# Speeding up IPv6 Transition: Discovering NAT64 and Learning Prefix for IPv6 Address Synthesis

Yi Ding
University of Helsinki
yi.ding@cs.helsinki.fi

Teemu Savolainen
Nokia Research Center
teemu.savolainen@nokia.com

Jouni Korhonen
Nokia Siemens Networks
jouni.korhonen@nsn.com

Markku Kojo
University of Helsinki
markku.kojo@cs.helsinki.fi

***Abstract*—During the transition from IPv4 to IPv6 hosts in IPv6-only networks need to communicate with IPv4 hosts as most of the Internet services are not yet supporting IPv6. Hosts with IPv6 access would benefit from discovering the presence of NAT64 and learning a prefix needed for IPv6 address synthesis. We propose two mechanisms and systematically evaluate the existing solutions in this area. Based on our comparison of the existing solutions and practical implementation experience, we recommend the heuristic discovery method which is now adopted in the IETF[1] transition toolbox for the IPv6 Internet.**

## I. Introduction

IPv6 transition is entering a critical phase by facing the pressure of IPv4 address space depletion as IANA[2] has assigned its last available IPv4 address pool in February 2011 [1]. The demand for IPv6 is growing strong especially in Europe and Asia where the proliferations of smartphones, IP connected sensor devices and new broadband subscribers are pleading for a larger address space that IPv4 simply cannot offer. For the merits of IPv6 besides sufficient address space the global IPv6 adoption rate is increasing steadily. However, most of the Internet services are still not ready for IPv6. For instance, only a few of the top 1,000 websites provide IPv6 addresses (AAAA records) in the Domain Name System (DNS) and merely 4% of service sites are accessible via an IPv6 native connection [2], [3]. Moreover, until February 27 of 2012 the percentage of registered domains with AAAA records is no more than 2.1% of all domains in the listed top level domains (3,247,614 out of 154,840,089) [4]. As pointed out in [5], [6], the key to speed up IPv6 transition is interoperability so that both IPv4 and IPv6 nodes are able to coexist and communicate with each other.

To achieve interoperability for the smooth adoption, IETF has developed a number of transition mechanisms and standardized recommended solutions. The once criticized Network Address Translation (NAT) mechanism is now gaining a positive role in the IPv6 transition to bridge the gap between the two 'uncollaborative' IP versions. IETF has developed NAT64 [7] for IPv6 to IPv4 protocol translation and DNS64 [8] to make IPv4-only nodes to appear as reachable over IPv6. DNS64 achieves this by returning a synthetic IPv6 address as a response to a query for an IPv4-only address. NAT64 together with DNS64 allows nodes in IPv6-only networks to communicate with nodes in IPv4-only networks. As a growing percentage of access network infrastructure is migrating to IPv6, NAT64 enables connectivity to a large number of existing IPv4 services from IPv6 networks. More users are therefore encouraged to adopt IPv6 as the first choice.

At the same time, owing to the inherited NAT limitations, NAT64 causes problems to various applications such as individual end user targeted web advertising. Hopefully NAT64 will indirectly push service and content providers to upgrade from IPv4 to IPv6 for better service via native IPv6.

When a NAT64 entity is deployed on the edge of an access network, an end host can benefit from learning the information of IPv6 synthesis for two major reasons. First, by discovering the presence of NAT64 the host can choose to avoid network translation by redirecting traffic to other channels. Second, by deriving the NAT64 prefix from a synthetic IPv6 address, the host can synthesize IPv6 addresses without the support of the DNS64 entity and also perform DNSSEC [9] validation successfully.

Several mechanisms for discovering the presence of NAT64 and learning the prefix used for address synthesis are hence proposed in the recent development. The "Framework for IPv4/IPv6 Translation" [10] describes an extensive set of scenarios for IPv4/IPv6 translation. Among these scenarios learning the NAT64 prefix is useful, when communication is initiated from an IPv6 network to the IPv4 Internet or to an IPv4 network [11]. 3GPP[3] has specified IPv6 as the standard addressing scheme. Therefore, the protocol translation scenario is of particular commercial interest in 3GPP based cellular networks (3GPP migration guidelines, scenario 3 [12]) as an alternative to dual-stack deployments. As of writing this paper first commercial IPv6-only cellular networks are already being launched.

In this paper, we study the area of discovering the NAT64 function in access networks and learning the NAT64 IPv6 prefix used for address synthesis. Our major contributions include:

- An extensive survey of solutions. We summarize our analysis in [11] and extend it further with learned deployment considerations.

[1]Internet Engineering Task Force, http://www.ietf.org
[2]Internet Assigned Numbers Authority, http://www.iana.org
[3]The 3rd Generation Partnership Project, http://www.3gpp.org

- Based on our systematic comparison of solutions, we recommend the heuristic discovery approach [13].
- We implement both of our proposals and share our experience in mobile networks to guide future development.

