Behave WG
Internet-Draft
Intended status: Standards Track
Expires: December 31, 2012

T. Savolainen
Nokia
J. Korhonen
Nokia Siemens Networks
D. Wing
Cisco Systems
June 29, 2012

Discovery of IPv6 Prefix Used for IPv6 Address Synthesis
draft-ietf-behave-nat64-discovery-heuristic-10.txt

Abstract

This document describes a method for detecting the presence of DNS64 and for learning the IPv6 prefix used for protocol translation on an access network. The method depends on the existence of a well-known IPv4-only domain name "ipv4only.arpa". The information learned enables nodes to perform local IPv6 address synthesis and to potentially avoid NAT64 on dual-stack and multi-interface deployments.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

As part of the transition to IPv6, NAT64 [RFC6146] and DNS64 [RFC6147] technologies will be utilized by some access networks to provide IPv4 connectivity for IPv6-only nodes [RFC6144]. DNS64 utilizes IPv6 address synthesis to create local IPv6 addresses for peers having only IPv4 addresses, hence allowing DNS-using IPv6-only nodes to communicate with IPv4-only peers.

However, DNS64 cannot serve applications not using DNS, such as those receiving IPv4 address literals as referrals. Such applications could nevertheless be able to work through NAT64, provided they are able to create locally valid IPv6 addresses that would be translated to the peers' IPv4 addresses.

Additionally, DNS64 is not able to do IPv6 address synthesis for nodes running validating DNSSEC-enabled DNS resolvers, but instead the synthesis must be done by the nodes themselves. In order to perform IPv6 synthesis, nodes have to learn the IPv6 prefix(es) used on the access network for protocol translation. The prefixes, which may be Network Specific Prefixes (NSP) or Well-Known Prefixes (WKP) [RFC6052], are referred in this document as Pref64::/n [RFC6146].

This document describes a best-effort method for applications and nodes to learn the information required to perform local IPv6 address synthesis. The IPv6 address synthesis procedure itself is out-of-scope of this document. An example application is a browser encountering IPv4 address literals in an IPv6-only access network. Another example is a node running a validating security-aware DNS resolver in an IPv6-only access network.

The knowledge of IPv6 address synthesis taking place may also be useful if DNS64 and NAT64 are used in dual-stack enabled access networks or if a node is multi-interfaced [RFC6418]. In such cases, nodes may choose to prefer IPv4 or an alternative network interface in order to avoid traversal through protocol translators.

It is important to note that use of this approach will not result in a system that is as robust, secure, and well-behaved as an all-IPv6 system would be. Hence it is highly recommended to upgrade nodes' destinations to IPv6 and utilize the described method only as a transition solution.

2. Requirements and Terminology

2.1. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Terminology

NAT64 FQDN: a fully qualified domain name for a NAT64 protocol translator.

Pref64::/n: the IPv6 prefix used for IPv6 address synthesis [RFC6146].

Pref64::WKA/n: the IPv6 address consisting of Pref64::/n and WKA at any of the locations allowed by RFC 6052 [RFC6052].

Well-Known IPv4-only Name (WKN): a fully qualified domain name, "ipv4only.arpa", well-known to have only A record(s).

Well-Known IPv4 Address (WKA): an IPv4 address that is well-known and present in an A record for the well-known name. Two well-known IPv4 addresses are defined for the Pref64::/n discovery purposes: 192.0.0.170 and 192.0.0.171.

3. Node Behavior

A node requiring information about the presence (or absence) of NAT64, and the Pref64::/n used for protocol translation SHALL send a DNS query for AAAA resource records of the Well-Known IPv4-only Name (WKN) "ipv4only.arpa". The node MAY perform the DNS query in both IPv6-only and dual-stack access networks.

When sending a DNS AAAA resource record query for the WKN, a node MUST set the "Checking Disabled (CD)" bit to zero [RFC4035], as otherwise the DNS64 server will not perform IPv6 address synthesis (Section 3 of [RFC6147]) and hence would not reveal the Pref64::/n used for protocol translation.

A DNS reply with one or more AAAA resource records indicates that the access network is utilizing IPv6 address synthesis. In some scenarios NXDOMAIN and NXRRSET hijacking performed by the access network may result in a false positive. One method to detect such hijacking is to query a Fully Qualified Domain Name (FQDN) that is known to be invalid (and normally return an empty response or an error response) and see if it returns a valid resource record. However, as long as the hijacked domain does not result in AAAA

resource record responses that contain well-known IPv4 address in any location defined by RFC6052, the response will not disturb Pref64::/n learning procedure.

