Internet Engineering Task Force (IETF)
Request for Comments: 7051
Category: Informational
ISSN: 2070-1721

J. Korhonen, Ed.
Broadcom
T. Savolainen, Ed.
Nokia

November 2013

# Analysis of Solution Proposals for Hosts to Learn NAT64 Prefix

## Abstract

Hosts and applications may benefit from learning if an IPv6 address is synthesized and if NAT64 and DNS64 are present in a network. This document analyzes all proposed solutions (known at the time of writing) for communicating whether the synthesis is taking place, what address format was used, and what IPv6 prefix was used by the NAT64 and DNS64. These solutions enable both NAT64 avoidance and local IPv6 address synthesis. The document concludes by recommending the standardization of the approach based on heuristic discovery.

## Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7051.

Table of Contents

## 1. Introduction

Hosts and applications may benefit from learning if an IPv6 address is synthesized, which would mean that 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 [RFC6146] and the DNS64 [RFC6147] devices.

The first issue is finding out whether a particular address is synthetic and therefore learning the presence of a NAT64. For example, a dual-stack 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.

The second issue is 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 analyzes all proposed solutions 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

Address Synthesis

Address synthesis is 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 an 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].

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 in the IPv6 namespace. DNS64 uses NSP and/or WKP in the synthesis process.

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 that hosts sent to addresses in the NSP and/or WKP are routed to NAT64.

NSP

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

## 3. Issues

This document analyzes different solutions with a focus on the following five issues:

Issue #1

The problem of distinguishing between synthesized and 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 the 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 a 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 the 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 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.

## 4. 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 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") in the IPv4/IPv6 translation framework document [RFC6144]. 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 a remote IPv4 network, and a host in IPv6 Internet can benefit very little from learning the 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 [RFC6147], 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 the 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 an 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 translation [MCAST-TRANSLATOR] [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 an IPv6 address out of the literal in the URI and use IPv6 to create a connection through NAT64.

- Updating the host's [RFC6724] 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 the Real Time Streaming Protocol (RTSP) [RFC2326] and the 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 [REFERRAL-PS]. Synthesizing IPv6 addresses does not necessarily make the situation any better as the synthesized addresses utilizing NSP 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 to firewalls). A node 'A' behind NAT64 may detect it is talking to a node 'B' through NAT64, in which case the node 'A' may want to avoid passing its IPv6 address as a referral to the node 'B'. The node 'B' on the IPv4 side of the NAT64 should not see the IPv6 address of a node 'A' from the IPv6 side of NAT64, and hence the node 'B' should not be able to pass IPv6 address referral to a node 'C'. Passing IPv4 presentation of the IPv6 address of the host 'A' to the node 'C' is bound to similar problems as passing a public IPv4 address of a host behind NAT44 as a referral. This analysis focuses on detecting NAT64 presence from the IPv6 side of NAT64.

## 5. Proposed Solutions to Learn about Synthesis and Network-Specific Prefix

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

#### 5.1.1. Solution Description

Section 3 of [RFC7050] 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 a AAAA record of a Well-Known IPv4-only Fully Qualified Domain Name (FQDN). If a host receives a negative reply, it knows that no DNS64 and NAT64 are in the network.

If a host receives a 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.

#### 5.1.2. Analysis and Discussion

The PROs of the proposal are listed below:

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

+ Solves Issue #4 via the lifetime of the DNS record.

+ 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 the point of view of 'legacy' hosts and servers.

+ 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 a 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).

#### 5.1.3. 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.

### 5.2. EDNS0 Option Indicating AAAA Record Synthesis and Format

#### 5.2.1. Solution Description

[SYNTH-FLAG-2011] defined a new Extension Mechanisms for DNS (EDNS0) option [RFC2671] that contained 3 flag bits (called SY-bits). The EDNS0 option served as an implicit indication of the presence of a DNS64 server and NAT64 device. The EDNS0 option SY-bit values other than '000' and '111' explicitly told the NSP prefix length. Only the DNS64 server could insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA resource record.

### 5.2.2. 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 the lifetime of the DNS record.

+ 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 the point of view of 'legacy' hosts and servers.

+ 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 is needed.

+ 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 EDNS0 extension mechanisms [RFC6891].

- Requires host resolver changes and 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 a solution for Issue #3.

- EDNS0 flags and options are typically hop-by-hop only, severely limiting the applicability of these approaches, unless the EDNS0-capable DNS64 is the first DNS server the end host talks to, as it

is otherwise not possible to guarantee that the EDNS0 option survives through all DNS proxies and servers in between.

#### 5.2.3. Summary

The solution based on the EDNS0 option 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.

