Practical evaluation of stateful NAT64/DNS64 translation

Article *in* Advances in Electrical and Computer Engineering · August 2011

DOI: 10.4316/aece.2011.03008

2 authors, including:

Mojca Ciglarič
University of Ljubljana
41 PUBLICATIONS **135** CITATIONS

SEE PROFILE

Practical evaluation of NAT64 and DNS64 translation

Nejc Škoberne[1] and Mojca Ciglarič[2]

[1]Viris d. o. o., Likozarjeva ulica 12, SI-1000 Ljubljana
[2]Faculty of Computer and Information Science,
University of Ljubljana, Tržaška cesta 25, SI-1000 Ljubljana
nejc@viris.si, mojca.ciglaric@fri.uni-lj.si

Abstract. The often suggested approach to coexistence of IPv4 and IPv6 protocols is dual-stack deployment, however in certain environments its implementation is infeasible. As NAT-PT (Network Address Translation – Protocol Translation) specifiction has been deprecated, NAT64 and DNS64 Internet Drafts have been proposed, supporting only IPv6-to-IPv4 translation scenario. Now the question of usability in real world arises. In this paper, we systematically test a number of widely used application-layer network protocols regarding how well they traverse Ecdysis, the first open source NAT64 and DNS64 implementation. We first propose three levels for assessing translation quality and according to these we practically evaluate 18 popular application layer protocols, among them HTTP, RDP, MSNP, IMAP... and discuss the shortcomings of such translation that might not be apparent at the first sight.

Keywords: NAT64, DNS64, network address translation, IPv6, Ecdysis, protocol translation quality

1 Introduction and Related Work

After the IPv6 protocol [13] had been adopted as the new-generation Internet protocol, the researchers started to investigate different approaches to transition from IPv4 to IPv6 protocol [2] [11] [19]. We call these approaches *transition mechanisms* and they can be roughly categorized into three categories: *tunneling*, *translation* and *dual-stack*. The three are overviewed in [21].

Tunneling is most often used when two isolated IPv6 -enabed 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, however new IPv4 header is formed and used on the way through IPv4 network. On the tunnel exit, IPv6 datagrams are decapsulated again. Of course also IPv4 communication through IPv6 tunnel is possible.

In dual stack scenario, the critical nodes (servers, routers) implement both protocol stacks, IPv4 and IPv6. Dual stack results in unnecessary high complexity of network backbone as well as end systems. This way, end systems may run

IPv6 and IPv4 applications, servers may receive connections from IPv4-only as well as IPv6-only hosts, routers may handle IPv4 and IPv6 traffic IPv4-only devices may communicate to other IPv4-only devices or to dual stack devices through the network using only IPv4, while simultaneously IPv6 or dual stack devices may communicate using IPv6 through the same network. The situation gets complicated when communication is requested between IPv4-only and IPv6-only entities.Translation of network protocols and addresses between IPv4 and IPv6 realms is neccessary to enable cross-version 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 datagrams need to be converted into IPv4 datagrams. This means that the IPv6 head is discarded and substituted with IPv4 head, whereas the contents of the fields with no clear mapping in the target protocol head are lost [**?**].

In 2010, the first open source implementation of NAT64 gateway and DNS64 server called Ecdysis [16], has been made publicly available. Until then, we could only theoretically discuss NAT64/DNS64 feasibility. Ecdysis is available for Linux and BSD operating systems. It includes IP translator and DNS application layer gateway, implemented within Unbound and Bind open-source DNS servers. In Linux, the IP translator is implemented as a kernel module using Netfilter facilities. In BSD operating system, it is available as a modification of the PF firewall. Ecdysis is modestly documented and not yet extensively tested in real-world environments. At the time of writing we are not aware of other NAT64/DNS64 implementations however there are some (partial) implementations of the obsolete NAT-PT (RFC 2766), eg. [1].

