

Expanding the Address space through REBEKAH-IP: An Architectural View

Bjorn Landfeldt*, Sanchai Rattananon**, and Aruna Seneviratne**

*School of Electrical and Information Engineering and
School of Information Technologies
The University of Sydney
Sydney 2006 Australia
Fax: +61 2 9351 3838
bjornl@staff.usyd.edu.au

**School of Electrical Engineering and Telecommunications
The University of New South Wales
Sydney 2052 Australia
san@mobqos.ee.unsw.edu.au, a.seneviratne@ee.unsw.edu.au

Abstract—The advent of 2.5 and 3G networks mark the beginning of the mobile Internet. The investments in these technologies will be the largest in the history of telecommunication. However, their success will largely depend on the availability of successful services and initially these will largely constitute existing services. These services all reside in the current IPv4 address space which make them inaccessible to mobile Internet terminals that will reside in the IPv6 address space.

There have been many proposals in literature for translating between the two address spaces, but as we will show, they all have shortcomings that make them unsuitable for large scale deployment in the mobile Internet. In this paper, we detail the algorithms and mechanisms of our proposal for overcoming the shortcomings of the previous proposals. We also present experimental results from early experiments that justify our implementation design decisions.

I. INTRODUCTION

When subscribers got access to second generation cellular networks, the popularity of mobility soon became evident. In many countries, the revenues from 2G networks far outweigh those from the fixed networks. In parallel, the population of Internet users world-wide has grown far beyond expectations. Together, these two services have the potential of making the most powerful platform for future communication services. The telecom industry is expecting an unprecedented growth in subscribers with the introduction of 2.5/3G networks since these systems are a realization of this platform and thus mark the beginning of the mobile Internet.

As with the fixed Internet, the success of the emerging mobile Internet will largely depend on the services that can be provided. Therefore the primary design goal is to ensure that

the infrastructure can support as wide a variety of applications and services as possible. However, the current Internet services are offered over a network, which uses IP version 4 (IPv4) for addressing and routing. It is well recognized and documented, that one of the major problems with IPv4 is its limited address space. Moreover, it is obvious that the current IPv4 address range will not be able to support the introduction of new mobile Internet terminals and therefore, the growth of these networks would suffer.

Several different ways of overcoming this problem have been proposed in the literature. The most obvious is the use of the new generation IP protocol, IPv6, which expands the address space from 32 to 128 bits. Therefore, the telecom industry has decided to use IPv6 for addressing in the rollout of 3G networks. However, IPv4 and IPv6 are not compatible, and therefore IPv6 hosts will not be able to communicate with IPv4 hosts directly. As a result, the mobile Internet offerings will initially not be able to leverage off the existing services offered in the IPv4 based Internet which is one of the major threats to the successful deployment of 2.5 and 3G networks.

One intermediate solution to this problem is to deploy an IPv4 - IPV6 address translation mechanism until the number of services in the IPv6 networks is large enough to be self sufficient.

II. EXISTING METHODS FOR EXTENDING THE ADDRESS SPACE

Over the past few years, a number of methods have been proposed for both extending the address space and for address translations [1]. However, as we will discuss in this paper, all methods proposed to date are limited and will not meet the requirements of operators and users. In addition, many existing applications will fail when these methods are deployed and under these conditions, the design criteria for new applications and protocols are very limiting [2, 5].

The existing methods for expanding the IPv4 address space are based around the concept of Network Address Translators (NATs) [3]. Several flavours of NATs have been proposed. The common feature of these different flavours is that they all hide addresses in one address realm and reuse addresses in another realm to enable communication between hosts in the two different address realms as described below.

A. Classical NAT

A classical NAT uses a set of public IP addresses it assigns to individual private nodes on a per-session basis. When a host within a private realm wants to contact a host in the public Internet, the NAT assigns one of its public addresses to the private host. The NAT then rewrites the sender address in the IP header of the outgoing packet with the assigned public address and makes the necessary changes to the checksum. Thus, to the hosts outside the private realm it appears as if the packet originated from another public domain host. On the return path, the NAT rewrites the destination address in the IP header with the private address and again makes the necessary changes to the checksum, so the packet is correctly routed within the private realm.