### 5.3. EDNS0 Flags Indicating AAAA Record Synthesis and Format

#### 5.3.1. Solution Description

[SYNTH-FLAG-2010] defined 3 new flag bits (called SY-bits) in the EDNS0 OPT [RFC2671] header that served as an implicit indication of the presence of a DNS64 server and NAT64 device. SY-bit values other than '000' or '111' explicitly told the NSP prefix length. Only the DNS64 server could insert the EDNS0 option and the required SY-bits combination into the synthesized AAAA resource record.

#### 5.3.2. 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 the lifetime of the DNS record.

+ 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 the point of view of 'legacy' hosts and servers.

+ 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 EDNS0 extension mechanisms [RFC6891].

- Consumes scarce flag bits from the EDNS0 OPT header.

- Requires a host resolver changes and 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 a solution for Issue #3.

- EDNS0 flags and options are typically hop-by-hop only, severely limiting the applicability of these approaches, unless the EDNS0-capable DNS64 is the first DNS server the end host talks to, as it is otherwise not possible to guarantee that the EDNS0 option survives through all DNS proxies and servers in between.

#### 5.3.3. 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.

### 5.4. DNS Resource Record for IPv4-Embedded IPv6 Address

#### 5.4.1. Solution Description

[DNS-A64] proposed a new DNS resource record (A64) that would be a record dedicated to storing a single IPv4-embedded IPv6 address [RFC6052]. Use of a dedicated resource record would allow a host to distinguish between real IPv6 addresses and synthesized IPv6 addresses. The solution requires the host to send a query for an A64 record. A positive answer with an 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.

### 5.4.2. Analysis and Discussion

The PROs of the proposal are listed below:

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

+ Solves Issue #4 via the lifetime of the DNS record.

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

+ Synthesized addresses can be used in authoritative DNS servers.

+ Maintains the reliability of the DNS model (i.e., a synthesized IPv6 address is presented as such and not as a 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 Issues #2 or #3 in any way.

- Requires a host resolver changes and 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 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 on 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.

### 5.4.3. Summary

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

## 5.5. Learning the IPv6 Prefix of a Network's NAT64 Using DNS

### 5.5.1. Solution Description

[LEARN-PREFIX] proposed two DNS-based methods for discovering the presence of a DNS64 server and a NAT64 device. It also proposed a mechanism for discovering the used NSP.

First, the document proposed that a host may learn the presence of a DNS64 server and a NAT64 device by receiving a TXT resource record with a well-known string (which the document proposes to be reserved by IANA) 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, the document proposed specifying a new URI-Enabled NAPTR (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 the host finds the provisioned domain name with the correct prefix length. The response to a successful U-NAPTR query contains the unicast IPv6 address and the prefix length of the NAT64 device.

### 5.5.2. Analysis and Discussion

The PROs of the proposal are listed below:

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

+ Solves Issue #4 via the lifetime of the DNS record.

+ Does not require host stack or resolver changes if the required logic and heuristics are 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 Name by IANA for the TXT resource record and/or standardization of a new U-NAPTR application.

- Requires a host resolver changes and 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 the PROs list.

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

- The U-NAPTR prefix-learning method requires provisioning of NSPs in the ".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 a solution for Issue #3.

Note: If the TXT record includes 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 PROs or CONs.

#### 5.5.3. Summary

The implementation of this solution requires some changes to the applications and resolvers in a similar fashion as in solutions in Sections 5.2, 5.3, and 5.4. Unlike the other DNS-based approaches, the U-NAPTR-based solution also requires provisioning information into the ".ip6.arpa." tree, which is no longer entirely internal to the provider hosting the NAT64/DNS64 service.

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

### 5.6. Learning the IPv6 Prefix of a Network's NAT64 Using DHCPv6

#### 5.6.1. Solution Description

Two individual IETF documents specified DHCPv6-based approaches.

[LEARN-PREFIX] described a new DHCPv6 [RFC3315] option (OPTION_AFT_PREFIX_DHCP) that would contain the IPv6 unicast prefix, IPv6 Any-Source Multicast (ASM) prefix, and IPv6 Source-Specific Multicast (SSM) prefix (and their lengths) for the NAT64.

[DHCPV6-SHARED-ADDRESS] proposed a DHCPv6 option that could be used to communicate to a requesting host the prefix used for building IPv4-converted IPv6 addresses together with the format type and

therefore also the used address synthesis algorithm. Provisioning the format type is required so as to be correctly handled by the NAT64-enabled devices deployed in a given domain.

### 5.6.2. Analysis and Discussion

The PROs of the proposal are listed below:

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

+ Does not involve the 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 the 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 aware of issues caused by NAT64/DNS64.