The NAT64 and DNS64 mechanism specifications, as described in the new Internet Drafts seem very consistent and ready for implementation. It is to be expected that with the anticipated depletion of IPv4 address space 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 the administrators to know what they can expect from translation mechanisms: is the behavior of the applications going to change under translation? Maybe only if they are using specific application layer protocols? How well are these protocols going to be translated? Is translation feasible for every protocol at all?

Therefore, the purpose of this paper is twofold: first, we will evaluate translation of different application-layer network protocol theoretically, and then we will empirically evaulate traversal of these application-layer network protocols over Ecdysis in a controlled environment. We will not, however, test other aspects of translation, such as scalability and performance. The paper will be of interest to application adminsitrators, 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 everybody interested in general IPv6 transition problems and technologies.

The paper is organized as follows. In the rest of this section we overview the area of 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 also theoretically assess translation quality of the chosen protocols. In the next step, we describe the test beds used for testing 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 selected set of application-layer protocols over Ecdysis NAT64/DNS64 translator. We conclude with practical implications and a discussion of the future research possibilities.

The research on IPv4 and IPv6 coexistence is scarce. Che and Lewis [**?**] compare transition mechanisms and evaluate their effectiveness regarding packet delay and packet loss in a simulated environment. The authors expose some migration challenges, however NAT64/DNS64 is not addressed. Martin [**?**] points out that the IPv4 to IPv6 transition strategy is incomplete and IPv4 to IPv6 migration process will be more difficult than originally thought, hence the need for additional translation mechanisms. No further discussion on translation is offered. AlJaafreh et al. [**?**], [5] and [4] discuss bidirectional mapping among native IPv4 and IPv6 networks, examine its performance by means of a network simulator and compare performance with tunneling and dual stack approach. Chen [**?**] proposes an ALG for SIP protocol, however it is not really 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 [**?**] overviews and compares the differences in approach to NAT as known in IPv4 networks and NAT in IPV6 and mixed networks. It appears that no research has been published directly related to our work, probably due to the fact that NAT64 and DNS64 translation methods are still to be massively implemented.

2 Translation Mechanisms

In this paper we focus on translation mechanisms. Translation is conceptually challenging and highly complex mechanism and IETF issued several internet drafts on this problem. Although NAT (Network Address Translation) is a well established concept in IPv4 world [17], the translation between IPv4 and IPv6 addresses brings a range of new problems, which different translation mechanisms – NAT64/DNS64, NAT-PT and NAPT-PT (Network Address Port Translation – Protocol Translation) [20], TRT (Transport Relay Translation) [14] etc. – try to address each in its own way. We can split the translation problem in two sub problems 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 translation. NAT46 (IPv4-to-IPv6) address translation 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 conversion of the network layer address information found inside the application payload into the address acceptable by the hosts on the other side of the firewall/NAT device. In other words, ALGs are application specific translation agents that allow an application on a host in one address realm to connect to its counterpart running on a host in different realm transparently [18].

The main advantage of translation is that the communicating devices need not be changed in any way, thus 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. This way, 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 translation mechanism was NAT-PT, which although adopted by some major vendors was too complex and has been made deprecated by RFC 4966 [6]. 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 [**?**], the main reason for its deprecation were operational complexities, but deeper discussion of its issues is out of the scope of this paper and can be found in the aforementioned RFC. NAT64 translation, however, is still useful in IPv6-only environments in order to access the IPv4 Internet and the underlying translation method is called NAT64/DNS64 (from now on, these terms will refer to translation methods, not translation sub problems). In 2008, the first NAT64 and DNS64 Internet Drafts have been proposed [10] [7] [15] [9] [8], supporting only some specific cases of IPv6-to-IPv4 translation. In addition, NAT64 draft currently only allows for translation of TCP and UDP transport-layer protocols and network-layer protocol ICMP. The application-layer protocols relying on transport-layer protocols other than TCP or UDP are not supposed to traverse a NAT64 translator.

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

Fig. 1. An example of NAT64 translation with the help of DNS64 server.

3 Experimental Method

First we have 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 metrics for protocol translation quality. We have also theoretically evaluated how well different types of protocols are expected to translate. 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 have also analyzed packet trace files and the reasons for incompatibility.

