On the Scalability of Addressing in Private Networks Using RPX

Sanchai Rattananon, Zhe Guang Zhou, Björn Landfeldt and Aruna Seneviratne

Original scientific paper

Abstract: **In recent times, the imminent lack of public IPv4 addresses has attracted the attention of both research community and industry. The cellular industry has decided to combat this problem by using IPv6 for all new terminals. However, the success of 3G network deployment will depend on the services offered to end users. Currently, almost all services reside in the public IPv4 address space, making them inaccessible to users in IPv6 networks. Thus, an intermediate translation mechanism is required. Previous studies on network address translation methods have shown that REBEKAH-IP with Port Extension (RPX) supports all types of services that can be offered to IPv6 terminals from the public IPv4 based Internet, and provides excellent scalability. However, this method suffers from an ambiguity problem which may lead to call blocking.**

In this paper, we present an improvement to RPX scheme in which the side effect is removed and fully scalable system. We firstly show the expected number of public IPv4 addresses utilization to the DNS of RPX server. This utilization is computed in terms of the probability of socket open requests from mobile terminals, the probability of call blocking and the estimated number of mobile terminals at the network initialization phase. The mathematical model is also provided as a guideline to determine the range of public IPv4 addresses allocated to an RPX gateway in a cellular network. In addition, the results are presented through a set of simulations. However, we proposed the RPX scheme to use a simple round robin scheduling algorithm is sub-optimal in terms of call blocking probability and further propose to use a priority queue algorithm to improve the scalability. In addition, we present extensive simulation results on the practical scalability of RPX with different traffic compositions to provide a guideline of the expected scalability in large-scale networks such as 3G networks.

Index terms: **IPv4, IPv6, Terminals, Algorithms**

I. INTRODUCTION

With the introduction of third generation cellular networks, the number of IP enabled devices is expected to grow rapidly to number of ways above the current level. Since the current version of IP, version 4, has a limited address space of 32-bits, it is expected that we will have used all available addresses in a foreseeable future [1]. In order to overcome this problem and to meet the demand for addresses, the cellular industry has decided to use the next generation IP, IP version 6 [2], which expand the address field to 128-bits as the primary network protocol in third generation (3G) cellular networks. However, in [3] it has been shown that the successful rollout of 3G will depend heavily on the services accessible through the new networks, and the added value compared with second generation networks such as GSM. It will be crucial for operators and vendors to obtain a good return on the investment for the network coverage to reach predicted levels. Unfortunately, the vast majority of services available to 3G users currently reside in the public IPv4 based Internet, rendering them inaccessible to IPv6 based 3G terminals since IPv6 and IPv4 are incompatible. This means that 3G terminals cannot directly communicate with IPv4 terminals and instead need to communicate via an intermediate translation mechanism. Because of this, there is a need for a short to medium term solution that will enable IPv6 terminals to access public IPv4 Internet services until IPv6 based services have been widely deployed enough for the majority of services to migrate into this space.

Over the past few years, there have been a number of proposals from the research community aimed at providing methods for both extending the address space and for performing translation between the Internet domains. These proposals can be divided into two major groups, namely Network Address Translators (NATs) [4] that aim to extend the address range of IPv4 and IP tunneling solutions [5, 6] that aim at enabling co-existence of IPv4 and IPv6. However, none of the proposed methods are transparent to the applications and impose different restrictions on the type of applications that can be used [7]. In addition, the proposals based on these two methods are limited and will not meet the requirements of operators and users for a successful deployment of 3G networks by the following of these criteria properties:

- It should be scalable (being able to support millions of new 2.5 and 3G terminals that are currently being added to the Internet).
- It should be application friendly (It should not restrict the applications that can be used together with the scheme).
- It should allow network initiated communication (make private terminals reachable from the public Internet).
- It should have relatively low impact on the current infrastructure (so it can be deployed).

For example, the first group of NATs uses different forms of address translation methods to enable traffic traversal between different address realms. A classical NAT [4] has a

Manuscript received May 21, 2006 and revised May 24, 2007 and November 17, 2007.

Sanchai Rattananon is with University of the Thai Chamber of Commerce, Thailand, (e-mail: Sunchai_rat@utcc.ac.th)

Zhe Guang Zhou and Aruna Seneviratne are both with National ICT Australia Limited, Australia (e-mail: zheguang@mobqos.ee.unsw.edu.au, Aruna.seneviratne@nicta.com.au)

Björn Landfeldt is with University of Sydney, Australia (e-mail: bjornl@it.usyd.edu.au).

1845-6421/07/5190

set of public IPv4 addresses that can be assigned to individual private nodes on a per-session basis. When a terminal, with private address X is located within a private realm and wants to contact a remote terminal on the public Internet, the NAT assigns one of its public addresses to the private terminal such as IPv4 address Y. The NAT then rewrites the sender address in the IP header of the outgoing packet with the assigned public address Y. Thus, to the terminals outside the private realm it appears as if the packet originated from another public domain terminal. On the return path, the NAT rewrites the destination address Y in the IP header with the private address X so the packet is correctly routed within the private realm as shown in Fig. 1.

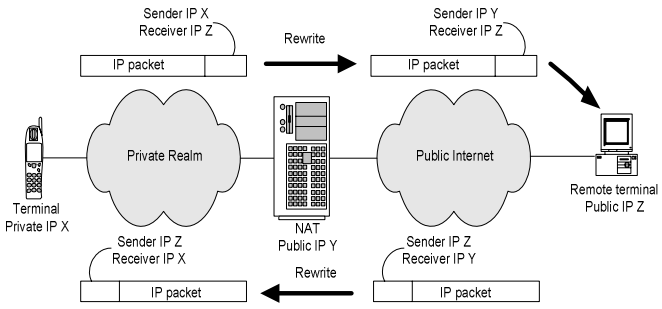

Fig. 1. Operation of Classical NAT.

