Implementing NAT Traversal with Private Realm Gateway

Jesús Llorente Santos, Raimo A. Kantola, Nicklas Beijar and Petri Leppäaho
Department of Communications and Networking
Aalto University
Helsinki, Finland
{jesus.llorente.santos, raimo.kantola, nicklas.beijar, petri.leppaaho}@aalto.fi

***Abstract*—A Network Address Translator (NAT) allows hosts in a private address space to communicate with servers in the public Internet. There is no accepted solution for an arbitrary host in the Internet to initiate a communication with a host located in a private address space despite the efforts to create one. This paper proposes to replace NATs with a new concept we call Private Realm Gateway (PRGW). Private Realm Gateway creates connection state based on incoming DNS queries towards the hosts in the private network. The state gives means for the private network operator to apply elaborate access control to packet flows arriving from the Internet to the private network. PRGW does not require changes in the hosts and the deployment can take place one network at a time. The paper shows that the PRGW is most applicable for connecting mobile and other wireless hosts to the Internet.**

Keywords- Network Address Translator; NAT traversal; DNS; reachability.

I. INTRODUCTION

The rapid growth of the Internet has made globally unique IP addresses a scarce resource. In February 2011 IANA announced the allocation of the last /8 address pools in favor of APNIC resulting in the depletion of its own address pool [1]. The adoption of mobile broadband also increases the demand for addresses. In 2008 the penetration of mobile broadband already outgrew the fixed broadband in subscriptions. By the end of 2011, ITU-T statistics [2] shows that around half of the Internet users were using mobile broadband.

The industry's reaction to the limitations of the IPv4 protocol was the adoption of middle boxes such as NATs. The boxes introduced the separation of private and public realms [3][4] allowing hosts in a private network to connect with the Internet by sharing a public IP address. This also introduced the so called *reachability problem* that prevents a public host from initiating, unilaterally, a connection with a private host that is located behind a NAT.

The extensive deployment of NATs in today's networks together with the slow penetration of IPv6 made the official institutions take a hand in the matter. The IETF has categorized the different NAT behaviors [5] and defined a set of recommendations on how to traverse NATs. The result is the UNilateral Self-Address Fixing (UNSAF) architecture [6]. This model comprises the STUN [7] protocol for determining the NAT behavior between the private and the public devices, the TURN [8] protocol for relaying information between them and the ICE [9] protocol that combines the STUN and TURN techniques to achieve optimum connectivity even under very challenging network conditions.

The benefit of UNSAF is that it operates under virtually any kind of network providing dynamic discovery of the optimal route and ultimately guarantees connectivity. If necessary, the information may be relayed through a third party TURN server if no direct communication is possible between the devices. On the other hand, there are several drawbacks related to this solution, especially when it is used in conjunction with mobile devices. (1) It clutters the applications with additional code that has no relation with the intended task, therefore increasing the complexity and the memory requirements. (2) It forces a mobile device to wake up at regular intervals and perform keep-alive operations to prevent binding expiration on the NAT device, consuming energy and draining the battery of the mobile device. (3) It introduces a significant delay during the session setup of a communication because the application must wait for the expiration of timeouts before it can discard an option; some scenarios may even take up to 25 seconds [10] with P2PSIP for a call establishment thus reducing the perceived quality of the system.

Alternative methods that enable traversal of NATs are UDP Hole Punching and Application Layer Gateways (ALG). An ALG is located on a NAT/firewall device and its goal is to enable end-to-end communication by performing protocol specific operations over the user data. The major disadvantages of the ALGs are that they are completely protocol specific and utterly dependent of the NAT/firewall implementation.

The purpose of this paper is to propose an architecture that makes hosts in the private address space globally reachable. The architecture does not require registration, keep-alive signaling or additional changes in the host. The scenario is optimal for mobile devices that sleep most of the time, which contributes to battery saving whilst staying reachable regardless of user interaction.

The paper is structured as follows. Section II presents the related work on the matter. Section III describes the requirements of the architecture. Section IV introduces the proposed solution. Section V covers the efficiency and

This work was partially supported by the European Celtic MEVICO project.