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

Note: 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 PROs or CONs.

### 5.6.3. Summary

The DHCPv6-based solution would be a good solution as it hooks into the 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 Third Generation Partnership Project (3GPP) networks do not mandate hosts or networks to implement or deploy DHCPv6.

### 5.7. Learning the IPv6 Prefix of a Network's NAT64 Using Router Advertisements

#### 5.7.1. Solution Description

Revision three of [LEARN-PREFIX] described a new Router Advertisement (RA) [RFC4861] option (OPTION_AFT_PREFIX_RA) that would contain 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 5.6.

#### 5.7.2. Analysis and Discussion

The PROs of the proposal are listed below:

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

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

The CONs of the proposal are listed below:

- Requires configuration and management of all access routers to emit correct information in the 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 the RA to upper layers.

- Requires changes to the host operating system's IP stack.

- An NSP change requires changes to the access router configuration.

- Requires standardization of a new option to the Router Advertisement, which is generally an 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 aware of issues caused by NAT64/DNS64.

Note: If the 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 PROs or CONs.

#### 5.7.3. Summary

The RA-based solution would be a good solution as it hooks into the 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 to the impact on both the network-side node and end host IP stack implementations.

Compared to the DHCPv6 equivalent solution in Section 5.6, the management overhead is greater with the RA-based solution. With the DHCPv6-based solution, the management can be centralized to a few DHCPv6 servers compared to the RA-based solution where each access router is supposed to be configured with the same information.

### 5.8. Using Application-Layer Protocols such as STUN

#### 5.8.1. Solution Description

Application-layer protocols, such as Session Traversal Utilities for NAT (STUN) [RFC5389], that 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 representations of STUN servers' IPv4 addresses, because the host has no way to directly use IPv4 addresses to contact STUN servers.

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

#### 5.8.2. Analysis and Discussion

This solution is relatively similar to the one described in Section 5.1, but instead of using DNS, it uses STUN to get input for heuristic algorithms.

The PROs of the proposal are listed below:

+ Can be used to solve Issues #1 and #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 the point of view of 'legacy' hosts and servers.

+ 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, a client can learn its external IPv4 address and an NSP/WKP with the same exchange.

+ Can be used to confirm the heuristics by synthesizing the IPv6 address of another STUN server or by synthesizing the IPv6 address of first STUN server after the host has heuristically determined NSP using the method in Section 5.1, i.e., the connectivity test could be done with STUN.

+ The true IPv4 destination address is used in NSP determination instead of the 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 a different IPv6 address from 'SENDING-TO'.

The CONs of the proposal are listed below:

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

- Requires standardization if done as an extension to STUN.

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

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

- Does not solve Issue #4 as the STUN server would not be aware of the 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.

#### 5.8.3. Summary

An approach based on STUN or a similar protocol 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 an actual IPv4 destination address is compared to the 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 5.1, synthesize the IPv6 representation of the STUN server's IPv4 address, and then test connectivity to the STUN server.

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

### 5.9. Learning the IPv6 Prefix of a Network's NAT64 Using Access-Technology-Specific Methods

#### 5.9.1. Solution Description

Several link layers on different access systems have attachment time signaling protocols for negotiating various parameters that are used later on with the established link-layer connection. Examples of such include the 3GPP Non-Access-Stratum (NAS) signaling protocol [NAS.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.

#### 5.9.2. Analysis and Discussion

The PROs of the proposal are listed below:

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

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

The CONs of the proposal are listed below:

- Requires configuration and management of all access routers/ gateways to emit correct information in "link/lower-layer" signaling. If NAT64 functionality is implemented into the access router/gateway 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.

- An NSP change may require changes to the 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.

#### 5.9.3. Summary

The solution based on access technology would be a good solution as 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 process, and changes would need to be done for all access technologies/systems that are used with NAT64.

Compared to the RA-equivalent solution in Section 5.7, the management overhead is equivalent or even less than the RA-based solution.

## 6. Conclusion

Our conclusion is to recommend publishing the Well-Known DNS Name heuristic discovery-based method as a Standards Track IETF document for applications and host implementors to implement as-is.

As a general principle, we prefer to have as minimal a solution as possible, avoid impacts to entities not otherwise involved in the protocol translation scheme, minimize host impact, and require minimal to no operational effort on the network side.

