# Practical Evaluation of Stateful NAT64/DNS64 Translation

Nejc ŠKOBERNE[1], Mojca CIGLARIČ[2]
[1]*Viris, Šmartinska 130, SI-1000 Ljubljana*
[2]*Faculty of Computer and Information Science, University of Ljubljana, Tržaška 25, SI-1000 Ljubljana*
*nejc@viris.si, mojca.ciglaric@fri.uni-lj.si*

***Abstract*—It is often suggested that the approach to IPv6 transition is dual-stack deployment; however, it is not feasible in certain environments. As Network Address Translation -- Protocol Translation (NAT-PT) has been deprecated, stateful NAT64 and DNS64 RFCs have been published, supporting only IPv6-to-IPv4 translation scenario. Now the question of usability in the real world arises. In this paper, we systematically test a number of widely used application-layer network protocols to find out how well they traverse Ecdysis, the first open source stateful NAT64 and DNS64 implementation. We practically evaluated 18 popular protocols, among them HTTP, RDP, MSNP, and IMAP, and discuss the shortcomings of such translations that might not be apparent at first sight.**

***Index Terms*—Network address translation, IP networks, Next-generation networking, Domain Name System, Protocols**

## I. Introduction and Related Work

After the IPv6 protocol [1] was adopted as the new-generation Internet protocol, researchers started to investigate different approaches to transition from IPv4 to IPv6 protocol [2]-[4]. These approaches are called *transition mechanisms,* and they can be roughly categorized into three groups: *tunneling*, *translation,* and *dual-stack*. The three are reviewed in [5].

Tunneling is most often used when two isolated IPv6-enabled networks or hosts communicate through an IPv4-only network. The tunnel endpoints perform encapsulation and decapsulation of IPv6 datagrams into IPv4 datagrams as their payload. The IPv6 header remains intact, but a new IPv4 header is formed and used on the way through the IPv4 network. On exiting the tunnel, IPv6 datagrams are decapsulated again. Of course, IPv4 communication is also possible through an IPv6 tunnel.

In the dual stack scenario, the network nodes implement both protocol stacks, IPv4 and IPv6. Dual stack results in an unnecessary highly complex network backbone and end systems. In this scenario, end systems may run IPv6 and IPv4 applications, and servers may receive connections from IPv4-only as well as IPv6-only hosts. The situation becomes complicated when communication is requested between IPv4-only and IPv6-only entities. Translation of network protocols and IP addresses between IPv4 and IPv6 realms is necessary to enable cross-family communication. For example, if a host from an IPv6-only network wants to access a web server operating in an IPV4-only environment, the original IPv6 packets need to be converted into IPv4 packets. This means that the IPv6 header is discarded and an IPv4 header is substituted, whereas the contents of the fields with no clear mapping in the target protocol header are lost [6].

In 2010, the first open source implementations of the stateful NAT64 gateway and DNS64 server called Ecdysis [7] were made publicly available. Until then, we could only theoretically discuss stateful NAT64/DNS64 feasibility. Ecdysis is available for Linux and OpenBSD operating systems. It includes a stateful IP translator and DNS application layer gateway, implemented within Unbound and Bind open source DNS servers. In Linux, the stateful IP translator is implemented as a kernel module using *netfilter* facilities. In the OpenBSD operating system, it is available as a modification of the PF firewall. Ecdysis appears to be modestly documented and not yet extensively tested in the real-world environments. At this time, we are not aware of other open source stateful NAT64/DNS64 implementations. However, there are some implementations of the obsolete NAT-PT (RFC 2766), e.g., [8] and a stateless NAT64 gateway implementation TAYGA [9], which performs 1-to-1 stateless IP/ICMP translation (SIIT) of IPv6 addresses into IPv4 addresses. The major drawback of stateless NAT64 is that each IPv6-only host requires its own (possibly temporary) IPv6-to-IPv4 address mapping. Since this is a substantial requirement, which is not present when performing stateful translation, we did not consider stateless translators in our research, and we will use the term "NAT64 translation" to refer to stateful NAT64 translation only.