TABLE I. NAT TRAVERSAL INITIATED BY A GLOBAL NODE

	Required changes			
Solution	***GH***	***PH***	***NAT***	***Additional requirements***
STUN TURN ICE	No	Yes	No	STUN and TURN servers
AVES	No	No	Yes	Modified DNS server, DDNS operations, Waypoint server
NAT-f	Yes	No	Yes	DDNS operations
P2P/UDP	Yes	Yes	No	Master server
UPnP	No	Yes	Yes	
Extended RSIP	No	Yes	Yes	RSA-IP Server, DDNS operations
DPRP	No	Yes	Yes	
NATS	No	No	Yes	NTS Server

GH – Host in a Public Network, PH – Host in a Private Node

performance, Section VI presents the scalability of the architecture, Section VII addresses security issues, Section VIII introduces an experimental proof-of-concept implementation and finally Section IX concludes the paper.

II. RELATED WORK

Traversal of NAT devices has been previously studied by other researches [11] together with new architectures [12]. Hereafter we summarize the results of the proposed solutions in Table I.

Many of the analyzed solutions require additional changes in at least one or even both of the hosts involved in the communication. Most of these solutions also need additional servers or the modification of existing elements in order to support explicit signaling, the creation and maintenance of state as well as specific packet forwarding.

In order to overcome the reachability problem caused by NATs, new architectures such as Customer Edge Switching (CES) [13] have been proposed. Conceptually CES aims at placing hosts in private networks enabling connectivity with additional security features enforced by the Customer Edge Traversal Protocol (CETP) [14]. This architecture is divided into User Networks (UN) and Service Provider Networks (SPN). The user data is tunneled between CES devices conveying IDs for the host identification in the private realm. The main benefits of the architecture are that no changes are required in the hosts or the protocols enabling communication between hosts in different private realms. The Private Realm Gateway that we propose in this paper can be seen in two different ways: (1) as a component of the Customer Edge Switch or (2) a standalone solution for NAT traversal.

III. REQUIREMENTS

Let us define the requirements so that a host with a public IP address can initiate communication with a host located in a private network, and vice versa. The PRGW must (1) support end-to-end communications with existing protocols (TCP, UDP and ICMP) and applications, (2) interwork with NATs and firewalls in both the public and private realms; provide additional mechanisms for protocols that fail to operate due to the presence of NATs. The goal is (3) to design a scalable model that accommodates a large number of users with a limited amount of public IP addresses and networking equipment, (4) to enable transparent and gradual deployment motivated by technological and economic factors so that the main players and the customers benefit from it and (5) a secure architecture that protects the private network and does not expose the hosts to attacks or malicious users.

Fig. 1. Communication scenario

IV. PRIVATE REALM GATEWAY

The solution devised proposes to replace the existing NATs with a new element called Private Realm Gateway (PRGW) that performs address translation operations between a private and a public realm for both outbound and inbound traffic.

A. System architecture

The PRGW interconnects a private network with a public network similarly to a NAT. The scenario is represented in Figure 1. In addition, it provides the hosts with domain name resolution and a default gateway for the public network. On the public side, the PRGW owns a pool P_{public} of public IP addresses. There are two policies for managing the outbound address allocation for the hosts: (1) arbitrary pooling, whereas an IP address is randomly chosen from the public pool and (2) fixed pooling, whereas each host receives a predefined IP address. The system provides mechanisms to prevent clashes performing an additional port translation if necessary.

The PRGW also includes a built-in DNS server as authoritative name-server for the given zone of authority. The DNS functionality enables the resolution of domain names of the served hosts and a public host to communicate with the private network via the Fully Qualified Domain Name (FQDN) of the intended host. Consider the PRGW as authoritative name-server for the domain *.foo*.

For each connection the PRGW stores state information in the forwarding table, comprising the local IP and port (*A*:*iPA*), the outbound IP and port (*R1*:*oPA*), the remote IP and port (*E1*:*oE1*), the protocol $P_{protocol}$ and the direction *D*. In order to delete expired connection state, each connection has a timeout $T_{timeout}$ indicating the lifetime and a timestamp $T_{timestamp}$ specifying the last time used. If the connection expires, the state is cleared. The purpose of the direction field is to indicate whether the traffic has flowed in a single direction (*D=outgoing* or *D=incoming*) or in both directions (*D=bidirectional*). This field can be utilized for security purposes. The circular pool algorithm described in Section IV.C adds additional state information.

The notation used in the figures is explained in Table II.

