Behavior Engineering for Hindrance Avoidance (BEHAVE)
Internet-Draft
Intended status: Informational
Expires: August 26, 2011

J. Korhonen, Ed.
Nokia Siemens Networks
T. Savolainen, Ed.
Nokia

February 22, 2011

# Analysis of solution proposals for hosts to learn NAT64 prefix draft-korhonen-behave-nat64-learn-analysis-02.txt

## Abstract

Hosts and applications may benefit from the knowledge if an IPv6 address is synthesized, which would mean a NAT64 is used to reach the IPv4 network or Internet. This document analyses a number of proposed solutions for communicating whether the synthesis is taking place, used address format, and the IPv6 prefix used by the NAT64 and DNS64. This enables both NAT64 avoidance and intentional utilization by allowing local IPv6 address synthesis.

## Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 26, 2011.

Table of Contents

## 1. Introduction

Hosts and applications may benefit from the knowledge of whether an IPv6 address is synthesized, which would mean a NAT64 is used to reach the IPv4 network or Internet. There are two issues that can be addressed with solutions that allow hosts and applications to learn the Network Specific Prefix (NSP) [RFC6052] used by the NAT64 [I-D.ietf-behave-v6v4-xlate-stateful] and the DNS64 [I-D.ietf-behave-dns64] devices.

Firstly, finding out whether a particular address is synthetic and therefore learning the presence of a NAT64. For example, a Dual-Stack (DS) host with IPv4 connectivity could use this information to bypass NAT64 and use native IPv4 transport for destinations that are reachable through IPv4. We will refer this as 'Issue #1' throughout the document.

Secondly, finding out how to construct from an IPv4 address an IPv6 address that will be routable to/by the NAT64. This is useful when IPv4 literals can be found in the payload of some protocol or applications do not use DNS to resolve names to addresses but know the IPv4 address of the destination by some other means. We will refer this as 'Issue #2' throughout the document.

Additionally three other issues have to be considered by a solution addressing the first two issues: whether DNS is required 'Issue #3', whether a solution supports changing NSP 'Issue #4', and whether multiple NSPs are supported (either of the same or different length) for load-balancing purposes 'Issue #5'.

This document analyses all known solution proposals known at the time of writing for communicating if the synthesis is taking place, used address format, and the IPv6 prefix used by the NAT64 and DNS64. Based on the analysis we conclude whether the issue of learning the Network-Specific Prefix is worth solving and what would be the recommended solution(s) in that case.

## 2. Terminology and Assumptions

NSP

Network-Specific Prefix: A prefix chosen by network administrator for NAT64/DNS64 to present IPv4 addresses in IPv6 namespace.

WKP

Well-Known Prefix: A prefix (64:ff9b::/96) chosen by IETF and configured by a network administrator for NAT64/DNS64 to present IPv4 addresses in IPv6 namespace.

NAT64

Network Address and protocol Translation mechanism for translating IPv6 packets to IPv4 packets and vice-versa: A network entity that a host or an application may want to either avoid or utilize. IPv6 packets hosts send to addresses in the NSP and/or WKP are routed to NAT64.

DNS64

DNS extensions for network address translation from IPv6 clients to IPv4 servers: A network entity that synthesizes IPv6 addresses and AAAA records out of IPv4 addresses and A records, hence making IPv4 namespaces visible into IPv6 namespace. DNS64 uses NSP and/or WKP in the synthesis process.

Address Synthesis

A mechanism, in the context of this document, where an IPv4 address is represented as an IPv6 address understood by a NAT64 device. The synthesized IPv6 address is formed by embedding an IPv4 address as-is into an IPv6 address prefixed with a NSP/WKP. It is assumed that the 'unused' suffix bits of the synthesized address are set to zero as described in Section 2.2 of [RFC6052].

Issue #1

The problem of distinguishing between a synthesized and a real IPv6 addresses, which allows a host to learn the presence of a NAT64 in the network.

Issue #2

The problem of learning the NSP used by the access network and needed for local IPv6 address synthesis.

Issue #3

The problem of learning the NSP or WKP used by the access network by a host not implementing DNS (hence applications are unable to use DNS to learn prefix).

Issue #4

The problem of supporting changing NSP. The NSP learned by the host may become stale for multiple reasons. For example, the host might move to a new network that uses different NSP, thus making the previously learned NSP stale. Also, the NSP used in the network may be changed due administrative reasons, thus again making previously learned NSP stale.

