4 may experience operational problems. Section 3 of RFC 6343 [36] mentions measurements reporting high TCP connection failure rate. There are 9 possible reasons were identified. We mention only two of them: e.g. firewalls may filter out protocol number 41, or some ISP may advertise 192.88.99.0/24 but not forward 6to4 traffic for “alien” networks, etc. Section 4 provides appropriate guidelines for vendors, network operators, and ISPs to eliminate the particular issues. Thus 6to4 may be used if all parties take enough care. Unfortunately, the communication of two computers may fail due to the malpractice of a third party because of asymmetric routing.

More details of the operation of the 6to4 technique can be found in the publication [37], and in the related RFCs ([24], [29] and [32]).

III. A Short Survey of Current Research Results

There are a lot of publications about IPv6 and several of them related to the transition to the IPv6 protocol.

There is a very good survey about the state of IPv6 adoption with measurement methods in [38]. The authors of the article used excellent methods for the survey, but the data in it is a little outdated today. A newer, and also very good survey can

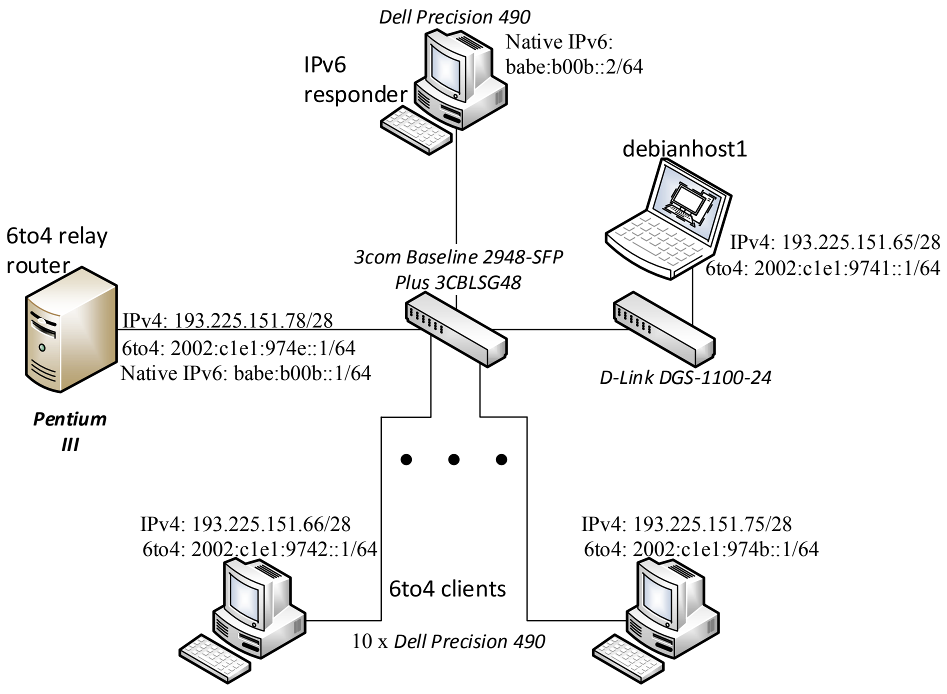

Fig. 1. Topology of the test network.

be found in [39]. The two papers give a good overview about the progress of the transition process.

There are several publications about comparison of different tunneling based transition methods.

In [40] the performance of both the ISATAP and the 6to4 tunneling solution is compared on a Windows XP and Windows Server 2003 based test-bed network. The authors used UDP streaming and ICMP to measure and compare the throughput, the End to End Delay (E2ED), the jitter and the Round Trip Time (RTT) performance characteristics. The final conclusion found the ISATAP protocol significantly more efficient.

Sans and Gamess carried out a performance comparison of the native IPv6 protocol and the following tunneling methods: ISATAP, 6to4, 6rd and Teredo on a test network was built on Linux computers and different numbers of Cisco routers [41]. The authors tested the throughput and the RTT with UDP and TCP protocol both on Ethernet and fast Ethernet network. They concluded, the best choice is native IPv6 but if native IPv6 cannot be used, ISATAP, 6to4, and 6rd are good possibilities. Selecting one tunneling technology over the other depends on many factors. Teredo was presented as the less good solution, whereas, Teredo is the only choice when the hosts to be connected are using private IPv4 addresses and are helped by a NAT server to reach the Internet.

Shah and Parvez performed simulations about the performance of native IPv6, dual stack, 6in4 and 6to4 [42]. The authors used OPNET Modeler (now Riverbed Modeler [43]) to investigate the TCP delay, throughput and response time of the different methods. Naturally, the native IPv6 produced the best results, whereas the second one was the 6to4.

There is a good comparison of the performance of the Windows Server 2008 and 2012 6to4 and 6in4 tunnels in [44]. The authors used UDP and TCP and three games to compare the throughput, the jitter and the delay of the two tunneling methods, but they did not collect data about the resource usage on the computers.

The comparison of the TCP and UDP throughput, RTT, and tunneling overhead with native IPv4, native IPv6 and 6to4 tunneling can be found in [45]. The authors concluded that the 6to4 tunneling mechanism is a suitable method in the early part of the transition period.

The characteristics of the tunneled IPv6 traffic on the border of the Czech national research and education network (CESNET) were investigated in [46], whereas the traffic of the FUNET operated public 6to4 relay was analyzed in [47].

Narayan and Tauch investigated the 6to4 and configured tunnel performance characteristics on two different Linux and Windows operating system [44-46] in a test network.

The performance characteristics of Linux sit, FreeBSD stf, and NetBSD stf based 6to4 relay implementations were investigated in [37].

The performance of and stability of Debian Linux sit, OpenWRT sit and FreeBSD stf were analyzed in our conference paper [51], which is now extended by Debian Linux v4tunnel and NetBSD stf.

