Transition to IPv6 with Realm Gateway 64

Jesús Llorente Santos, Raimo Kantola
Department of Communications and Networking
Aalto University
Helsinki, Finland
{jesus.llorente.santos, raimo.kantola}@aalto.fi

***Abstract*—The IPv4 address space has been depleted and the usage of IPv6 is still very limited, however increasing. Smooth coexistence of IPv4 and IPv6 can support the development of the next generation Internet. During the transition there will be IPv4-only, IPv6-only and dual-stack hosts and network segments. This paper presents Realm Gateway 64 (RGW64) – a solution for interconnecting heterogeneous network realms as defined by the IETF, which does not require changes in end-hosts. RGW64 relies on stateful DNS64/NAT64 translation and DNS resolution for establishing inbound connections. An analysis of the scalability and the security is also presented. The paper shows that RGW64 is suitable for operators who want to gradually migrate customer networks to IPv6 yet maintaining reachability with the IPv4 Internet.**

Keywords—IPv6 transition; IPv6 translation; NAT; NAT64; DNS64; Realm Gateway.

I. INTRODUCTION

The IPv4 address depletion turned into reality as the Internet Assigned Numbers Authority (IANA) allocated the last address blocks available in favour of the Asia-Pacific Network Information Centre (APNIC) back in 2011 [1]. APNIC stated that in order to sustain the ongoing growth of the Internet, the industry has to move quickly towards IPv6 [2]. As of 2013, the mobile broadband subscriptions were estimated worldwide at 2.1 billion [3]. The rapid increase of mobile users continues fuelling the demand for addresses.

There have been many efforts to tackle the issues of the address exhaustion and to extend the lifetime of IPv4. The industry's reaction was the adoption of the Network Address Translator (NAT) [4, 5]. NAT allows hosts in private networks to connect to the Internet by sharing a single public IP address. However, NAT also introduces a problem of reachability, preventing the Internet hosts from unilaterally initiating connections with the hosts in the private networks. In addition, there are a number of protocols whose functionality is heavily affected by NAT [6]. The IETF standardized a set of recommendations called UNilateral Self-Address Fixing (UNSAF) architecture [7]. The UNSAF relies on STUN [8], TURN [9] and ICE [10] in order to allow hosts to communicate under virtually any kind of network conditions. However, the discovery algorithms are executed by the hosts, increasing the complexity of the applications and the computation required.

Despite the fact that IPv6 has been around for a number of years, the adoption globally is very slow. Cisco [11] indicates that Belgium is currently leading the rank in IPv6 penetration with a relative index of 10/10, followed by Switzerland (9.1), Germany (7.8) and USA (6.9). Likewise, Google [12] reports that today, only 3% of the users access their services via IPv6. However, the share has tripled in the past year showing symptoms of increasing adoption.

There are many reasons that have interfered with the deployment of IPv6: (1) It is not compatible with IPv4 and requires hardware upgrades and configuration in network and home equipment, with the added cost; (2) NATs are a widely deployed inexpensive solution that have become de facto standard among Internet Service Providers (ISP) for Internet connectivity in Small Office Home Office (SOHO); (3) NAT provides a centralized aggregation point where security mechanisms such as firewall can be applied and traffic analysed; (4) IPv6/IPv4 interworking mechanism do not allow reaching IPv6 single-stack servers from IPv4 single-stack clients resulting in all servers having to have the IPv4 protocol and also an IPv4 address; (5) the fact that servers must have IPv4 lowers the motivation to have IPv6 in clients when clients can be NATed; (6) when a company cannot obtain IPv4 addresses for its servers, it can delegate the problem to a cloud server provider; and (7) NAT is a well understood technology whereas IPv6 still results in confusing the users.

Nevertheless, IPv6 is coming and some people believe that emerging services will have no other choice than running on IPv6. Naturally, the question arises: how can we interconnect the current IPv4 with the future IPv6 world? As a result, a number of IPv6 transition and co-existence mechanisms have already been developed. These are categorized in: (1) translation, (2) tunnelling, and (3) dual-stack.

This paper proposes a solution called Realm Gateway 64 (RGW64) as a new IPv6 transition technology based on translation. RGW64 grants communications of hosts in IPv4 and IPv6 address realms via protocol translation. In addition, it provides to single-stack Internet hosts the ability to unilaterally initiate connections to single-stack servers in customer networks without complex NAT traversal mechanism. The solution is also suitable for deployment in SOHO networks provided that it does not require modifications or patches to the existing hosts or protocols.

The paper is structured as follows. Section II describes the related work. Section III explains the Realm Gateway 64. Section IV covers the evaluation of the functionality, performance and security of the solution. Section V introduces the discussion and finally Section VI presents the conclusions.

This work has been performed in the framework of the CELTIC project CP7-011 MEVICO and CP2012/2-5 SIGMONA.

(PrePrint) accepted for publication in IEEE International Conference on Communications (ICC 2015), London, UK, 8-12 June 2015.