Issue #5

The problem of supporting multiple NSPs. A network may be configured with multiple NSPs for address synthesis. For example, for load-balancing purposes each NAT64 device in the same network could be assigned with their own NSP. It should be noted that learning a single NSP is enough for an end host to successfully perform local IPv6 address synthesis but to avoid NAT64 the end host needs to learn all NSPs used by the access network.

## 3. Background

Certain applications, operating in protocol translation scenarios, can benefit from knowing the IPv6 prefix used by a local NAT64 of the attached access network. This applies to the Framework document [I-D.ietf-behave-v6v4-framework] Scenario 1 ("IPv6 network to IPv4 Internet"), Scenario 5 ("An IPv6 network to an IPv4 network"), and Scenario 7 ("The IPv6 Internet to the IPv4 Internet"). Scenario 3("The IPv6 Internet to an IPv4 network") is not considered applicable herein as in that case a NAT64 is located at the front of remote IPv4 network and host in IPv6 Internet can benefit very little of learning NSP IPv6 prefix used by the remote NAT64. The NAT64 prefix can be either a Network Specific Prefix (NSP) or the Well-known Prefix (WKP). Below is (an incomplete) list of various use cases where it is beneficial for a host or an application to know the presence of a NAT64 and the NSP/WKP:

- Host-based DNSSEC validation: as is documented in DNS64 [I-D.ietf-behave-dns64] section 5.5. point 3, synthetic AAAA records cannot be successfully validated in a host. In order to utilize NAT64 a security-aware and validating host has to perform DNS64 function locally and hence it has to be able to learn WKP or proper NSP.

- Protocols that use IPv4 literals: in IPv6-only access native IPv4 connections cannot be created. If a network has NAT64 it is possible to synthesize IPv6 address by combining the IPv4 literal and the IPv6 prefix used by NAT64. The synthesized IPv6 address

can then be used to create an IPv6 connection.

- Multicast translations [I-D.venaas-behave-mcast46][I-D.venaas-behave-v4v6mc-framework].

- URI schemes with host IPv4 address literals rather than domain names (e.g., http://192.0.2.1, ftp://192.0.2.1, imap://192.0.2.1, ipp://192.0.2.1)): a host can synthesize IPv6 address out of the literal in URI and use IPv6 to create connection through NAT64.

- Updating host's [RFC3484] preference table to prefer native prefixes over translated prefixes: this is useful as applications are more likely able to traverse through NAT44 than NAT64.

DNS64 cannot serve applications that are not using DNS or that obtain referral as an IPv4 literal address. One example application is the Session Description Protocol (SDP) [RFC4566], as used by Real Time Streaming Protocol (RTSP) [RFC2326] and Session Initiation Protocol (SIP) [RFC3261]. Other example applications include web browsers, as IPv4 address literals are still encountered in web pages and URLs. Some of these applications could still work through NAT64, provided they were able to create locally valid IPv6 presentations of peers' IPv4 addresses.

It is a known issue that passing IP address referrals, often fails in today's Internet [I-D.carpenter-referral-ps]. Synthesizing IPv6 addresses does not necessarily make the situation any better as the synthesized addresses are not distinguishable from public IPv6 addresses for the referral receiver. However, the situation is not really any different from the current Internet as using public addresses does not really guarantee reachability (for example due firewalls). Therefore, we think that it is up to the referral originating host to somehow identify that the IPv6 address is made-up.

# 4. Proposed solutions to learn about synthesis and Network-Specific Prefix

## 4.1. EDNS0 option indicating AAAA Record synthesis and format

### 4.1.1. Solution description

Section 3 of [I-D.korhonen-edns0-synthesis-flag] defines a new EDNS0 option [RFC2671], which contain 3 flag bits (called SY-bits). The EDNS0 option serves as an implicit indication of the presence of DNS64 server and the NAT64 device. The EDNS0 option SY-bit values other than '000' and '111' explicitly tell the NSP prefix length.

Only the DNS64 server can insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA Resource Record.

#### 4.1.2. Analysis and discussion

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1 and is designed to explicitly solve Issue #2.

+ Solves issue #4 via DNS record lifetime.

+ Can partially solve issue #5 if multiple synthetic AAAA records are included in the response and all use same format.

+ The solution is backward compatible from 'legacy' hosts and servers point of view.