This method of translating between realms is simple but suffers from three drawbacks which will harm the deployment of 2.5 and 3G networks. Firstly, it is a mechanism of single direction connection initiation. It does not allow a terminal in the public Internet to initiate session to a terminal within the private realm because NAT would not advertise private addresses to the public realm. Thus, a packet from a public realm host cannot be routed to the private address. Secondly, it does not scale since every time a private terminal wants to communicate to a terminal in the public network, a public IPv4 address is reserved exclusively for that terminal's use. Thirdly, some applications and protocols such as ICQ, Real Player and Session Initial Protocol (SIP) carry private network addresses in their payloads. Thus, when the application layer signalled address is a private realm address, despite the NAT changing the addresses in the IP packet header, the addresses in the payload remain unchanged. The receiver detects a non-routable address that will result in a malfunction. In order to overcome this problem, it is necessary to deploy Application Layer Gateways (ALGs) [7] in the NAT. These ALGs needs to be application specific to correctly decipher the addressing information embedded in the payload and substitute the signalled private address with the appropriate public address. However, the ALG function is limited to supporting only a small subset of selected application.

Moreover, Network Address Port Translation, NAPT [4] improves the scalability of the classical NAT by enabling multiple private realm terminals to share a single public IPv4 address. This is achieved by using the transport identifier (e.g. TCP and UDP port numbers) in the translation procedure. This port multiplexing enables the NAPT to share one IPv4 address among several private terminals. In [8], Audet et al. detail a proposal to ascribe the address and port mapping behavior of the NAPT method called Endpoint-Independent Mapping, which reuses the same public address and port mapping to service multiple private terminals simultaneous to communicate to any public terminal. As a result, the method has a good scalability but it is still impossible for terminals within the public realm to initiate sessions to terminals within the private realm. Furthermore, NAPT also requires the use of ALGs to deal with applications which embed IP addressing information in the IP payload.

The second group of extension methods use different forms of tunnelling mechanisms to forward data between two different address realms. The ngtrans working group within the IETF [9] has proposed a number of methods for transition from IPv4 to IPv6 based networks. The work has primarily concerned itself with methods for co-existence of the two protocols and not interoperability. However, through the use of a combination of tunnelling mechanisms [10] it is possible to interconnect IPv6 and IPv4 domains. In [5], Borella et al. present a proposal of the Realm Specific IP (RSIP) scheme that performs IP tunnelling. RSIP scheme takes a different approach than the above-mentioned NATs to provide connectivity between different realms. It uses a client server type architecture, where the client and the server, are aware of the different realms. Consider the following operational scenario of RSIP as shown in Fig. 2, the client first request a public IPv4 address from the RSIP server and the server leases a public realm address X to the client. The client then constructs the message using the leased public address X as the source address. To get the packets with the public realm addresses through the private realm, clients tunnel the packets to the server by using the destination address M which is a private realm address. The server decapsulates the tunnel header and passes them to the public realm. Thus, a RSIP client makes use of a public address when communicating with a host in a public realm, thereby alleviating the need for rewriting of addresses at an intermediate point. An advantage of this scheme is that there is no need to deploy ALGs for applications since public realm addresses are used by the private clients when constructing application data packets. However, the tunnelling mechanism does not extend the public IPv4 address space and still requires public IPv4 addresses for the communication between terminals in the public realm and private realms. Therefore, the mechanism does not solve the immediate need for address expansion until IPv6 gains enough momentum to start easing the burden on IPv4 addressing

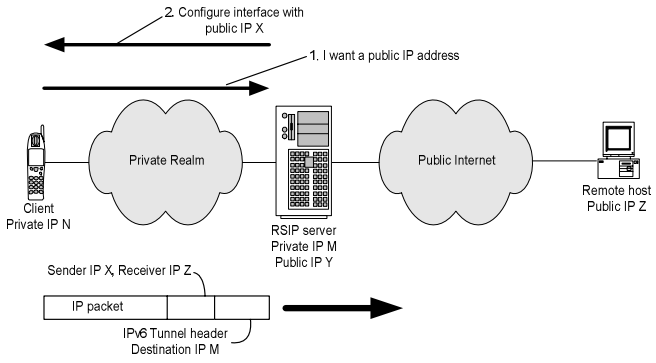

Fig. 2. Operation of RSIP.

To overcome these limitations, we proposed a method called REBEKAH-IP with Port Extension (RPX) [11-13] which meets the demands for cellular operators whilst ensuring the support of three main criteria, namely scalability, public realm initiated communication and application friendliness. However, this method suffers from an ambiguity problem which may lead to call blocking. In this paper, we present an improvement to RPX scheme in which the side effect is removed and fully scalable system. We firstly provide a mathematical model that allocates the expected number of public IPv4 addresses utilization to the RPX server. The calculations consider the probability of socket open requests from mobile terminals, the probability of call blocking and the estimated number of mobile terminals at the network as the main factors to estimate public IPv4 address utilization.

However, the exact scalability of RPX will depend on the traffic composition in terms of the duration of IP flows. In addition, we proposed RPX scheme to use a round robin scheduling algorithm for the assignment of addresses. Thus, there is an additional limitation to the number of flows that the RPX scheme can support. Under heavy load conditions it is possible that a single public IPv4 address may run out of free combinations well in advance of other IPv4 addresses causing call blocking to some terminals. This happens because the private terminals are only configured with a single public IPv4 address for any subsequent connections. Thus, if the terminals using a certain IPv4 address should have many long-lived connections, there is a chance is that they will exhaust the number of free combinations for that address well before another address with fewer long-lived connections. The selection of an algorithm for assigning addresses to private terminals is pivotal in avoiding this effect. Therefore, in this paper we also propose to use a priority queue algorithm instead of a round robin scheduling algorithm for RPX server, in order to overcome the problem mentioned above and further increase the scalability of RPX scheme. In addition, we show that the priority queue algorithm outperforms the round-robin algorithm and perform extensive simulations to investigate the expected number of terminals a 3G network can support in the face of actual IP traffic. The results can act as a guide to estimate the number of terminals networks can support given a range of available public IPv4 addresses available to an operator.

The rest of this article is organized as the follows. We first detail background information on the original REBEKAH-IP proposal and REBEKAH-IP with Port Extension (RPX) in Section II. Then the mathematical model is provided used to determine the expected number of public IPv4 addresses utilization in the network together with simulation results in Section III. Section IV provides the round robin and priority queue algorithms for assigning addresses and port numbers within RPX and also presents simulation results of different traffic models to highlight their performance. Finally, the conclusion is presented in Section V.

II. BACKGROUND

In [11], we proposed a scheme called Realm Base Kluge Address Heuristic-IP (REBEKAH-IP) and its extension REBEKAH-IP with Port Extension (RPX) [13] for expanding the public IPv4 address space that scale satisfactorily while preserving the Internet model of catering for a multitude of service.