II. Related Work

For helping IPv6 deployment, IETF [13] defines two mechanisms that are able to operate on top of the current IPv4 infrastructure: Dual-stack and IPv6 tunnelling. Dual-stack deployments require hosts and network equipment to support both IPv4 and IPv6, thus allowing a gradual migration towards an IPv6-only architecture. On the other hand, the purpose of tunnelling is to create IPv6 links over the existing IPv4 network segments that are not yet compatible with IPv6. Point-to-point tunnelling can be configured either manually or via Generic Routing Encapsulation (GRE) [14]. There is an additional subset of tunnelling techniques that enable more flexibility and automation compared to the manual approach: (1) 6to4 [15] creates unique IPv6 addresses out of globally unique IPv4 addresses, prepending the reserved prefix `2002::/16`; the model is complemented with 6to4 relays/routers reachable via anycast addresses for forwarding the data between the end hosts. (2) 6rd [16] is built on 6to4; for a service provider 6rd enables rapid deployment of IPv6 and the operation of its own relay servers by utilizing a different IPv6 prefix than 6to4. (3) Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) [17] utilises the underlying IPv4 network as link layer for IPv6, thus enabling communication between dual-stack nodes over the current infrastructure. (4) Teredo [18] brings in IPv6 connectivity to hosts located in IPv4 networks, even behind NAT, by tunneling IPv6 packets in UDP/ IPv4 packets; the tunnel is terminated on the other end by Teredo relays, which un-encapsulate the IPv6 packets and forward them to the intended destination.

Translation mechanisms have been proposed by the IETF. The first attempt was NAT-PT [19] and relied on Stateless IP/ICMP Translation (SIIT) [20]; it was recently deprecated [21]. In 2001, IETF released a Framework for IPv4/IPv6 Translation [22], which brought together protocol and address translation, stateful operations and Application Layer Gateways (ALG). These techniques are referred as DNS64/NAT64. The acceptance of these solutions seems to increase as more proprietary and open source solutions become available. In the proprietary segment: Microsoft UAG [23], Cisco ASR 1000 [24], or Juniper [25]. In open source we find, Ecdysis [26], TAYGA [27], or ISC Bind [28] whose version 9.8 already implements DNS64 and address synthesis. Related works also include performance evaluation of NAT64 implementations [29] and experience of IPv6-only network behind NAT64 [30].

Unfortunately, the cited solutions cannot properly handle inbound connections towards the customer network hosts. New architectures have been proposed [31] to overcome the reachability problems of address translators. However, many of them require changes in at least one of the communicating hosts, the presence of additional servers, or the modification of the existing network equipment, which hinders adoption.

Customer Edge Switching (CES) [32] proposes a new architecture that, like NAT, places hosts in private networks behind edge nodes. CES uses Customer Edge Traversal Protocol [33] to grant policy based end-to-end connectivity between hosts by tunnelling data edge-to-edge regardless of the core technologies, requiring no changes to either hosts or protocols.

III. Realm Gateway 64

In Private Realm Gateway (PRGW) [34] we presented the algorithm namely Circular Pool of Public Addresses as a solution for traversing NATs without modifications to either hosts or protocols. The algorithm is based on name resolutions and allows arbitrary legacy Internet hosts to unilaterally initiate connections to servers located in the private address space without requiring complex NAT traversal mechanisms. PRGW features a stateful NAT44 architecture with address and port-dependent mapping and filtering [35] that contributes to the paradigm of communications due to the depletion of public IPv4 addresses. However, the Internet keeps growing and IPv6, reluctantly at first, continues to pierce the IPv4 networking hegemony; IPv6 is coming, slow but steady.

Realm Gateway 64 (RGW64) is a proposal for the co-existence of IPv4 and IPv6. RGW64 features a number of mechanisms for enabling technology agnostic communications between end hosts and enhances the algorithms devised in PRGW in order to boost scalability and improve the security.

We focus on developing algorithms that reutilize widely-adopted protocols and procedures to contribute to a better utilisation of the current technology. These algorithms can be applied at the boundaries of the customer networks without requiring modification or software patches to the end devices. For the data-path, our solution uses either a dedicated Python implementation or OpenvSwitch. However, the analysis of the data-path is intentionally excluded from the scope of this paper.

A. Stateless vs. Stateful implementation

As PRGW is stateful and tightly coupled with DNS, we deemed appropriate for RGW64 to leverage a modified stateful version of DNS64 [36] and NAT64 [37]. The RGW64 is therefore implemented as a middle-box; the DNS64 provides name resolution and address synthesis whereas the NAT64 performs packet forwarding between the different realms as well as the protocol translation. The state information is shared between the DNS64 and the NAT64, thus allowing optimal decision-making and a very high degree of IPv4 address reuse. A stateless model would require every IPv6 host to have an assigned public IPv4 address, which is no longer possible due to address exhaustion.

Table I [38] presents a comparison of the NAT64 operating modes in stateless and stateful manner.

TABLE I. NAT64 Operating Modes

Stateless NAT64	Stateful NAT64
1:1 IPv6-to-IPv4 translations	N:1 IPv6-to-IPv4 translations
No conservation of IPv4 addresses	Conservation of IPv4 addresses
End-to-end address transparency	Address overloading lacks end-to-end transparency
No state or binding per translation	Requires state and binding per translation
Mandatory IPv4-translatable-IPv6 address assignment	Arbitrary IPv6 address assignment
Requires manual or dynamic host configuration for IPv6 addressing	It can use either manual, dynamic or stateless configuration for IPv6

(PrePrint) accepted for publication in IEEE International Conference on Communications (ICC 2015),
London, UK, 8-12 June 2015.