A node MUST look through all of the received AAAA resource records to collect one or more Pref64::/n. The Pref64::/n list might include the Well-Known Prefix 64:ff9b::/96 [RFC6052] or one or more Network-Specific Prefixes. In the case of NSPs, the node SHALL determine the used address format by searching the received IPv6 addresses for the WKN's well-known IPv4 addresses. The node SHALL assume the well-known IPv4 addresses might be found at the locations specified by RFC6052 section 2.2. If the well-known IPv4 addresses are not found within the standard locations, it indicates that the network is not using standard address format and the Pref64::/n cannot be determined. Developers can over time learn of IPv6 translated address formats that are extensions or alternatives to the standard formats. Developers MAY at that point add additional steps to the described discovery procedure. The additional steps are outside the scope of the present document.

In case a node does not receive positive DNS reply to AAAA resource record query, the node MAY perform DNS A resource record query for the well-known name. If the node receives positive reply to the DNS A resource record query it means the used recursive DNS server is not DNS64 server.

The node MUST ensure a 32-bit well-known IPv4 address value is present only once in an IPv6 address. In case another instance of the value is found inside the IPv6 address, the node SHALL repeat the search with the other well-known IPv4 address.

If only one Pref64::/n was present in the DNS response: a node SHALL use that Pref64::/n for both local synthesis and for detecting synthesis done by the DNS64 server on the network.

If more than one Pref64::/n was present in the DNS response: a node SHOULD use all of them when determining whether other received IPv6 addresses are synthetic. The node MUST use all learned Pref64::/n when performing local IPv6 address synthesis, and use the prefixes in the order received from DNS64 server. That is, when the node is providing a list of locally synthesized IPv6 addresses to upper layers, IPv6 addresses MUST be synthesized by using all discovered Pref64::/n in the received order.

In the case of a negative response (NXDOMAIN, NXRRSET) or a DNS query timeout: a DNS64 server is not available on the access network, the access network filtered out the well-known query, or something went wrong in the DNS resolution. All unsuccessful cases result in a node

being unable to perform local IPv6 address synthesis. In the case of timeout, the node SHOULD retransmit the DNS query like any other DNS query the node makes [RFC1035]. In the case of a negative response (NXDOMAIN, NXRRSET), the node MUST obey the Time-To-Live [RFC1035] of the response before resending the AAAA resource record query. The node MAY monitor for DNS replies with IPv6 addresses constructed from the WKP, in which case if any are observed the node SHOULD use the WKP as if it were learned during the query for the well-known name.

To save Internet resources if possible, a node should perform Pref64::/n discovery only when needed (e.g., when local synthesis is required, a new network interface is connected to a new network, and so forth). The node SHALL cache the replies it receives during the Pref64::/n discovery procedure and it SHOULD repeat the discovery process ten seconds before the Time-To-Live of the Well-Known Name's synthetic AAAA resource record expires.

3.1. Validation of Discovered Pref64::/n

If a node is using an insecure channel between itself and a DNS64 server, or the DNS64 server is untrusted, it is possible for an attacker to influence the node's Pref64::/n discovery procedures. This may result in denial-of-service, redirection, man-in-the-middle, or other attacks.

To mitigate against attacks, the node SHOULD communicate with a trusted DNS64 server over a secure channel, or use DNSSEC. NAT64 operators SHOULD provide facilities for validating discovery of Pref64::/n via a secure channel and/or DNSSEC protection.

It is important to understand that DNSSEC only validates that the discovered Pref64::/n is the one that belongs to a domain used by NAT64 FQDN. Importantly, the DNSSEC validation does not tell if the node is at the network where the Pref64::/n is intended to be used. Furthermore, DNSSEC validation cannot be utilized in the case of WKP.

3.1.1. DNSSEC Requirements for the Network

If the operator has chosen to support nodes performing validation of discovered Pref64::/n with DNSSEC, the operator of the NAT64 device MUST perform the following configurations.

1. Have one or more Fully Qualified Domain Names for the NAT64 translator entities (later referred as NAT64 FQDN). In the case of more than one Pref64::/n being used in a network, e.g., for load-balancing purposes, it is for network administrators to decide whether a single NAT64's fully-qualified domain name maps to more than one Pref64::/n, or whether there will be a dedicated

NAT64 FQDN per Pref64::/n.