A. Realm Base Kluge Address Heuristic-IP (REBEKAH-IP)

REBEKAH-IP [11, 12] was introduced to overcome some of the drawbacks of previous translation proposals [4], while allowing full connectivity. The scheme was designed to allow both terminal and network initiated communication and supports for all types of public Internet services to IPv6 terminals without the need for Application Layer Gateways (ALG). It requires minor changes to existing infrastructure with the addition of a REBEKAH-IP server (RS) acting as a border gateway at the edge of the private network connecting to the public Internet.

The REBEKAH-IP server (RS) integrates features from two existing NAT proposals [4], namely RSIP and Bi-directional NAT, and extends the combination of Layer 3 (Network Layer) and Layer 4 (Transport Layer) switching functions. Therefore, the routing proposed in this scheme is based on a four tuple (sender and receiver IP address and port numbers) rather than only a parameter pair (destination IP address and sender port number). The number of unique combinations between the four tuple can be used to distinguish between flows and thereby increase the scalability of REBEKAH-IP, far surpassing those of the previous NAT proposals. In this scheme, we also use a pool of public IPv4 addresses to configure private terminals while allowing the public IPv4 addresses to be reused as long as the combination is unique as the identifier for a single flow.

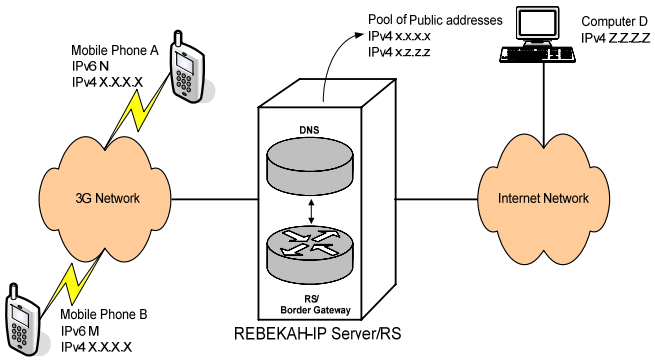

Fig. 3. All components view of REBEKAH-IP.

A simplistic view of REBEKAH-IP is shown in Fig. 3. Similar to RSIP [5], there is a REBEKAH-IP server (RS) that delegates public IPv4 addresses to private terminals on demand.

In addition, there is a purpose built DNS for interoperating in both IPv4 and IPv6 address spaces in a similar manner to Bi-directional NAT [4]. Thus, the DNS function within the server is responsible for resolving a private address and also the assignment of public IPv4 addresses to the private realm terminals. The terminals in the private realm

implement a specific REBEKAH-IP client (RC) for address resolution and session setup similar to RSIP scheme [5], and the terminals in the public realm are left unchanged for backward compatibility unless they reside inside another REBEKAH-IP realm.

Furthermore, we consider the following operational scenario as shown in Fig. 3. When client A/RC connects to a REBEKAH-IP server and requests to be assigned a public IPv4 address with an opened ephemeral port number as sender port, and informs the RS server of both the IP address and port number of the public realm terminal to which it wishes to communicate. The selection process is simple. The DNS function within the RS server has a pool of public IPv4 addresses and it selects addresses from this pool using a round-robin algorithm. The RS server then queries the DNS with predetermined parameters (destination address, sender and destination port numbers) as input. It takes the predetermined parameters into account and searches for a unique combination of the three parameters together with the public IPv4 addresses from the pool. In the example, the RS server obtains public IPv4 address (x.x.x.x) from the DNS function and assigns it to A/RC to open the socket. Once this is done, the client A/RC and the RS server establish an IPv6 tunnel between themselves to route the IPv4 traffic through the IPv6 domain, the same as RSIP. When client B/RC requests a public IPv4 address, the DNS function is able to pick and assign to it the same IPv4 address (x.x.x.x) as long as the four tuple is kept unique.

However, once a private realm terminal has been assigned a public IPv4 address it will maintain this address until all communication channels are closed. This avoids assigning multiple public IPv4 addresses to one private terminal. Thus, there is a small possibility that more than one private terminal will try to open a connection to the same public terminal using the same four-tuple (sender and receiver IP address and port numbers). In this case it will be impossible to distinguish between the different flows resulting in request rejection (i.e. this is the same terminology of call blocking in the rest of this paper) for the all but the first terminal. Even though there is a very low probability of this happening [11], it is an undesirable property of the system. The reader is referred to [11, 12] for additional details on REBEKAH-IP scheme.

B. REBEKAH-IP with Port Extension (RPX)

The problem with the REBEKAH-IP scheme stems from the usage of ephemeral (sender) ports when applications open sockets. Since there is no control over the port allocation, it is impossible for a REBEKAH-IP server to predict the sender port that a private terminal will use for a certain flow. In order to overcome this problem, we proposed a REBEKAH-IP with Port Extension, RPX [13]. RPX involves modification of the REBEKAH-IP scheme to incorporate centralised management of both public IPv4 addresses and port numbers, since new terminals can be shipped with special support and it is possible to have them implement more optimised versions of the REBEKAH-IP scheme. Therefore, the DNS function of RPX is able to decide on not only public IPv4 addresses but also source ports to assign private terminals for use. Thus, instead of querying the DNS for a public IPv4 address only when setting up a connection, as in the RSIP scheme [5], in our proposal the private terminal will obtain the sender port number to use for the socket as well as the public IPv4 address to use. This way, RPX will be able to fully avoid possible clashes between sessions and to unambiguously extend the IPv4 address space. Thus, the RPX scheme is capable of unambiguously supporting a maximum of:

$$\phi = N_{IP} \times (2^{a} - N_{IP}) \times (2^{p} - R)^{2} \qquad (1)$$

flows where N_{IP} is the number of publicly available IPv4 addresses to the DNS, p is the number of bits in the port range, a is the number of bits in the IP address range and R is the number of ports excluded from the assignment by the IANA [14].