The stateful NAT64 and DNS64 mechanism specifications, as described in the latest RFCs [22], [25] seem very consistent and ready for implementation. It is to be expected that with the anticipated depletion of IPv4 address space at the regional registry level there will be more and more IPv6-only networks, which will need access to IPv4 services in the rest of the Internet. Since many applications rely on the underlying application layer protocols, it is important for administrators to know what they can expect from translation mechanisms: is the behavior of the applications going to change during translation? Will this perhaps occur only if they are using specific application layer protocols? How well are these protocols going to be translated? Is translation feasible for every application-layer protocol?

In an attempt to answer these questions, the purpose of this paper is twofold: first, we evaluate translation of different application-layer network protocols theoretically,

Operation part-financed by the European Union, European Social Fund.
Digital Object Identifier 10.4316/AECE.2011.03008

and then we empirically evaluate traversal of these application-layer network protocols over Ecdysis in a controlled environment. We did not, however, test other aspects of translation, such as scalability and performance. The paper will be of interest to application administrators, system administrators, to network infrastructure architects and to the designers and constructors of other NAT64/DNS64 translators and ALGs. Moreover, it might be of interest to anybody interested in general IPv6 transition problems and technologies.

The paper is organized as follows. In the rest of this section we review IPv6 transition mechanisms and comment on related literature. In Section 2, we propose an objective measure for evaluating the quality of NAT64/DNS64 traversal of application-layer network protocols and theoretically assess translation quality of the chosen protocols. In the next section, we describe the test beds used for assessing the traversal of protocols over the Ecdysis translator. Then, we present the test results and point out the most interesting observations. Finally, we critically evaluate, in accordance with the proposed objective measure, the quality of traversal of a selected set of application-layer protocols over the Ecdysis NAT64/DNS64 translator. We conclude with practical implications and a discussion of future research possibilities.

The research on IPv4 and IPv6 coexistence is scarce. Che and Lewis [10] compare transition mechanisms and evaluate their effectiveness regarding packet delay and packet loss in a simulated environment. The authors expose some migration challenges, but they do not address NAT64/DNS64. Martin [11] points out that the IPv4-to-IPv6 transition strategy is incomplete and the IPv4-to-IPv6 migration process will be more difficult than originally thought, so there is a need for additional translation mechanisms. He offers no further discussion on translation. AlJa'afreh et al. [12]-[14] discuss bidirectional mapping among native IPv4 and IPv6 networks, examine its performance by means of a network simulator, and compare performance with tunneling and a dual stack approach. Chen [15] proposes an ALG for SIP protocol; however, it is not implemented, and the authors do not properly address its performance. One of the most recognized names in the area of transitional mechanisms specifications is Wing. His paper [6] reviews and compares the different known approaches to NAT in IPv4 networks and in IPv6 and mixed networks. It appears that no research has been published that is directly related to our work, probably due to the fact that NAT64 and DNS64 translation methods are yet to be massively implemented.

## II. Translation Mechanisms

In this paper we focus on stateful IPv6-to-IPv4 translation mechanisms. Stateful translation is a conceptually challenging and highly complex mechanism, and IETF issued several RFCs and Internet drafts on this problem. Although Network Address Translation (NAT) is a well-established concept in the IPv4 world [16], the translation between IPv4 and IPv6 addresses brings a range of new problems, which different translation mechanisms – NAT64/DNS64, NAT-PT and NAPT-PT [17], Transport Relay Translation (TRT)) [18], etc. – try to address each in its own way. We can split the translation problem into two subproblems with regard to the direction of translation: from larger IPv6 to smaller IPv4 address space (NAT64) and translation of network addresses from smaller IPv4 to larger IPv6 address space (NAT46). The latter is much harder to achieve since IPv4 address space is smaller (IPv4 uses 32-bit addresses, while IPv6 uses 128-bit addresses). A device using IPv4 address space can therefore address (using one of the translation mechanisms) only a small subset of the IPv6 address space. Furthermore, some of the application-layer protocols (called *control/data* protocols) make direct use of the IP addresses in their payload, so not only the message header needs to be translated. The payload is usually not inspected by the translator itself. Consequently, another translation-related entity needs to be implemented: application layer gateway (ALG). The key functionality of the ALG is the conversion of the network layer address information found inside the application payload into the address so it is acceptable to the hosts on the other side of the NAT device. In other words, ALGs are application-specific translation agents that allow an application on a host in one address realm to transparently connect to its counterpart running on a host in a different realm [19].