IV. TESTED IMPLEMENTATIONS

The following widely used open source [52] (also called free software [53]) operating systems and their 6to4 implementations were chosen for the tests: Debian Linux sit and v4tunnel [54], OpenBSD gif interface [55], FreeBSD stf interface [56], NetBSD stf interface [57], OpenWRT 6to4 plus kmod-sit packages [58]. The open source software can be freely used by anyone, and their licenses allow the performance benchmarks. These two arguments were the most important ones in our selection of the implementations for testing.

The following software versions were used:

- Debian 7.1.0_x86 – sit
- Debian 7.1.0_x86 – v4tunnel
- OpenWRT (Attitude Adjustment) 12.09_x86 – sit
- FreeBSD 9.1_x86 – stf
- NetBSD 6.1.2_x86 – stf

It was found during the preliminary tests that the OpenBSD system does not support the 6to4 transition mechanism.

V. TEST ENVIRONMENT

A. *Topology of the network*

An isolated test network was built for the performance and the stability measurements. The topology of the network can be seen in Fig. 1. Due to the isolation, any IPv4 and IPv6 addresses could be used on the network. The computer on the top of the figure played the role of the "internet" and responded all of the queries, and the queries were generated by the 10 client computers which can be seen on the bottom of the figure. These computers played the role of the large number of the clients. The clients sent their queries by 6to4 through the 6to4 relay router to the "internet" computer. These queries were generated different levels of load on the 6to4 relay computer during the measurement process. The load was tuned by the number of the active clients. The laptop and the connecting switch on the right side of the figure were used to control the experiments.

B. *Hardware configurations*

1000Base-TX connections were used on all of the network segments.

A specially low performance computer was built for the 6to4 relay computer so that the client computers could produce high enough load for overloading it. The main goal of the measurements was the comparison of the different implementations and not any hardware related investigation.

The configuration of the 6to4 relay computer was:

- Intel D815EE2U motherboard
- 800MHz Intel Pentium III (Coppermine) processor
- 128MB, 100MHz SDRAM
- Two TP-LINK TG-3269 REV 3.0 Gigabit PCI Ethernet NICs

All of the ten clients and the responder computer were Dell Precision 490 workstations with same configuration:

- DELL 0GU083 motherboard with Intel 5000X chip-set
- Two Intel Xeon 5140 2.33GHz dual core processors (in the responder: Intel Xeon 5160 3GHz)
- 4x1GB 533MHz DDR2 SDRAM (accessed quad channel)
- Broadcom NetXtreme BCM5752 Gigabit Ethernet controller (PCI Express)

C. *Software configurations*

Debian Linux 6.0.7 with 2.6.32-5-amd64 kernel and OpenBSD 5.3 64 bit version were installed on the clients, and the responder, respectively.

On the responder, NAT66 was used to simulate server computers with different IPv6 addresses. The following commands were used in the /etc/pf.conf file on the responder:

```
set timeout interval 2
set limit states 400000
pass in on bge0 inet6 from any to \
    2001:738:2c01:8000::/64 rdr-to babe:b00b::2
```

All of the client computers used sit or stf interfaces with the following setting in the /etc/network/interfaces file:

```
  auto sit0
  iface sit0 inet6 static
  address 2002:c1e1:9742::1- …974b::1
  netmask 64
gateway ::193.225.151.78
```

VI. MEASUREMENT METHOD

The load was generated by `ping6` commands with the following Bash shell script:

```
#!/bin/bash
i=`cat /etc/hostname | grep -o '[0-9]'`
for b in {0..255}
do
  rm -rf $b
  mkdir $b
  for c in {0..252..4}
  do
    ping6 2001:738:2c01:8000::193.$i.$b.$c \
      -c8 -i0 >> $b/6to4-193-$i-$b-$c &
    ping6 2001:738:2c01:8000::193.$i.$b.$c \
      -c8 -i0 >> $b/6to4-193-$i-$b-$c &
    ping6 2001:738:2c01:8000::193.$i.$b.$((c+1)) \
      -c8 -i0 >> $b/6to4-193-$i-$b-$((c+1)) &
    ping6 2001:738:2c01:8000::193.$i.$b.$((c+1)) \
      -c8 -i0 >> $b/6to4-193-$i-$b-$((c+1)) &
    ping6 2001:738:2c01:8000::193.$i.$b.$((c+2)) \
      -c8 -i0 >> $b/6to4-193-$i-$b-$((c+2)) &
    ping6 2001:738:2c01:8000::193.$i.$b.$((c+2)) \
      -c8 -i0 >> $b/6to4-193-$i-$b-$((c+2)) &
    ping6 2001:738:2c01:8000::193.$i.$b.$((c+3)) \
      -c8 -i0 >> $b/6to4-193-$i-$b-$((c+3)) &
    ping6 2001:738:2c01:8000::193.$i.$b.$((c+3)) \
      -c8 -i0 >> $b/6to4-193-$i-$b-$((c+3))
  done
done
```

During the preliminary measurements, the script was tuned to generate about 100% load on the CPU of the 6to4 relay computer with 10 clients.

The variable `i` contains the serial number of the actual client. The script contains two nested `for` cycles. The outer cycle with variable `b` from 0 to 255 runs 256 times, while the inner cycle with variable `c` from 0 to 252 (with stepping interval 4) runs 64 times. The core of the script contains 4 pairs of concurrent `ping6` commands. Each pair of them send out 8 ICMPv6 echo requests with almost zero interval, in parallel, whereas the first 7 of them are started asynchronously with the `&` parameter. The last `ping6` command at the end of the cycle is started normally thus the cycle waits for the execution of it. In a measurement, one client sends out 256*64*8*8= 1048576 ICMP echo requests in total to 256*64*4= 65536 different IP addresses.

In the series of measurements, the number of the clients was increased from one to ten. On the 6to4 relay computer, the `vmstat` command was used to log the CPU and memory consumption. For proper operation of the `vmstat`, -10 nice value was used.

We note that having no timeout specified, the ping command waited two RTTs and then it considered the missing replies as lost. As the RTTs were small, our packet loss rate can be considered as an upper bound of rate of the ultimately lost packets.