The rest of paper is organized as follows. Section II introduces the architecture and major functions of NAT64/DNS64. Section III provides an extensive survey of solutions. In Section IV we analyze all proposals and provide our recommendation. The implementation of our proposals is covered in Section V. Finally, we conclude the paper and discuss future work in Section VI.

## II. NAT64 AND DNS64

NAT64 [7] and DNS64 [8] enable nodes in IPv6-only networks to communicate with nodes in IPv4-only networks. A typical 3GPP cellular network-based NAT64/DNS64 setup is presented in Fig. 1. NAT64 performs stateful translation very much like ordinary IPv4 NAT, except that in addition to address translation it also translates packet headers between IPv6 and IPv4. DNS64 is an essential part of the overall protocol translation system, making IPv4-only reachable nodes to look like being reachable over IPv6. DNS64 does this by combining IPv4-only node's real IPv4 address and the prefix of the NAT64 entity together as a synthetic IPv6 address. The synthetic IPv6 addresses are delivered to an IPv6-only node inside synthetic AAAA records, hence making the node to believe that the destination is IPv6-reachable, while in reality it is not. Standardized address formats are used in the process [14]. The NAT64 prefix can be a Network-Specific Prefix (NSP) which is assigned to a provider and chosen by the network administrator or a Well-Known Prefix (WKP) defined by IETF as 64:ff9b::/96 to represent IPv4 addresses in IPv6 namespace (i.e. an IPv4-converted IPv6 address).

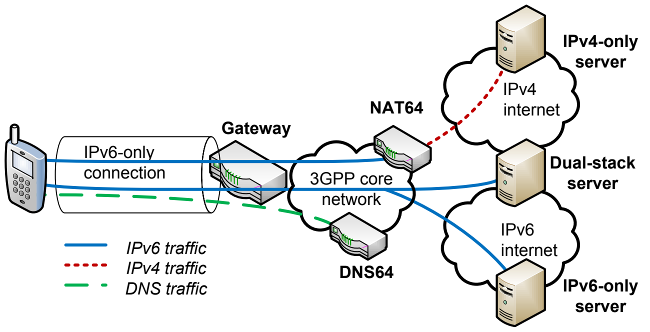

Figure 1: NAT64 and DNS64 in cellular network

The NAT64/DNS64 system works very well when applications on IPv6-only connected hosts use only DNS to resolve domain names for IPv6 addresses in their communications and the hosts do not implement validating DNSSEC-capable resolvers [9]. Applications that need to create connections using IPv4 address literals face obvious problems when the node has IPv6-only connectivity. Furthermore, DNSSEC-capable validating resolvers cannot successfully validate synthetic IPv6 addresses received from DNS64 [8].

The solution for allowing a host to cope with IPv4 address literals and perform DNSSEC validation in networks having NAT64 services is obvious: the host needs to implement local IPv6 address synthesis functionality. However, before a host can perform local IPv6 address synthesis it has to learn the NAT64 IPv6 prefix used in a current access network.

Once a host, with any tool, has learned the IPv6 prefix of the NAT64 in the attached network, the host can synthesize IPv6 addresses from IPv4 addresses as described in the DNS64 specification [8]. This means the host can synthesize IPv6 addresses out of (possible DNSSEC validated) A records or from IPv4 address literals in URLs (e.g. *http://192.0.2.1/index.html*) or from other sources.

In order to avoid man-in-the-middle attacks, it must be possible to secure the NAT64 prefix learning procedure on need basis. If an attacker is able to influence what NAT64 IPv6 prefix nodes configure, it will be possible to cause at least denial-of-service and traffic redirection attacks for flooding or eavesdropping purposes. In the worst case the node performs an IPv6 address synthesis by combining a DNSSEC [9] validated IPv4 address with the NAT64 IPv6 prefix and thinks the resulting address is securely synthesized.

NAT64 works best with applications that do not embed IP address literals in the application protocol payload. Applications that embed IP address literals in their payload are likely to fail in the presence of NAT64 unless Application Layer Gateways (ALGs) are implemented on NAT64s.

## III. Survey on Solutions

### A. Motivation, Challenges, and Solution Categories

As dual-stack OSs and multiple interfaces have become the main stream, discovering the presence of NAT64 in the access network can assist such hosts to bypass NAT64 via a native IPv4 connection or other interfaces. At the same time, learning the NAT64 prefix can support hosts in constructing valid IPv6 synthesis addresses and hence solving the problem of embedded IPv4 address literals, which has become the major cause of connection disruption for IPv6-only hosts in accessing current Internet services [3]. Such information also helps applications that do not utilize DNS but obtain destination addresses via other means.