This method of translating between realms is simple but suffers from three drawbacks. Firstly, it does not allow a host in the public Internet to connect to a host within the private realm. Secondly, it does not scale since every time a private host wants to connect to a public realm host, a public IP address is reserved exclusively for that host's use.

Thirdly, many applications and protocols embed IP address information in IP packets payloads. Thus, when the application layer signalled address is a private realm address, despite the NAT changing the IP header, the application will read the wrong address in the payload and consequently malfunction. In order to overcome this problem it is necessary to deploy application layer gateways (ALGs) at the NAT. These ALGs need to be application specific to correctly decode the payload and substitute the private address with the appropriate public address. Therefore, ALGs are difficult to deploy and manage in public cellular networks, unless the ALG function will be limited to supporting only a small subset of selected applications.

This problem is made even worse as the application developers very often have no commercial relationship with either the vendors or the operators. Seldom is there an incentive for the application developers to develop ALGs and in most cases the developers do not have the necessary knowledge and/or skills to incorporate them into different vendors equipment. The vendors on the other hand do not develop the applications and cannot assume responsibility for maintaining ALGs for all existing and future networked applications.

There are other flavours of traditional NAT, which were designed to meet specific criteria.

B. NAT

Network Address Port Translation NAT, helps the

scalability problem by enabling multiple private realm hosts to share a single IP address. This is done by using the transport protocol port information in the translation procedure. When a host in the private realm wants to connect to a host in the public realm, the NAT assigns a free port on the public realm interface to the connection and uses the original and assigned ports together with the IP addresses for the translation. This way the NAT can share one IP address among several private hosts.

C. Bi-directional NAT

Bi-directional NAT overcomes the problem of hosts within the public realm not being able to reach hosts within a private realm through the use of an ALG in the DNS. The ALG makes it possible to understand both address types and to make translations between the two.

D. NAT-PT

Another flavour of NAT, NAT-PT, adds specific protocol translation between IPv4 and IPv6 [4]. NAT-PT operates similarly to standard NATs with the addition of a set of IPv4-IPv6 specific translation functions.

E. Realm Specific IP (RSIP)

RSIP takes a different approach than the above-mentioned flavours of NATs to provide connectivity between different realms. RSIP uses a client-server type architecture, where the server is aware of the different realms and can distinguish between the two and the clients correspond to the hosts in the private realm. The server essentially leases a public realm address to the clients so the clients will use the public realm address to construct packets. To get the packets with the public realm addresses through the private realm, clients tunnel the packets to the server. The server decapsulates the tunnel header and passes them to the public realm. Hence, RSIP makes use of public addresses for both parties when communicating between the private and public realms, and does not perform any address translation.

There are two versions of RSIP. Realm Specific Address IP RSA-IP, corresponds to classical NAT and assigns a public IP address to a single host. Realm Specific Address and Port IP RSAP-IP, corresponds to NAT and configures multiple hosts with the same public IP address and multiplexes on ports instead of IP addresses.

An advantage of this scheme is that there is no need to deploy ALGs for applications since the clients are configured with public realm addresses.

III. COMPARISON

Method	Allows public realm initiated traffic	Scalable	Requires ALG
Classic NAT	No	No	Yes
NAPT	No	Yes	Yes
Bi-directional NAT	Yes	No	Yes
Bi-directional NAT-PT	Yes	No	Yes
RSA-IP	No	No	No
RSAP-IP	No	Yes	No

Table I, a comparison of NAT solutions

Table I compares the existing methods in terms of scalability, whether they require the use of an ALG and their ability to perform public realm initiated communication, all of which are of crucial importance for a successful deployment of 2.5 and 3G networks.

As can be seen from the table, none of the existing proposals meets all these demands.

IV. REALM BASED KLUGE ADDRESS HEURISTIC-IP (REBEKAH-IP)

We have developed an architecture that overcomes the problems with the previously proposed solutions. Our proposal REBEKAH-IP integrates RSIP and Bi-directional NAT and introduces some extensions. RSIP was chosen because it eliminates the need for ALGs. This choice implies changes to the clients to incorporate an RSIP client. However, in the case of 2.5 and 3G network clients, we are at a stage of introducing new equipment that can be furnished and configured without needing to be backwards compatible. Therefore this is not a critical issue.