VII. MEASUREMENT RESULTS

The results are presented in similar tables for all the tested 6to4 implementations. A detailed explanation is given for the first table only – the others are to be interpreted in the same way.

A. Debian 7.1.0_x86 – sit

The results have been listed in Table I. The first row shows the number of clients that executed the test script at the same time. The potential load on the 6to4 relay was proportional with the number of the clients, but the actual number of the packets was less than that, because the measurement script does not start a new iteration until the 8th ping6 command is finished. The second row contains the packet loss ratio. Rows 3, 4 and 5 show the average, the standard deviation and the maximum value of the response time, respectively. The average and the standard deviation of the CPU utilization of the 6to4 relay computer are shown in the Rows 6 and 7. Row 8 contains the memory consumption of the 6to4 process on the relay computer. (This parameter can be measured with high uncertainty, because its value is very low and other processes than the 6to4 relay implementation may also influence the size of the used memory of the computer.) The last row shows the number of forwarded packets per seconds.

The graphical representation of the forwarded packets per second and the CPU utilization are shown in Fig. 2.

Evaluation of the results:

Despite the fact that packet loss occurred in all cases, the proportion of it was always very low and it increased with more clients. (The maximum value of it was 0.061% with ten clients, which means about 6 packets from 10.000 packets were lost.)

The average, the standard deviation and the maximum value of the response times were increasing with higher load on the 6to4 relay computer, but the average value did not exceed 1.63 milliseconds with ten clients.

The CPU utilization were increasing continuously, but not linearly.

The deviation of the CPU utilization were higher with 4, 5, 6 and 7 clients than with other number of clients, which

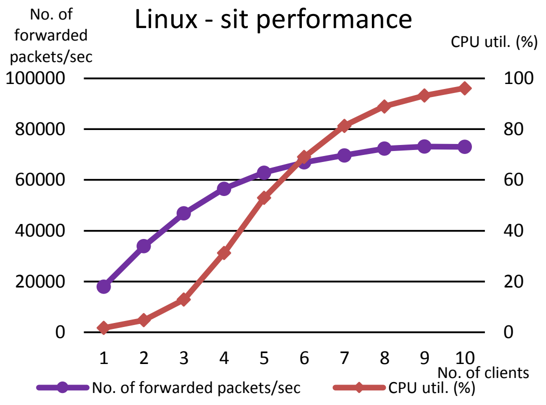

Fig. 2. Linux sit forwarded packets and CPU utilization.

TABLE I
DEBIAN LINUX – SIT 6TO4 RELAY PERFORMANCE RESULTS

Number of clients		1	2	3	4	5	6	7	8	9	10
Packet loss (%)		0.002	0.006	0.008	0.013	0.020	0.035	0.035	0.037	0.048	0.061
Response time (ms)	Average	0.287	0.353	0.445	0.566	0.710	0.868	1.043	1.209	1.411	1.626
	Std. dev.	0.174	0.248	0.353	0.423	0.509	0.588	0.685	0.722	0.832	0.864
	Maximum	27.900	28.400	28.500	28.900	29.400	30.700	31.100	34.100	32.800	39.600
CPU Utilization (%)	Average	1.756	4.821	12.933	31.243	52.964	69.049	81.319	88.941	93.206	96.132
	Std. dev.	1.944	2.811	5.619	12.215	16.379	16.493	12.690	9.817	5.289	7.388
Memory consumption (kB)		10.855	10.418	10.363	10.594	10.824	10.996	10.855	10.994	10.828	11.137
Traffic volume (packets/sec)		18051	33953	46856	56534	62853	66947	69663	72304	73129	73050

TABLE II
DEBIAN LINUX – V4TUNNEL 6TO4 RELAY PERFORMANCE RESULTS

Number of clients		1	2	3	4	5	6	7	8	9	10
Packet loss (%)		0.003	0.006	0.008	0.011	0.018	0.033	0.036	0.039	0.047	0.060
Response time (ms)	Average	0.287	0.351	0.444	0.579	0.709	0.865	1.007	1.198	1.389	1.632
	Std. dev.	0.174	0.251	0.334	0.428	0.508	0.588	0.690	0.776	0.842	0.887
	Maximum	27.800	27.700	28.700	29.920	24.000	30.100	31.300	35.100	33.900	32.800
CPU Utilization (%)	Average	1.915	4.886	14.202	30.927	51.121	69.555	80.392	89.042	93.441	96.444
	Std. dev.	1.727	3.037	6.871	12.412	16.664	14.790	13.807	10.084	7.934	5.461
Memory consumption (kB)		10.664	10.559	10.910	10.555	10.855	10.728	10.730	10.602	11.102	11.438
Traffic volume (packets/sec)		18083	34062	47079	55828	62788	67181	71315	72759	74025	72792

indicates some fluctuation in the utilization.

The memory consumption was almost constant and very low, and the maximum value of it was 11.14kB with ten clients.

The traffic volume increased until the system reached its limit with 9 clients. With 10 clients, the number of transferred packets were slightly decreased from 73129 to 73050.

B. Debian 7.1.0_x86 – v4tunnel

The results have been listed in Table II, whereas the graphical representation of the forwarded packets per second and the CPU utilization are shown in Fig. 3.

Evaluation of the results:

The packet loss ratio was always very low and it strictly increased with the number of clients.

The average and the standard deviation value of the response times were increasing with higher load on the 6to4 relay computer, and the average value reached its maximum value with ten clients (1.632 ms).

The CPU utilization were increasing continuously, but not linearly.

The standard deviation of the CPU utilization were higher with 4, 5, 6 and 7 clients than with other number of clients, which indicates some fluctuation in the utilization.

The memory consumption was almost constant and very low, and the maximum value of it was 11.44kB with ten clients.

The traffic volume increased until the system reached its limit with 9 clients. With 10 clients, the number of transferred packets were decreased from 74025 to 72792.