3.1 Selection of Protocols to be Tested

The selection of protocols (see Table 2) was made according to our personal experience. We considered three groups of users – corporate users, mobile users and home users (these groups are not disjoint, however, 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 client, instant messaging client, soft phone (VoIP) client, terminal console client and VPN client. We considered each application and figured out which application-layer protocols does it use. DNS protocol is not included since IPv6-only machines behind the NAT64 translator **must** use the DNS64 server in order to be able to connect to the IPv4 world. Use of DNS servers other than DNS64 is irrelevant. For each of the selected protocols, we evaluated the quality of NAT64 traversal first theoretically and than also empirically, using the protocol in our test bed and inspecting what happens with protocol messages after Ecdysis traversal.

3.2 Translation Quality Metrics

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

Table 1. List of selected application-layer protocols

Protocol Acronym	Protocol Name
BitTorrent	BitTorrent (peer-to-peer file sharing protocol)
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMAP	Internet Message Access Protocol
NTP	Network Time Protocol
POP3	Post Office Protocol (version 3)
RDP	Remote Desktop Protocol
Skype	Skype (peer-to-peer VoIP protocol)
MSNP	Microsoft Notification Protocol
SIP	Session Initiation Protocol
SMB/CIFS	Server Message Block/Common Internet File System
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
TELNET	Terminal Network
OpenVPN	OpenVPN (Virtual Private Network)
IPsec	IPsec (IP security)
PPTP	Point-to-Point Tunneling Protocol

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

Conditionally Translated Protocols. Conditionally translated protocols are protocols, which need some kind of special treatment in order to be successfully translated over a NAT64 translator. With 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 server or at client side. After the special treatment is applied, at least core protocol functionality is successfully provided over a NAT64 translator.

Poorly Translated Protocols. If nothing could be done in order to make a protocol traverse a NAT64 translator and any special treatment is infeasible to implement, the protocol is classified as poorly translated.

When we had the translation quality metrics defined, we were able to state some predictions or hypotheses about the chosen protocols translation quality. These hypotheses will need to be empirically confirmed in our testbed:

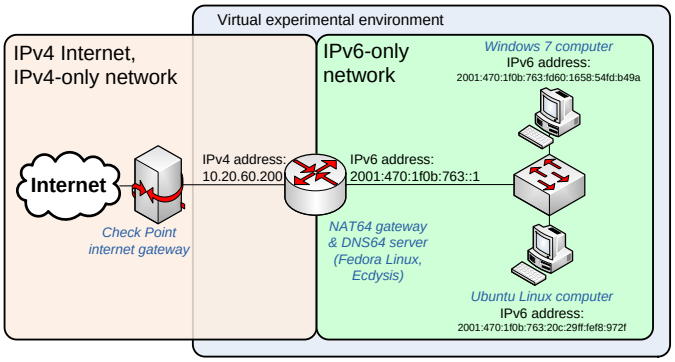

Fig. 2. The experimental environment for translation quality evaluation

– 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 conditionally translated class. For example, SIP needs an ALG even when performing IPv4 network address translation [17].

– The protocols using transport layer protocols other than TCP and UDP (for example GRE or ESP) will classify as poorly translated, since NAT64 translator only implements translation of TCP, UDP and ICMP-based protocols.

3.3 Testbed and Experimental Details

The test bed scheme and topology are shown in Fig. 2. The experimental environment was virtualized and consisted of a NAT64 translator and DNS64 server (Fedora Linux with Ecdysis), DHCPv6 server and two IPv6-only clients, one with Ubuntu Linux and the other with Windows 7 operating system. Ecdysis implementation of NAT64 and DNS64 methods only supports the "IPv6-to-IPv4" scenarios, which means that IPv6-only machines can establish connections towards the IPv4-only Internet. Consequently we tested the translation in IPv6-to-IPv4 direction. We decided to use BIND because of its widespread use, although Ecdysis also supports Unbind DNS server. Router Advertisement daemon `radvd` was announcing the prefix `2001:470:1f0b:763::/64` on the internal IPv6-only network. DHCPv6 server was used to configure 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 for the user, the application performed as if there were no translation, the test was finished at that point. Otherwise, when the protocol would not belong to the Well Trans-