The main advantage of translation is that the communicating devices need not be changed in any way, so translation may be applicable with all kinds of legacy devices, where either hardware, operating system or applications do not permit pure IPv6 deployment or more advanced transition mechanisms. However, translation also exhibits a conceptual disadvantage: it could break the end-to-end connectivity, which is considered a core concept of the Internet. Therefore, the role of Internet users behind translation devices is reduced to Internet "consumers" only, and, consequently, they cannot make their own content and services available to other Internet users.

The first standardized IPv6-to-IPv4 translation mechanism was NAT-PT, and although it was adopted by some major vendors, it was too complex and has been deprecated by RFC 4966 [20]. Its intention was to provide NAT64 together with NAT46 translation and DNS ALG in one complex device. Moreover, DNS ALG was tightly coupled with the translator itself (with a direct interface to the translator). According to Wing [6], the main reason for its deprecation was operational complexities, and deeper discussion of its issues is outside the scope of this paper and can be found in the aforementioned RFC. Stateful NAT64 translation, however, is still useful in IPv6-only environments in order to access the IPv4 Internet, and the underlying translation method is called stateful NAT64/DNS64. (From now on, these terms will refer to translation methods, not translation subproblems.) In 2008 the first stateful NAT64, DNS64, and other IPv6-to-IPv4 translation-related Internet drafts were proposed, some of which are now RFCs [21]-[25], supporting only some specific cases of IPv6-to-IPv4 translation. In addition, Stateful NAT64 RFC only allows for translation of TCP and UDP transport-layer protocols and network-layer protocol ICMP. In other words, according to specifications the application-layer protocols relying on transport-layer protocols other than TCP or UDP are not supposed to traverse a stateful NAT64 translator.

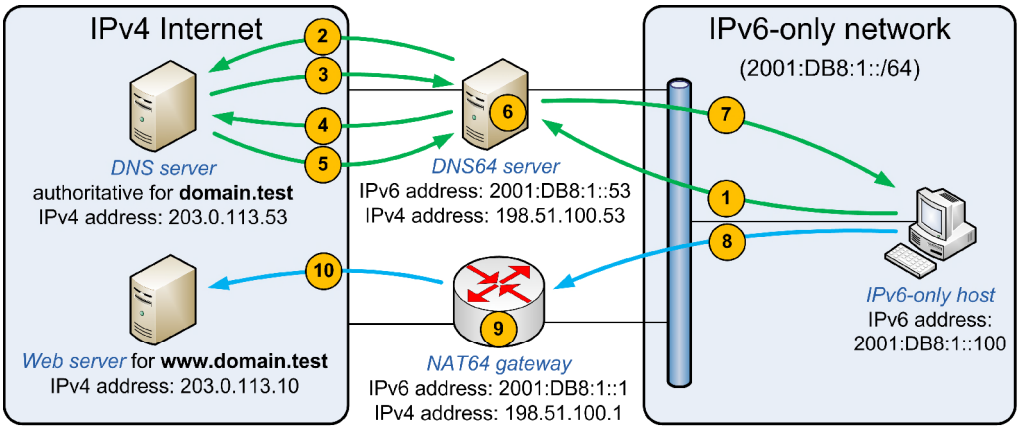


Figure 1. IPv6-only client accessing IPv4 Internet through stateful NAT64

Figure 1 shows a typical scenario of an IPv6-only client communicating to the IPv4-only Internet over a NAT64 gateway, using a DNS64 server:

1. DNS query: AAAA record for **www.domain.test**?
2. DNS query: AAAA record for **www.domain.test**?
3. DNS answer: no AAAA record for **www.domain.test**.
4. DNS query: A record for **www.domain.test**?
5. DNS answer: A record for **www.domain.test** is **203.0.113.10**.
6. DNS64 server synthesizes AAAA record, i.e. **64:FF9B::203.0.113.10**.
7. DNS64 answer: the IPv6 address of **www.domain.test** is **64:FF9B::203.0.113.10**.
8. TCP/UDP packet: SRC address: **2001:DB8:1::100**, DST address: **64:FF9B:203.0.113.10**.
9. NAT64 translation.
10. TCP/UDP packet: SRC address: **198.51.100.1**, DST address: **203.0.113.10**.