Even though a terminal is not position to establish sessions at the complete range of 2^{32} addresses (minus private, reserved, broadcast and multicast address range) according to the IPv4 address assignment procedures [15], the most of these addresses can be used by a terminal for inter-domain communication. Thus, with an example from a cellular network, if N_{IP} = 1000, p = 16, a = 32 and the number of reserved ports R = 1024, the maximum number of flows RPX can support becomes: $(2^{16} - 1024)$ x 1000 x $(2^{32} - 1000)$ x $(2^{16} - 1024) = 1.8 \times 10^{22}$ flows. Note however that this figure requires all connections to be made to different processes in the public Internet. In addition, since each terminal is limited to a maximum of 2^{P}–R connections, the minimum number of terminals needed to reach this number of flows is:

$$n = N_{IP} \times (2^{a} - N_{IP}) \times (2^{p} - R) \qquad (2)$$

which with the above parameters yields 2.8×10^{7} terminals. In addition, if the number of connections should be made to the same public server process (and the server can handle an unlimited number if connections), the theoretical minimum number of flows RPX can uniquely identify becomes:

$$\phi = N_{IP} \times (2^{p} - R) \qquad (3)$$

with the same parameters as above this yields: 1000 x $(2^{16} - 1024) = 6.5 \times 10^{7}$ flows, the same as classical NAT with port translation (NAPT) [4]. In reality, the number of flows an RPX system will be able to support will vary in-between these two extremes.

B.1 Private Terminal Implementation

The private realm terminals will have to support RPX by special functions that are not implemented in current operating systems. Even though this is a negative aspect of the scheme, we argue that it does not have a major impact on the deployment of the scheme for the following reasons:

- First, 3G networks are in the beginning of the rollout phase and since the vast majority of terminals for these networks are yet to be deployed, the challenge is limited to incorporating this function from the beginning (no retro fitting required). For the existing terminals, it is possible to deploy RPX enabled terminals in parallel so that the gateways treat the RPX and non-RPX terminals differently.
- Second, if RPX is deployed in a small-scale domain, it is possible to upgrade existing hosts within the domain while shifting them from their current environment into an RPX environment.

The great advantage is that existing services and network infrastructure such as routers in the public Internet do not have to be modified in any way in order for RPX to be deployed. The terminals have to be able to signal the DNS and expect configuration information in return. They also have to be able to configure themselves with the returned information and possibly also override an application's attempt to specify sender port for a socket.

In Fig. 4, we show a flow chart of the steps taken by a private domain terminal when it wants to open a connection to another terminal in either the private or public realm. After the terminal gets a reply from the DNS the record type determines how the terminal is configured. However, there is a new type of record which is called a SRV record [16] that can be used to contain not only an IP address corresponding to an Fully Qualified Domain Names (FQDN) [17], but also any information; hence we can add public address and port information to this record. If the record is an SRV record, the connection is destined for a public domain terminal and the RPX scheme comes into play. If on the other hand the returned record is a standard A or AAAA record, the terminal uses a standard socket creation process using its private address and a randomly assigned ephemeral port number. This way, RPX only comes into play when traversing the border between the private and public realms.

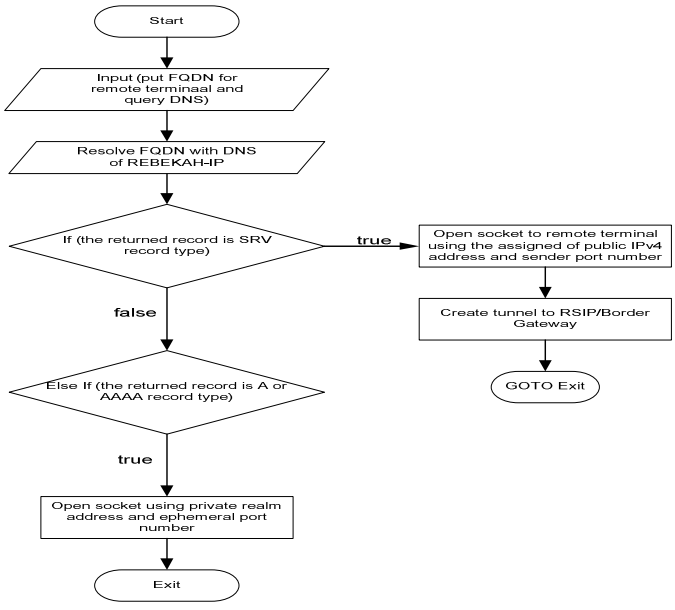

Fig. 4. Flowchart of private terminal signaling function.

An overview of the prototype implementation of RPX scheme set up is shown in Fig. 5. The figure shows the steps taken to configure the private terminal with the obtained parameters (the assigned public IPv4 address and sender port number) to initiate communication to a public realm terminal. In addition, it also shows the steps taken for the DNS to relay the query to the foreign DNS server and resolve the FQDN.

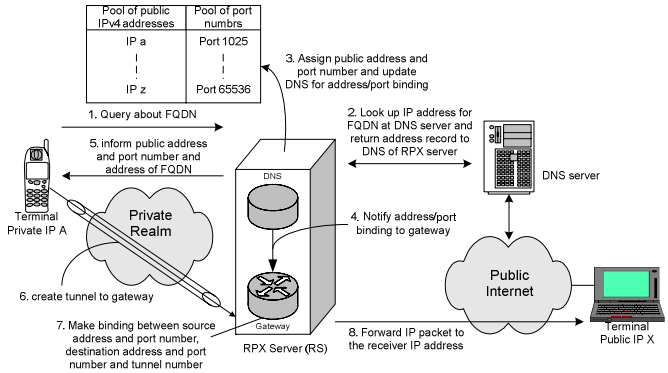

Fig. 5. Operation of RPX.

III. Proviiding The Number Of Public IPv4 Addresses Utilization

Form the equation (1) to (3) it can be seen that the scalability of the RPX scheme depends on the number of public IPv4 addresses (N_{IP}) that are allocated to the RPX DNS. Therefore, the purpose of this section is to determine the range of public IPv4 addresses utilized by an RPX server in a cellular network. This utilization is computed in terms of the probability of socket open requests from terminals, the probability of call blocking and the estimated number of mobile terminals at the network initialization phase as the main factors to estimate IPv4 address utilization. In addition, the results are presented through a set of simulations.

A. System Design and Mathematical Model

In our model, the network is divided into different connection areas. We assumed that mobile terminals are free to move between different areas. These areas are analogous to sub-networks in IP terminology. Each area has a dedicated RPX server (RS) acting as a border gateway. Finally, we also assumed that all areas belong to a common administrative domain as shown in Fig. 6.