C. OpenWRT (Attitude Adjustment) 12.09_x86 – sit

The results have been listed in Table III., whereas the graphical representation of the forwarded packets per second and the CPU utilization are shown in Fig. 4.

Evaluation of the results:

The packet loss ratio was always very low and it strictly increased with the number of clients. The maximum value of it was 0.089% with ten clients.

The average and the standard deviation value of the response times were increasing with higher load on the 6to4 relay computer, but the average value did not exceed 2.16 milliseconds with ten clients.

The CPU utilization with two clients was 4.5 times greater

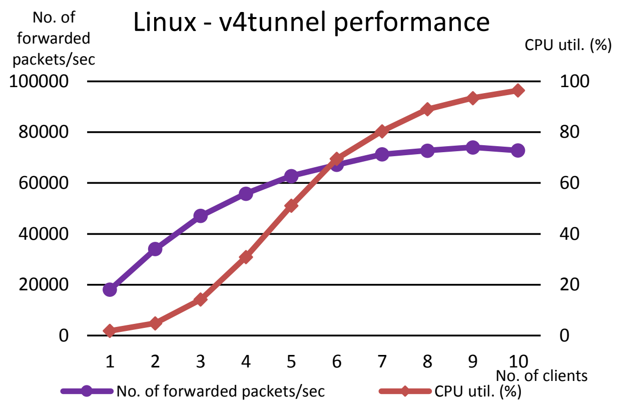

Fig. 3. Linux v4tunnel forwarded packets and CPU utilization.

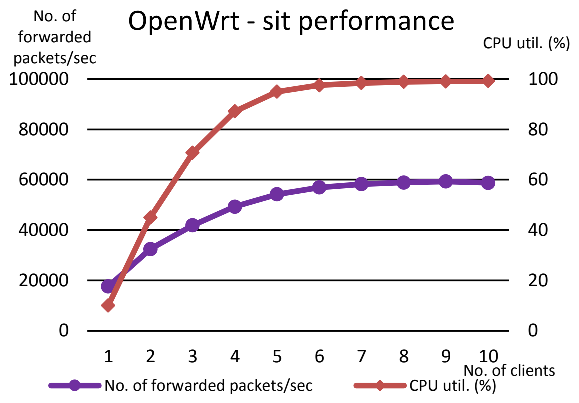

Fig. 4. OpenWrt sit forwarded packets and CPU utilization.

TABLE III
OpenWRT (Attitude Adjustment) 12.09_x86 – sit 6to4 relay performance results

Number of clients		1	2	3	4	5	6	7	8	9	10
Packet loss (%)		0.004	0.006	0.007	0.013	0.018	0.026	0.036	0.064	0.079	0.089
Response time (ms)	Average	0.314	0.402	0.568	0.733	0.909	1.118	1.358	1.616	1.873	2.160
	Std. dev.	0.161	0.239	0.330	0.420	0.508	0.583	0.652	0.705	0.773	0.829
	Maximum	25.000	25.300	25.500	25.500	26.500	27.100	27.000	27.100	27.300	28.100
CPU Utilization (%)	Average	10.067	45.015	70.713	87.188	94.979	97.540	98.467	98.916	99.066	99.288
	Std. dev.	3.188	5.593	5.828	9.376	7.954	7.462	4.991	4.567	4.824	4.410
Memory consumption (kB)		10.316	10.414	10.359	10.727	10.469	10.324	10.746	10.492	10.066	10.469
Traffic volume (packets/sec)		17595	32488	41906	49270	54196	56920	58272	58928	59332	58763

TABLE IV
FreeBSD 9.1_x86 – stf 6to4 relay performance results

Number of clients		1	2	3	4	5	6	7	8	9	10
Packet loss (%)		0.013	0.008	0.010	0.012	0.013	0.015	0.017	0.018	0.019	0.019
Response time (ms)	Average	0.315	0.456	0.681	0.941	1.268	1.637	2.011	2.385	2.740	3.126
	Std. dev.	0.111	0.171	0.314	0.404	0.450	0.457	0.463	0.466	0.480	0.490
	Maximum	22.200	9.220	12.800	15.400	17.600	18.100	18.800	18.500	19.600	19.400
CPU Utilization (%)	Average	51.525	77.110	88.994	96.380	98.482	99.435	99.395	99.371	99.462	99.859
	Std. dev.	6.899	5.140	6.465	7.398	7.593	3.447	5.336	6.445	5.971	0.475
Memory consumption (kB)		0.008	0.012	0.012	0.273	0.395	0.398	0.445	0.406	0.500	0.492
Traffic volume (packets/sec)		17594	30656	37613	41982	43681	43892	43875	43819	43970	43737

than the value with one client. Then the slope was reduced, until the CPU approached its maximum capacity with 6 clients.

The standard deviation of the CPU utilization were under 10% in each case, which indicates consistent utilization of the CPU.

The memory consumption was almost constant and very low.

The traffic volume increased until the system reached its limit with 9 clients. With 10 clients, the number of transferred packets were decreased by 0.97% from 59332 to 58763.

D. *FreeBSD 9.1_x86 – stf*

The results have been listed in Table IV., whereas the graphical representation of the forwarded packets per seconds and the CPU utilization are shown in Fig. 5.

Evaluation of the results:

The packet loss ratio was always very low and starting from two clients it increased with the number of clients, whereas the value of it was the same with one and five clients. The maximum value of it was 0.019% with ten clients.

The average and the standard deviation value of the response times were increasing with higher load on the 6to4 relay computer, but the average value did not exceed 3.13 milliseconds with ten clients. The maximum value of the response times showed some fluctuation

One client could generate 51.53% load on the CPU. The CPU utilization was increasing continuously, but not linearly, until the CPU reached its almost maximum capacity (99.44%) with 6 clients.

The standard deviation of the CPU utilization was under 10% in each case, whereas it was very small (0.46%) with ten clients. This phenomenon indicates consistent utilization of the CPU.

The memory consumption was extremely low and it was growing almost continuously.