+ Even if the solution is bundled with DNS queries and responses, a standardization of a new DNS record type is not required, rather just defining a new EDNS0 option.

+ EDNS0 option implementation requires changes only to DNS64 servers.

+ Does not require additional provisioning or management as the EDNS0 option is added automatically by the DNS64 server to the responses.

+ Does not involve additional queries towards the global DNS infrastructure as EDNS0 logic can be handled within the DNS64 server.

The CONs of the proposal are listed below:

- Requires end hosts to support [RFC2671] EDSN0 extension mechanism.

- Requires host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length.

- Requires a modification to DNS64 servers to include the EDNS0 option to the synthesized responses.

- Does not provide solution for issue #3.

#### 4.1.3. Summary

The EDNS0 option based solution works by extending the existing EDNS0 Resource Record. Although the solution has host resolver and DNS64 server impacts, the changes are limited to those entities (end host, applications) that are interested in learning the presence of NAT64 and the used NAT64 prefix. The provisioning and management overhead is minimal if not non-existent as the EDNS0 options are synthesized in a DNS64 server in a same manner as the synthesized AAAA Resource Records. Moreover, EDNS0 does not induce any load to DNS servers because no new RRType query is defined.

### 4.2. EDNS0 flags indicating AAAA Record synthesis and format

#### 4.2.1. Solution description

Section 3 of [EDNS0-Flag] defines 3 new flag bits (called SY-bits) into EDNS0 OPT [RFC2671] header, which serve as an implicit indication of the presence of DNS64 server and a NAT64 device. SY-bit values other than ’000’ or ’111’ explicitly tell the NSP prefix length. Only the DNS64 server can insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA Resource Record.

#### 4.2.2. Analysis and discussion

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1 and is designed to explicitly solve Issue #2.

+ Solves issue #4 via DNS record lifetime.

+ Can partially solve issue #5 if multiple synthetic AAAA records are included in the response and all use same format.

+ The solution is backward compatible from ’legacy’ hosts and servers point of view.

+ EDNS0 option implementation requires changes only to DNS64 servers.

+ Does not require additional provisioning or management as the EDNS0 option is added automatically by the DNS64 server to the responses.

+ Does not involve additional queries towards the global DNS infrastructure as EDNS0 logic can be handled within the DNS64 server.

The CONs of the proposal are listed below:

- Requires end hosts to support [RFC2671] EDSN0 extension mechanism.

- Consumes scarce flag bits from EDNS0 OPT header.

- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length.

- Requires a modification to DNS64 servers to include the EDNS0 option to the synthesized responses.

- Does not provide solution for issue #3.

#### 4.2.3. Summary

This option is included here for the sake of completeness. The consumption of three bits of the limited EDNS0 OPT space can be considered unfavorable and hence is unlikely to be accepted.

### 4.3. DNS Query for a Well-Known Name

#### 4.3.1. Solution description

Section 3 of [I-D.savolainen-heuristic-nat64-discovery] describes a host behavior for discovering the presence of a DNS64 server and a NAT64 device, and heuristics for discovering the used NSP. A host requiring information for local IPv6 address synthesis or for NAT64 avoidance sends a DNS query for an AAAA record of a Well-Known IPv4-only Fully Qualified Domain Name (FQDN). If a host receives a negative reply, it knows there are no DNS64 and NAT64 in the network.

If a host receives AAAA reply, it knows the network must be utilizing IPv6 address synthesis. After receiving a synthesized AAAA Resource Record, the host may examine the received IPv6 address and use heuristics, such as "subtracting" the known IPv4 address out of synthesized IPv6 address, to find out the NSP.

The Well-Known Name may be assigned by IANA or provided some third party, including application or operating system vendor. The IPv4 address corresponding to the Well-Known Name may be resolved via A query to Well-Known Name, assigned by IANA, or hard-coded.

#### 4.3.2. Analysis and discussion

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1 and Issue #2.

+ Solves issue #4 via DNS record lifetime.

+ Can partially solve issue #5 if multiple synthetic AAAA records are included in the response. Can find multiple address formats.

+ Does not necessarily require any standards effort.

+ Does not require host stack or resolver changes. All required logic and heuristics can be implemented in applications that are interested in learning about address synthesis taking place.

+ The solution is backward compatible from ’legacy’ hosts and servers point of view.