## III. Experimental Methods

First, we selected a representative set of the most commonly used application protocols. In order to consistently evaluate traversal of application-layer protocols over a NAT64/DNS64 translator, we established a test environment and defined an ordinal scale as a metric for protocol translation quality. We also theoretically evaluated how well different types of protocols are expected to translate. Afterwards, the protocols were tested one by one. The IPv6-only clients would connect to the IPv4 Internet using the tested application-layer protocol. When the protocol did not traverse NAT64 seamlessly, we also analyzed packet trace files in order to find an explanation for the incompatibility.

### A. Selection of Protocols to be Tested

The selection of protocols (see Table 1) is based on the authors' subjective knowledge of the area. We considered three groups of users – corporate users, mobile users, and home users (there is some overlap in these groups, but we assume that their union represents the great majority of Internet or computer network users) – and collected a set of applications these users use most frequently (according to our personal experience). These applications are: the operating system itself, e-mail client, Internet browser, terminal services client, peer-to-peer (P2P) client, instant messaging (IM) client, soft phone (VoIP) client, terminal client, and VPN client. We considered each application and figured out which application-layer protocols it uses. DNS protocol is not included since IPv6-only machines behind the NAT64 translator **have to** use the DNS64 server in order to be able to connect to the IPv4 world. We considered use of DNS servers other than DNS64 to be irrelevant.

For each of the selected protocols, we evaluated the quality of NAT64 traversal, first theoretically and then empirically, using the protocol in our test bed and inspecting what happens with protocol messages after Ecdysis traversal.

TABLE 1: LIST OF SELECTED APPLICATION-LAYER PROTOCOLS

| Acronym | Protocol Name | Application |
|---|---|---|
| BitTorrent | BitTorrent (P2P file sharing protocol) | P2P client |
| FTP | File Transfer Protocol | Internet browser |
| HTTP | Hypertext Transfer Protocol | Internet browser |
| HTTPS | Hypertext Transfer Protocol Secure | Internet browser |
| IMAP | Internet Message Access Protocol | e-mail client |
| NTP | Network Time Protocol | operating system |
| POP3 | Post Office Protocol (version 3) | e-mail client |
| RDP | Remote Desktop Protocol | terminal client |
| Skype | Skype (P2P VoIP protocol) | VoIP, IM client |
| MSNP | Microsoft Notification Protocol | VoIP, IM client |
| SIP | Session Initiation Protocol | VoIP client |
| CIFS | Common Internet File System | operating system |
| SMTP | Simple Mail Transfer Protocol | e-mail client |
| SSH | Secure Shell | terminal client |
| TELNET | Terminal Network | terminal client |
| OpenVPN | OpenVPN (Virtual Private Network) | VPN client |
| IPsec | IPsec (IP security) | VPN client |
| PPTP | Point-to-Point Tunneling Protocol | VPN client |

### B. Translation Quality Metrics

In this section, we suggest the ordinal scale, categorizing the protocols into three translation quality classes. If only the IPv6 header would need to be replaced with an IPv4 protocol header, all the protocols would translate well. However certain protocols include network-related data (i.e., IP addresses) in the application payload, which complicates the translation. Our suggested classification rules are outlined below.