B. Domain Name System in a nutshell

DNS provides resolution of domain names and IP addresses. The domain names are expressed as Fully Qualified Domain Names (FQDN); the IP addresses can be IPv4 or IPv6. A DNS resolution requires a record and a record type. The record consists of the FQDN or IP address to be resolved; the record type indicates the type of the resolution to be performed:

- *A*: Resolution of IPv4 addresses.
  a.foo. IN A 192.0.2.100
- *AAAA*: Resolution of IPv6 addresses.
  a.foo. IN AAAA 2001:DB8::192.0.2.100
- *CNAME*: Canonical names pointing to other domains.
  a.foo. IN CNAME another-host.foo.
- *SRV*: Service location including ports and protocols.
  _ssh._tcp.a.foo. ttl IN SRV p w 22 a.foo.

C. Enhancing the Circular Pool Algorithm

The Circular Pool is used to establish new incoming connections to hosts located in the private network. The Circular Pool contains a number of public IP addresses. Upon an incoming domain resolution to the FQDN of the private host, the algorithm reserves one of these IP addresses, generating temporary binding state for subsequent packet forwarding to the private network. The reserved IP address is released when the first data packet of the flow is received; the address can be then reutilized. To minimize the effect of caching, DNS responses are generated with a zero TTL value, so that neither DNS servers nor end hosts reuse the dynamic address for another connection.

SRV records are very versatile and provide a simple location of specific services, e.g. SIP, XMPP, etc. The SRV records can be overloaded with several entries, each one with different priorities, weight, or destination target, which is useful for load balancing. The use of SRV records would allow locally managed on-demand dynamic bindings with very high degree of address reuse and efficiency. Unfortunately, the use of SRV is limited to only a few applications with very little impact for our purposes.

In order to bridge the gap between the A and the SRV record types, in this paper we introduce the concept of Service FQDN (SFQDN), which consists of a domain name that includes the protocol and port number used by the data connection. A standard SSH service running in host A, `a.foo`, can be represented with SFQDN as `tcp22.a.foo`. It is important to understand that the utilization of SFQDN does not require changes in the hosts or the protocols; as SFQDN only refers to a domain name where alias records (CNAME) can be used.

The new version of the algorithm parses the SFQDN and creates binding state. The state ($H{:}iP_H$, $R_X{:}oP_H$, $P_{protocol}$, $T_{timeout}$) is unique and includes the IP address and port of the private host ($H{:}iP_H$), the IP address and port on the public side of the RGW64 ($R_X{:}oP_H$), the protocol ($P_{protocol}$) and the lifetime of the entry ($T_{timeout}$). Theoretically, this means that a RGW64 with a single public IP address could establish simultaneously 128K connections (2^{16} *ports x 2 protocol*). However in practice, the inbound traffic patterns will be a mix of FQDN and SFQDN respectively. This produces a negative effect on the efficiency of the system, provided that an address allocated via FQDN cannot be reutilised until release, not even by SFQDN.

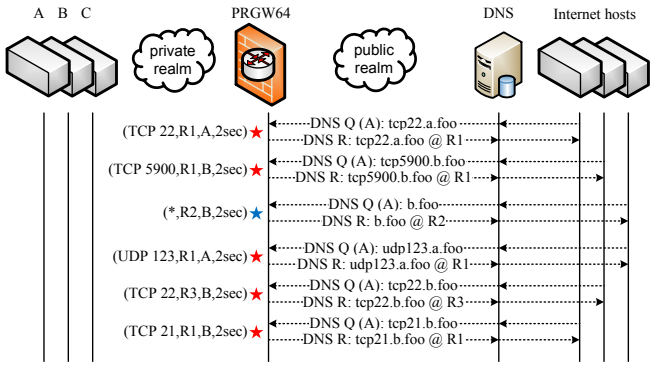

Fig. 1. Circular Pool Address Allocation

Figure 1 depicts the scenario where a RGW64 device interconnects a customer network with a public network; the RGW64 has a pool of 3 public IPv4 addresses for connecting to the Internet (R_1-R_2-R_3). The figure illustrates the operation of the allocation algorithm when Internet hosts attempt to initiate connections to the private hosts (A/B/C) and no data connections have yet been established. The RGW64 receives a SFQDN DNS query for an SSH connection with host A at `tcp22.a.foo` and the public address R_1 is allocated; following a new request for a VNC connection with host B at `tcp5900.b.foo` that reuses the R_1 address. Thereafter, a generic FQDN request for host B at `b.foo` allocates the entire public address R_2, which cannot be reused by another connection until it is released. Then, a request for a NTP connection for host A arrives at `udp123.a.foo` to what the algorithm allocates, once again, the address R_1, thus conserving the public addressing. Next, a request for host B at `tcp22.b.foo` allocates a new address R_3 provided that R_1 is already occupied for that combination of port/protocol and that R_2 cannot be reused and is still allocated. Finally, a request for host B at `tcp21.b.foo` reuses the address R_1 as it is still available for the combination of port/protocol required. Inter-realm data packets are forwarded between realms by the RGW64 on a stateful manner. A packet that originates in the public Internet and does not belong to any ongoing connection is matched to the temporary binding state of the Circular Pool and forwarded towards the private network, releasing the temporary state and creating a new connection.