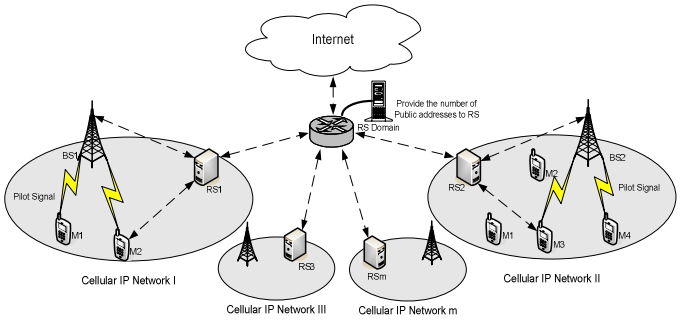

Fig. 6. System design for distributing RPX servers.

As the RS servers are distributed among the sub-networks, the operator needs to manage and allocate a subset of the publicly available IPv4 address pool to each server in each sub-network. The allocation method depends on the probability of requests, the probability of call blocking and the estimated number of mobile terminals within the sub-network.

Assume there are a total of N mobile terminals in the network domain and let N_j be the total number of mobile terminals within sub-network j with the condition $1 \le N_j \le N$. The handoff frequency for a new terminal and the probability of the terminal moving out of the network are assumed equal at the network initialization phase. Therefore, we only need to calculate the expected public IPv4 address utilization for network j by considering the estimated number of existing terminals within the area. Given that the probability of requests for assigning addresses and port numbers, $P_{r,j}$ is equal among all mobile terminals so that the probability of call blocking, $P_{B,j}$ is given by:

$$P_{B,j} = \begin{cases} 1 - \dfrac{N_A}{N_R} & , for\ N_R > N_A \\ 0 & , otherwise \end{cases} \quad (4)$$

where N_A is the maximum number of unambiguously supported flows by the RPX scheme and N_R is the total number of requests from the mobile terminals in the network j.

We express the number of public IPv4 addresses a sub-network j utilizes as follows. Let the average rate for opening new sockets be λ, the average socket holding time be t and the port range for each public IPv4 address be m.

Then, equation (4) could be written as a function of the expected public IPv4 addresses utilization as follows:

$$N_{IP,j} = \frac{K \cdot N_j}{2^{32} \cdot m^2} \cdot \left(1 - P_{B,j}\right) \quad (5)$$

where $N_{IP,j}$ is the expected public IPv4 address utilization for network j and $K = \lambda t P_{r,j}$.

In addition, if number of connections would be made to the same server process according to equation (3), then equation (5) would be become:

$$N_{IP,j} = \frac{K \cdot N_j}{m} \cdot \left(1 - P_{B,j}\right) \quad (6)$$

In reality, the expected public IPv4 address utilization for sub-network j will be varied in-between the two equations (5) and (6). Thus, these equations provide the upper and lower bounds of the expected number of public IPv4 addresses that the network operator will assign to each RPX server to support mobile terminals in each sub-network.

However, the sum of allocated public IPv4 addresses for each sub-network must be less than or equal to the total number of available public IPv4 addresses from the network domain pool $N_{total\_of\_IP}$. Therefore, the sum of allocated public IPv4 addresses for m networks with $\hat{N}_{IP}$ addresses each is defined as:

$$\hat{N}_{IP} = \sum_{j=1}^{K} N_{IP,j} \le N_{total\_of\_IP} \quad (7)$$

In the case when $\hat{N}_{IP} > N_{total\_of\_IP}$, we also provide a negotiation method for solving the problem as follows:

$$\overline{N}_{IP,j} = \left(1 - \frac{N_{IP,exceed}}{\hat{N}_{IP}}\right) \cdot N_{IP,j} \quad (8)$$

where $\overline{N}_{IP,j}$ is the new number of allocated public IPv4 addresses that the network domain has allocated to sub-network j when $\overline{N}_{IP} \in N_{total\_of\_IP}$ and $N_{IP,exceed} = \hat{N}_{IP} - N_{total\_of\_IP}$ is the total number of allocated public IPv4 addresses exceeding the total number of available public IPv4 addresses within the network domain.

For example, we assume that there are two sub-networks, sub-network 1 and sub-network 2. Assume that the total number of publicly available IPv4 addresses within the RS domain is 1000 and also that the result of the allocated public IPv4 addresses for sub-network 1 and sub-network 2, from equation (6), are 500 and 600 respectively. When the summation of these results of allocated public IPv4 addresses is greater than the total number of available public IPv4 addresses from the domain, the operator has to re-allocate addresses according to equation (8). Therefore, the new resulting number of allocated public IPv4 addresses becomes 450 and 550 for sub-network 1 and 2 respectively. This solution will slightly increase the call blocking probability for both sub-networks because of the decreasing number of requested public IPv4 addresses, as can be seen from equation (6). However, the negotiation method balances the blocking probability in both sub-networks in terms of address sharing and utilization, which is also dependent on the address assignment requests and the number of mobile terminals of each sub-network according to equation (5) and (6). In addition, the RS domain also has a sufficient number of public IPv4 addresses to allocate for both sub-networks.

Moreover, the conclusion of the negotiation method above works well for the initialization phase of the sub-networks. Once, the public IPv4 address pool allocation process is complete, all sub-networks are able to operate with the allocated public IPv4 address pool from the RS domain. After this, if the RS domain still has a set of free public IPv4 addresses. This set can be further allocated for the new sub-networks by the RS domain. When the new sub-networks are created, the negotiation method will be applied only to the set of free public IPv4 addresses if these free addresses are not

enough to allocate for all new sub-networks. Thus, the operation of all existing sub-networks will remain un-affected.

B. Simulation Results

In order to obtain realistic input values to the mathematical model, we examined traffic using parameters from previous work [11]. The average number of sockets opened per second and the average socket holding time for a mobile terminal were 0.015 and 17 seconds respectively. In addition, the available number of ports for each IP address was $2^{16} - 1024$ as specified in [18].