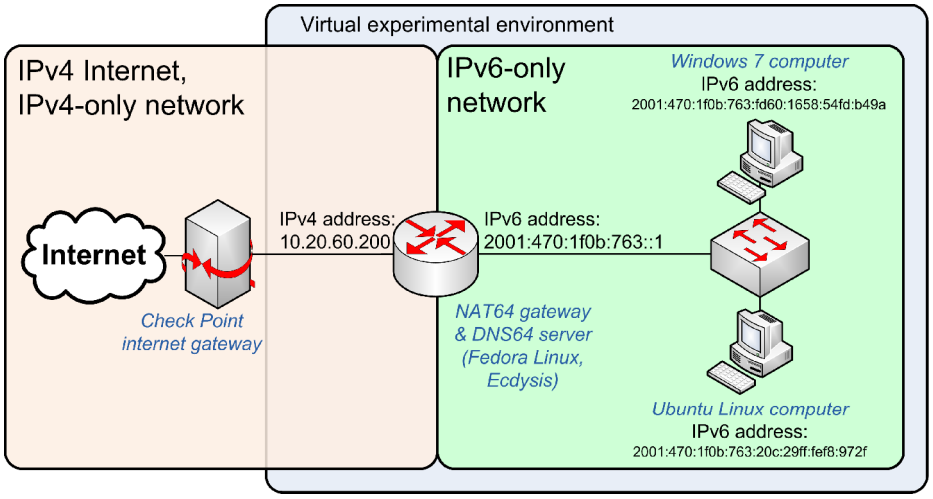


Figure 2. The experimental environment for translation quality evaluation

*1) Well Translated Protocols*

Well translated protocols are the protocols that do not need any special treatment when traversing the NAT64 translator. They do not need the support of a dedicated ALG or any other additional translation logic either at the server or at the client side.

*2) Conditionally Translated Protocols*

Conditionally translated protocols are protocols requiring some kind of special treatment in order to be successfully translated over a NAT64 translator. By special treatment we mean either a dedicated ALG, constructed only for the purpose of NAT64 translation for the specific protocol, or any modifications or limitations at the server or at the client side. After the special treatment is applied, at least core protocol functionality is successfully provided over a NAT64 translator.

*3) Poorly Translated Protocols*

If nothing could be done in order to make a protocol traverse a NAT64 translator and any special treatment is not feasible to implement, the protocol is classified as poorly translated.

When we had the translation quality metrics defined, we were able to formulate some predictions or hypotheses about the chosen protocols' translation quality. We wanted to confirm these predictions empirically in our testbed:

- The protocols seamlessly traversing IPv4 NAT translators will also be well translated over a NAT64 translator.
- The protocols embedding network-layer addresses (IPv4 or IPv6) in their payload will belong to a conditionally translated class. For example, SIP needs an ALG even when performing IPv4 network address translation [16].
- The protocols using transport layer protocols other than TCP and UDP (for example, GRE or ESP) will classify as poorly translated, since the NAT64 translator only implements translation of TCP-, UDP-, and ICMP-based protocols.

*C. Testbed and Experimental Details*

The testbed scheme and topology are shown in Figure 2. The experimental environment was virtualized and consisted of a NAT64 translator and a DNS64 server (Fedora Linux with Ecdysis), a DHCPv6 server, and two IPv6-only clients, one with Ubuntu Linux and the other with a Windows 7 operating system. NAT64 and DNS64 methods only support the IPv6-to-IPv4 scenarios, which means that IPv6-only machines can establish connections with the IPv4-only Internet. Consequently, we tested the translation in the IPv6-to-IPv4 direction over Ecdysis. We decided to use BIND because of its widespread use, although Ecdysis also supports the Unbound DNS server. Router Advertisement daemon *radvd* was announcing the prefix 2001:470:1f0b:763::/64 on the internal IPv6-only network. A DHCPv6 server was used to configure the DNS server on DHCPv6 clients.

For each of the selected application layer protocols, the IPv6-only clients would connect to the IPv4 Internet using a tested protocol from within a client application. If the protocol translated seamlessly, which means that from the user's perspective the application performed as if there were no translation, the test was finished at that point. Otherwise, if the protocol did not translate well, we needed to analyze packet trace files and identify the reasons for the incompatibility.

## IV. Results

The test results have mostly confirmed our hypotheses. The control/data protocols, which use network-layer addresses in the payload, belong to Conditionally Translated or Poorly Translated classes. Results are shown in Table 2. However, some of the results require additional explanation, which is provided below.