Based on our investigation [11], we identify two major challenges in this domain. First, how to minimize the implementation impact on hosts and network entities. This is the key to deploy and integrate the proposed mechanism into existing networks. Second, how to enable security. The solution should provide means for ensuring that NAT64 prefix detection is not compromised. To assist analysis, we divide existing solutions into five categories:

- Solutions independent of existing deployed technologies.
- Application layer solutions.
- Network layer solutions.
- Link layer solutions relying on access technology assets.
- Out-of-band solutions relying on manual configuration.

## B. Technology Independent Solutions

A solution using heuristic methods [13] was designed when it became evident that it may be impossible to have all access networks with NAT64 to explicitly provide prefix information required for IPv6 address synthesis. The solution is simple: a node shall make an AAAA record DNS query against a well-known name (such as *ipv4only.arpa* or any other domain name with required properties of top level domain high availability and existence in foreseen future) that the node knows not to have any IPv6 addresses. Hence, any IPv6 address received in a DNS response reveals there is a DNS64 in the network.

The IPv6 prefix and the synthetic address format used by the DNS64 and the NAT64 is discovered by searching for an IPv4 address of the well-known name inside the received IPv6 address. The location of IPv4 address determines the used address format [14] and what is the prefix used by the access network. The learned prefix and address format is then used for local IPv6 address synthesis. As the learned prefix and format are cached by the node, the time-to-live (TTL) value of AAAA synthetic record determines how often a node needs to repeat the heuristic discovery to update the cached information.

## C. Application Layer Solutions

At the application layer, a new `EDNS0` option serving as an implicit indication of NAT64/DNS64 presence has been proposed [15]. Three bits are defined in the option to convey the prefix length used for IPv6 synthesis. In this solution, both hosts and DNS64 servers need to understand the new option. Upon receiving valid requests from hosts, only DNS64 servers are able to insert the EDNS0 option with the required bits into the synthesized AAAA Resource Record.

In the version -00 of [15], a similar solution was proposed to add three bits into the EDNS0 OPT header [16] to notify the prefix length used for synthesis as an implicit indication of NAT64/DNS64 presence. This solution also required hosts and DNS64 servers to understand the new flag bits contained in EDNS0 OPT header.

Based on DNS, a new Resource Record (RR), A64, is proposed in [17] to store a synthetic IPv6 address. The dedicated A64 record allows hosts to distinguish between real IPv6 addresses and synthetic IPv6 addresses. To obtain A64 record, hosts need to send a query asking for the new record. Positive response from DNS indicates that the resolved address is synthesized rather than native IPv6 address.

Two DNS based mechanisms are proposed in [18] to discover the presence of NAT64 and learn the NAT64 prefix. The first mechanism utilizes a TXT based Resource Record to convey a predefined string containing the NAT64 unicast IPv6 address and its prefix length. The second mechanism relies on a new U-NAPTR application to conduct U-NAPTR query similar to reverse DNS query. The domain name used in U-NAPTR query is derived from host's IPv6 address in the ".ip6.arpa." tree and response to a successful query contains the unicast IPv6 address and the prefix length of the NAT64.

Another application layer solution is to utilize application layer signaling protocols such as Session Traversal Utilities for NAT (STUN) [19]. With STUN, a host first needs to resolve the synthetic IPv6 address of the target STUN server residing in the IPv4 Internet. After learning the synthetic IPv6 address of the STUN server, a STUN client on the host sends a request to the STUN server with a new `SENDING-TO` attribute that tells the server the IPv6 destination address the client sent the request to. The server responds with another new attribute `RECEIVED-AS` containing server's IPv4 address the request arrived at. By comparing the addresses in `SENDING-TO` and `RECEIVED-AS`, the host can derive the NAT64 prefix information.

## D. Network Layer Solutions

At the network layer, two DHCP based mechanisms are proposed to hook the discovery and learning process into general IP configuration phase [18], [20]. The `OPTION_AFT_PREFIX_DHCP` option proposed in [18] contains necessary information to derive NAT64 prefix, including the IPv6 unicast prefix, IPv6 any-source multicast prefix, source-specific multicast prefix, and their lengths. Another option described in [20] provides similar information for hosts to allow them synthesize IPv6 addresses that can be routed through NAT64.

A new router advertisement (RA) option defined as `OPTION_AFT_PREFIX_RA` is proposed in [21] to convey information similar to the DHCP based solution, including IPv6 unicast prefix, IPv6 any-source multicast prefix, source-specific multicast prefix, and the length of prefixes for NAT64.

## E. Link Layer Solutions

Many access technologies allow configuration information to be delivered during link layer establishment phase, such as 3GPP Non-Access-Stratum (NAS) signaling protocol [22]. It is possible to utilize such signaling to deliver and negotiate parameters including all NSPs with their respective prefix length via generic protocol configuration options. The lack of such options would be treated as implicit indication that no NAT64 is deployed in the access network.