Table 2. Translation quality of the selected protocols

	Translation Quality	Translation Quality
BitTorrent	Conditionally	Conditionally
FTP	Conditionally	Conditionally
HTTP	Well	Well
HTTPS	Well	Well
IMAP	Well	Well
NTP	Well	Well
POP3	Well	Well
RDP	Well	Well
Skype	Poorly	Poorly
MSNP	Poorly	Poorly
SIP	Poorly	Poorly
SMB/CIFS	Well	Well
SMTP	Well	Well
SSH	Well	Well
TELNET	Well	Well
OpenVPN	Conditionally	Poorly
IPsec	Poorly	Poorly
PPTP	Poorly	Poorly

lated Protocols class, we would also analyze packet trace files and identified the reasons for incompatibility.

4 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.

4.1 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 in its place). As shown in Table 2, we empirically proved the translation quality for all well-translated protocols, however there are two exceptions:

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 (for example http://111.112.113.114) , which is not accessible to IPv6-only HTTP clients using NAT64 translator since the address is not known to DNS64 server (see the typical scenario depicted in Figure 1). In [22] Wing proposes using a HTTP proxy to handle such traffic as a workaround. Althought it overcomes the described problem, operating a HTTP proxy interfaced

to IPv4 Internet is not a trivial task, is more resource-intensive, complicates the network topology and increases 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=111.112.113.114).

SMB/CIFS. In our experiment, we could only empirically evaluate file transfer. The Microsoft NetBIOS-over-TCP/IP (NBT) name resolution and service location protocol couldn't be tested, since Microsoft's implementation does not support IPv6 [12]. Link-local Multicast Name Resolution protocol (LLMNR) also couldn't be tested, since it uses link-local multicast addresses, which are only valid within one network segment. Since NAT64 translator, terminates the network segment, LLMNR traffic cannot traverse the translator. Therefore, using DNS for name resolution is suggested when using SMB/CIFS protocol with NAT64 translation.

4.2 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.

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.

BitTorrent. The problem of BitTorrent traversal over NAT64 was identified by Wing [**?**]: although BitTorrent packets would traverse NAT64 and reach their destnation, 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. This way BitTorrent clients wouldn't 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.

FTP. Some considerations about IPv6-to-IPv4 translation of FTP can be found in [**?**]. 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.

4.3 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.

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 if IPv6-ready Skype will be able to pass NAT64 translators, since we do not know how Skype will implement IPv6.

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

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

IPsec. To encapsulate transport-layer protocols, IPsec uses ESP (Encapsulating Security Payload), 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 another method exists enabling IPsec-protected datagrams to pass through IPv4 NAT translators. It is called NAT-T (NAT Traversal in IKE (Internet Key Exchange)), but it was not tested in our research.

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

5 Conclusion

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 [**?**] indicate that ALGs might be released in near future, although further research is still needed on ALG design considerations.

6 Acknowledgements

Operation part-financed by the European Union, European Social Fund.