Functions were selected from Bi-directional NAT to enable public realm initiated communication. Thus, REBEKAH-IP contains the RSIP components as well as one ALG, namely a DNS/ALG, which distinguishes between the address spaces in the two realms.

A. Key Mechanism

The extensions address the issues associated with the RSIP server using a pool of public IP addresses that are assigned to private hosts. The address expansion is not achieved by assigning ports to hosts as in previous NAT port translation schemes. Instead, the multiplexing is achieved by finding a sender and receiver *IP address and port number combination* that is unique before a public IP address is assigned to a private host.

Communication between hosts in the Internet requires hosts to have unique IP addresses for packet routing. In addition, port numbers are used to multiplex data from individual processes residing on the same hosts. This way, processes are uniquely identified on the Internet.

Even though this scheme is deterministic and robust, it has

the disadvantage that it occupies a large range of process identifiers for each host. In fact, with two fixed IP addresses and two 16 bit port number ranges there are 2^{32} possible process identifiers occupied by two hosts even if they only communicate between two processes.

REBEKAH-IP utilises the identification space more efficiently by avoiding unique assignment of public IP addresses to individual private hosts and instead making sure that the entire combination of sender and receiver addresses and ports are unique. Therefore, two private hosts can share the same public IP address as long as they do not communicate with the same peer using the same sender and receiver port numbers.

In [6] we show how this mechanism scales extremely well in terms of number of concurrent sessions it can support and therefore avoids the restrictions of previous address based schemes while at the same time enabling almost all existing services.

B. Components

A simplistic view of REBEKAH-IP is shown in Figure I.

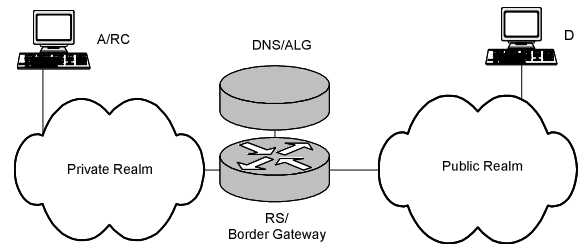


Fig I. The REBEKAH-IP Architecture

Similar to RSIP there is a server (RS) that delegates public IP addresses to private hosts on demand. In addition, there is a purpose built DNS that integrate an ALG function for interoperating in both IPv4 and IPv6 address spaces similar to Bi-directional NAT.

The hosts in the private realm implement a specific REBEKAH-IP client (RC) for address resolution and session setup similar to RSIP, and the hosts in the public realm are left unchanged for backward compatibility unless they reside inside another private (REBEKAH-IP) realm.

Below, we describe the function of the architecture in regards to both private and public realm initiated communication.

C. Private Realm Initiated Communication

Consider the following scenario, private host A wants to connect to public host D. If A has been assigned a public address already, it uses this for the communication. Otherwise, the RC at A connects to its DNS/ALG and requests a public IP address. The DNS/ALG has a pool of public addresses from which it can make a selection for the requested session. The selection is made so that there is no existing identical

combination of sender/receiver IP addresses and source and destination ports of a session in progress. The steps are as follows:

The RC connects to the DNS/ALG and requests to be assigned a public IP address. The DNS/ALG searches its database for existing sessions with a combination of sender/receiver IP addresses and ports that is not already in use. If a match is found, the sender IP address is returned to the RC. The RC then configures the host with the assigned IP address. Finally, the DNS/ALG signals the RS to set up an IP tunnel to the private host and maps it to the session so that the IPv4 packets are correctly routed through the IPv6 domain.

If a match is not found the DNS/ALG can either wait until a combination becomes free, the request times out or simply return a fail message to the RC.

There is a possible ambiguity that can occur when two private realm hosts want to connect to the same public realm host, for example an email server. If both connections are made to the same process and the two private realm hosts use the same ephemeral port, the RS cannot distinguish between the two connections. In this case, the DNS/ALG simply has to deny the second session request.