TABLE II. NOTATION

Element	Definition
R*n*	An address *n* from the pool P_{public}
A, B	The IP address of the host A and B, respectively
iP*n*, oP*n*	The port number in the private and public realm, respectively, for a connection of host *n*
(*n*:iP*n*)	A socket consisting of an IP address and port number in host *n*
(*n*:iP*n*)>(*m*:oP*m*)	A data packet transmission from socket *n* towards socket *m*
★	Creation of waiting state on incoming DNS query
☆	Creation of connection state
●	Inbound translation modifies the *destination* IP address and port with the private host information
◉	Outbound translation modifies the *source* IP address and port with the public host information

B. Outbound connections

An outbound connection from a private host to a public host is handled in the same fashion as with NATs. Upon receiving the first packet from the private host, the PRGW performs an outbound translation and creates connection state. All outbound packets are forwarded to the public network using the connection state. Upon receiving a response from the public host, the PRGW looks up the connection state for the packet. After performing an inbound translation the packet is forwarded to the private host. Figure 2 illustrates the depicted scenario.

C. Inbound connections

PRGW utilizes a method we call *Circular Pool of Public Addresses* for accepting incoming connections. Addresses from the pool P_{public} are reserved based on incoming DNS queries to the FQDN of the private host. This method requires temporary state called *waiting state* that maps the reserved address to the private host. The waiting state has a *holding timeout* $T_{htimeout}$ specifying its validity.

A public host initiates a connection by performing a name resolution for the FQDN of the private host. Upon reception of a DNS query, the PRGW extracts a public IP address *Rn* from the pool P_{public} and sends it in the DNS response. The PRGW also creates a waiting state that maps the allocated address *Rn* to the host indicated by the FQDN. The offered address cannot be re-allocated before it is returned to the pool.

Upon receiving an incoming data packet that does not match with any existing connection state, the PRGW checks if the destination address matches with a waiting state. Connection state is created that maps the remote host to the local host obtained from the waiting state. The waiting state is cleared and the allocated public address is returned to the pool for future use. If the holding timeout expires, the waiting state is automatically cleared and the address is returned to the pool.

All packets between the public and the private hosts are forwarded by applying inbound and outbound translations based on the connection state.

Figure 3 illustrates the scenario where public hosts are able to initiate connections with private hosts behind the PRGW. The pool P_{public} consists of two addresses: R1 and R2. Note that

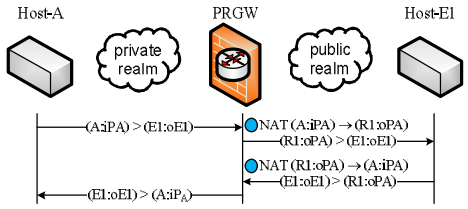

Fig. 2. Outbound connection

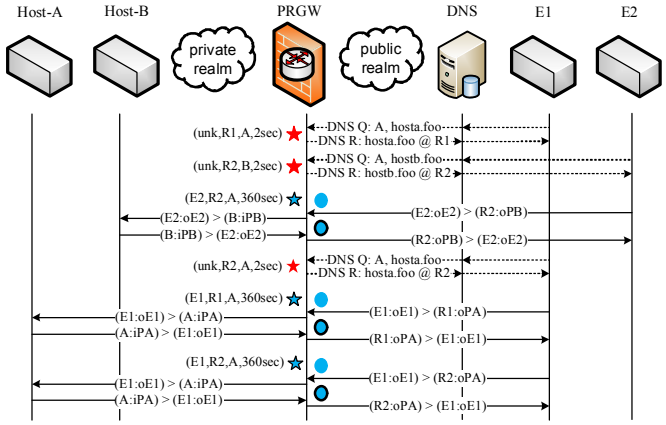

Fig. 3. Inbound connection

from E1's perspective, the private host A becomes reachable via two different IP addresses, R1 and R2.

V. EFFICIENCY AND PERFORMANCE

The system displays no restrictions regarding the number of ongoing connections. However, there is a scalability limitation that only applies to new inbound connections.

The solution is able to process a new inbound connection as long as the circular pool is not depleted and at least one address can be allocated. When the depletion of the pool occurs, the system enters a *blocking state*. During this phase the model is unable to establish inbound connections until the waiting connections are completed or are timed out. The ongoing traffic is not affected by the blocking state.