Fig. 7 shows the number of utilized public IPv4 address ($N_{IP,j}$) versus the number of mobile terminals (N_j) when using different probabilities of call blocking ($P_{B,j}$). The probability of socket open requests from the number of terminals ($P_{r,j}$) was set to one in the network. The results show that the public IPv4 address utilized increases with the increasing number of mobile terminals that can be supported by RPX in the network, as expected. However, the address utilization can also be contained by varying the probability of the call blocking. This figure illustrates the utilization with a blocking probability of 0%, 5% and 10% respectively.

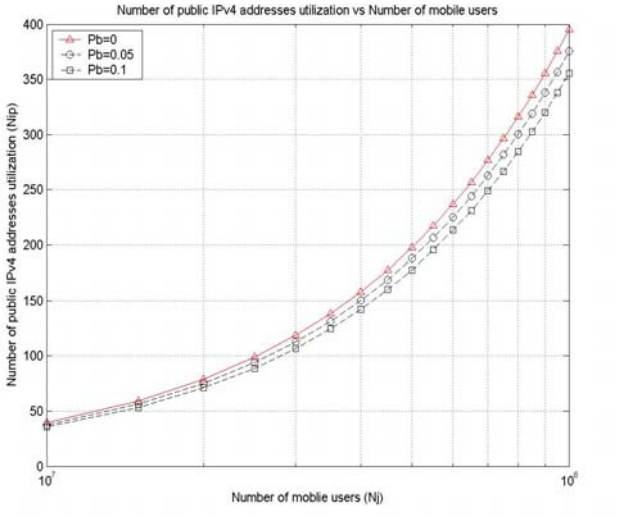

Fig. 7. Public IPv4 address utilization and number of mobile terminals with $Pr, j = 1$.

Fig. 8 shows the public IPv4 addresses ($N_{IP,j}$) utilization versus the number of mobile terminals (N_j) when the probability of socket open requests from the number of terminals ($P_{r,j}$) is varied. The call blocking probability was set to a constant of 5% in the simulation. The figure illustrates how the number of utilized public IPv4 addresses will be reduced as the probability of request decreases since the call blocking probability ($P_{B,j}$) is dependent on the probability of socket open requests from the mobile terminals ($P_{r,j}$) with the condition $P_{B,j}$ α $P_{r,j}$. Furthermore, we can see that the number of terminals that the RPX server can support increases as the probability of socket open requests decreases with a fixed public IPv4 addresses utilization.

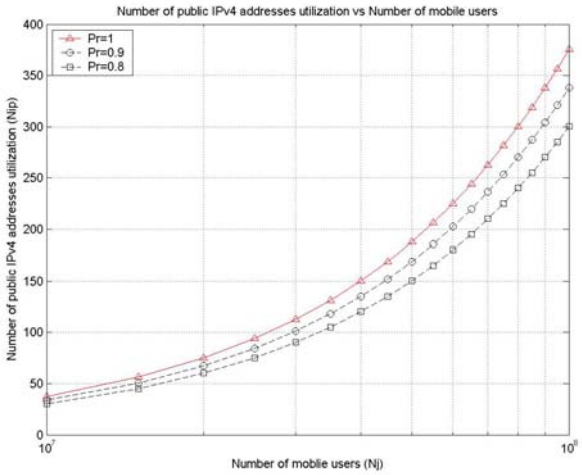

Fig. 8. Public IPv4 addresses utilization and number of mobile terminals with $P_B = 5\%$.

In this section, the results from our simulations allow us to draw some conclusions regarding the number of public IPv4 addresses utilized by the RPX scheme as follows:

Firstly, the results in Fig. 7 and 8 were obtained with the assumption that all connections were opened up to the same server process in the public Internet as simplified in equation (6). Therefore, the results illustrate the maximum number of public IPv4 addresses utilized, while scaling the expected number of mobile terminals in the network from ten to one hundred million hosts under the assumed conditions of call blocking probability and number of sockets of each host. From the results above it can be seen that RPX can provide excellent scalability in terms of supporting a large number of private terminals, while only utilizing a small number of public IPv4 addresses, meaning that cellular 3G operators may operate with a realistic value of 1000 IPv4 addresses available for allocation.

Secondly, the formulas used in the simulations above enable us to give a good estimation of the number of IPv4 addresses utilized in order to achieve reasonable estimates of RPX in actual deployment, and indicate that the scalability of RPX is very promising.

IV. Algorithm For Distributing Addresses And Port Numbers

The RPX scheme was designed to use a round robin scheduling algorithm in the address assignment to IPv6 host according to REBEKAH-IP scheme [11]. In the implementation of the original round-robin algorithm to allocate public IPv4 addresses and port numbers we used a simple method by which the IPv4 addresses are arranged in a linked list and stepped through sequentially for address allocation. If there is no free port for a specific address, the next address in the list is selected instead. The address range is in a circular list. When the end of the address list is reached, the server starts assigning addresses from the beginning of the list. However, this method is susceptible to IPv4 address blocking or call blocking as described above.

To overcome this problem, we propose another algorithm, the Minimum-Oriented Priority Queue algorithm [19] for distributing IPv4 addresses and port numbers from the address pool. The priority queue tries to balance the port utilization of each IPv4 address by looking for the IPv4 address with the least number of occupied port numbers and assigning an IPv4 address/port combination from this address.

In this section, we perform extensive simulations to investigate the expected number of terminals a 3G network can support in the face of actual IP traffic for both round robin and priority queue algorithms, and also present simulation results of different traffic models to highlight their performance by given a range of available public IPv4 addresses available to an operator.

A.. Simulation Model and Results

We have made a simulation study to investigate the call blocking probability of the two algorithms in large scale scenarios to determine their comparative performance as well as obtaining a good indication of the RPX scalability in real life cellular network scenarios. We have used the simulation model shown in Fig. 9.

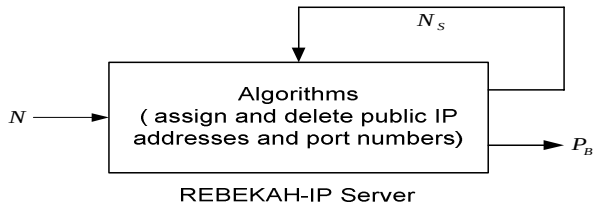

Fig. 9. Simulation model.

In Fig. 9, the number of new mobile terminals is N, the number of existing mobile terminals in the system is N_S and the call blocking probability is P_B. In our simulations we have constructed three different traffic scenarios and investigated the scalability of both the round robin and priority queue algorithms.