## F. Out-of-Band Solutions

For the final category of possible solutions the information needed for the IPv6 address synthesis is delivered via some other means than over the node's Internet connection. This may include, for example, manual configuration of nodes, usage of some provisioning tools such as the Access Network Discovery and Selection Function (ANDSF) [23], or any other protocol designed for network administration purposes.

## IV. Systematic Evaluation and Discussions

## A. Evaluation Metrics and Solution Classification

In order to compare and evaluate existing solutions, we identify and define eight evaluation metrics.

1) Active detection - hosts purposely try to detect NAT64.

Table I: Comparison Table for NAT64 Prefix Learning Solutions

| Solution | Active detection | Explicit via configuration | Adaptation to changes | Multiple NSP support | Without DNS protocol | Transparent to network | Host system changes | Secure learning |
|---|---|---|---|---|---|---|---|---|
| Heuristic discovery | Yes | No | Moderate | Yes | No | No if DNSSEC | No | With DNSSEC |
| EDNS0 option/OPT flags | Yes | No | Fast | Yes | No | No | Yes | No |
| DNS A64 RR | Yes | No | Fast | Yes | No | No | Yes | No |
| STUN | Yes | No | Moderate | No | Possible | No | No | No |
| DNS TXT/U-NAPTR RR | Yes | No | Fast | Yes | No | No | No | No |
| DHCPv6 | No | Yes | Moderate | Yes | Yes | No | Yes | No |
| RA | No | Yes | Fast | Yes | Yes | No | Yes | No |
| L2 | No | Yes | Undefined | Yes | Yes | No | Yes | With secure L2 |
| Out-of-Band | No | Yes | Undefined | Yes | Yes | Possible | No | Possible |

2) Explicit via configuration - hosts learn prefix passively as part of the host configuration procedure.
3) Capability to adapt to dynamic changes of NSP.
4) Support for multiple NSPs.
5) Independent of DNS protocol support on hosts.
6) Transparent to existing services and network entities.
7) Requiring changes to operating system services/kernel.
8) Secure learning of NAT64 prefix.

As our focus is on deployability, we selected the metrics that cover the essential aspects of this domain, including functionality, transparency, and deployment impact. Table I summarizes the existing solutions against our evaluation metrics.

The active detection is enabled in heuristic discovery and all application layer solutions since hosts are able to initiate the NAT64 detection on demand by calling necessary function or application. Meanwhile, network layer solutions, link layer solutions, and out-of-band solutions rely on the network-generated information passively obtained from the access network during host configuration. Such solutions are identified as explicit via configuration.

Regarding to dynamic adaptation to the change of NSP, the analyzed solutions differ on how fast they can adapt to possible changes in the NAT64 prefix. In the cases where the NAT64 prefix detection is based on reception of IPv6 prefix information from DNS, a solution must repeat the detection at latest when the received DNS record's time to live (TTL) expires. If the solution repeats the detection only when the TTL is about to expire, we call the rate of adaptation `moderate`. The heuristic discovery, STUN, DHCPv6-based solution are included in the moderate category, as the received information can be associated with a lifetime. Some of the solutions are able to adapt to possible changes in the NAT64 prefix faster than the timeout triggered moderate solutions. We categorize these as `fast`. The fast solutions include RA-based approach where the network can trigger the NAT64 prefix detection simply by sending a new RA. We also consider the EDNS0 and DNS-based solutions fast, as in those cases the host will discover new NAT64 prefix at the moment it receives synthetic DNS reply to any IPv6 DNS query. For link layer and out-of-band solutions, as currently they are speculative approaches without clear defined proposals, we identify them as `undefined`.

Due to the restriction of relying on STUN functionality, only the STUN approach is not able to support multiple NSPs in the access network, while other solutions provide such support.

For DNS protocol independence, all DNS related solutions are inherently excluded. Meanwhile, DHCP, RA, link layer, and out-of-band solutions are able to function without DNS protocol support. For STUN, it is possible that a host can learn the IPv6 presentation for the STUN server's IPv4 address without using DNS, e.g. by using DHCPv6. It is hence identified as possible.

Concerning transparency to network entities, all solutions introduce modifications to existing entities in the network, except the heuristic discovery and out-of-band solution. The heuristic discovery is a transparent procedure for the access network, except when it is protected via DNSSEC. To enable DNSSEC protection for the heuristic discovery, administrator of a NAT64 entity needs to allocate a fully qualified domain name for the NAT64 and also maintain a set of related PTR and DNSSEC signed AAAA records. A host then uses these DNS records to protect the heuristic discovery procedures, but becomes dependent on network support. The out-of-band mechanisms may be transparent to the network from the Internet Protocol suite point of view, but obviously do require deployment and management of the out-of-band mechanisms themselves.