Fresh DNS queries arriving at the system during the blocking state are discarded. This action triggers the retransmission of the query in the originating host, allowing multiple chances for establishing the connection during a high load period. Retransmission is preferable to the alternative of answering the query with a message indicating server failure, name error or refusal, as this approach might develop into cache-poisoning of the DNS servers and prevent new queries towards the PRGW.

A. Efficiency analysis

The limitations of the Private Realm Gateway are tied to the size of the circular pool and the holding time $T_{holding}$ for an address. The holding time can be calculated as the time elapsed since the waiting state is created until the first data packet of the connection is received, whereas the address is returned to the pool. The holding time is dependent on the name resolution process as well as the rate of packet loss in the network. Figure 4 represents the components of the holding time in a typical scenario with a single DNS server. Assuming a constant

transmission delay D_n of the links n=1..3, the holding time can be expressed based on the Round Trip Time (RTT) as

$$T_{holding} = D_1 + D_2 + D_3 \approx 1.5 \text{ RTT}. \quad (1)$$

The holding timeout $T_{htimeout}$ must be larger than the maximum expected holding time $T_{holding}$ so that waiting state does not expire before the first data packet. On the other hand, it should be sufficiently short so that a public address is not reserved for a connection that may not occur.

Equation 2 gives the number of connections established per time unit based on the size of the pool and the holding time, from now on referred to as service time. Retransmissions and packet loss are intentionally excluded from the calculation; as a result the equation yields the upper bound of the system operating under ideal circumstances.

$$\eta = \frac{Pool\ size}{Service\ time} \quad (2)$$

The effects of packet loss are diverse. Consider Figure 4 and the following packet loss. (1) If the DNS query is dropped, the PRGW does not allocate an address and the host retransmits the query without consequences for PRGW. (2) If the DNS response is dropped after the PRGW has allocated an address, the host retransmits the query forcing the PRGW to allocate yet a new address, if available. The first address expires after the holding timeout decreasing the overall efficiency. (3) If the first data packet from the host is dropped, the PRGW keeps the address on hold for the holding timeout. If the packet is retransmitted, its admission is still subjected to the same time window created upon address allocation which may result in either acceptance or rejection from the PRGW.

Further optimization of the model is challenging since it is entirely related to the DNS resolution process and the difficulty to detect retransmitted DNS queries as DNS servers tend to allocate a new ID to retransmitted queries.

B. Experimental evaluation of efficiency

A prototype was designed in order to test the efficiency of the model and analyze the impact of the different parameters in play. The testing scenario comprises a private host, a PRGW and a public host that issues DNS queries directly to the PRGW. The delay between the PRGW and the host is artificially generated. In this case the minimum holding time is of $T_{holding} = 2D = 1$ RTT which matches with the introduced delay D. The maximum number of retransmissions is set to 4, as in most of today's operating systems. The results obtained are somewhat constrained by the available computing power, resulting in slightly lower performance results.

Figure 5 evaluates the effect of submitting a circular pool of 5 addresses with a constant offered load of 60 new connections per second towards the served private network.

Figure 6 complements Figure 5 by representing the individual success rates per DNS resolution attempt under the same network conditions with pool sizes of 3, 5 and 7 addresses.