D. Address synthesis from DNS records

The synthesis of IP addresses is paramount for IPv4/IPv6 interworking. The challenge lies in representing IPv6 addresses in the IPv4 address space, provided their differences in size. We refer to these synthetic addresses as *proxy addresses*. In Customer Edge Switching [32] we determined that it was possible to communicate between address realms across IP and non-IP tunnels. At a host level, the remote IPv4 endpoints are represented with an arbitrary proxy address, which is used for sending and receiving the data of a specific connection.

In the IPv4 addressing scheme, the proxy addresses are allocated from the private address space [5]. With a stateful allocation, scalability is guaranteed as it allows about 18 million active connections per host, at any time. In IPv6, however, the proxy addresses are represented using the Unique Local Address (ULA) [39]. An ULA address uses the prefix `fc00::/8` and a 40-bit pseudo-randomly generated global-ID to avoid collisions.

(PrePrint) accepted for publication in IEEE International Conference on Communications (ICC 2015),
London, UK, 8-12 June 2015.

Because the IPv6 ULA is not guaranteed to be globally unique, it can be generated locally. In addition, RGW64 cannot use the Well-Known prefix `64:ff9b::/96` [40] due to the fact that it must not be used to represent non-global IPv4 addresses.

E. IPv4 / IPv6 Communicating Scenarios

RGW64 focuses on the communication scenarios between private and public networks as defined by IETF in the Framework for IPv4/IPv6 Translation [22]. These scenarios are represented in Figure 2.

1) IPv6 network and IPv4 Internet

Represented in Fig.2 as case #1, this communication model covers the scenarios 1 and 2 of [22]. This is typical for the case where a single stakeholder creates a green-field IPv6 deployment and wishes to remain connected to the IPv4 Internet. (1) Outbound: the hosts in the customer network issue IPv6-AAAA name resolutions; the DNS64 resolves or falls back to IPv4-A resolution in case of failure. The DNS64 synthesizes an IPv6 ULA proxy address and creates binding state for the NAT64. (2) Inbound: connections are established by the Circular Pool [34] and require the allocation of an IPv6 ULA proxy address to represent the Internet IPv4 host.

2) IPv4 network and IPv6 Internet

Represented in Fig.2 as case #2, this communication model covers the scenarios 3 and 4 of [22]. This is the case of a constrained legacy IPv4 network that requires communicating with the global IPv6-only Internet. (1) Outbound: the DNS64 synthesizes IPv4 private proxy addresses to represent the IPv6 Internet hosts and generates binding state for the NAT64. (2) Inbound: although they could be established by the Circular Pool, we assume the IPv4-only hosts already have IPv6 addresses assigned in the middle-box; therefore a simple address translation 6to4 is required.

3) IPv4 network and IPv6 network

Represented in Fig.2 as case #3, this communication model covers the scenarios 5 and 6 of [22]. Communication is granted by DNS64 for proxy address allocation and the NAT64 that performs the protocol translation and packet forwarding.

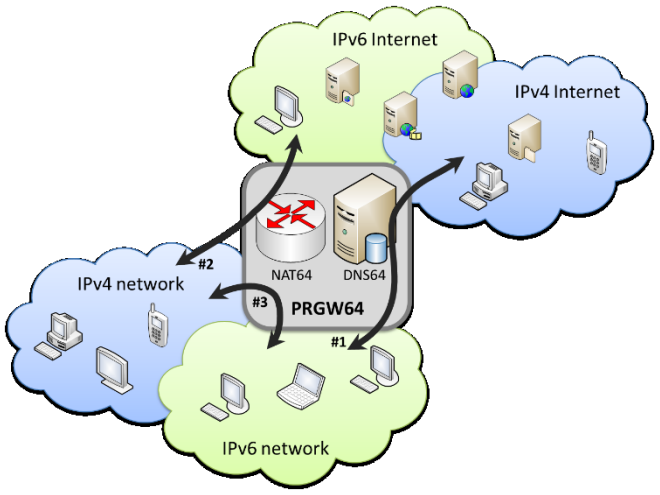

Fig. 2. IPv4 / IPV6 Communicating Scenarios

IV. Evaluation

We continued the development of the PRGW implementation in Python in order to test and validate the correctness of the algorithms defined for RGW64. We categorize the results attending to: (1) connectivity, (2) scalability and (3) security.

A. Evaluation of the connectivity

Table II summarizes the connectivity tests conducted in our prototype in order to evaluate the compatibility of RGW64 with most common protocols and application. The results obtained are in accordance with the IETF document on *IPv6-Only Experiences* [30]; where an IPv6 green-field is deployed and connected to the public Internet via a stateful NAT64 device developed by Ericsson. The authors [30] tested a variety of protocols and applications concluding that IPv6-Only networks served by NAT64 are feasible today. In addition, they attributed connectivity issues to the presence of IPv4 literals in signalling messages and non-IP agnostic application code. However, these issues can be addressed with the corresponding Application Layer Gateway (ALG) [41]. A good practice for protocol design is to leverage FQDNs instead of IPv4 literals as the end-point identifier to avoid troubles caused by middle boxes. RGW64 also includes an ICMP64 ALG that follows the directives of the IETF for *IP/ICMP Translation Algorithm* [42].