Changes to a host system are inevitable for EDNS0 and DNS A64 because the resolver functionality on the host need to be modified to support the new proposals. DHCPv6, RA, and link layer solutions also introduce modifications on the hosts to enable NAT64 discovery and prefix learning. Since heuristic discovery, STUN, and DNS TXT solutions can utilize the existing services on hosts to acquire necessary information with the support from network side, they do not require host modifications. The out-of-band mechanisms are likely to be implemented at the application layer and hence would not necessitate host system changes.

The secure discovery of a NAT64 prefix is challenging for all solutions. The heuristic discovery includes an algorithm that could ensure secure prefix discovery but requires use of DNSSEC on a host and signed FQDNs for NAT64 entities [13]. Furthermore, the algorithm is still subject to changes. The link layer solutions are likely to be able to provide truly secure discovery assuming the link layer itself is secured. For out-of-band solutions secure discovery is also an option.

### B. Evaluation of Solutions

Concerning each solution category, the heuristic discovery can be deployed without explicit support by a host or network. Hosts interested in learning IPv6 synthesis can proactively do the "discovery" at any time. Although the solution is dependent on DNS protocol, it offers moderate adaptation to NSP changes and supports multiple NSPs. With the advantage of secure discovery, this solution can be used also as a fall-back mechanism when explicit methods such as EDNS0 and DNS A64 are not supported in the access network.

Among the application layer solutions, all DNS based proposals (EDNS0 OPT flags, EDNS0 option, A64 RR, and TXT/U-NAPT RR) are light weight in general but suffer from limitations. For instance, A64 Resource Record lacks the capability to learn NAT64 prefix for synthesis. The main concern against such solutions are two fold. First, there are already deployed DNS64 implementations without support for such solutions, which will cause backward compatibility issues. Second, changes are required on the host side DNS resolvers. While this is not the case in all situations, the need for possible resolver library or API modifications is strong enough reason to favor other solutions over the DNS based methods. Furthermore, having all DNS64 servers to support explicit WKP/NSP discovery (ENDS0, A64, and DNS based approaches) is difficult to arrange. The STUN solution also involves changes and management of network entities that are not part of NAT64/DNS64 deployment.

For network layer solutions, both DHCPv6 and RA are efficient in that they are hooked to general IP configuration phase without involving DNS. Both solutions allow easy updates when configuration information changes. However, both solutions introduce changes to both host IP stack and network components, which is generally unfavored. The DHCPv6 approach requires that the NAT64, DHCPv6, and possibly even DNS64 servers are all synchronized. While DHCPv6 utilizes centralized management model, RA based solution is expensive in operation as configuration need to be placed and maintained on all access routers. Furthermore, both DHCPv6 and RA involve entities that do not necessarily need to be aware of NAT64 operations. Changes to DHCP and Neighbor Discovery protocols demand extra standardization efforts.

What becomes to link layer solutions, in theory it would be possible to define access technology specific methods to pass on information required for local IPv6 address synthesis. This approach, however, would come with several disadvantages, such as standards changes required for all access technologies, upgrades needed to entities involved in the link layer establishment procedures, configuration of information required for IPv6 address synthesis to all access routers and gateways, and also host changes for passing information from the link layer to a higher layer. Due to these complex dependencies and required changes, the link layer based approaches are considered to be out of question.

The out-of-band solutions, which currently are undefined according to authors' knowledge, could in theory work very well in specific deployment scenarios, but not in the general case. For example, manual configuration does not scale, ANDSF is essentially used only for 3GPP deployment scenarios and hence would not be enough for general purpose nodes, and special protocols used for network administration would help applications only when applications are actually executed in nodes that support such special protocols and that are under the network administration services.

### C. Recommendation and Discussions

Guided by the principle of minimizing impact on both host and network sides, we recommend the heuristic discovery solution for its ease of deployment. For IETF standardization, the main argument to favor it over other solutions, especially the EDNS0 option, is that people will implement and deploy similar solution in any case. Therefore, having a common standardized solution is for common benefit. At the same time, we are aware of that standardizing a well-known domain name has a particular operational concern. For instance, who is going to operate the infrastructure for the well-known domain name? A domain name server hosting the well-known domain name has practically similar availability requirements as root or top level domain name servers. Furthermore, the well-known domain name has to be operational for unforeseen number of years to come. These are not trivial issues especially for a non-profit service. Basically the only viable solution is to host the well-known domain name under the IANA managed name space, which is used for other Internet wide basic infrastructure operations, i.e., under the `.arpa` domain. In addition, all proactive NAT64 prefix determination proposals including our EDNS0 solution can benefit from such well-known domain name.