+ Hosts or applications interested in learning about synthesis and the used NSP can do the "discovery" proactively at any time, for example every time the host attaches to a new network.

+ Does not require explicit support from the network using NAT64

The CONs of the proposal are listed below:

- Requires hosting of a DNS resource record for the Well-Known Name.

- Does not provide solution for issue #3.

- This method is only able to find one NSP even if a network is utilizing multiple NSPs (issue #5) (unless DNS64 includes multiple synthetic AAAA records in response).

#### 4.3.3. Summary

This is the only approach that can be deployed without explicit support from the network or the host. This approach could also complement explicit methods and be used as a fallback approach when explicit methods are not supported by an access network.

### 4.4. DNS Resource Record for IPv4-Embedded IPv6 address

#### 4.4.1. Solution description

Section 4 of [I-D.boucadair-behave-dns-a64] defines a new DNS Resource Record (A64) that is a record specific to store a single IPv4-Embedded IPv6 address [RFC6052]. Using a dedicated Resource Record allows a host to distinguish between real IPv6 addresses and synthesized IPv6 addresses. The solution requires host to send a query for an A64 record. Positive answer with A64 record informs the requesting host that the resolved address is not a native address but an IPv4-Embedded IPv6 address. This would ease the local policies to prefer direct communications (i.e., avoid using IPv4-Embedded IPv6 addresses when a native IPv6 address or a native IPv4 address is available). Applications may be notified via new or modified API.

#### 4.4.2. Analysis and discussion

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1 and #5.

+ Solves issue #4 via DNS record lifetime.

+ The solution is backward compatible from ’legacy’ hosts and servers point of view.

+ Synthesized addresses can be used in authoritative DNS servers.

+ Maintains the reliability of the DNS model (i.e., a synthesised IPv6 address is presented as such and not as native IPv6 address).

+ When both IPv4-Converted and native IPv6 addresses are configured for the same QNAME, native addresses are preferred.

The CONs of the proposal are listed below:

- Does not address Issue #2 or #3 in any way.

- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized.

- Requires a standardization of a new DNS Resource Record type (A64), and the implementation of it in both resolvers and servers.

- Requires a coordinated deployment between different flavors of DNS servers within the provider to work deterministically.

- Additional load the DNS servers (3 Queries, A64, AAAA and A, may be issued by a dual-stack host).

- Does not help to identify synthesized IPv6 addresses if the session does not involve any DNS queries.

#### 4.4.3. Summary

While the proposed solution delivers explicit information about address synthesis taking place solving the Issue #1, a standardization of a new DNS record type might turn out a too overwhelming task for a solution for a temporary transition phase. Defining a new record type increases load towards DNS server as the host issues parallel A64, AAAA and A queries.

### 4.5. Learning the IPv6 Prefix of a Network's NAT64 using DNS

#### 4.5.1. Solution description

Section 4.1 of [I-D.wing-behave-learn-prefix] actually proposes two DNS-based method for discovering the presence of a DNS64 server and a NAT64 device, and then a mechanism for discovering the used NSP. First, a host may learn the presence of a DNS64 server and a NAT64 device, by receiving a TXT Resource Record with a well-known (TBD IANA registered?) string followed by the NAT64 unicast IPv6 address and the prefix length. The DNS64 server would add the TXT Resource Record into the DNS response.

Second, Section 4.1 of [I-D.wing-behave-learn-prefix] also proposes specifying a new U-NAPTR [RFC4848] application to discover the NAT64's IPv6 prefix and length. The input domain name is exactly the same as would be used for a reverse DNS lookup, derived from the host's IPv6 in the ".ip6.arpa." tree. The host doing the U-NAPTR queries may need multiple queries until finds the provisioned domain name with the correct prefix length. The response to a successful U-NATPR query contains the unicast IPv6 address and the prefix length of the NAT64 device.

#### 4.5.2. Analysis and discussion

[Editor' note: can this be made to solve issue #5 by having multiple NSPs in TXT record?]

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1 and Issue #2.

+ Solves issue #4 via DNS record lifetime.

+ Does not require host stack or resolver changes if the required logic and heuristics is implemented in applications that are interested in learning about address synthesis taking place.

The CONs of the proposal are listed below:

- Requires standardization of a well-known names from IANA for TXT Resource Record and/or a standardization of a new U-NAPTR application.