TABLE II. Protocol Compatibility

Application in realm		Protocol	Direction	Result
Private	**Public**			
Netcat 4/6 client/server	Netcat 4/6 client/server	TCP & UDP	In & Out	Success
Ping/Echo 4/6	Ping/Echo 4/6	ICMP	In & Out	Success ALG
Traceroute 4/6	Traceroute 4/6	ICMP Error	In & Out	Success ALG
NTP client 4/6	NTP server 4/6	UDP	Out	Success
SSH 4/6 client/server	SSH 4/6 client/server	TCP	Both	Success
HTTP client 4/6	HTTP server 4/6	TCP	Outgoing	Success
HTTP server 4/6	HTTP client	TCP	Incoming	Success Proxy
FTP 4/6 client/server	FTP 4/6 client/server	TCP	Both	Untested ALG
SIP client/server	SIP client/server	UDP	Both	Untested ALG
Skype	Skype	TCP & UDP	Both	Fail

B. Evaluation of the scalability

We measure the scalability based on the number of new connections that can be established per unit of time with an acceptable level of success at the first DNS request. We see no fundamental differences in the operations required when it comes to synthesizing DNS responses compared to other DNS64 implementations, provided the outbound connections solely rely on DNS64/NAT64 standard procedures. We do, however, acknowledge a possible bottleneck when leveraging the Circular Pool for creating inbound connections.

We introduced the concept of *service time* $T_{service}$, as the time duration that a public IP address R_X is reserved for establishing an inbound connection. $T_{service}$ spans the time from the creation of the temporary binding state until the first data packet of the connection is received, when R_X is released and returned to the pool. The upper bound of the system measures the number of

(PrePrint) accepted for publication in IEEE International Conference on Communications (ICC 2015), London, UK, 8-12 June 2015.

new connections established per unit time (η); the result is entirely based on the size of the address pool N_{IP} and the $T_{service}$. For FQDN the performance can be calculated as shown in Eq. 1.

$$\eta = \frac{Nip}{Tservice} \quad (1)$$

In RGW64 we introduce the concept of SFQDN boosting the address reuse for accepting new connections. The allocation is, however, tightly coupled with the service distribution of the incoming traffic. The algorithm has the potential to allocate the entire port number space (2^{16}) for the supported transport protocols N_{proto} (TCP and UDP). To get a more accurate upper bound we define N_{port} as the number of ports used/admitted in the private network. Eq. 2 represents the upper bound for the number of admitted connections per time unit when SFQDN is used for all connections.

$$\eta' = \frac{Nport\ x\ Nproto\ x\ Nip}{Tservice} \quad (2)$$

We developed a simulator to verify the feasibility of the new algorithm and avoid the computing limitations of the actual prototype. The test scenario comprises a RGW64 simulator node, an address pool of 3 public IP addresses, and a host to initiate ~~the~~ inbound connections. The network delay between the nodes is artificially generated whereas the maximum number of DNS reattempts is set to 4. We have designed a number of different fictional traffic patterns in order to present the reader with a comparison and the effects of different traffic distributions. In this paper, we present the following:

- Test 1: The 100% of traffic is generated using the FQDN of the destination host. The allocated IP address cannot be reused until the data connection has been established.
- Test 2: A 50% of traffic is generated using FQDN and the remaining 50% uses SFQDN. The SFQDN traffic is geometrically distributed in five services contributing to higher address reuse.
- Test 3: This is a variant of Test 2 with 50% of FQDN traffic and the remaining 50% modelled with SFQDN. The SFQDN traffic is geometrically distributed in two series of five services each, thus resulting in a wider distribution of traffic than the previous scenario.
- Test 4: The 100% is modelled as SFQDN, distributed in two geometric series of five services each as in Test 3.

Figure 3 shows the results after stressing the simulator with a network delay of 50 ms and a constant load of 60 connections per second. The traffic load was generated following an exponential distribution. The figure reveals that with the new algorithm in place there are great differences in the success rate when the traffic offered is initiated with an SFQDN query compared to the traditional FQDN. Figure 4 represents in detail the proportion of failing connections depending on the initiation method and traffic shapes of test cases 2 and 3. The figure reveals that an SFQDN connection attempt is more likely to succeed compared to a standard FQDN, due to the address reuse being more scalable with SFQDNs than with FQDNs. When using all SFQDN traffic, the upper bound of the scalability is determined by the most popular service.