*A. Well Translated Protocols*

For well translated protocols, it is sufficient that only the IP header is replaced by the translator (IPv4 header is removed and IPv6 header is inserted). As shown in Table 2, we empirically proved the translation quality for all well-translated protocols; however, there were two exceptions:

TABLE 2: TRANSLATION QUALITY OF THE SELECTED PROTOCOLS

| Acronym | Theoretical Translation Quality | Empirical Translation Quality |
|---|---|---|
| BitTorrent | Conditionally Translated | Conditionally Translated |
| FTP | Conditionally Translated | Conditionally Translated |
| HTTP | Well Translated | Well Translated |
| HTTPS | Well Translated | Well Translated |
| IMAP | Well Translated | Well Translated |
| NTP | Well Translated | Well Translated |
| POP3 | Well Translated | Well Translated |
| RDP | Well Translated | Well Translated |
| Skype | Poorly Translated | Poorly Translated |
| MSNP | Poorly Translated | Poorly Translated |
| SIP | Poorly Translated | Poorly Translated |
| CIFS | Well Translated | Well Translated |
| SMTP | Well Translated | Well Translated |
| SSH | Well Translated | Well Translated |
| TELNET | Well Translated | Well Translated |
| OpenVPN | Conditionally Translated | Poorly Translated |
| IPsec | Poorly Translated | Poorly Translated |
| PPTP | Poorly Translated | Poorly Translated |

*1) HTTP*

Although we classified HTTP as a Well Translated protocol, it should be pointed out that a small percentage of HTTP URIs contain an IPv4 address literal as the hostname (e.g., http://203.0.113.1), which is not accessible to IPv6-only HTTP clients using a NAT64 translator since the address is not known to the DNS64 server (see the typical scenario depicted in Figure 1).

An examination of Alexa's top 1 million domains at the end of August 2009 showed that 2.38% of the HTML in their home pages contained IPv4 address literals. Also, of the top 1 million websites at the end of August 2009, 0.35% were IPv4 address literals.

In [26] Wing proposes using a HTTP proxy to handle such traffic as a workaround. Although it overcomes the described problem, operating an HTTP proxy interfaced to IPv4 Internet is not a trivial task, is more resource-intensive, complicates the network topology, and increases the attack surface. Moreover, proxy still cannot handle IPv4 address literals located in the URL path or query string (for example, http://www.example.com/?host=203.0.113.1).

*2) CIFS*

In our experiment, we were able to empirically evaluate file transfer only. The Microsoft NetBIOS-over-TCP/IP (NBT) name resolution and service location protocol could not be tested, since Microsoft's implementation does not support IPv6 [27]. Link-local Multicast Name Resolution protocol (LLMNR) also could not be tested, since it uses link-local multicast addresses, which are only valid within the same network segment. Since the NAT64 translator terminates the network segment, LLMNR traffic cannot traverse the translator. Therefore, we suggest using DNS for name resolution when using a CIFS protocol with NAT64 translation.

## B. Conditionally Translated Protocols

Conditionally translated application-layer protocols require dedicated application layer gateways for NAT64 traversal. We proved empirically that none of the conditionally translated protocols in Table 2 were able to traverse NAT64 seamlessly.

*1) OpenVPN*

Since OpenVPN does not currently officially support IPv6 endpoints, the connections could not be established during the test. However, since OpenVPN only uses UDP datagrams, we expect the traversal over NAT64 should be seamless when the endpoints are upgraded with IPv6 capability, and OpenVPN will be promoted to the Well Translated class.

*2) BitTorrent*

The problem of BitTorrent traversal over NAT64 was identified by Wing [28]: although BitTorrent packets would traverse NAT64 and reach their destination, an IPv6-only BitTorrent peer cannot use IPv4 addresses obtained from its tracker. To do so, the client software would need to prefix the IPv4 address with the prefix of an IPv6/IPv4 translator that will perform the necessary address family translation on behalf of the IPv6-only client.

As an alternative to Wing's suggestion, introduction of an ALG is possible, performing application-layer deep packet inspection and IPv4-to-IPv6 address translation on the fly. In this way, BitTorrent clients would not need changes. The payload of HTTP sessions between BitTorrent client and tracker could be modified, either transparently or by means of a HTTP proxy. Note that the proposed solution is only feasible when using HTTP protocol, since it is not possible for an ALG or proxy to decrypt HTTPS payload.