The traffic volume increased until the system reached its limit with 6 clients. From this point the throughput of the system started very slightly fluctuating. The maximum value of the number of transferred packets per second was 43970 with 9 clients.

The relay did not show significant decrease in its throughput even in serious overload situations thus it complied with the graceful degradation principles [59].

E. *NetBSD 6.1.2_x86 – stf*

The results have been listed in Table V., whereas the graphical representation of the forwarded packets per seconds and the CPU utilization are shown in Fig. 6.

Evaluation of the results:

The proportion of the packet loss ratio strictly increased until 5 clients, where it started to decrease monotonically. This phenomenon is strange, but the packet loss ratio was always very low.

The average, the standard deviation and the maximum value of the response times were increasing with some fluctuation, but the average value did not exceed 2.52 milliseconds with ten clients.

One client could generate 38.96% load on the CPU. The CPU utilization was increasing continuously, but only by smaller and smaller value.

The standard deviation of the CPU utilization was under 10% in each case, which indicates consistent utilization of the CPU.

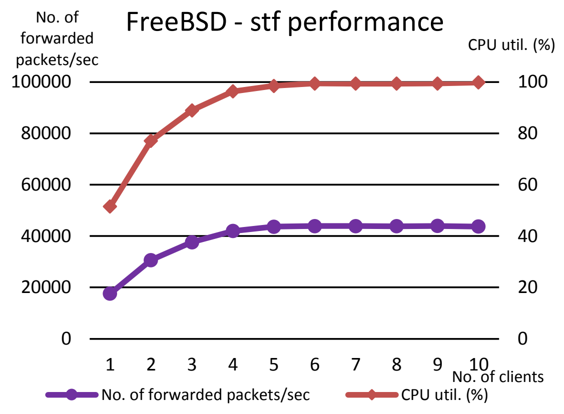

Fig. 5. FreeBSD stf forwarded packets and CPU utilization.

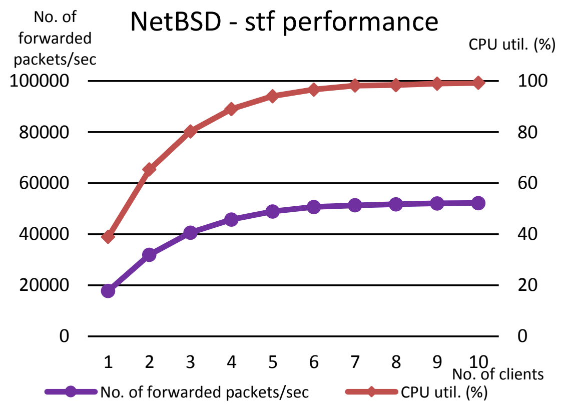

Fig. 6. NetBSD stf forwarded packets and CPU utilization.

TABLE V
NETBSD 6.1.2_X86 – STF 6TO4 RELAY PERFORMANCE RESULTS

Number of clients		1	2	3	4	5	6	7	8	9	10
Packet loss (%)		0.011	0.016	0.028	0.047	0.056	0.051	0.044	0.038	0.031	0.031
Response time (ms)	Average	0.301	0.418	0.603	0.823	1.061	1.326	1.620	1.908	2.210	2.519
	Std. dev.	0.186	0.236	0.319	0.403	0.499	0.571	0.631	0.681	0.707	0.712
	Maximum	5.760	11.500	13.600	16.900	18.900	21.400	21.100	21.700	22.200	24.300
CPU Utilization (%)	Average	38.957	65.382	80.290	89.055	94.130	96.671	98.259	98.435	99.020	99.306
	Std. dev.	4.519	6.229	9.771	3.769	5.878	6.664	3.759	5.751	6.243	4.642
Memory consumption (kB)		0.016	0.027	0.055	0.148	0.191	0.203	0.695	0.336	0.480	0.180
Traffic volume (packets/sec)		17797	31937	40639	45745	48913	50686	51345	51750	52062	52202

The memory consumption was extremely low and it was growing with some fluctuation.
The traffic volume strictly increased.

VIII. COMPARISON OF THE RESULTS

To facilitate the comparison of the properties of the different 6to4 relay implementations, we represented the packet loss ratio, the response time, number of forwarded packets per second and the average value of the CPU utilization in graphical form in Figures 7, 8, 9 and 10, respectively.
It is visible at first sight that the Linux sit and v4tunnel produced almost the same results in all of the four represented areas.

All of the tested implementations proved to be reliable and the packet loss ratios of the different implementations were always low. The packet loss ratio of the Linux and OpenWrt implementations increased with the number of clients, whereas the NetBSD stf produced the highest packet loss with 5 clients. We note that even these low packet loss rates may cause significant loss of TCP performance. For example 0.08% packet loss may result in about 50% decrease of TCP performance at 80ms RTT, see the calculations of [60].

All of the implementations proved their stability under overload situations.

Linux v4 tunnel forwarded the most packets per second, but the performance of it started to visibly decrease in overload situation, whereas the Linux sit system only differs slightly. The OpenWrt sit performance is the next one, and the two BSD systems are the last competitors in the performance comparison. FreeBSD stf produced 43970 maximum throughput, whereas Linux v4tunnel had 74025 maximum packets per second. This means Linux outperformed the FreeBSD system by 1.68 times.

All of the implementations use negligibly small amount of memory, which is usually proportional to the generated load.

With one client, all of the implementations forwarded similar number of packets, but with significantly different CPU utilization, which property can explain the high degree of difference in the performance with more clients. Linux sit 6to4 relay implementation used 1.76% of CPU with one client, whereas FreeBSD stf used 51.53%, which means about 29 times difference.