Multi-homing is a general problem when multiple NAT64/DNS64 boxes are deployed on different access networks. As stated in [24] the synthesized AAAA record is guaranteed to be valid only on the network from which the knowledge is learned. Nodes with multiple interfaces should distinguish the learned information from different networks to avoid potential conflict and connection failure.

## V. Heuristic Discovery and EDNS0

### A. Protocol Implementations

We implemented both EDNS0 [15] and heuristic [13] based NAT64 discovery and NAT64 prefix determination methods. For the implementation we took two approaches. First one is a standalone solution (a `ping64` network diagnostic tool) that implements both EDNS0 and heuristic discovery within the application but leaves the DNS resolver untouched. Second one is a modification to `glibc` and its `getaddrinfo()` DNS resolver function. For the EDNS0 solution to work, we also modified Ecdysis' open-source implementation of a `BIND9` based DNS64 [25] on Linux to understand our ENDS0 additions. The client side of standalone solution is implemented on Mac OS X 10.6 Snow Leopard and the integrated solution is implemented on Ubuntu 10.04 with Linux kernel 2.6.32 and `glibc-2.13` for resolver function.

The EDNS0 approach required a definition of a new EDNS0 option to inform the end host that IPv6 address synthesis took place and convey the NAT64 prefix length information. The option is shown in Fig. 2.

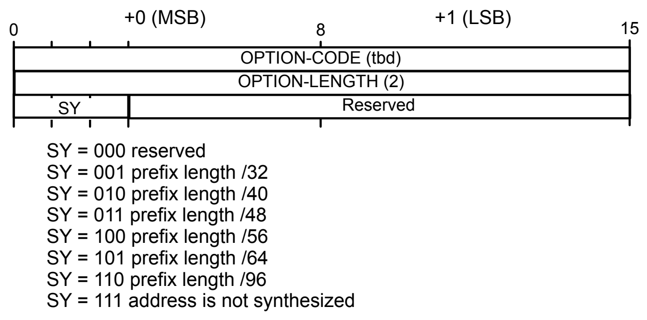
+0 (MSB)
+1 (LSB)
0
8
15
OPTION-CODE (tbd)
OPTION-LENGTH (2)
SY
Reserved
SY = 000 reserved
SY = 001 prefix length /32
SY = 010 prefix length /40
SY = 011 prefix length /48
SY = 100 prefix length /56
SY = 101 prefix length /64
SY = 110 prefix length /96
SY = 111 address is not synthesized

Figure 2: EDNS0 option for NAT64 prefix determination

The host indicates support for the feature by including the EDNS0 option into the DNS query. The absence of the EDNS0 option in the reply means that either no synthesis takes place or the DNS64 entity does not support the feature. Either way, when the EDNS0 option is missing, the host cannot conclude for certain whether the AAAA response is synthetic or not.

Our patched DNS64 server includes the EDNS0 option with prefix length information in the DNS response when address synthesis takes place and the host indicates support for the EDNS0 feature. With the prefix length information the host can easily extract the NAT64 prefix out of the synthetic IPv6 address.

For the heuristic discovery, we configured a fully qualified domain name (FQDN) with an A record only into a DNS server under our management. Step A in Fig. 3 illustrates how NAT64 prefix is learned out of a received synthetic IPv6 address. The step B shows how the learned information is used to construct a synthetic IPv6 address. Before receiving the synthetic IPv6 address from DNS64, the node has learned with DNS A record query or by static configuration the well-known name's IPv4 address (e.g., *127.127.127.127*). The node learns the network is performing IPv6 address synthesis when the node receives an IPv6 address, in this example *2001:db8::7f:7f7f:7fab:cdef*, to AAAA record DNS query for the well-known IPv4-only name. The node searches for the IPv4 address inside the received synthetic IPv6 address, and in this example the IPv4 address can be found encoded into bits 72-103. The location of the IPv4 address indicates that the NAT64 prefix is 64-bits. Therefore, the node extracts the first 64 bits of the IPv6 address to be used as the NAT64 prefix. As per [14] the 'u' and 'suffix' fields should have been set to zero by DNS64, and hence by the current standards the node cannot do anything but initialize these to zero during local IPv6 address synthesis. In the future when some meaning for 'u' and 'suffix' fields is possibly defined the node doing the local synthesis will need to be adjusted as well. Fig. 3 step B shows the result of local synthesis of an IPv6 address that maps to the IPv4 address of 192.0.2.1 and how the 'u' and 'suffix' fields are set to zero as per standard.