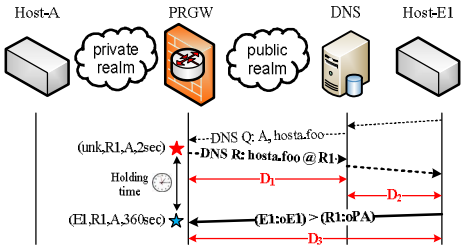

Fig. 4. Holding time in Circular Pool

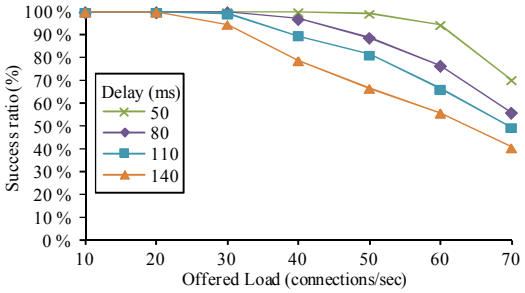

Fig. 5. Success ratio for a pool of 5 addresses

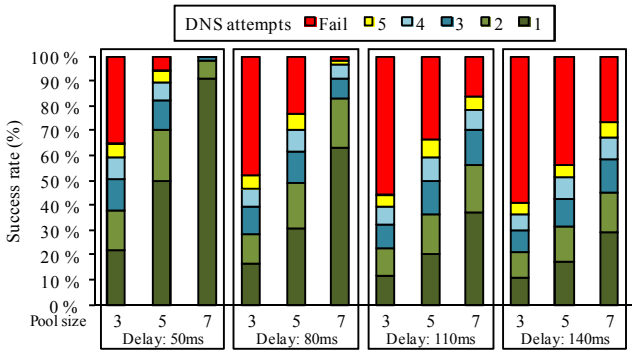

Fig. 6. Effects of delay and pool size with constant load

In Figure 7 we represent the evolution of the carried load as a function of the offered load for a circular pool of 5 addresses. Three phases can be identified. (1) Ramp-up: operation under light load with high success rate. (2) Peak-capacity: the system reaches the maximum number of accepted connections and starts to drop some requests. (3) Fade-down: the high load of connections causes congestion, DNS retransmissions are at their peak and the number of failing connection increases decreasing as well the overall efficiency.

Figure 8 presents the success ratio, comparing different models such as the theoretical upper bound, the B-Erlang and the tested results. The circular pool contains 5 addresses and the service time is fixed to 110 ms. The B-Erlang model is derived from the Erlang formula for Poisson arrivals and indicates the blocking probability of a call given certain amounts of resources without retransmissions. Figure 8 shows that the measured performance is close to the upper bound. The difference to Erlang-B can be explained by the 4 re-attempts that are typical in DNS.

VI. SCALABILITY

In order to contribute to a better understanding of the proposed model, let us introduce a case example. Consider a

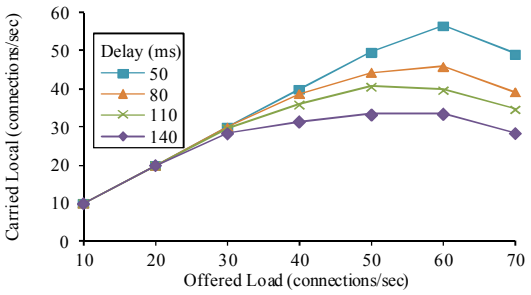

Fig. 7. Carried load vs. offered load for pool of 5 addresses

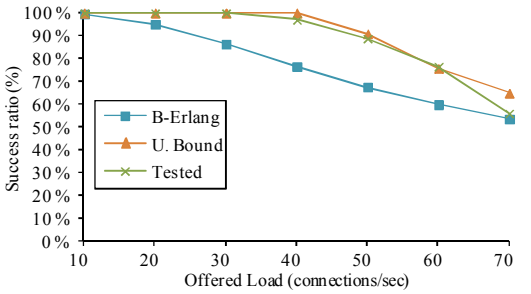

Fig. 8. Comparison of models

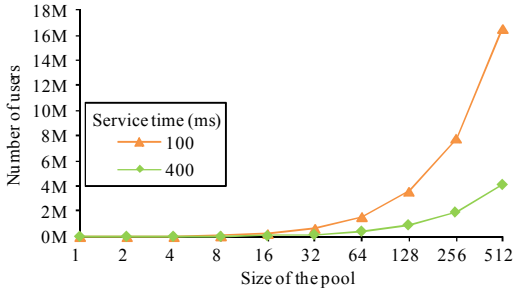

Fig. 9. System scalability based on service time

service provider provisioning a Session Initiation Protocol (SIP) based telephony service, where the customers are located behind a PRGW. The purpose is to determine the required number of public IP addresses and the number of users that it can serve during the busy hour. To that end, we use a model based on the Erlang-B formula. With a blocking probability of 0.1% the offered load is represented in Eq. 3. The parameter h corresponds with the service time, i.e. the time that is required to establish the call after the user has picked up the phone. The parameter λ indicates the arrival of calls per time unit and has been set to 1 call per hour and user. Note that the Erlang-B model does not account for retransmissions thus always provides a lower bound for a given system.

$$E = h\lambda \text{ [Erlang]} \quad (3)$$

Figure 9 represents the scalability of the system in millions of users as a function of the service time and the size of the pool. The results indicate that with a C-block of addresses it is possible to serve around 7.5 Million users. The results also give means to another type of analysis; the λ parameter indicates the number of calls per user during the busy hour, but the system only shows limitations on incoming calls, which allows us to state that the supported number of users actually doubles up to