IX. CONCLUSION

The 6to4 protocol is a useful transition technique in a situation, where two IPv6 enabled hosts have to communicate over an IPv4 only network. All of the tested open source 6to4 relay implementations are reliable solutions in production networks, but the two Linux based ones showed the best

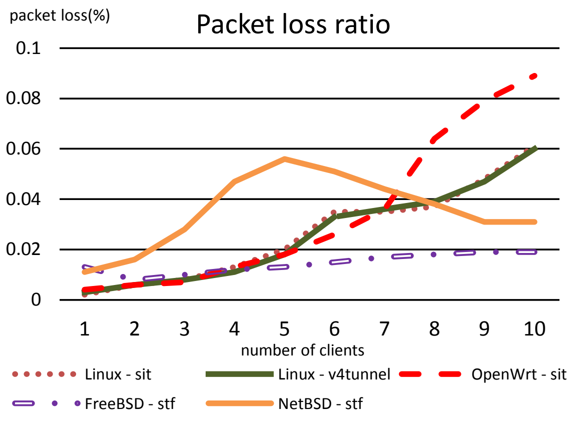

Fig. 7. Packet loss ratio of the different 6to4 implementations.

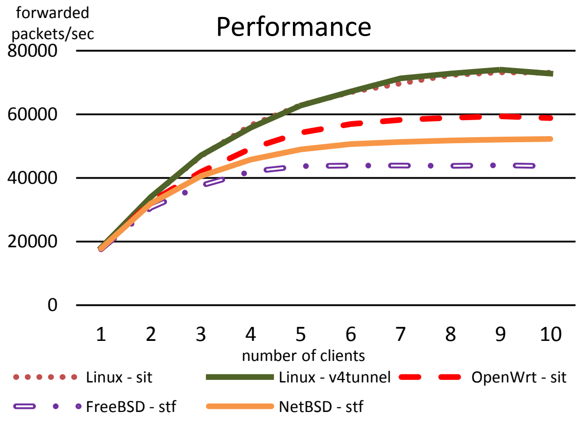

Fig. 9. Performance of the different 6to4 implementations.

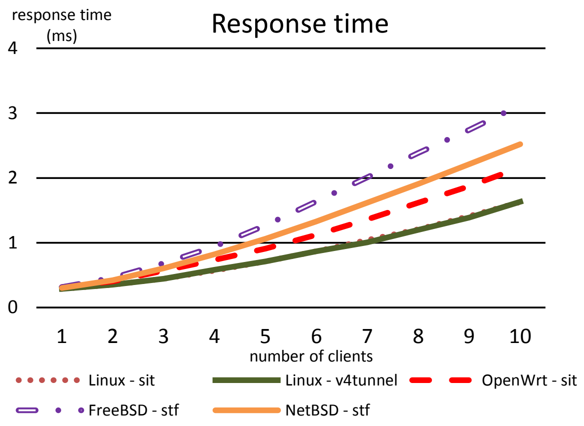

Fig. 8. Response time of the different 6to4 implementations.

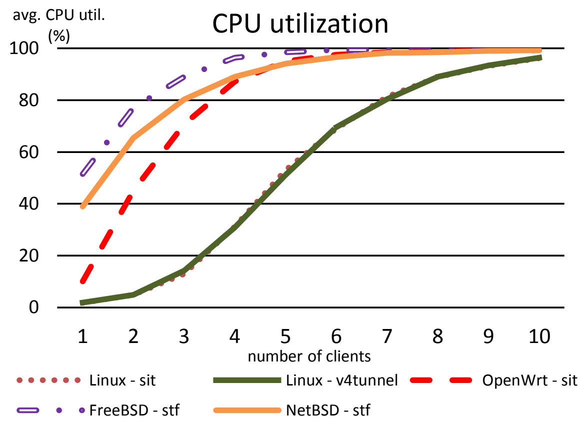

Fig. 10. Average CPU utilization of the different 6to4 implementations.

performance characteristics, whereas the OpenWrt based one was the second to them. In an environment, where BSD systems are preferred, the two BSD based implementations are usable solutions as well.

The authors hope that their work has contributed to the early adoption of the IPv6 protocol and the published results and methodology are valuable for both researchers and network professionals.

REFERENCES

[1] S. Bradner and A. Mankin, "The recommendation for the IP next generation protocol", IETF, January 1995. (RFC 1752) Available: https://tools.ietf.org/html/rfc1752

[2] M. Waiser, "Whatever happened to the Next-Generation Internet?", Communications of the ACM, vol. 44, no. 9. pp. 61-69, 2001.

[3] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification", IETF, December 1998. (RFC 2460) Available: https://tools.ietf.org/html/rfc2460

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

[5] IEEE-USA, "Next generation internet: IPv4 address exhaustion, mitigation strategies and implications for the U.S.", IEEE-USA White Paper, 2009. Available: http://www.ieeeusa.org/policy/whitepapers/IEEEUSAWP-IPv62009.pdf

[6] V. Fuller and T. Li, "Classless Inter-domain Routing (CIDR): The internet address assignment and aggregation plan", IETF, August 2006. (RFC 4632) Available: https://tools.ietf.org/html/rfc4632

[7] P. Srisuresh and K. Egevang, "Traditional IP network address translator (Traditional NAT)", IETF, January 2001. (RFC 3022) Available: https://tools.ietf.org/html/rfc3022

[8] S. Jiang, D. Guo, and B. Carpenter, "An incremental carrier-grade NAT (CGN) for IPv6 transition", IETF, June 2011. (RFC 6264) Available: http://tools.ietf.org/html/rfc6264

[9] M. Mueller, "Scarcity in IP addresses: IPv4 address transfer markets and the regional internet address registries", Internet Governance Project, July 2008. Available: http://www.internetgovernance.org/wordpress/wp-content/uploads/IPAddress_TransferMarkets.pdf

[10] G. Huston, "IPv4 address report", Available: http://www.potaroo.net/tools/ipv4/index.html

[11] L. Smith and I. Lipner, "Free pool of IPv4 address space depleted", Number Resource Organization, February 2011. Available: https://www.nro.net/news/ipv4-free-pool-depleted

[12] E. Nordmark and R. Gilligan, "Basic transition mechanisms for IPv6 hosts and routers", IETF, October 2005. (RFC 4213) Available: https://tools.ietf.org/html/rfc4213