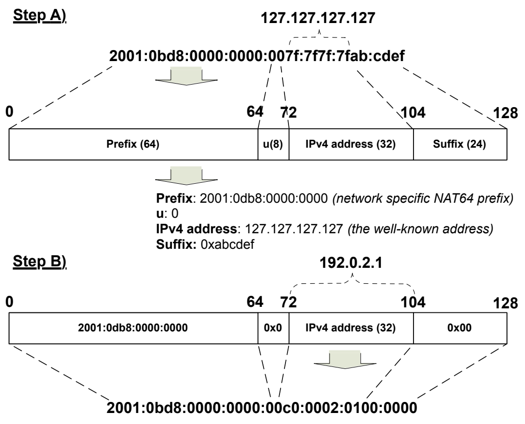
Step A)
127.127.127.127
2001:0bd8:0000:0000:007f:7f7f:7fab:cdef
0
64
72
104
128
Prefix (64)
u(8)
IPv4 address (32)
Suffix (24)
Prefix: 2001:0db8:0000:0000 (network specific NAT64 prefix)
u: 0
IPv4 address: 127.127.127.127 (the well-known address)
Suffix: 0xabcdef
Step B)
192.0.2.1
0
64
72
104
128
2001:0db8:0000:0000
0x0
IPv4 address (32)
0x00
2001:0bd8:0000:0000:00c0:0002:0100:0000

Figure 3: NAT64 prefix determination and IPv6 address synthesis. IP numbering is for exemple purposes only.

The `ping64` program allows us to test NAT64 and prefix determination. It also supports pinging IPv4 destinations from IPv6-only networks using IPv4 literal destinations (such as *192.0.2.1*). In this case the program first issues a DNS query using the standard `libresolv`[4] API for the known IPv4-only FQDN. If the DNS reply includes an EDNS0 option with the prefix length information, it is used for learning the NAT64 prefix and for subsequent IPv6 address synthesis. If the DNS64 server does not support EDNS0, `ping64` employs the heuristic algorithm and uses the known octet pattern to learn the NAT64 prefix as illustrated in Fig. 3.

For sake of completeness, we also implemented a backward compatible patch to the `glibc` and its `getaddrinfo()` DNS resolver function. The application calling the `getaddrinfo()` function for a name resolution would eventually receive the SY bits in `struct addrinfo.ai_flags` for each returned IPv6 address. The patched `glibc` practically implemented equivalent functionality as the `ping64` but within the host wide (dynamically linked) library, thus making it implicitly available for all applications linking against `glibc`. Naturally, an arbitrary application has to know how to make use of the extended `getaddrinfo()` functionality.

In addition, we implemented an `AI_POLICYTABLE` hint flag in the `getaddrinfo()` for application to express its willingness to generally prefer native IPv4 connectivity over IPv6 connectivity through NAT64. This is important when a dual-stack host in a dual-stack access network would receive both A and synthetic AAAA records for the queried FQDN. If the application sets the `AI_POLICYTABLE` hint upon (any) name resolution request, then the patched `getaddrinfo()` would modify the *default address selection policy table* [26] so that *IPv4-mapped* IPv6 addresses are preferred over synthetic IPv6 addresses, thus making the host select IPv4 transport over IPv6 through a NAT64.

[4]Standard `libresolv` originates from BIND distribution.

### *B. Implementation Experience*

In our implementation the EDNS0 option on the host resolver side (at the application level including DNS query and reply processing) is around 130 lines of uncommented C-code and on the DNS64 server side the patch is of much less code. Implementation was straightforward. The `glibc` approach with a *policy table* modifications required a considerable effort and a number of code changes. This is mainly due to the complexity of the `glibc` integration and additional communication into the kernel space for *policy table* modifications. On the other hand, the heuristic-based solution is straightforward to implement requiring less than 100 lines of uncommented C-code.

Based on our feasibility tests in operator networks, the immediate benefit of the heuristic-based solution over the EDNS0-based solution is its independence of both resolver and DNS64 modifications. The scalability and interoperability are the main challenges for solutions with impact on both hosts and network entities. For instance, the EDNS0 solution is tightly coupled with the existing deployment in terms of selected DNS64/NAT64 software. Even though an implementation should strictly follow the protocol standard, a DNS resolver for one system setup may encounter troubles in communicating with other DNS64 servers if they exhibit vendor specific features, e.g., for optimization purposes. The advantage of the heuristic discovery becomes dominant in this respect due to its minimal impact on external components.

## VI. Conclusions and Future Work

Our work provides an empirical basis for evaluating incentives and coordination issues surrounding the existing and upcoming NAT implementation strategies to speed up the transition from IPv4 to IPv6. According to our best knowledge, we are the first to provide an extensive and systematic evaluation of all known proposals in this domain. The impact of our work is reflected in the adoption of heuristic discovery method by IETF as the main track of development. Our implementation also offers a first-hand experience in real networks, revealing potential pitfalls and assisting in understanding of what should be taken into account in protocol design in the transition phase. In the future we intend to investigate the security aspects of the recommended solution.