The division of tasks for the DNS/ALG and the RS is an implementation specific issue. In our implementation we have made the decisions based on signalling delays as shown in section E.

D. Public Realm Initiated Communication

When host *D* wants to connect to host *A* it starts by querying the DNS/ALG about host *A*'s IP address. If host *A* already has an assigned address, the DNS/ALG returns this address and notifies the RSIP process of the expected binding from *D*, including *D*'s address and *A*'s address. This way, the RSIP server can make the binding to the correct tunnel upon the receipt of the first packet from *D*. If *A* has not been assigned an address, the DNS/ALG employs the same algorithm as in the previous case to find an address without any binding from host *D*.

There is a possible failure scenario arising from an ambiguity in this case too. If host *D* queries the DNS/ALG for the address to two private hosts *A* and *B*, and they both have the same public address, the RS cannot distinguish between the two connections. Therefore, the DNS/ALG should also keep a list of queries from public hosts and the resulting public address responses. When a query is made, the DNS/ALG goes through the list to see if there is a collision (pending session set-up) and if so, the DNS/ALG will simply disregard the query. The queries will be removed from the list upon session set-up and also time stamped to allow repeated queries after a specified period. When the query is removed, the RSIP server is notified so it too can remove the expected binding if a connection has not been made. This way, the system maintains stability if a query was made without any corresponding connection.

It is a requirement of the RSIP server to disallow any connection attempt for which it does not have a binding

request from the DNS/ALG. This is because the public addresses are not unique to hosts, and the DNS/ALG query is the method to separate private hosts sharing a public address. The exception is, if there is an existing connection between two hosts. In this case, the incoming connection is bound to the private host with the existing connection, since the new connection most likely is a data channel corresponding to the already existing signaling channel. After the binding, the two connections are not a part of the ambiguity problem. As stated before, the situation might arise when there are two open sessions to the same host where neither session is bound to a corresponding data connection. In this case, the RSIP server has to reset the TCP connections.

In order to illustrate the function of the DNS/ALG we provide flowcharts of the decision algorithm below:

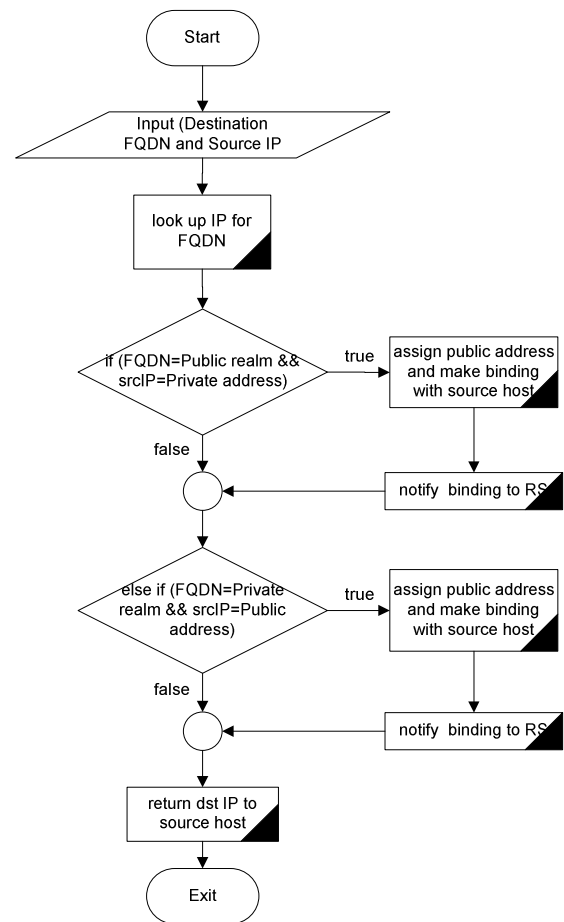


Figure II, Flowchart of DNS/ALG function.

There is a problem with DNS resolving and caching. Since DNS resolving and caching builds on the ability to bind a query by caching the address of the querying host at the DNS, it is essential that the public realm host makes the query directly and not via another DNS. To ensure this, a standard mechanism at the DNS/ALG is utilised.