[13] 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) Available: https://tools.ietf.org/html/rfc6147

[14] 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) Available: https://tools.ietf.org/html/rfc6146

[15] G. Lencse and S. Répás, "Performance analysis and comparison of different DNS64 implementations for Linux, OpenBSD and FreeBSD" in *Proc. 27th IEEE International Conference on Advanced Information Networking and Applications (AINA-2013),* Barcelona, 2013, pp. 877-884, doi: 10.1109/AINA.2013.80

[16] G. Lencse and S. Répás, "Performance analysis and comparison of the TAYGA and of the PF NAT64 implementations" in *Proc. 36th International Conference on Telecommunications and Signal Processing (TSP-2013)*, Rome, 2013, pp. 71-76, doi: 10.1109/TSP.2013.6613894

[17] S. Répás, T. Hajas and G. Lencse, "Application compatibility of the NAT64 IPv6 transition technology" in *Proc. 37th International Conference on Telecommunications and Signal Processing (TSP-2014)*, Berlin, 2014, pp. 49-55, DOI: 10.1109/TSP.2015.7296383

[18] A. Conta and S. Deering, "Generic packet tunneling in IPv6 specification", IETF, December 1998. (RFC 2473) Available: http://tools.ietf.org/html/rfc2473

[19] SixXS - IPv6 Deployment & Tunnel Broker, https://www.sixxs.net/main/

[20] Hurricane Electric Free IPv6 Tunnel Broker, https://tunnelbroker.net/

[21] R. Despres, "IPv6 rapid deployment on IPv4 infrastructures (6rd)", IETF, January 2010. ISSN: 2070-1721 (RFC 5569) Available: https://tools.ietf.org/html/rfc5569

[22] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)", IETF, February 2006. (RFC 4380) Available: https://tools.ietf.org/html/rfc4380

[23] F. Templin, T. Gleeson and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", IETF, March 2008. (RFC 5214) Available: https://tools.ietf.org/html/rfc5214

[24] B. Carpenter and K. Moore, "Connection of IPv6 domains via IPv4 clouds", IETF, February 2001. (RFC 3056) Available: https://tools.ietf.org/html/rfc3056

[25] P. Wu, Y. Cui, J. Wu, J. Liu, and C. Metz, "Transition from IPv4 to IPv6: A state-of-the-art survey", IEEE Communications Surveys & Tutorials, vol. 15, no. 3. pp. 1407-1424, 2013, doi: 10.1109/SURV.2012.110112.00200

[26] R. Gilligan and E. Nordmark, "Transition mechanisms for IPv6 hosts and routers", IETF, August 2000. (RFC 2893) Available: https://tools.ietf.org/html/rfc2893

[27] M. Cotton, L. Vegoda, R. Bonica and B. Haberman, "Special-purpose IP address registries", IETF, April 2013. ISSN: 2070-1721 (RFC 6890) Available: https://tools.ietf.org/html/rfc6890

[28] C. Partridge, T. Mendez and W. Milliken, "Host anycasting service", IETF, November 1993. (RFC 1546), Available: https://tools.ietf.org/html/rfc1546

[29] C. Huitema, "An anycast prefix for 6to4 relay routers", IETF, June 2001. (RFC 3068) Available: https://tools.ietf.org/html/rfc3068

[30] D. Malone, "Counting 6to4 relay routers", SIGCOMM Computer Communication Review, vol. 36, no. 1. pp. 79-82, 2006, doi: 10.1145/1111322.1111340

[31] RIPEstat, https://stat.ripe.net

[32] P. Savola and C. Patel, "Security considerations for 6to4", IETF, December 2004. (RFC 3964), Available: https://tools.ietf.org/html/rfc3964

[33] W. Townsley and O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)", IETF, August 2010, (RFC 5969), Available: https://tools.ietf.org/html/rfc5969

[34] O. Troan and G. Van de Velde, "Request to move connection of IPv6 domains via IPv4 clouds (6to4) to historic status", February, 2011, (expired internet draft), Available: https://tools.ietf.org/html/draft-troan-v6ops-6to4-to-historic-00

[35] O. Troan and B. Carpenter, ed, "Deprecating the anycast prefix for 6to4 relay routers", May 2015, (RFC 7526), Available: https://tools.ietf.org/html/rfc7526

[36] B. Carpenter, "Advisory Guidelines for 6to4 Deployment", August 2011, (RFC 6343), Available: https://tools.ietf.org/html/rfc6343

[37] G. Lencse and S. Répás, "Performance analysis and comparison of 6to4 relay implementations", International Journal of Advanced Computer Science and Applications, vol. 4, no. 9. pp. 13-21, 2013, doi: 10.14569/IJACSA.2013.040903

[38] M. Nikkhah, R. Guérin, Y. Lee and R. Woundy, "Assessing IPv6 through web access a measurement study and its findings" in *Proc. Seventh Conference on emerging Networking EXperiments and Technologies (CoNEXT '11)*, Tokyo, 2011, doi: 10.1145/2079296.2079322

[39] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil and M. Bailey, "Measuring IPv6 adoption" in *Proc. ACM conference on SIGCOMM (SIGCOMM '14)*, Chicago, 2014, pp. 87-98. doi: 10.1145/2619239.2626295

[40] M. Aazam, A.M. Syed, S.A.H. Shah, I. Khan and M. Alam, "Evaluation of 6to4 and ISATAP on a test LAN" in *Proc. IEEE Symposium on Computers & Informatics (ISCSI 2011)*, Kuala Lumpur, 2011, pp. 46-50. doi: 10.1109/ISCI.2011.5958881

[41] F. Sans and E. Gamess, "Analytical performance evaluation of native IPv6 and several tunneling technics using benchmarking tools" in *Proc. XXXIX Latin American Computing Conference (CLEI 2013)*, Naiguata, 2013, pp. 1-9. doi: 10.1109/CLEI.2013.6670610