2. Each NAT64 FQDN MUST have one or more DNS AAAA resource records containing Pref64::WKA/n (Pref64::/n combined with WKA).

3. Each Pref64::WKA/n MUST have a PTR resource record that points to the corresponding NAT64 FQDN.

4. Sign the NAT64 FQDNs' AAAA and A resource records with DNSSEC.

3.1.2. DNSSEC Requirements for the Node

A node SHOULD prefer a secure channel to talk to a DNS64 server, whenever possible. In addition, a node that implements a DNSSEC validating resolver MAY use the following procedure to validate discovery of the Pref64::/n.

1. Heuristically find Pref64::/n candidates by making a AAAA resource record query for "ipv4only.arpa" by following the procedure in Section 3. This will result in IPv6 addresses consisting of Pref64::/n combined with WKA, i.e. in Pref64:: WKA/n. For each Pref64::/n that the node wishes to validate, the node performs the following steps.

2. Send DNS PTR resource record query for the IPv6 address of the translator (for "ip6.arpa"), using the Pref64::WKA/n learned on the step 1. CNAME and DNAME results should be followed according to the rules in RFC 1034 [RFC1034], RFC 1034 [RFC1035], and RFC 6672 [RFC6672]. The ultimate response will include one or more NAT64 FQDNs.

3. The node SHOULD compare the domains of learned NAT64 FQDNs to a list of the node's trusted domains and choose a NAT64 FQDN that matches. The means for node to learn the trusted domains is implementation-specific. If the node has no list of trusted domains, the node MAY query the user whether the domain can be trusted and MAY remember the answer for future use. If the node has no trust for the domain, the discovery procedure is not secure and the remaining steps described below MUST NOT be performed.

4. Send a DNS AAAA resource record query for the NAT64 FQDN.

5. Verify the DNS AAAA resource record contains Pref64::WKA/n addresses received at the step 1. It is possible that the NAT64 FQDN has multiple AAAA records, in which case the node MUST check if any of the addresses matches the ones obtained in step 1. The node MUST ignore other responses and not use them for local IPv6

address synthesis.

6. Perform DNSSEC validation of the DNS AAAA response.

After the node has successfully performed the above five steps, the node can consider Pref64::/n validated.

3.2. Connectivity Check

After learning a Pref64::/n, the node SHOULD perform a connectivity check to ensure the learned Pref64::/n is functional. It could be non-functional for a variety of reasons -- the discovery failed to work as expected, the IPv6 path to the NAT64 is down, the NAT64 is down, or the IPv4 path beyond the NAT64 is down.

There are two main approaches to determine if the learned Pref64::/n is functional. The first approach is to perform a dedicated connectivity check. The second approach is to simply attempt to use the learned Pref64::/n. Each approach has some trade-offs (e.g., additional network traffic or possible user-noticable delay), and implementations should carefully weigh which approach is appropriate for their application and the network.

The node SHOULD use a implementation-specific connectivity check server and a protocol of the implementation's choice, but if that is not possible, a node MAY do a PTR resource record query of the Pref64::WKA/n to get a NAT64 FQDN. The node then does an A resource query of the NAT64 FQDN, which will return zero or more A resource records pointing to connectivity check servers used by the network operator. A negative response to the PTR or A resource query means there are no connectivity check servers available. A network operator that provides NAT64 services for a mix of nodes with and without implementation-specific connectivity check servers SHOULD assist nodes in their connectivity checks by mapping each NAT64 FQDN to one or more DNS A resource records with IPv4 address(es) pointing to connectivity check server(s). The Pref64::/n -based connectivity check approach works only with NSP, as it is not possible to register A record for each different domain using WKP. The network operator MUST disable ICMPv6 rate limiting for connectivity check mesages.

In case of multiple connectivity check servers being available for use, the node chooses the first one, preferring implementation-specific servers.

The connectivity check protocol used with implementation-specific connectivity check servers is implementation-specific.