When a public realm host makes a query about the IP address of a private realm host, the DNS/ALG does not return

the address directly. Instead, it returns the address of the authoritative domain server for the host's domain. The authoritative server resides on a different address than the hierarchical domain server. This way, the public realm host is forced to directly connect to the DNS/ALG on a specified IP address, and the source address of the query can be obtained by the authoritative DNS/ALG.

Both hosts and DNSs can cache results from name resolutions. If a host caches an IP address from a previous session and then the host will try to open a socket to the private client without querying the DNS/ALG first. In this case, the RSIP server will reset the session and the connection will fail.

In order to alleviate this problem, the authoritative DNS/ALG should return TTL 0 with every reply so that the querying host knows not to cache the result.

There is an issue regarding the scalability of introducing another round trip for the DNS lookup. However, a recent report [7] shows that a large portion of the Internet core traffic is due to failed DNS queries. The introduction of another round trip is small in comparison to this traffic. The report also states that caching of DNS results in a decrease the load on the root servers. In our case, we allow caching of hierarchical DNS server addresses, which are then queried without asking root servers. Then the authoritative server address is returned to the DNS resolver, which connects directly. Therefore, in our case we will not introduce a higher load on the root servers.

E. Implementation and Experimental Results

We made a prototype implementation of REBEKAH-IP to verify that the theoretical algorithms can be realised and to gain experience in how best to implement the functions. Our experimental set-up was according to Figure IV, with a DNS/ALG, a RS two private and a public realm host.

The implementation was written entirely in JAVA on Red Hat Linux PCs.

Our main concern regarding the implementation was how to separate the functions for the DNS/ALG and the RS in order to minimise the signalling delay for the session set-up. We therefore constructed three case scenarios using three separate implementations. In all scenarios below, *Host 1* attempts to establish a connection with *Host 3*.

In the first scenario, the signalling follows paths 1 and 3 in Figure IV. *Host 1* connects to the DNS/ALG and requests a public IP address for the session. The DNS/ALG implements the address selection algorithm and returns selected address to *Host 1*. *Host 1*, then contacts the RS and requests the creation of a tunnel and a binding of the session to the tunnel.

In the second scenario, the signalling path follows paths 1, 2 and 3 in Figure IV. In this scenario, the DNS/ALG signals the address binding information to the RS instead of *Host 1*. The RS then replies to the *Host 1* RC with the success of the request and the public IP address to use. Finally, the RS creates a tunnel to *Host 1*.

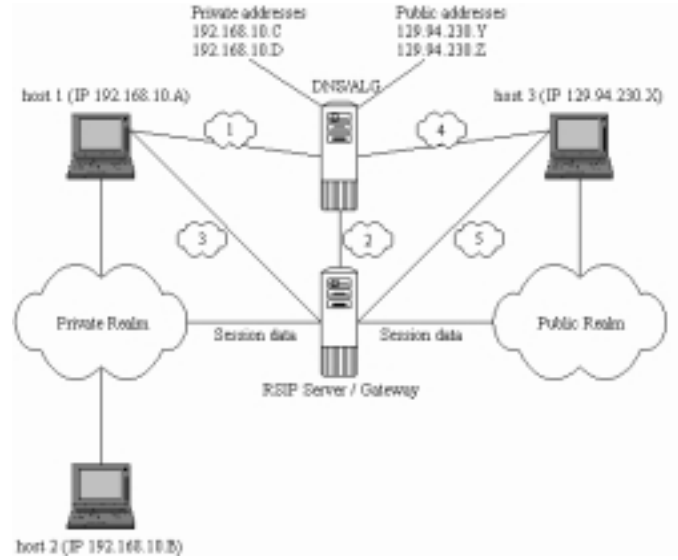


Figure IV, Prototype Implementation

The third scenario is similar to scenario 2. The difference is that the RS acknowledges the binding request from the DNS/ALG which in turn contacts the RC on *Host 1* with the public IP address and the success of the request.

For reference, we also implemented RSA-IP and ran the same connection set-up latency tests in the same environment. Table II shows the experienced latencies in ms.