## Acknowledgement

This work has been carried out in the WiBrA project by University of Helsinki, Nokia, Nokia Siemens Networks, and TeliaSonera, funded in part by TEKES Finland. We also would like to thank Aki Nyrhinen and Ilpo Järvinen for their implementation support.

## References

[1] K. Claffy, "Tracking IPv6 evolution: Data We Have and Data We Need," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 3, 2011.

[2] A. Keränen and J. Arkko, "Some Measurements on World IPv6 Day from End-User Perspective," IETF, Internet Draft draft-keranen-ipv6day-measurements-01, July 2011, work in progress.

[3] J. Arkko and A. Keränen, "Experiences from an IPv6-Only Network," IETF, Internet Draft draft-arkko-ipv6-only-experience-04, October 2011, work in progress.

[4] M. Leber, "Global IPv6 Deployment Progress Report, http://bgp.he.net/ipv6-progress-report.cgi." [Online]. Available: http://bgp.he.net/ipv6-progress-report.cgi

[5] D. Geer, "To Boost Adoption, IPv6 Proponents Back Once-Shunned Technology," in *IEEE Technology News, 2008*.

[6] M. Shin *et al.*, "An Empirical Analysis of IPv6 Transition Mechanisms," in *Proc. of ICACT 2006*.

[7] M. Bagnulo, P. Matthews, and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers," Internet RFCs, ISSN 2070-1721, RFC 6146, April 2011.

[8] M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers," Internet RFCs, ISSN 2070-1721, RFC 6147, April 2011.

[9] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," Internet RFCs, ISSN 2070-1721, RFC 4033, March 2005.

[10] F. Baker, X. Li, C. Bao, and K. Yin, "Framework for IPv4/IPv6 Translation," Internet RFCs, ISSN 2070-1721, RFC 6144, April 2011.

[11] J. Korhonen and T. Savolainen, "Analysis of solution proposals for hosts to learn NAT64 prefix," IETF, Internet Draft draft-ietf-behave-nat64-learn-analysis-02, December 2011, work in progress.

[12] 3GPP, "TS23.975, IPv6 migration guidelines, http://www.3gpp.org/ftp/Specs/html-info/23975.htm," June 2011. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/23853.htm

[13] T. Savolainen, J. Korhonen, and D. Wing, "Discovery of a Network-Specific NAT64 Prefix using a Well-Known Name," IETF, Internet Draft draft-ietf-behave-nat64-discovery-heuristic-05, January 2012, work in progress.

[14] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, and M. Li, "IPv6 Addressing of IPv4/IPv6 Translators," Internet RFCs, ISSN 2070-1721, RFC 6052, October 2010.

[15] J. Korhonen and T. Savolainen, "EDNS0 Option for Indicating AAAA Record Synthesis and Format," IETF, Internet Draft draft-korhonen-edns0-synthesis-flag-02, February 2011, work in progress.

[16] P. Vixie, "Extension Mechanisms for DNS (EDNS0)," Internet RFCs, ISSN 2070-1721, RFC 2671, August 1999.

[17] M. Boucadair and E. Burgey, "A64: DNS Resource Record for IPv4-Embedded IPv6 Address," IETF, Internet Draft draft-boucadair-behave-dns-a64-02, September 2011, work in progress.

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

[19] J. Rosenberg *et al.*, "Session Traversal Utilities for NAT (STUN)," Internet RFCs, ISSN 2070-1721, RFC 5389, October 2008.

[20] M. Boucadair *et al.*, "Dynamic Host Configuration Protocol (DHCPv6) Options for Shared IP Addresses Solutions," IETF, Internet Draft draft-boucadair-dhcpv6-shared-address-option-01, December 2009, work in progress.

[21] D. Wing, X. Wang, and X. Xu, "Learning the IPv6 Prefix of a Network's IPv6/IPv4 Translator," IETF, Internet Draft draft-wing-behave-learn-prefix-03, July 2009, work in progress.

[22] "Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS)," 3GPP, TS 24.301 8.8.0, December 2008.

[23] "TS24.302, Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks, http://www.3gpp.org/ftp/Specs/html-info/24302.htm," 3GPP, TS24.302.

[24] M. Blanchet and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement," Internet RFCs, ISSN 2070-1721, RFC 6418, November 2011.

[25] Viagenie, "Ecdysis: open-source implementation of NAT64 and DNS64, http://ecdysis.viagenie.ca/." [Online]. Available: http://ecdysis.viagenie.ca/

[26] R. Draves, "Default Address Selection for Internet Protocol version 6 (IPv6)," Internet RFCs, ISSN 2070-1721, RFC 3484, February 2003.