- Requires a host resolver changes and a mechanism/additions to the host resolver API (or flags, hints etc) to deliver a note to the querying application that the address is synthesized and what is the NSP prefix length. However, it is possible that the U-NAPTR application logic is completely implemented by the application itself as noted in PROs list.

- U-NAPTR prefix learning method may entail multiple queries.

- U-NAPTR prefix learning method requires provisioning of NSPs in ".ip6.arpa." tree.

- RFC5507 [RFC5507] specifically recommends against reusing TXT Resource Records to expand DNS.

- Requires configuration on the access network’s DNS servers.

- Does not provide solution for issue #3.

#### 4.5.3. Summary

The implementation of this solution requires some changes to the applications and resolvers in a similar fashion as in solutions in Section 4.1, Section 4.2 and Section 4.4. Unlike the other DNS-based approaches, the U-NAPTR-based solution also requires provisioning information into the ’.ip6.arpa.’ tree which is not any more entirely internal to the provider hosting the NAT64/DNS64 service.

The iterative approach of learning the NAT64 prefix in U-NAPTR-based solution may result in multiple DNS queries, which can be considered more complex and inefficient compared to other DNS-based solutions.

### 4.6. Learning the IPv6 Prefix of a Network's NAT64 using DHCPv6

#### 4.6.1. Solution description

Two individual drafts specify DHCPv6 based approaches.

Section 4.2 of [I-D.wing-behave-learn-prefix] describes a new DHCPv6 [RFC3315] option (OPTION_AFT_PREFIX_DHCP) that contains the IPv6 unicast prefix, IPv6 ASM prefix, and IPv6 SSM prefix (and their lengths) for the NAT64.

Section 4 of [I-D.boucadair-dhcpv6-shared-address-option] defines a DHCPv6 option that can be used to communicate to a requesting host the prefix used for building IPv4-Converted IPv6 addresses together with the format type. Provisioning the format type is required so as to be correctly handled by the NAT64-enabled devices deployed in a given domain.

#### 4.6.2. Analysis and discussion

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1, Issue #2, Issue #3 and Issue #4 via DHCPv6 information lifetime.

+ Does not involve DNS system. Therefore, applications that would not normally initiate any DNS queries can still learn the NAT64 prefix.

+ DHCPv6 is designed to provide various kinds of configuration information in a centrally managed fashion.

The CONs of the proposal are listed below:

- Change of NSP requires change to DHCPv6 configuration.

- Requires at least Stateless DHCPv6 client on hosts.

- Requires support on DHCPv6 clients, which is not trivial in all operating systems.

- The DHCPv6-based solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

- A new DHCPv6 option is required and the corresponding changes to both DHCPv6 clients and servers.

If DHCPv6 would include multiple NSPs issue #5 could be solved as well, but only if nodes as a group would select different NSPs hence supporting load-balancing. As this is not clear this item is not yet listed under PRO nor CON.

#### 4.6.3. Summary

The DHCPv6-based solution would be a good solution in a sense it hooks into general IP configuration phase, allows easy updates when configuration information changes and does not involve DNS in general. Use of DHCPv6 requires configuration changes on DHCPv6 clients and servers and in some cases may also require implementation changes. Furthermore, it is not obvious that all devices that need translation services would implement stateless DHCPv6. For example, cellular 3GPP networks do not mandate hosts or network to implement or deploy DHCPv6.

### 4.7. Learning the IPv6 Prefix of a Network's NAT64 using Router Advertisements

#### 4.7.1. Solution description

Section 3.3 of [RA-Learn-Prefix] describes a new Router Advertisement (RA) [RFC4861] option (OPTION_AFT_PREFIX_RA) that contains the IPv6 unicast prefix, IPv6 ASM prefix, and IPv6 SSM prefix (and their lengths) for the NAT64. The RA option is essentially the same as for DHCPv6 discussed in Section 4.6.

#### 4.7.2. Analysis and discussion

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1, Issue #2, and Issue #3.

+ Can solve Issue #4 if lifetime information can be communicated.

The CONs of the proposal are listed below:

- Requires configuration and managements of all access routers to emit correct information in RA. This could, for example, be accomplished somehow by piggybacking on top of routing protocols (which would then require enhancements to routing protocols)

- In some operating systems it may not be trivial to transfer information obtained in RA to upper layers

- Requires changes to host operating system’s IP stack

- NSP change requires changes to access router configuration

- Requires standardization of a new option to Router Advertisement that is generally unfavored approach

- The RA-based solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

If RA would include multiple NSPs issue #5 could be solved as well, but only if nodes as a group would select different NSPs hence supporting load-balancing. As this is not clear this item is not yet listed under PRO nor CON.