A.1 Input Parameters

We have made a number of assumptions and based our simulation parameters on previous work [20] as follows. Assume a very large population of N mobile nodes in the system and a number of nodes N_S that the system can accommodate with the condition $1 \leq N_S \leq N$. The new call rate is uniformly distributed over the mobile service area and the average call rate in the simulation is independent from the number of calls in progress and the number of new calls after a socket is closed.

The call blocking probability will depend on the constitution of the traffic in terms of the channel holding times. Therefore, in our simulations, we used three traffic models, voice traffic, web traffic and long-lived connections in which the node keeps the socket open for one hour. We simulated three different scenarios with different traffic compositions as follows:

- In the first scenario, we generated 10% voice traffic, 80% web traffic with 4 sockets per session and 10% long-lived connections.
- In the second scenario, we generated 10% voice traffic, 80% web traffic with 1 socket per session and 10% long-lived connections.
- In the third scenario, we generated 30% voice traffic, 50% web traffic with 1 socket per session and 20% long-lived connections.

We used the following parameters for our simulations:

- The nodes randomly generated traffic proportional to the three cases above.
- The time to open a socket was 1 second according to [20].
- The average socket holding time in a mobile node in the three different categories were: voice call 1 minute, web browser 17 seconds according to [11] and long-lived connections 1 hour.
- The average waiting time before a mobile terminal would make a new call after closing a socket was: voice call 1 hour, web browser 10 minutes and long-lived connection 3 hours.
- The pool of public IPv4 addresses was set to 5 and the number of ports was 1000 per IPv4 address.

A.2 Simulation Results

Using the above input parameters, the simulations were conducted to compare the performance of the round robin algorithm and the priority queue algorithm with respect to the number of terminals the system could support (N_S) before the first blocking occurred and also until the call blocking probability (P_B) reached 1%, 2% and 3%.

Fig. 10 and Fig. 11 illustrate the number of mobile terminals (N_S) versus the total number of combinations of IPv4 addresses and port numbers (the product of the number of IPv4 addresses and the number of ports per IPv4 address) from the different scenarios above. The figures show the number of terminals in the system that RPX can support and the number of successful addressing assignments before the first call block took place using both round robin and priority queue algorithms.

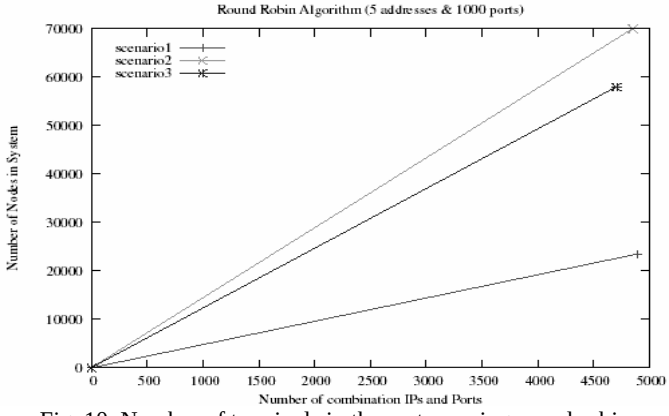

Fig. 10. Number of terminals in the system using round robin algorithm, when the first call blocking occurs.

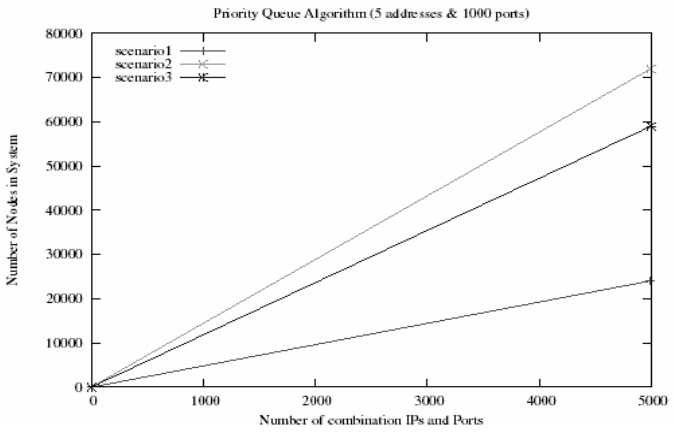

Fig. 11. Number of terminals in the system using priority queue algorithm, when the first call blocking occurs.

The following conclusions can be made regarding the two algorithms. Firstly, the round robin algorithm implements a weaker balancing strategy, which results in a smaller number of assigned combinations of IPv4 addresses and port numbers before call blocking is experienced. Secondly, we can deduce that the round robin algorithm can support a smaller number of mobile terminals (N_S) than the priority queue algorithm, which is intuitive since the priority queue algorithm performs better in terms of call blocking probability.

Fig. 12 and Fig. 13 show the distribution of allocated ports for each IPv4 address in a pool when the first call blocking occurs. The figures clearly illustrate how the priority queue algorithm distributes the load between addresses better than the round robin algorithm thereby achieving a better address utilization.

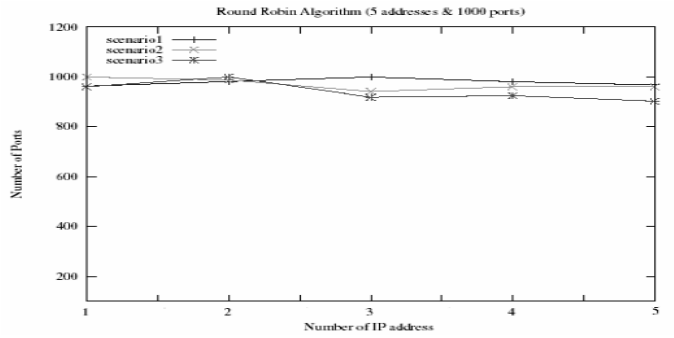

Fig. 12. Number of ports in use for each public IPv4 address using round robin algorithm.

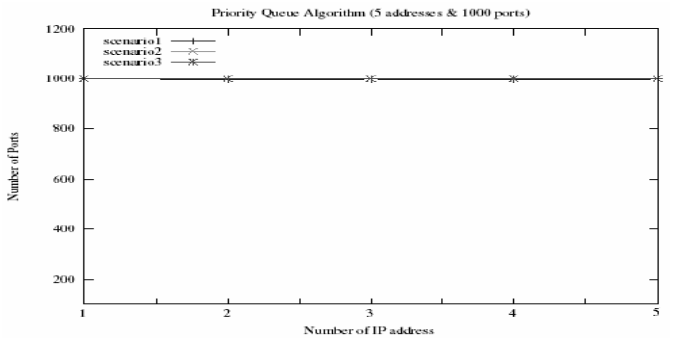

Fig. 13. Number of ports in use for each public IPv4 address using priority queue algorithm.