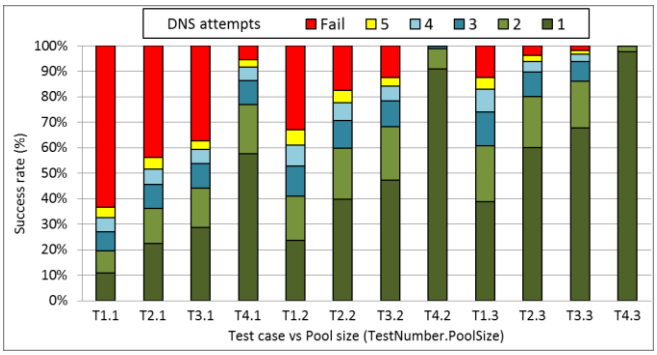

Fig. 3. Traffic distribution impact on performance

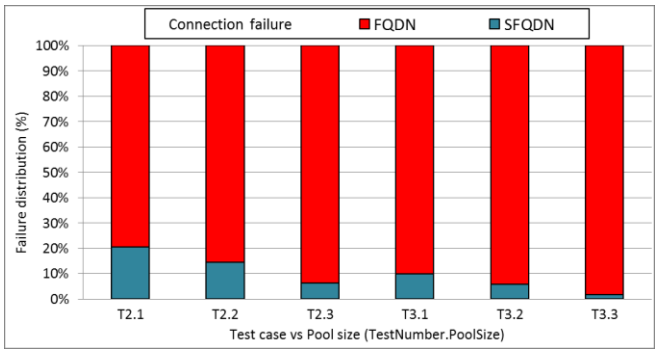

Fig. 4. Connection failure distribution FQDN vs SFQDN

C. *Evaluation of the security*

Private Realm Gateway presents a system architecture that does not introduce any explicit security vulnerability compared to the current Internet model and the management of incoming connections. These considerations are based on: (1) the private network and its hosts remain *hidden* at plain IP level, being only accessible via FQDN thus allowing fine-grained resource allocation; (2) the Circular Pool prevents direct exposure of a host compared to traditional port forwarding in NATs. This section only introduces a general overview of the security and the mechanisms devised to protect the system. A more detailed analysis of the security is the focus of another study [43].

Without the proper counter measures our architecture would be vulnerable to a number of security threats. These include mainly: (1) DoS/DDoS to the Circular Pool algorithm, which exhausts the available resources and prevents from creating new connections; and (2) connection hijacking, which causes attackers, potentially botnet-controlled networks, to steal previously allocated connections to legitimate hosts and leak malicious traffic into the private network. We developed the following mechanisms to address these issues [43]:

1) *DNS exploitation/abuse/misuse*

These attacks aim to exhaust the Circular Pool, allocating all the available addresses, thus pushing the pool to the blocking state that impedes the system from creating new inbound connections. To this end, we classify DNS servers (querying DNS clients) according to the service level agreement (SLA) and track record in white, grey and black lists. The white-list includes trusted servers with specific SLA; the black list includes banned servers to which no service is offered; the grey list conceptually includes every other server.

(PrePrint) accepted for publication in IEEE International Conference on Communications (ICC 2015), London, UK, 8-12 June 2015.

These lists are dynamically maintained based on the server behaviour; it is possible to promote and demote servers between different lists, i.e. a white server can be demoted to grey if the SLA is repeatedly exceeded or temporarily black listed in the event of an attack. White listed servers are bound to a set of pre-conditions prior to establishing a *contract* with a remote ISP. Among them, we highlight ingress filtering and DNS over TCP. When an ISP performs ingress filtering, the spoofed traffic is blocked, so the remaining traffic can be attributed to the ISP customers i.e. *evidence* of misbehaviour can be collected. Introducing DNS over TCP provides a security advantage against spoofed DNS traffic over UDP, since the TCP handshake will never complete with spoofed sources across the Internet.

Additionally, it would be possible to use DNS extensions [44] between DNS servers in order to carry very valuable information such as the originating client IP address. The ability to track connection requests from remote users enables better decision making in the Circular Pool algorithm. Abusing hosts can be easily tracked and punished whereas well behaved hosts benefit from better fairness.

2) Inherent Internet vulnerabilities

These vulnerabilities constitute the most common threats on the Internet nowadays. These include address spoofing, host scanning or DoS/DDoS originated from botnet-controlled networks. Although our architecture does not seem particularly vulnerable to these threats we have devised some solutions that contribute to a safer Internet.

Connection hijacking is a concern. Security is at risk if a non-legitimate host can claim the temporary connection state. This is addressed with a TCP handshake including a cookie; therefore guaranteeing that a connection is established with a non-spoofed host. The Realm Gateway uses a TCP relay for the subsequent TCP flow. Additional heuristic algorithms look for mismatching packets, packets that do not belong to any ongoing connection, to trigger alerts and search for evidence of anomalous patterns, i.e. bot controlled networks. It can be argued that the introduction of SFQDN, possibly hidden by a CNAME record, hardens the security by making it more difficult for an attacker to hijack a connection.

Network planning can have a positive effect on security if traffic originating from different networks could be routed via different network interfaces. However, this would result in smaller and less efficient pools of addresses, which becomes an issue that needs to be solved at the network provisioning stage.

V. DISCUSSION

The connectivity results are encouraging. There were only some issues with Skype, yet fixable. Compatibility with NAT-friendly application using TCP, UDP or ICMP is fully achieved. On the other hand, protocols like FTP or SIP require additional Application Layer Gateways. The case of FTP and NAT64 has already been addressed [45]. For SIP, we believe the modifications required in our SIP ALG [41] are not extensive. Unfortunately, we could not find a suitable IPv6-only SIP application for testing; albeit we expect successful operations.