*3) FTP*

Some considerations about IPv6-to-IPv4 translation of FTP can be found in [29]. Disparate implementations of the newer FTP commands EPSV and EPRT are largely inconsistent, which causes inconsistent behavior of FTP even when not traversing NAT64 and makes ALG construction significantly harder.

## C. Poorly Translated Protocols

At present, these protocols are not able to traverse NAT64, either due to their closedness and lack of IPv6 support or due to incompatibility with any NAT in general.

*1) Skype*

Skype is a proprietary peer-to-peer protocol for VoIP and Instant Messaging communication. Currently it does not support IPv6. It is impossible to predict whether IPv6-ready Skype will be able to pass NAT64 translators, since we do not know how Skype will implement IPv6.

*2) MSNP*

Microsoft Notification Protocol is a proprietary P2P protocol unable to traverse the NAT64 translator. MSNP only supports IPv6 in P2P communication among users. In communication with server IPv6 it is not yet supported.

*3) SIP*

SIP protocol uses bidirectional UDP communication among peers. Since NAT64 only supports connections from IPv6 towards the IPv4 world, SIP over NAT64 is not viable.

*4) IPsec*

To encapsulate transport-layer protocols, IPsec uses Encapsulating Security Payload (ESP), which is not supported by NAT64 specification (only TCP, UDP and ICMP are supported on lower layers). IPv4 NATs usually implement "IPsec Pass-Through" functionality. However, there exists another method enabling IPsec-protected datagrams to pass through IPv4 NAT translators. It is called NAT Traversal (NAT-T) in Internet Key Exchange (IKE), which encapsulates ESP packets into UDP datagrams that can traverse traditional NATs. However, we did not test NAT-T.

*5) PPTP*

PPTP has a problem similar to that encountered with IPsec. On transport layer, PPTP uses Generic Routing Encapsulation (GRE) protocol. Since GRE is not supported by NAT64, PPTP is unable to traverse a NAT64 translator.

## V. CONCLUSIONS

The experiment mostly confirmed our expectations about NAT64/DNS64 traversal quality of different application-layer protocols. Most of the protocols daily used by average Internet users are categorized as Well Translated. FTP, SIP, PPTP, and Skype are conditionally or poorly translated; however, where the lack of IPv4 dictates the use of IPv6-only networks, we will hopefully be able to cope without them. New Internet drafts [29] indicate that ALGs might be released in the near future, although further research is still needed on ALG design considerations.

From the users' point of view, NAT64/DNS64 translation might not provide an experience comparable to native IPv6- or IPv4-only connectivity, especially since VoIP and IM applications are extensively used in home environments. However, in the corporate environment, most business applications will continue to perform as before.

## REFERENCES


[1] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt.

[2] H. Afifi and Laurent Toutain, "Methods for IPv4-IPv6 Transition", in *Proc. IEEE Symposium on Computers and Communications*, Sharm El Sheik, 1999, pp. 478–484. [Online]. Available: http://dx.doi.org/10.1109/ISCC.1999.780953.

[3] J. Bi, J. Wu and X. Leng, "IPv4/IPv6 Transition Technologies and Univer6 Architecture", International Journal of Computer Science and Network Security, vol. 7, pp. 232–243, Jan. 2007.

[4] M. Tatipamula, P. Grossetete and H. Esaki, "IPv6 Integration and Coexistence Strategies for Next-Generation Networks", IEEE Communications Magazine, vol. 42, pp. 88–96, Jan. 2004. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2004.1262167.

[5] D. G. Waddington and F. Chang, "Realizing the Transition to IPv6", IEEE Communications Magazine, vol. 40, pp. 138–148, Jun. 2002. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2002.1007420.

[6] D. Wing, "Network Address Translation: Extending the Internet Address Space", IEEE Internet Computing, vol. 14, pp. 66–70, Jul. 2010. [Online]. Available: http://dx.doi.org/10.1109/MIC.2010.96.

[7] S. Perreault, "Ecdysis: Open-Source Implementation of a NAT64 Gateway", Feb. 2010. [Online]. Available: http://ecdysis.viagenie.ca.

[8] Microsoft Corporation, "Using Integrated NAT64 and DNS64 with Forefront UAG DirectAccess", Feb. 2010. [Online]. Available: http://technet.microsoft.com/en-us/library/ee809079.aspx.