#### 4.7.3. Summary

The RA-based solution would be a good solution in a sense it hooks into general IP configuration phase, allows easy updates when configuration information changes and does not involve DNS in general. However, generally introducing any changes to the Neighbor Discovery Protocol that are not absolutely necessary are unfavored due the impact on both network node side and end host IP stack implementations.

Compared to the DHCPv6 equivalent solution in Section 4.6 the management overhead is greater with RA-based solution. In case of DHCPv6-based solution the management can be centralized to few DHCPv6 servers compared to RA-based solution where each access router is supposed to be configured with the same information.

### 4.8. Using application layer protocols such as STUN

#### 4.8.1. Solution description

Application layer protocols, such as Session Traversal Utilities for NAT (STUN) [RFC5389], which define methods for endpoints to learn their external IP addresses could be used for NAT64 and NSP discovery. This document focuses on STUN, but the protocol could be something else as well.

A host must first use DNS to discover IPv6 representation(s) of STUN server(s) IPv4 address(es), because the host has no way to directly use IPv4 addresses to contact to STUN server(s).

After learning the IPv6 address of a STUN server the STUN client sends a request to the STUN server containing new ’SENDING-TO’ attribute that tells to the server the IPv6 address the client sent the request to. In a reply the server includes another new attribute called ’RECEIVED-AS’, which contains server’s IP address the request arrived on. After receiving the reply the client compares ’SENDING-TO’ and ’RECEIVED-AS’ attributes to find out an NSP candidate.

#### 4.8.2. Analysis and discussion

This solution is relatively similar as described in section 4.3, but instead of using DNS uses STUN to get input for heuristic algorithms.

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1 and Issue #2.

+ Does not require host changes or supportive protocols such as DNS or DHCPv6. All required logic and heuristics can be implemented in applications that are interested in learning about address synthesis taking place.

+ The solution is backward compatible from ’legacy’ hosts and servers point of view.

+ Hosts or applications interested in learning about synthesis and the used NSP can do the "discovery" proactively at any time, for example every time the host attaches to a new network.

+ Does not require explicit support from the network using NAT64.

+ Can possibly be bundled to existing STUN message exchanges as new attributes and hence client can learn its external IPv4 address and a NSP/WKP with the same exchange

+ Can be used to confirm the heuristics by synthesizing IPv6 address of another STUN server or by synthesizing IPv6 address of first STUN server after host has heuristically determined NSP using method from section 4.3. I.e. the connectivity test could be done with STUN.

+ True IPv4 destination address is used in NSP determination instead of IPv4 address received from DNS. This may increase reliability.

+ The same STUN improvement could also be used to reveal NAT66 on the data path, if the 'RECEIVED-AS' would contain different IPv6 address than 'SENDING-TO'.

The CONs of the proposal are listed below:

- Requires a server on the network to respond the queries.

- Requires standardization if done as extension to STUN.

- The solution involves changes and management on network side nodes that are not really part of the NAT64/DNS64 deployment (or issues caused by their existence).

- Does not solve issue #3 if STUN server's synthetic IPv6 address is provisioned via DNS.

- Does not solve issue #4 as the STUN server would not be aware of learned NSP's validity time.

- Does not solve issue #5 as the STUN server would not be aware of multiple NSP prefixes.

- Heavyweight solution especially if an application does not otherwise support STUN.

#### 4.8.3. Summary

The STUN, or similar, protocol based approach is a second way to solve the problem without explicit access network support. The heuristics for NSP discovery would still be in the client, however, the result may be more reliable as actual IPv4 destination address is compared to IPv6 address used in sending. The additional benefit of STUN is that the client learns its public IPv4 address with the same message exchange. STUN could also be used as the connectivity test tool if the client would first heuristically determine NSP out of DNS as described in section 4.3, synthesize IPv6 representation of STUN server's IPv4 address, and then tests connectivity to the STUN server.

As an additional benefit the STUN improvement could be used for NAT66 discovery.