Experiment	Scenario 1	Scenario 2	Scenario 3	RSA-IP
1	750	730	566	579
2	720	685	645	564
3	694	636	726	566
4	586	570	520	541
5	688	671	650	554
6	613	540	638	549
7	745	680	612	638
8	635	520	499	632
9	665	693	633	486
10	826	785	810	632
Average	692.2	651	629.9	574

Table II, Private Realm Initiated Address Assignment Delays in ms.

As expected, REBEKAH-IP introduces a slightly longer set-up delay than RSA-IP due to the more complex algorithms involved. However, the setup in scenario 3 is close in performance. The values from these experiments are indicative only and will vary drastically depending on path delays and server implementation/computational resources.

Experiment	Scenario 1	Scenario 2	Scenario 3
1	810	753	599
2	538	702	539
3	622	641	635
4	611	582	637
5	816	652	727
6	590	680	663
7	587	861	625
8	813	572	579
9	770	595	574
10	591	528	628
Average	674.8	656.6	620.6

Table III, Public Realm Initiated Address Assignment Delays in ms.

We also verified the private realm initiated experiments by conducting the corresponding experiments with public initiated communication. In these experiments we had the same three scenarios with the difference that Host 3 triggered the configuration of Host 1, by making a DNS query for its IP address. RSA-IP does not support public realm initiated communication, so there was no comparison made for these experiments.

The results of the experiments are shown in Table III. The results verify our findings from the private realm initiated experiments and also shows that the performance of private and public realm initiated address assignment are comparable. Public realm initiated address assignment delay will also depend on the path delays for the DNS lookup.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have described the architecture of a novel method for expanding the IPv4 address space and pave the way for service delivery in the emerging mobile Internet. The method, REBEKAH-IP overcomes the limitations on service provisioning that previous proposed methods impose. It is of vital importance for the development of the mobile Internet that communication platforms and technologies make it easy to provide a multitude of services that users find useful and interesting or else the success of these systems will suffer.

We have in this paper detailed the mechanisms that make up REBEKAH-IP and showed the viability of the architecture through an experimental implementation.

In the future, we intend to work on ways of minimising the ambiguity problems and to make a full investigation of the scalability of REBEKAH-IP in different scenarios. Even though the architecture is promising in the sense that it meets the immediate demands of the cellular industry, much work remains to fully detail the implications of using it and the ways it can be improved.

The address assignment algorithms presented in this paper are generic to allow compatibility with currently deployed hosts. New hosts can be shipped with special support and it is

possible to have them implement more optimised versions of REBEKAH-IP. It is possible to have the DNS/ALG/RS function decide on not only IP addresses but also source ports, thus fully avoid possible clashes between sessions. However, such a solution must remain local for specific networks and even though possible to be made interoperable with legacy Internet hosts, it is not a generic solution.

We intend to continue working on such a solution and to investigate how it will interact with legacy Internet hosts.

ACKNOWLEDGEMENTS

The authors would like to acknowledge that the original ideas of REBEKAH-IP were produced by Dr. Landfeldt at Ericsson Research Networks and Systems in Stockholm, 2001. Also, this work is partially funded by the Ericsson Research.

REFERENCES

- [1] "IP Network Address Translator (NAT) Terminology and Considerations". P. Srisuresh, M. Holdrege., RFC 2663, August 1999.
- [2] "Network Address Translator (NAT)-Friendly Application Design Guidelines". D. Senie., RFC 3235, January 2002.
- [3] "The IP Network Address Translator (NAT)". K. Egevang, P. Francis, RFC 1631, May 1994
- [4] "Network Address Translation - Protocol Translation (NAT-PT)". G. Tsirtsis, P. Srisuresh., RFC 2766, February 2000.
- [5] "Architectural Implications of NAT". T. Hain, RFC 2993, November 2000.
- [6] "Providing Scalable and Deployable Addressing in Third Generation Cellular Networks", B.Landfeldt, S.Rattananon and A. Seneviratne, to appear in IEEE Wireless Communications Magazine, special issue on 3G Networks, spring 2003.
- [7] Jayeon Jung, Emil Sit, Hari Balarishnan and Robert Morris, "DNS Performance and the Effectiveness of Caching", ACM SIGCOMM Internet Measurement Workshop 2001