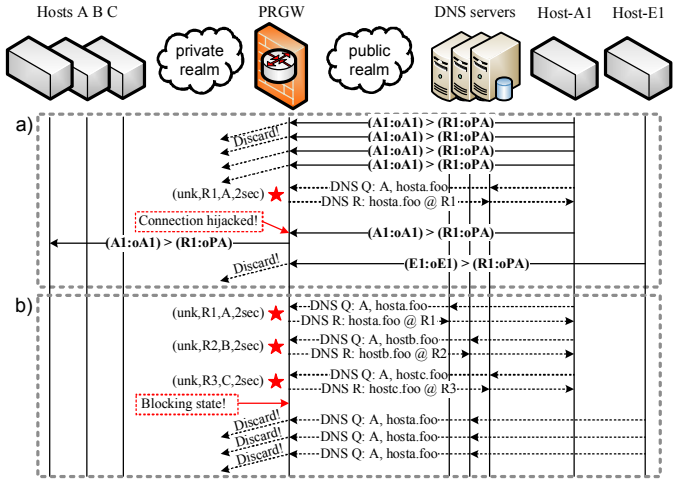

Fig. 10. Potential attacks in Circular Pool

15 million for the busy hour. The results indicate an asymptotic growth of the efficiency as the size of the address pool grows. Furthermore, it is reasonable to state that the marginal cost of adding a subscriber decreases as long as the pool grows.

VII. Security

The PRGW architecture not only does not introduce any explicit weaknesses that may endanger the hosts but it also includes additional features compared to NATs. (1) The private network is not exposed and the hosts remain *hidden* being only accessible via FQDNs. (2) The Circular Pool prevents a host from being exposed to the public network as in the case of static forwarding in NATs.

Some concerns may arise based on the premise that the proposed solution relies heavily on domain resolutions to operate successfully. The system may be exposed to misuse or abuse of the DNS service despite of the attempts of securing the architecture. We have identified scenarios that imply certain level of hazard [15].

1) Connection hijacking: Figure 10.a represents an attacker constantly sending packets to an address of the pool. The packets are dropped due to lack of forwarding state. A future allocation of this address by a legitimate host creates a waiting state which potentially could be matched by the attacker's packet resulting in connection hijacking. The countermeasure is to blacklist users whose packets repeatedly do not match any connection state. Malicious traffic can be reported to a *Trust Management System* for further actions [16].

2) *Denial of Service:* The performance is compromised if the system enters the *blocking state* as shown in Figure 10.b. An attacker could exploit the allocation policy of the Circular Pool by requesting multiple FQDNs leading to the depletion of the pool. The countermeasure is to define policies that limit the number of waiting states available for a private host based on the current allocation of the pool and the originator of the DNS query.

VIII. Experimental Proof-of-Concept

In a Python prototype we implemented a complete solution of the PRGW architecture, and tested connections using

TABLE III. PROTOCOL COMPATIBILITY

Application in realm		Protocol	Direction	Result
Private	**Public**			
Netcat client/server	Netcat client/server	TCP & UDP	Both	Success
Ping request	Ping response	ICMP	Outgoing	Success
Ping response	Ping request	ICMP	Incoming	Success
-	Ping req / Dig	DNS & ICMP	Incoming	Success
NTP client	NTP server	UDP	Both	Success
SSH client/server	SSH client/server	TCP & UDP	Both	Success
Skype	Skype	TCP & UDP	Both	Success
Traceroute	Traceroute	ICMP error	Both	ALG
HTTP client	HTTP server	TCP	Outgoing	Success
HTTP server	HTTP client	TCP	Incoming	Proxy
FTP client	FTP server	*Active mode*	Outgoing	ALG
FTP client	FTP server	*Passive mode*	Outgoing	Success
FTP server	FTP client	*Active mode*	Incoming	Success
FTP server	FTP client	*Passive mode*	Incoming	ALG
SIP client/server	SIP client/server	UDP	Both	ALG

various protocols. Each test can produce one of three outcomes: (1) *success*: the protocol worked as expected, (2) *ALG*: an application layer gateway is required for complete interoperability, and (3) *proxy*: a new element in the form of a proxy server is required to grant smooth operations. The results are summarized in Table III.