Of the different issues, we give the most weight to Issues #1 and #2. We do not give much weight to Issue #3, as cases where hosts need to synthesize IPv6 addresses but do not have DNS available seem rare to us. Even if an application does not otherwise utilize DNS, it ought to be able to trigger a simple DNS query to find out WKP/NSP. Issue #4 is handled by the majority of solutions, and Issue #5 is

considered to be mostly insignificant as even if individual hosts would use only one NSP at a time, different hosts would be using different NSPs, hence supporting load-balancing targets. Only one of the discussed solutions, see Section 5.6, supports 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 synthesis; hence, using DNS also for delivering the synthesis information sounds logical. The fact that the synthesis information fate-shares the information received in the DNS response is a valuable attribute and reduces the possible distribution of stale prefix information. However, having all DNS64 servers support explicit WKP/NSP discovery (ENDS0, A64, and DNS SRV record approaches) is difficult to arrange. The U-NAPTR-based approach would require provisioning information into the ".ip6.arpa." tree, which would not be entirely internal for the provider. Use of DHCPv6 would involve additional trouble configuring DHCPv6 servers and ensuring DHCPv6 clients are in place; it would also involve ensuring that the NAT64 and DHCPv6 (and possibly 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 (they only need to know DNS server addresses). Finally, regarding the use of STUN, a host does not need to implement STUN whereas DNS is, in practice, required anyway. Also, the STUN protocol would need to be changed on both the host and network side to support the discovery of NAT64 and WKP/NSP.

## 7. Security Considerations

The security considerations are essentially similar to those described in DNS64 [RFC6147]. The document also talks about man-in-the-middle and denial-of-service attacks caused by forging of information required for IPv6 synthesis from corresponding IPv4 addresses. Forgery of information required for IPv6 address synthesis may allow an attacker to insert itself as a middle man or to perform a denial-of-service attack. The DHCPv6 and RA-based approaches are vulnerable to forgery as the attacker may send forged RAs or act as a rogue DHCPv6 server (unless DHCPv6 authentication [RFC3315] or Secure Neighbor Discovery (SEND) [RFC3971] are used). If the attacker is already able to modify and forge DNS responses (flags, addresses of known 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, for example, the host cannot trust DHCPv6, it cannot

trust the DNS server learned via DHCPv6 either, unless the host has a way to authenticate all DNS responses (e.g., via DNSSEC [RFC4033]).

## 8. Contributors

The following individual contributed text to this document.

Mohamed Boucadair
France Telecom
Rennes, 35000
France

EMail: mohamed.boucadair@orange-ftgroup.com

## 9. Acknowledgements

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

Jouni Korhonen would like to specifically thank Nokia Siemens Networks as he completed the majority of this document while employed there.

## 10. References

### 10.1. Normative References

[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.

[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.

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

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, November 2013.

## 10.2. Informative References

[DHCPV6-SHARED-ADDRESS]
Boucadair, M., Levis, P., Grimault, J., Savolainen, T., and G. Bajko, "Dynamic Host Configuration Protocol (DHCPv6) Options for Shared IP Addresses Solutions", Work in Progress, December 2009.

[DNS-A64] Boucadair, M. and E. Burgey, "A64: DNS Resource Record for IPv4-Embedded IPv6 Address", Work in Progress, September 2010.

[LEARN-PREFIX]
Wing, D., "Learning the IPv6 Prefix of a Network's IPv6/IPv4 Translator", Work in Progress, October 2009.

[MCAST-TRANSLATOR]
Venaas, S., Asaeda, H., SUZUKI, S., and T. Fujisaki, "An IPv4 - IPv6 multicast translator", Work in Progress, December 2010.

[NAS.24.301]
3GPP, "Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS)", 3GPP TS 24.301 8.8.0, December 2010, <http://www.3gpp.org/ftp/Specs/html-info/24301.htm>.

[REFERRAL-PS]
Carpenter, B., Jiang, S., and Z. Cao, "Problem Statement for Referral", Work in Progress, February 2011.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

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

[RFC6144] Baker, F., Li, X., Bao, C., and K. Yin, "Framework for IPv4/IPv6 Translation", RFC 6144, April 2011.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, April 2013.

[SYNTH-FLAG-2010]
Korhonen, J. and T. Savolainen, "EDNS0 Option for Indicating AAAA Record Synthesis and Format", Work in Progress, July 2010.

[SYNTH-FLAG-2011]
Korhonen, J. and T. Savolainen, "EDNS0 Option for Indicating AAAA Record Synthesis and Format", Work in Progress, February 2011.

[V4V6MC-FRAMEWORK]
Venaas, S., Li, X., and C. Bao, "Framework for IPv4/IPv6 Multicast Translation", Work in Progress, June 2011.

Authors' Addresses

Jouni Korhonen (editor)
Broadcom
Porkkalankatu 24
FIN-00180 Helsinki
Finland

EMail: jouni.nospam@gmail.com

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

EMail: teemu.savolainen@nokia.com