[42] J. L. Shah and J. Parvez, "An examination of next generation IP migration techniques: Constraints and evaluation" in *Proc. International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT-2014)*, Kanyakumari District, 2014, pp. 776-781. doi: 10.1109/ICCICCT.2014.6993064

[43] Riverbed Modeler, http://www.riverbed.com/products/performance-management-control/network-performance-management/network-simulation.html
[44] D. Hadiya, R. Save and G. Geetu, "Network performance evaluation of 6to4 and configured tunnel transition mechanisms: An empirical test-bed analysis" in *Proc. 6th International Conference on Emerging Trends in Engineering and Technology (ICETET-13)*, Nagpur, 2013, pp. 56-60. doi: 10.1109/ICETET.2013.14
[45] N. Bahaman, E. Hamid and A.S. Prabuwono, "Network performance evaluation of 6to4 tunneling" in *Proc. 2012 International Conference on Innovation Management and Technology Research ((ICIMTR)*, Malacca, 2012, pp. 263-268. doi: 10.1109/ICIMTR.2012.6236400
[46] M. Elich, P. Velan, T. Jirsik and P. Celeda, "An investigation into teredo and 6to4 transition mechanisms: Traffic analysis" in *Proc. IEEE 38th Conference on Local Computer Networks Workshops (LCN 2013 Workshops)*, Sydney, 2013, pp. 1018-1024. doi: 10.1109/LCNW.2013.6758546
[47] S. Pekka," Observations of IPv6 traffic on a 6to4 relay", SIGCOMM Computer Communication Review, vol. 35, no. 1. pp. 23-28, 2005. doi: 10.1145/1052812.1052821
[48] S. Narayan and S. Tauch, "Network performance evaluation of IPv4-v6 configured tunnel and 6to4 transition mechanisms on windows server operating systems" in *Proc. 2010 International Conference on Computer Design and Applications (ICCDA 2010)*, Qinhuangdao, 2010, pp. V5-435-V5-440. doi: 10.1109/ICCDA.2010.5540939
[49] S. Narayan and S. Tauch, "IPv4-v6 configured tunnel and 6to4 transition mechanisms network performance evaluation on Linux operating systems" in *Proc. 2nd International Conference on Signal Processing Systems (ICSPS 2010)*, Dalian, 2010, pp. V2-113-V2-117. doi: 10.1109/ICSPS.2010.5555209
[50] S. Narayan and S. Tauch, "IPv4-v6 transition mechanisms network performance evaluation on operating systems" in *Proc. 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2010)*, Chengdu, 2010, pp. 664-668. doi: 10.1109/ICCSIT.2010.5564141
[51] S. Répás, V. Horváth and G. Lencse, "Stability Analysis and Performance Comparison of Three 6to4 Relay Implementations" in *Proc. 38th International Conference on Telecommunications and Signal Processing (TSP 2015)*, Prague, July 9-11, 2015, pp. 82-87. DOI: 10.1109/TSP.2015.7296228
[52] Open Source Initiative, "The open source definition", http://opensource.org/docs/osd
[53] Free Software Fundation, "The free software definition", http://www.gnu.org/philosophy/free-sw.en.html
[54] Debian, http://www.debian.org/
[55] OpenBSD, http://www.openbsd.org/
[56] FreeBSD, http://www.freebsd.org/
[57] NetBSD, http://www.netbsd.org/
[58] OpenWrt, https://openwrt.org/
[59] NTIA ITS, "Definition of 'graceful degradation' ", Available: http://www.its.bldrdoc.gov/fs-1037/dir-017/_2479.htm
[60] Network Throughput Calculator, http://wintelguy.com/wanperf.pl

Sándor Répás received his BA in business administration and management from the Corvinus University of Budapest in 2009 and MSc in electrical engineering from the Széchenyi István University, Győr in 2013.
He is a full time PhD student in information technology at the Széchenyi István University. The main field of his research is the IPv6 implementation technologies. His other favorite topics are computer networking, information security, and critical information infrastructure protection. He has several certificates from Cisco, ISACA, Microsoft, MikroTik, Novell, and other vendors.
Mr. Répás is a student member of the Association for Computer Machinery (ACM), and member of the Information Scientific Association for Infocommunications Hungary (HTE), and the John von Neumann Computer Society.

Viktor Horváth received his BSc in electrical engineering at Széchenyi István University in Győr in 2014. He had been working at the Department of Telecommunications as a graduate student during his thesis research. The area of his research included performance analysis of IPv6 transition technologies, router boards and several Linux and BSD operating system. Nowadays he is mostly interested in computer security field. He is an IT security engineer at one of the most professional value added security distributor company in Hungary. Horváth's other favorite topics are computer networking, wireless networking and secure mobile device management. He has several vendor specific certificate from MobileIron, SafeNet, Unitrends, Opswat and others. During his work he got familiar with several IT security vendor and solution. His main responsibilities include professional enterprise level IT support, IT Infrastructure Management and administration, trainings, technical presentations, site surveys and security infrastructure integration. He took part in most of the reasonable IT security focused events in Hungary where he was responsible for the IT infrastructure behind the "scene". These days he is involved in several project at multinational companies.

Gábor Lencse received his MSc in electrical engineering and computer systems from the Technical University of Budapest in 1994, and his PhD in 2001.
He has been working for the Department of Telecommunications, Széchenyi István University in Győr since 1997. Now, he is an Associate Professor. He teaches Computer networks and the Linux operating system. He is responsible for the specialization of the information and communication technology of the BSc level electrical engineering education. He is a founding member and also a core member of the Multidisciplinary Doctoral School of Engineering Sciences, Széchenyi István University. The area of his research includes discrete-event simulation methodology, performance analysis of computer networks and IPv6 transition technologies. He has been working part time for the Department of Networked Systems and Services, Budapest University of Technology and Economics (the former Technical University of Budapest) since 2005. There he teaches Computer architectures and Computer networks.
Dr. Lencse is a member of the Institute of Electronics, Information and Communication Engineers (IEICE).