The scalability in RGW64 is dependent on the incoming traffic patterns, FQDN vs SFQDN, and inter-arrival times. Through experimentation we know that order of the DNS queries, the popularity distribution of SFQDN services, the exact arrival process and the number of addresses in the pool have an impact on the exact numeric result. Nevertheless, SFQDN constitutes and improvement over the previous FQDN model and only requires changes in the naming scheme, applications do not need to be changed. For simplicity, security and deployment it is also possible to hide the complexity of the SFQDN with an alias name using the CNAME records of DNS. We intend to carry on addressing this question in future work.

VI. CONCLUSIONS

In this paper we have presented a stateful transition mechanism for IPv6 named Realm Gateway 64 (RGW64) as a solution that does not require additional third-party elements or changes in hosts or application. The RGW64 implements the DNS64/NAT64 functionality, relying on protocol and address translation at the border of address realms. Our solution introduces an innovative way of synthesizing DNS records for establishing communications between IPv4 and IPv6 hosts, which can be located in either customer or public networks.

The Realm Gateway offers the possibility for an Internet host to unilaterally initiate connections to servers located in private address space without NAT traversal mechanisms. RGW64 features a new allocation algorithm of the Circular Pool for enabling inbound connections to the hosts. The scalability and the security of our solution are enhanced with the addition of service descriptors to domain names. The identification of the service in the destination host grants the RGW64 a higher address reuse of public IP addresses helping with the address exhaustion problem. In particular, the concept of Service FQDN improves scalability so that if all traffic uses SFQDN, the most popular application protocol determines the upper bound of the number of new connections per time unit. The arrivals of less popular protocols has no impact on the upper bound because the temporary state can reuse the addresses previously allocated.

Our results reveal that the Circular Pool scales well to reasonable arrival rates of flows but experiences difficulties with extremely high rates of arrivals of new flows. The traffic patterns and distribution of new flows have a direct effect on the performance. RGW64 is able to accommodate a large number of single-stack servers in private realms. However, we recommend using dual stack solutions in servers that are planned to serve very high arrival rates.

Several contributions have been made to the scope of security [43]. The techniques presented focus on tackling attacks directed to the DNS system, DoS/DDoS attacks to the Realm Gateway and also includes mechanisms for detecting address spoofing. All these methods contribute to a safer architecture.

The deployment of Realm Gateway 64 focuses primarily on customer or SOHO networks. RGW64 constitutes a suitable solution for a service provider that gradually initiates the migration towards IPv6, one network at a time, with the replacement of the existing NAT boxes. Realm Gateway 64 continues our efforts of developing scalable algorithms that improve the Internet access to customer networks and increases the security of the communications across the Internet; all of these without requiring changes to either hosts or applications.

(PrePrint) accepted for publication in IEEE International Conference on Communications (ICC 2015),
London, UK, 8-12 June 2015.

REFERENCES

[1] ICANN, "Available pool of unallocated IPv4 internet address now completely emptied," Press release, 3 Feb. 2011 .

[2] APNIC, "Two /8s allocated to APNIC from IANA," 1 Feb. 2011 [Online]. Available: https://www.apnic.net/publications/news/2011/delegation.

[3] ITU-T Statistics. Internet: http://www.itu.int/ITU-D/ict/statistics, [Accessed July 24, 2013].

[4] P. Srisuresh and K. Egevang, *Traditional IP Network Address Translator (Traditional NAT),* IETF RFC 3022, Jan. 2001.

[5] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. d. Groot and E. Lear, *Address Allocation for Private Internets,* IETF RFC 1918, Feb. 1996.

[6] M. Holdrege and P. Srisuresh, *Protocol Complications with the IP Network Address Translator,* IETF RFC 3027, Jan. 2001.

[7] L. Daigle and IAB, *IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation,* IETF RFC 3424, Nov. 2002.

[8] J. Rosenberg, R. Mahy, P. Matthews and D. Wing, *Session Traversal Utilities for NAT (STUN),* IETF RFC 5389, Oct. 2008.

[9] R. Mahy, P. Matthews and J. Rosenberg, *Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),* IETF RFC 5766, Apr. 2010.

[10] J. Rosenberg, *Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,* IETF RFC 5245, Apr. 2010.

[11] Cisco Systems. "IPv6 statistics." Internet: http://6lab.cisco.com/stats/, [Accessed March 24, 2014].

[12] Google. "IPv6 statistics." Internet: http://www.google.com/intl/en/ipv6/statistics.html, [Accessed March 24, 2013].

[13] E. Nordmark and R. Gilligan, *Basic Transition Mechanisms for IPv6 Hosts and Routers,* IETF RFC 4213, Oct. 2005.

[14] D. Farinacci, T. Li, S. Hanks, D. Meyer and P. Traina, *Generic Routing Encapsulation (GRE),* IETF RFC 2784, Mar. 2000.

[15] B. Carpenter and K. Moore, *Connection of IPv6 Domains via IPv4 Clouds,* IETF RFC 3056, Feb. 2001.