References

1. Using integrated NAT64 and DNS64 with Forefront UAG DirectAccess (feb 2010), http://technet.microsoft.com/en-us/library/ee809079.aspx
2. Afifi, H., Toutain, L.: Methods for IPv4-IPv6 Transition. In: ISCC. pp. 478–484. IEEE Computer Society (1999), http://csdl.computer.org/comp/proceedings/iscc/1999/0250/00/02500478abs.htm
3. Al-Dabass, D., Nagar, A., Tawfik, H., Abraham, A., Zobel, R.N. (eds.): EMS 2008, Second UKSIM European Symposium on Computer Modeling and Simulation, Liverpool, England, UK, 8-10 September 2008. IEEE Computer Society (2008)
4. AlJa'afreh, R., Mellor, J., Awan, I.: Implementation of IPv4/IPv6 BDMS Translation Mechanism. In: Al-Dabass et al. [3], pp. 512–517
5. AlJa'afreh, R., Mellor, J., Awan, I.U.: A Comparison Between the Tunneling Process and Mapping Schemes for IPv4/IPv6 Transition. In: AINA Workshops. pp. 601–606. IEEE Computer Society (2009)
6. Aoun, C., Davies, E.: Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status. RFC 4966 (Informational) (jul 2007), http://www.ietf.org/rfc/rfc4966.txt
7. Bagnulo, M., Matthews, P., van Beijnum, I.: Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. draft-ietf-behave-v6v4-xlate-stateful-12 (Standards Track) (jul 2010), http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate-stateful-12
8. Bagnulo, M., Sullivan, A., Shinkuro, Matthews, P., van Beijnum, I.: DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. draft-ietf-behave-dns64-10 (Standards Track) (jul 2010), http://tools.ietf.org/html/draft-ietf-behave-dns64-10
9. Baker, F., Li, X., Bao, C., Yin, K.: Framework for IPv4/IPv6 Translation. draft-ietf-behave-v6v4-framework-09 (Informational) (may 2010), http://tools.ietf.org/html/draft-ietf-behave-v6v4-framework-09
10. Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., Li, X.: IPv6 Addressing of IPv4/IPv6 Translators. draft-ietf-behave-address-format-09 (Standards Track) (jul 2010), http://tools.ietf.org/html/draft-ietf-behave-address-format-09
11. Bi, J., Wu, J., Leng, X.: IPv4/IPv6 Transition Technologies and Univer6 Architecture. International Journal of Computer Science and Network Security 7(1), 232–243 (2007), http://paper.ijcsns.org/07_book/html/200701/200701106.html
12. Davies, J.: TCP/IP Fundamentals for Microsoft Windows (feb 2008), http://www.microsoft.com/downloads/details.aspx?FamilyID=c76296fd-61c9-4079-a0bb-582bca4a846f&displaylang=en
13. Deering, S., Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard) (dec 1998), http://www.ietf.org/rfc/rfc2460.txt, updated by RFCs 5095, 5722, 5871
14. Hagino, J., Yamamoto, K.: An IPv6-to-IPv4 Transport Relay Translator. RFC 3142 (Informational) (Jun 2001), http://www.ietf.org/rfc/rfc3142.txt
15. Li, X., Bao, C., Baker, F.: IP/ICMP Translation Algorithm. draft-ietf-behave-v6v4-xlate-20 (Standards Track) (may 2010), http://tools.ietf.org/html/draft-ietf-behave-v6v4-xlate-20
16. Perreault, S.: Ecdysis: Open-Source Implementation of a NAT64 Gateway (feb 2010), http://ecdysis.viagenie.ca
17. Srisuresh, P., Egevang, K.: Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational) (Jan 2001), http://www.ietf.org/rfc/rfc3022.txt

18. Srisuresh, P., Holdrege, H.: IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663 (Informational) (aug 1999), http://www.ietf.org/rfc/rfc2663.txt
19. Tatipamula, M., Grossetete, P., Esaki, H.: IPv6 Integration and Coexistence Strategies for Next-Generation Networks. IEEE Communications Magazine 42(1), 88–96 (jan 2004), http://dl.comsoc.org/comsocdl/?article=1224193
20. Tsirtsis, G., Srisuresh, P.: Network Address Translation - Protocol Translation (NAT-PT). RFC 2766 (Historic) (Feb 2000), http://www.ietf.org/rfc/rfc2766.txt, obsoleted by RFC 4966, updated by RFC 3152
21. Waddington, D.G., Chang, F.: Realizing the Transition to IPv6. IEEE Communications Magazine 40(6), 138–148 (jun 2002), http://dl.comsoc.org/comsocdl/?article=221365
22. Wing, D.: Coping with IP Address Literals in HTTP URIs with IPv6/IPv4 Translators. draft-wing-behave-http-ip-address-literals-02 (Informational) (mar 2010), http://tools.ietf.org/html/draft-wing-behave-http-ip-address-literals-02