The connectivity check protocol used with connectivity check servers

pointed to by NAT64 FQDN's A resource records is ICMPv6 [RFC4443]. The node performing a connectivity check against these servers SHALL send an ICMPv6 Echo Request to an IPv6 address synthesized by combining discovered Pref64::/n with an IPv4 address of the server as specified in RFC6052. This will test the IPv6 path to the NAT64, the NAT64's operation, and the IPv4 path all the way to the connectivity check server. If no response is received for the ICMPv6 Echo Request, the node SHALL send another ICMPv6 Echo Request, a second later. If still no response is received, the node SHALL send a third ICMPv6 Echo Request two seconds later. If an ICMPv6 Echo Response is received, the node knows the IPv6 path to the connectivity check server is functioning normally. If, after the three transmissions and three seconds since the last ICMPv6 Echo Request, no response is received, the node learns this Pref64::/n might not be functioning, and the node MAY choose a different Pref64::/n (if a different are available), choose to alert the user, or proceed anyway hoping the problem is temporal or only with the connectivity check itself. After all, the ICMPv6 is by design unreliable and failure to receive ICMPv6 responses may not indicate anything else than network failure to transport ICMPv6 messages through.

If no separate connectivity check is performed before local IPv6 address synthesis, a node MAY monitor success of connection attempts performed with locally synthesized IPv6 addresses. Based on success of these connections, and based on possible ICMPv6 error messages received (such as Destination Unreachable Message), the node MAY cease to perform local address synthesis and MAY restart Pref64::/n discovery procedures.

3.2.1. No Connectivity Checks Against ipv4only.arpa

Clients MUST NOT send a connectivity check to the address returned in the ipv4only.arpa query. This is because, by design, no server will be operated on the Internet at that address as such. Similarly, network operators MUST NOT operate a server on that address. The reason this address isn't used for connectivity checks is that operators who neglect to operate a connectivity check server will allow that traffic towards the Internet where it will be dropped and cause a false negative connectivity check with the client (that is, the NAT64 is working fine, but the connectivity check fails because a server is not operating at "ipv4only.arpa" on the Internet and a server is not operated by the NAT64 operator). Instead, for the connectivity check, an additional DNS resource record is looked up and used for the connectivity check. This ensures that packets don't unnecessarily leak to the Internet and reduces the chance of a false negative connectivity check.

3.3. Alternative Domain Names

Some applications, operating systems, devices, or networks may find it advantageous to operate their own DNS infrastructure to perform a function similar to "ipv4only.arpa", but using a different resource record. The primary advantage is to ensure availability of the DNS infrastructure and ensure the proper configuration of the DNS record itself. For example, a company named Example might have their application query "ipv4only.example.com". Other than the different DNS resource record being queried, the rest of the operations are anticipated to be identical to the steps described in this document.

3.4. Message Flow Illustration

A figure below gives an example illustration of a message flow in the case of prefix discovery utilizing Pref64::/n validation. The figure shows also a step where the procedure ends if no Pref64::/n validation is performed.

In this example, three Pref64::/n are provided by the DNS64 server. The first Pref64::/n is using NSP, in this example "2001:db8:42::/96". The second Pref64::/n is using NSP, in this example "2001:db8:43::/96". The third Pref64::/n is using WKP. Hence, when the Pref64::/n are combined with WKA to form Pref64::WKA/n, the synthetic IPv6 addresses returned by DNS64 server are "2001:db8:42::192.0.0.170", "2001:db8:43::192.0.0.170", and "64:ff9b::192.0.0.170". The synthetic addresses could also contain the IPv4 address 192.0.0.171.