The tests reveal that most of the applications are able to successfully traverse the PRGW in both directions without difficulties. The prototype indicated that the FTP, SIP and ICMP protocols require additional processing. The main problem of FTP and SIP lies within the protocol itself; the IP addresses conveyed within the user data are no longer valid across realms. Specific ALGs have been developed ensuring the compatibility of these protocols [15] [17].

Incoming HTTP connections towards the private network are enabled via a third-party element; an HTTP reverse-proxy server is embedded within the PRGW. This server accepts all the incoming web traffic and redirects it based on the HTTP headers that contain the intended private host.

IX. CONCLUSIONS

The Private Realm Gateway makes nodes in a private network reachable from a public network using the FQDN as a host identifier. Unlike the UNSAF solution recommended by IETF, PRGW scales well to all types of devices, especially to mobile hosts for which it is important to preserve battery life while remaining globally reachable. For outgoing connections, the solution resembles the NAT functionality and follows the *address and port-dependent mapping and filtering* behavior [5]. The solution uses a novel method called the Circular Pool to overcome the reachability problem. Dynamic state is created upon receiving DNS queries enabling the forwarding of subsequent data towards the private network.

Private Realm Gateway provides a transparent framework that enables end-to-end communications and does not require any changes in the hosts or registrations and keep-alive of any kind. Furthermore, no additional compatibility problems are introduced compared to NATs. On the contrary, ALGs and proxy servers enable the operation of difficult protocols.

The scalability is an important feature of the architecture, enabling a large number of users to be globally reachable with just a few public IP addresses thus contributing to alleviate the address exhaustion problem of today's Internet. However, we are aware of the limitations of the circular pool solution for the case of connecting heavy duty servers with a very high level of new flow arrivals per second particularly when the connection would be processed by the circular pool.

Private Realm Gateway aims at replacing NATs and can be gradually and transparently deployed. In addition, it can complement Customer Edge Switching by providing legacy interworking and thereby enabling gradual adaptation of a model that provides reinforced security and end-to-end trust.

REFERENCES

[1] ICANN, *Available Pool of Unallocated IPv4 Internet Addresses Now Completely Emptied*, Press release, 3 Feb. 2011.

[2] ITU-T ICT Developments 2009. *Free statistics*. [Retrieved on 24 Sep. 2012] Available: http://www.itu.int/ITU-D/ict/statistics

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

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

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

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

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

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

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

[10] G. Camarillo, J. Mäenpää, A. Keränen and V. Andersson, "Reducing Delays Related to NAT Traversal in P2PSIP Session Establishments," in *Proc. IEEE Consumer Communications and Networking Conference, CCNC 2011*, pp.549-553, Las Vegas, NV, USA, 9-12 Jan. 2011.

[11] Y. Miyazaki, H. Suzuki, A. Watanabe, "Proposal of a NAT traversal system independent of user terminals and its implementation," in *Proc. IEEE Region 10 Conference, TENCON 2008*, pp.1-4, Hyderabad, India, 19-21 Nov. 2008.

[12] R. Kantola, M. Luoma and J. Manner, "Future Internet is by Ethernet," in *World Computer Congress*, Brisbane, Australia, 20-23 Sep. 2010.

[13] R. Kantola, "Implementing Trust-to-Trust with Customer Edge Switching,", *AMCA in connection with AINA 2010*, Perth, Australia, 20-23 April 2010.

[14] R. Kantola, N. Beijar, Z. Yan and M. Pahlevan, *Customer Edge Traversal Protocol (CETP)*, Work in progress, Sep. 2012, [Retrieved on 24 Sep. 2012] Available: http://www.re2ee.org/.

[15] J. Llorente, *Private Realm Gateway*, M.Sc. Thesis, Aalto University, Helsinki, Finland, Sep. 2012.

[16] Z. Yan and R. Kantola, "Unwanted Traffic Control via Global Trust Management," in *Proc. 10th IEEE Int. Conf. on Trust, Security and Privacy in Computing and Communications, TrustCom 2011,* pp.647-654. Changsha, China, 16-18 Nov 2011.

[17] P. Leppäaho, *Design of Application Layer Gateways for Collaborative Firewalls*, M.Sc. Thesis, Aalto University, Helsinki, Finland, May 2012.