[16] W. Townsley and O. Troan, *IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) -- Protocol Specification,* IETF RFC 5969, Aug. 2010.

[17] F. Templin, T. Gleeson and D. Thaler, *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP),* IETF RFC 5214, Mar. 2008.

[18] C. Huitema, *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs),* IETF RFC 4380, Feb. 2006.

[19] G. Tsirtsis and P. Srisuresh, *Network Address Translation - Protocol Translation (NAT-PT),* IETF RFC 2766, Feb. 2000.

[20] E. Nordmark, *Stateless IP/ICMP Translation Algorithm (SIIT),* IETF RFC 2765, Feb. 2000.

[21] C. Aoun and E. Davies, *Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status,* IETF RFC 4966, July 2007.

[22] F. Baker, X. Li, C. Bao and K. Yin, *Framework for IPv4/IPv6 Translation,* IETF RFC 6144, Apr. 2011.

[23] Microsoft. "Forefront UAG DirectAccess deployment guide." Internet: http://technet.microsoft.com/en-us/library/dd857320.aspx, Feb 1, 2011 [Accessed March 15, 2014].

[24] Cisco Systems. "Cisco ASR 1000 Series Aggregation Services Routers." [Accessed March 15, 2014].

[25] Juniper. "Stateful NAT64 Overview." Internet: http://www.juniper.net/techpubs/en_US/junos10.4/topics/concept/ipv6-nat64-ipv4-depletion-overview.html, Aug 11, 2010 [Accessed March 15, 2014].

[26] Ecdysis. "open-source implementation of a NAT64 gateway." Internet: http://ecdysis.viagenie.ca/, [Accessed March 15, 2014].

[27] TAYGA. "Simple, no-fuss NAT64 for Linux." Internet: http://www.litech.org/tayga/, [Accessed March 15, 2014].

[28] Internet System Consortium. "BIND." Internet: http://www.isc.org/, [Accessed March 15, 2014].

[29] G. Lencse and S. Repas, "Performance analysis and comparison of the TAYGA and of the PF NAT64 implementations," in *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on,* pp. 71-76, 2013.

[30] J. Arkko and A. Keranen, *Experiences from an IPv6-Only Network,* IETF RFC 6586, Apr. 2012.

[31] D. Farinacci, V. Fuller, D. Meyer and D. Lewis, *The Locator/ID Separation Protocol (LISP),* IETF RFC 6830, Jan. 2013.

[32] R. Kantola, "Implementing Trust-to-Trust with Customer Edge Switching," in *Proc. 24th IEEE International Conference on Advanced Information Networking and Applications Workshops (AINA 2010),* pp. 1092-1099, Perth, Australia, 20-23 Apr. 2010.

[33] H. Kabir, R. Kantola, and J. Llorente, "Security Mechanisms for a Cooperative Firewall," in in Internatinal Symposium on Cyberspace Safety and Security (CSS), Paris, Aug. 2014.

[34] J. Llorente, R. Kantola, N. Beijar and P. Leppäaho, "Implementing NAT Traversal with Private Realm Gateway," in *Proc. IEEE International Conference on Communications (ICC 2013),* pp. 2174-2179, Budapest, Hungary, 9-13 June 2013.

[35] F. Audet and C. Jennings, *Network Address Translation (NAT) Behavioral Requirements for Unicast UDP,* IETF RFC 4787, Jan. 2007.

[36] M. Bagnulo, A. Sullivan, P. Matthews and I. v. Beijnum, *DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers,* IETF RFC 6147, Apr. 2011.

[37] M. Bagnulo, P. Matthews and I. v. Beijnum, *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers,* IETF RFC 6146, Apr. 2011.

[38] Cisco Systems, "NAT64 technology: Connecting IPv6 and IPv4 networks," white paper, Tech. Rep. C11-676278-02, Apr. 2012 .

[39] R. Hinden and B. Haberman, *Unique Local IPv6 Unicast Addresses,* IETF RFC 4193, Oct. 2005.

[40] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair and X. Li, *IPv6 Addressing of IPv4/IPv6 Translators,* IETF RFC 6052, Oct. 2010.

[41] P. Leppäaho, N. Beijar, R. Kantola and J. Llorente, "Traversal of the Customer Edge with NAT-Unfriendly Protocols," in *Proc. IEEE International Conference on Communications (ICC 2013),* pp. 1526-1531, Budapest, Hungary, 9-13 June 2013.

[42] X. Li, C. Bao and F. Baker, *IP/ICMP Translation Algorithm,* IETF RFC 6145, Apr. 2011.

[43] H. Kabir, J. Llorente and R. Kantola, "Securing the Private Realm Gateway," 2014, unpublished.

[44] J. Damas, M. Graff and P. Vixie, *Extension Mechanisms for DNS (EDNS(0)),* IETF RFC 6891, Apr. 2013.

[45] I. v. Beijnum, *An FTP Application Layer Gateway (ALG) for IPv6-to-IPv4 Translation,* IETF RFC 6384, Oct. 2011.

(PrePrint) accepted for publication in IEEE International Conference on Communications (ICC 2015), London, UK, 8-12 June 2015.