### 4.9. Learning the IPv6 Prefix of a Network's NAT64 using Access Technology Specific Methods

#### 4.9.1. Solution description

Several link layers on different access systems have an attachment time signaling protocols to negotiate various parameter used later on the established link layer connection. Examples of such include 3GPP Non-Access-Stratum (NAS) signaling protocol [3GPP.24.301] among other link layers and tunneling solutions. There, using NAS signaling it could be possible to list all NSPs with their respective prefix lengths in generic protocol configuration option containers during the network access establishment. The lack of NSPs in protocol configuration option containers would be an implicit indication that there is no NAT64 present in the network.

#### 4.9.2. Analysis and discussion

The PROs of the proposal are listed below:

+ Can be used to solve Issue #1, Issue #2, Issue #3 and Issue #5.

+ Can solve Issue #4 if lifetime information is also communicated.

The CONs of the proposal are listed below:

- Requires configuration and managements of all access routers/ gateway to emit correct information in "link/lower layer" signaling. In a case the NAT64 functionality is implemented into the access router/gateway itself that terminates the generic protocol configuration exchange, then the configuration management can be automated.

- In some operating systems it may not be trivial to transfer information obtained in "link/lower layers" to upper layers.

- NSP change may require changes to access router/gateway configuration.

- Requires standardization of a new configuration parameter exchange/container for each access system of interest. The proposed solution is indeed specific to each access technology.

#### 4.9.3. Summary

The Access Technology-based solution would be a good solution in a sense it hooks into general network access establishment phase, allows easy updates when configuration information changes and does not involve DNS in general. However, generally introducing any changes to the link/lower layers is a long and slow router, and yet is access technology/system specific.

Compared to the RA equivalent solution in Section 4.7 the management overhead is equivalent or even less than RA-based solution.

## 5. Conclusion

Our conclusion is to recommend publishing the Well-Known DNS Name heuristic-based method (see Section 4.3) as an Informational IETF document for applications and host implementors to implement as-is. If Standards Track work is seen beneficial, then our recommendation is the standardization of ENDS0 option. The reasoning for our conclusion is discussed in the following paragraphs.

Of the different issues we give most weight for issues #1 and #2. We are not giving much weight for the Issue #3 'DNS should not be required', as cases where hosts need to synthesize IPv6 addresses but do does not have DNS available seem rare for us. Even if application does not otherwise utilize DNS, it ought to be able to trigger simple DNS query to find out WKP/NSP. Issue #4 is handled by majority of solutions. Issue #5 is considered to be mostly insignificant from an individual hosts point of view as it would use only one NSP at a time, while different hosts could be using different NSPs, hence supporting load-balancing targets. None of the discussed solutions support learning of possible new or indicating support for multiple algorithms for address synthesis other than the one described in [RFC6052].

The DNS64 entity has to be configured with WKP/NSP in order for it to do synthetization and hence using DNS also for delivering the synthetization information sounds logical. The fact that the synthetization information fate-shares the information received in the DNS response is a valuable attribute and reduces the possible distribution of stale prefix information. On the contrary, use of DHCPv6 would require additional trouble configuring DHCPv6 servers and ensuring DHCPv6 clients are in place, and furthermore that the NAT64, DHCPv6 (and possible even some DNS64) servers are all in sync. RA-based mechanisms are operationally expensive as configuration would have to be placed and maintained in the access routers. Furthermore, both DHCPv6 and RA based mechanisms involve entities that do not otherwise need to be aware of protocol translation (only need to know DNS server addresses).

Of the DNS-based mechanisms we favor EDNS0 option due to its lightweight nature. All the A64, DNS SRV, and ENDS0 approaches would require standardization and deployment efforts that may be excessive compared to the size of the problem. The U-NAPTR-based approach would require provisioning information into the '.ip6.arpa' tree which would not be entirely internal for the provider.

The two heuristic-based approaches could be taken into use at once and would provide benefits in networks utilizing protocol translation, but on the long run their usefulness depends how well networks will deploy explicit methods.

## 6. Security Considerations

The security considerations are essentially similar to what is described in DNS64 [I-D.ietf-behave-dns64]. Forgery of information required for IPv6 address synthesis may allow an attacker to insert itself as middle man or to perform denial-of-service attack. The DHCPv6 and RA based approaches are vulnerable for the forgery as the attacker may send forged RAs or act as a rogue DHCPv6 server (unless DHCPv6 authentication or SEND are used). If the attacker is already able to modify and forge DNS responses (flags, addresses of know IPv4-only servers, records, etc), ability to influence local address synthesis is likely of low additional value. Also, a DNS-based mechanism is only as secure as the method used to configure the DNS server's IP addresses on the host. Therefore, if the host cannot trust e.g. DHCPv6 it cannot trust the DNS server learned via DHCPv6 either, unless the host has a way to authenticate all DNS responses.