The validation is not done for the WKP, see Section 3.1.

```
   Node                                           DNS64 server
     |                                                 |
     |  "AAAA" query for "ipv4only.arpa"               |
     |------------------------------------------------>| "A" query for
     |                                                 | "ipv4only.arpa"
     |                                                 |---------------->
     |                                                 |
     |                                                 | "A" response:
     |                                                 | "192.0.0.170"
     |                                                 | "192.0.0.171"
     |                                                 |<----------------
     |                                +-----------------------------+
     |                                | "AAAA" synthesis using      |
     |                                | three Pref64::/n.           |
     |                                +-----------------------------+
     |  "AAAA" response with:                          |
     |  "2001:db8:42::192.0.0.170"                     |
```

```
     |  "2001:db8:43::192.0.0.170"                       |
     |  "64:ff9b::192.0.0.0170"                          |
     |<--------------------------------------------------|
     |                                                   |
  +------------------------------------------------+     |
  | If Pref64::/n validation is not performed, a |     |
  | node can fetch prefixes from AAAA responses  |     |
  | at this point and skip the steps below.      |     |
  +------------------------------------------------+     |
     |                                                   |
     |  "PTR" query #1 for "2001:db8:42::192.0.0.170     |
     |-------------------------------------------------->|
     |  "PTR" query #2 for "2001:db8:43::192.0.0.170     |
     |-------------------------------------------------->|
     |                                                   |
     |  "PTR" response #1 "nat64_1.example.com"          |
     |<--------------------------------------------------|
     |  "PTR" response #2 "nat64_2.example.com"          |
     |<--------------------------------------------------|
     |                                                   |
  +------------------------------------------------+     |
  | Compare received domains to a trusted domain |     |
  | list and if matches are found, continue.     |     |
  +------------------------------------------------+     |
     |                                                   |
     |  "AAAA" query #1 for "nat64_1.example.com"        |
     |-------------------------------------------------->|
     |  "AAAA" query #2 for "nat64_2.example.com"        |
     |-------------------------------------------------->|
     |                                                   |
     | "AAAA" resp. #1 with "2001:db8:42::192.0.0.170    |
     |<--------------------------------------------------|
     | "AAAA" resp. #2 with "2001:db8:43::192.0.0.170    |
     |<--------------------------------------------------|
     |                                                   |
  +------------------------------------------------+     |
  | Validate AAAA responses and compare the IPv6 |     |
  | addresses to those previously learned.       |     |
  +------------------------------------------------+     |
     |                                                   |
  +------------------------------------------------+     |
  | Fetch the Pref64::/n from the validated      |     |
  | responses and take into use.                 |     |
  +------------------------------------------------+     |
     |                                                   |
```

Pref64::/n discovery procedure

4. Operational Considerations for Hosting the IPv4-Only Well-Known Name

The authoritative name server for the well-known name SHALL have DNS record Time-To-Live (TTL) set to at least 60 minutes in order to improve effectiveness of DNS caching. The exact TTL value will be determined and tuned based on operational experiences.

The domain serving the well-known name MUST be signed with DNSSEC. See also section 7.

5. Operational Considerations for DNS64 Operator

A network operator of DNS64 server can guide nodes utilizing heuristic discovery procedures by managing the responses DNS64 server provides.

If the network operator would like nodes to utilize multiple Pref64::/n, the operator needs to configure DNS64 server to respond with multiple synthetic AAAA records. As per section Section 3 the nodes can then use them all.

There are no guarantees on what of the Pref64::/n nodes will end up using. If the operator wants nodes to specifically use certain Pref64::/n or periodically change the Pref64::/n they use, for example for load balancing reasons, the only guaranteed method is to make DNS64 server to return only single synthetic AAAA resource record, and have Time-To-Live of that synthetic record such that node repeats the Pref64::/n discovery when required Section 3.

Besides choosing how many Pref64::/n to respond and what Time-To-Live to use, DNS64 servers MUST NOT interfere with or perform other special procedures for the queries related to the well-known name.

5.1. Mapping of IPv4 Address Ranges to IPv6 Prefixes

The RFC 6147 [RFC6147] allows DNS64 implementations to be able to map specific IPv4 address ranges to separate Pref64::/n. That allows handling of special use IPv4 addresses [RFC5735]. The example setup where this might be used is illustrated in Figure 1. The NAT64 "A" is used when accessing IPv4-only servers in the datacenter, and the NAT64 "B" is used for Internet access.

```
                    NAT64 "A" ----- IPv4-only servers in a datacenter
                   /
IPv6-only node----<
                   \
                    NAT64 "B" ----- IPv4 Internet
```

Figure 1: NAT64s with IPv4 Address Ranges

The heuristic discovery method described herein does not support learning of the possible rules used by DNS64 server for mapping specific IPv4 address ranges to separate Pref64::/n. Therefore, nodes will use the same discovered Pref64::/n to synthesize IPv6 addresses out from any IPv4 address. This can cause issues for routing and connectivity establishment procedures. The operator of the NAT64 and the DNS64 ought to take this into account in the network design.

The network operators can help IPv6-only nodes by ensuring the nodes do not have to work with IPv4 address literals for which special mapping rules are used. That is, the IPv4-only servers addressed from the special IPv4 address ranges ought to have signed AAAA records, which allows IPv6-only nodes to avoid local address synthesis. If the IPv6-only nodes are not using DNSSEC, then it is enough if the network's DNS64 server returns synthetic AAAA resource records pointing to IPv4-only servers. Avoiding the need for IPv6-only node to perform address synthesis for IPv4 addresses belonging to special ranges is the best approach to assist nodes.