[9] N. Lutchansky, "TAYGA: Simple, no-fuss NAT64 for Linux", Dec. 2010. [Online]. Available: http://http://www.litech.org/tayga/.

[10] X. Che and D. Lewis, "IPv6: Current Deployment and Migration Status", International Journal of Research and Reviews in Computer Science, vol. 1, pp. 22–29, Jun. 2010.

[11] O. Martin, "Where is the Internet heading to?", Journal of Physics: Conference Series, vol. 219, part 6, 2010. [Online]. Available: http://dx.doi.org/10.1088/1742-6596/219/6/062019.

[12] R. AlJa'afreh, J. Mellor and I. Awan, "Implementation of IPv4/IPv6 BDMS Translation Mechanism", in *Proc. UKSIM European Symposium on Computer Modeling and Simulation*, Liverpool, 2008, pp. 512–517. [Online]. Available: http://dx.doi.org/10.1109/EMS.2008.71.

[13] R. AlJa'afreh, J. Mellor and I. Awan, "Evaluating BDMS and DSTM Transition Mechanisms", in *Proc. UKSIM European Symposium on Computer Modeling and Simulation*, Liverpool, 2008, pp. 488–493. [Online]. Available: http://dx.doi.org/10.1109/EMS.2008.60.

[14] R. AlJa'afreh, J. Mellor and I. Awan, "A Comparison Between the Tunneling Process and Mapping Schemes for IPv4/IPv6 Transition", in *Proc. International Conference on Advanced Information Networking and Applications*, Bradford, 2009, pp. 601–606. [Online]. Available: http://dx.doi.org/10.1109/WAINA.2009.209.

[15] W.-E. Chen, Q. Wu, Y.-B. Lin, Y.-C. Lo, "Design of SIP Application Level Gateway for IPv6 Translation", Journal of Internet Technology, vol. 5, pp. 343–348, Apr. 2004.

[16] P. Srisuresh and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3022.txt.

[17] G. Tsirtsis and P. Srisuresh, "Network Address Translation – Protocol Translation (NAT-PT)", RFC 2766, 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2766.txt.

[18] J. Hagino and K. Yamamoto, "An IPv6-to-IPv4 Transport Relay Translator", RFC 3142, 2001. [Online]. Available: http://www.ietf.org/rfc/rfc3142.txt.

[19] P. Srisuresh and H. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2663.txt.

[20] C. Aoun and E. Davies, "Reasons to Move the Network Address Translator – Protocol Translator (NAT-PT) to Historic Status", RFC 4966, 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4966.txt.

[21] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, 2010. [Online]. Available: http://www.ietf.org/rfc/rfc6052.txt.

[22] M. Bagnulo, P. Matthews and I. van Beijnum, "Stateful NAT64: Network Address Translation and Protocol Translation for IPv6 Clients to IPv4 Servers", RFC 6146, 2011. [Online]. Available: https://www.ietf.org/rfc/rfc6146.txt.

[23] X. Li, C. Bao and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6145.txt.

[24] F. Baker, X. Li, C. Bao and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6144.txt.

[25] M. Bagnulo, A. Sullivan, P. Matthews and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6144.txt.

[26] D. Wing, "Coping with IP Address Literals in HTTP URIs with IPv6/IPv4 Translators", draft-wing-behave-http-ip-address-literals-02 (IETF Internet Draft, expired), 2010. [Online]. Available: http://tools.ietf.org/html/draft-wing-behave-http-ip-address-literals-02.

[27] J. Davies, "TCP/IP Fundamentals for Microsoft Windows", Microsoft Corporation, Feb. 2008.

[28] D. Wing, "Referrals Across an IPv6/IPv4 Translators", draft-wing-behave-nat64-referrals-01 (IETF Internet Draft, expired), 2009. [Online]. Available: http://tools.ietf.org/html/draft-wing-behave-nat64-referrals-01.

[29] I. van Beijnum, "An FTP ALG for IPv6-to-IPv4 translation", draft-ietf-behave-ftp64-07 (IETF Internet Draft), 2011. [Online]. Available: http://tools.ietf.org/html/draft-ietf-behave-ftp64-07.