## 7. IANA Considerations

This document is informative and has no actions to IANA.

## 8. Contributors

The following people have contributed text to this document.

Mohamed Boucadair
France Telecom
Rennes, 35000
France

Email: mohamed.boucadair@orange-ftgroup.com

## 9. Acknowledgements

Authors would like to thank Dan Wing and Christian Huitema especially for the STUN idea for their valuable comments and discussions.

## 10. Informative References

[3GPP.24.301]
3GPP, "Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS)", 3GPP TS 24.301 8.8.0, December 2010.

[EDNS0-Flag]
Korhonen, J. and T. Savolainen, "EDNS0 Flags Indicating AAAA Record Synthesis and Format", draft-korhonen-edns0-synthesis-flag-00 (work in progress), June 2010.

[I-D.boucadair-behave-dns-a64]
Boucadair, M. and E. Burgey, "A64: DNS Resource Record for IPv4-Embedded IPv6 Address", draft-boucadair-behave-dns-a64-02 (work in progress), September 2010.

[I-D.boucadair-dhcpv6-shared-address-option]
Boucadair, M., Levis, P., Grimault, J., Savolainen, T., and G. Bajko, "Dynamic Host Configuration Protocol (DHCPv6) Options for Shared IP Addresses Solutions", draft-boucadair-dhcpv6-shared-address-option-01 (work in progress), December 2009.

[I-D.carpenter-referral-ps]
Carpenter, B., Jiang, S., and B. Zhou, "Problem Statement for Referral", draft-carpenter-referral-ps-01 (work in progress), August 2010.

[I-D.ietf-behave-dns64]
Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-dns64-11 (work in progress), October 2010.

[I-D.ietf-behave-v6v4-framework]
Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", draft-ietf-behave-v6v4-framework-10 (work in progress), August 2010.

[I-D.ietf-behave-v6v4-xlate-stateful]
Bagnulo, M., Matthews, P., and I. Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-v6v4-xlate-stateful-12 (work in

progress), July 2010.

[I-D.korhonen-edns0-synthesis-flag]
Korhonen, J. and T. Savolainen, "EDNS0 Option for Indicating AAAA Record Synthesis and Format", draft-korhonen-edns0-synthesis-flag-02 (work in progress), February 2011.

[I-D.savolainen-heuristic-nat64-discovery]
Savolainen, T. and J. Korhonen, "Discovery of a Network-Specific NAT64 Prefix using a Well-Known Name", draft-savolainen-heuristic-nat64-discovery-01 (work in progress), February 2011.

[I-D.venaas-behave-mcast46]
Venaas, S., Asaeda, H., SUZUKI, S., and T. Fujisaki, "An IPv4 - IPv6 multicast translator", draft-venaas-behave-mcast46-02 (work in progress), December 2010.

[I-D.venaas-behave-v4v6mc-framework]
Venaas, S., Li, X., and C. Bao, "Framework for IPv4/IPv6 Multicast Translation", draft-venaas-behave-v4v6mc-framework-02 (work in progress), December 2010.

[I-D.wing-behave-learn-prefix]
Wing, D., "Learning the IPv6 Prefix of a Network's IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-04 (work in progress), October 2009.

[RA-Learn-Prefix]
Wing, D., Wang, X., and X. Xu, "Learning the IPv6 Prefixes of an IPv6/IPv4 Translator", draft-wing-behave-learn-prefix-03 (work in progress), July 2009.

[RFC2326] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[RFC2671] Vixie, P., "Extension Mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5507] IAB, Faltstrom, P., Austein, R., and P. Koch, "Design Choices When Expanding the DNS", RFC 5507, April 2009.

[RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, October 2010.

## Authors' Addresses

Jouni Korhonen (editor)
Nokia Siemens Networks
Linnoitustie 6
FI-02600 Espoo
Finland

Email: jouni.nospam@gmail.com

Teemu Savolainen (editor)
Nokia
Hermiankatu 12 D
FI-33720 Tampere
Finland

Email: teemu.savolainen@nokia.com