If the IPv6-only nodes have no other choice than use IPv4-address literals belonging to special IPv4 address ranges, and the IPv6-only node will perform local synthesis by using the discovered Pref64::/n, then the network ought to ensure with routing that the packets are delivered to the correct NAT64. For example, a router in the path from IPv6-only host to NAT64s can forward the IPv6 packets to correct NAT64 as illustrated in Figure 2. The routing could be based on the last 32-bits of the IPv6 address, but the network operator can also use some other IPv6 address format allowed by RFC 6052 [RFC6052], if it simplifies routing setup. This setup requires additional logic on the NAT64 providing connectivity to special IPv4 address ranges: it needs to be able to translate packets it receives that are using the Pref64::/n used with Internet connections.

```
                    NAT64 "A" ----- IPv4-only servers in a datacenter
                   /
IPv6-only host----router
                   \
                    NAT64 "B" ----- IPv4 Internet
```

Figure 2: NAT64s with Assisting Router

6. Exit Strategy

A day will come when this tool is no longer needed. At that point best suited techniques for implementing exit strategy will be documented.

A node SHOULD implement a configuration knob for disabling the Pref64::/n discovery feature.

7. Security Considerations

The security considerations follow closely those of RFC 6147 [RFC6147]. The possible attacks are very similar in the case where attacker controls DNS64 server and returns tampered IPv6 addresses to a node and in the case where attacker causes the node to use tampered Pref64::/n for local address synthesis. The DNSSEC cannot be used to validate responses created by a DNS64 server the node has no trust relationship with. Hence this document does not change the big picture for untrusted network scenarios. If an attacker mangles with Pref64::/n used by a DNS64 server or a node, the traffic generated by the node will be delivered to an altered destination. This can result in either a denial-of-service (DoS) attack (if the resulting IPv6 addresses are not assigned to any device), a flooding attack (if the resulting IPv6 addresses are assigned to devices that do not wish to receive the traffic), or an eavesdropping attack (in case the altered NSP is routed through the attacker).

The zone serving the well-known name has to be protected with DNSSEC, as otherwise it will be too attractive a target for attackers who wish to alter nodes' Pref64::/n discovery procedures.

A node SHOULD implement a validating DNSSEC resolver for validating the A response of the well-known name query. A node without a validating DNSSEC resolver SHOULD request validation to be performed by the recursive DNS server and use a secure channel when communicating with the DNS64 server.

For Pref64::/n discovery validation, the access network SHOULD sign the NAT64 translator's fully qualified domain name. A node SHOULD use the algorithm described in Section 3.1 in order to validate discovered Pref64::/n.

Lastly, the best mitigation action against Pref64::/n discovery attacks is to add IPv6 support for nodes' destinations and hence

reduce the need to perform local IPv6 address synthesis.

8. IANA Considerations

According to procedures described in RFC3172 this document requests IANA to reserve a second level domain from the .ARPA zone for the well-known domain name. The well-known domain name could be, for example, "ipv4only.arpa".

The well-known name needs to map to two different global IPv4 addresses. The addresses are to be taken from the 192.0.0.0/24 address block, and can be, for example 192.0.0.170 and 192.0.0.171. The addresses are to be documented to be of global scope, but they do not need to be routable in local or global scopes.

9. Acknowledgements

Authors would like to thank Dmitry Anipko, Cameron Byrne, Aaron Yi Ding Christian Huitema, Washam Fan, Peter Koch, Stephan Lagerholm, Zhenqiang Li, Simon Perreault, Marc Petit-Huguenin, Andrew Sullivan, and Dave Thaler, for significant improvement ideas and comments.

10. References

10.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[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.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, June 2012.

10.2. Informative References

[RFC5735] Cotton, M. and L. Vegoda, "Special Use IPv4 Addresses", BCP 153, RFC 5735, January 2010.

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

[RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, November 2011.

Appendix A. Example of DNS Record Configuration

The following BIND-style examples illustrate how A and AAAA records could be configured by a NAT64 operator.

The examples use Pref64::/n of 2001:db8::/96 and the example.com domain.

The PTR record for reverse queries (Section 3.1.1 bullet 3):

```
$ORIGIN 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.IP6.ARPA.
@         IN      SOA   ns1.example.com. hostmaster.example.com. (
                        2003080800 12h 15m 3w 2h)
          IN      NS    ns.example.com.

          IN      PTR   nat64.example.com.
```

If example.com does not use DNSSEC, the following configuration file could be used. Please note tat nat64.example.com has both an AAAA record with the Pref64::/n and an A record for the connectivity check (Section 3.1.1 bullet 2).

```
   example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
                         2002050501 ; serial
                         100        ; refresh (1 minute 40 seconds)
                         200        ; retry (3 minutes 20 seconds)
                         604800     ; expire (1 week)
                         100        ; minimum (1 minute 40 seconds)
                         )

   example.com.  IN NS  ns.example.com.

   nat64.example.com.
                 IN AAAA  2001:db8:0:0:0:0:0:0
   nat64.example.com.
                 IN A  192.0.2.1
```

If the example.com does use DNSSEC, the following configuration file could be used for A and AAAA records:

```
   example.com.  IN SOA  ns.example.com. hostmaster.example.com. (
                         2002050501 ; serial
                         100        ; refresh (1 minute 40 seconds)
                         200        ; retry (3 minutes 20 seconds)
                         604800     ; expire (1 week)
                         100        ; minimum (1 minute 40 seconds)
                         )

   example.com.  IN RRSIG SOA  5 2 100 20090803071330 (
                         20090704071330 17000 example.com.
                         TVgWsNQvsFmeNHAeccGi7+UI7KwcE9TXPuSvmV9yyJwo
                         4FvHkxVC1H+98EtrmbR4c/XcdUzdfgn+q+lBqNsnbAit
                         xFERwPxzxbX0+yeCdHbBjHe7OuOc2Gc+CH6SbT2lKwVi
                         iEx3ySqqNoVScoUyhRdnPV2A1LV0yd9GtG9mI4w= )

   example.com.  IN NS  ns.example.com.
   example.com.  IN RRSIG NS  5 2 100 20090803071330 (
                         20090704071330 17000 example.com.
                         Xuw7saDDi6+5Z7SmtC7FC2npPOiE8F9qMR87eA0egG0I
                         B+xFx7pIogoVIDpOd1h3jqYivhblpCoDSBQb2oMbVy3B
                         SX5cF0r7Iu/xKP8XrV4DjNiugpa+NnhEIaRqG5uoPFbX
                         4cYT51yNq70I5mJvvajJu7UjmdHl26ZlnK33xps= )

   nat64.example.com.
                 IN AAAA  2001:db8:0:0:0:0:0 nat64.example.com.
                 IN RRSIG SOA  5 2 100 20090803071330 (
                         20090704071330 17000 example.net.
                         TVgWsNQvsFmeNHAeccGi7+UI7KwcE9TXPuSvmV9yyJwo
                         4FvHkxVC1H+98EtrmbR4c/XcdUzdfgn+q+lBqNsnbAit
                         xFERwPxzxbX0+yeCdHbBjHe7OuOc2Gc+CH6SbT2lKwVi
                         iEx3ySqqNoVScoUyhRdnPV2A1LV0yd9GtG9mI4w= )

   nat64.example.com.
                 IN A  192.0.2.1
```

Appendix B. About the IPv4 Address for the Well-Known Name

The IPv4 addresses for the well-known name cannot be non-global IPv4 addresses as listed in the Section 3 of [RFC5735]. Otherwise DNS64 servers might not perform AAAA record synthesis when the well-known prefix is used, as stated in Section 3.1 of [RFC6052]. However, the addresses do not have to be routable or allocated to any real node, as no communications will be initiated to these IPv4 address.

Allocation of at least two IPv4 addresses improves the heuristics in cases where the bit pattern of the primary IPv4 address appears more than once in the synthetic IPv6 address (NSP prefix contains the same

bit pattern as the IPv4 address).

If no well-known IPv4 addresses would be statically allocated for this method, the heuristic would require sending of an additional A query to learn the IPv4 addresses that would be then searched from inside of the received IPv6 address.

Authors' Addresses

Teemu Savolainen
Nokia
Hermiankatu 12 D
FI-33720 Tampere
Finland

Email: teemu.savolainen@nokia.com

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

Email: jouni.nospam@gmail.com

Dan Wing
Cisco Systems
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com