Fig. 14 shows the number of mobile terminals (N_S) that could be supported by the system while the call blocking probability (P_B) was increased from 0% to 3%. The figure illustrates the performance of both the priority queue and round robin algorithms in terms of call blocking probability from the different scenarios above. The results show that the priority queue algorithm still outperforms the round robin algorithm.

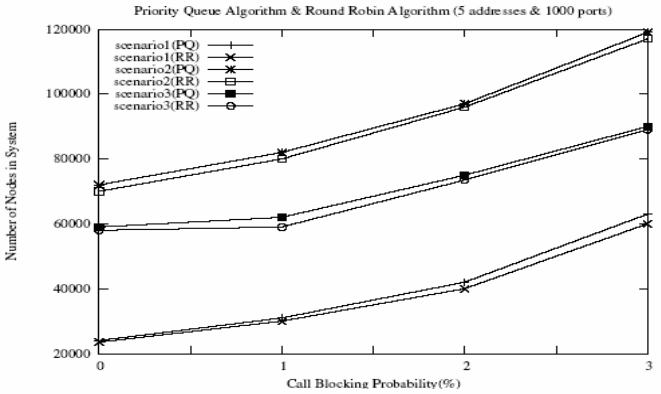

Fig. 14. Number of terminals in the system for both round robin and priority queue algorithms, when the call blocking reaches 1%, 2% and 3%.

Fig. 15 shows the number of mobile terminals (N_S) that the system can support as a function of the fraction of long-lived nodes using the two algorithms. The parameters used were derived from scenario 2 where the number of voice call nodes was kept at a constant 10% and the number of web traffic terminals and long-lived terminals were changed. The results illustrate the relative performance of the two algorithms over a broad range of traffic compositions. From the graph we can see that even though the priority queue algorithm is constantly better performing than the round robin algorithm, it too will be affected by the traffic composition.

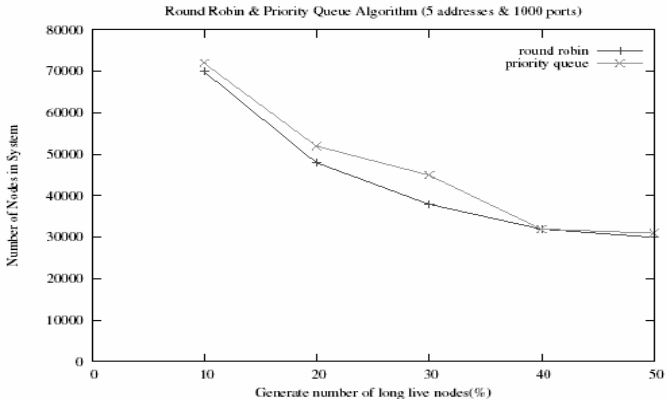

Fig. 15. Number of terminals in the system for both round robin and priority queue algorithms, when the long-lived nodes are increased.

We also carried out a simulation study to verify that the behavior with a small number of IPv4 addresses that was indeed preserved within the context of a real cellular network. We therefore carried out a simulation with 10 million terminals using 1000 IPv4 addresses in the RPX address pool

and investigated the distribution of occupied ports over the IPv4 addresses. The results using the above mentioned scenarios can be seen in Fig. 16 and 17 for the round robin and the priority queue algorithms respectively. The figures reveal that the behavior is indeed similar and that the round robin scheme is somewhat more unbalanced than the priority queue scheme as in the previous results. It is also worth noting that given these traffic scenarios, we can derive an indication of the scalability of RPX and verify that it comes very close to the theoretical best behavior since the priority queue scheme performs very well in terms of load balancing.

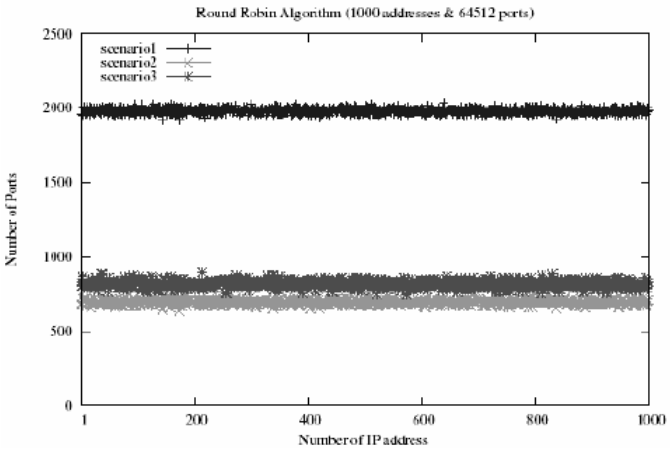

Fig. 16. Number of ports used for each IPv4 address using round robin algorithm, when there are 10 million terminals in the system.

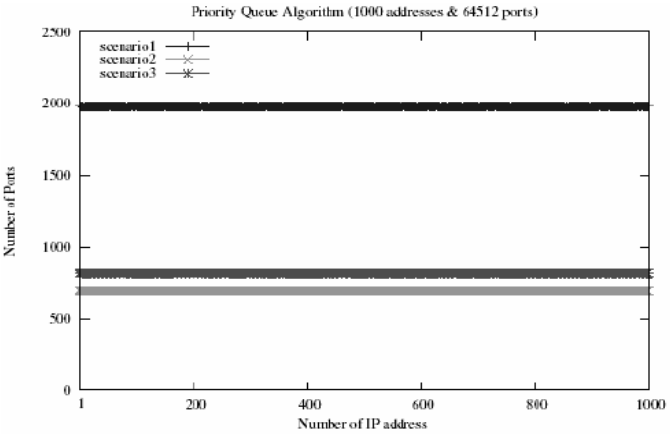

Fig. 17: Number of ports used for each IPv4 address using priority queue algorithm, when there are 10 million terminals in the system.

In addition, in these scenarios, the network utilizes only 3.5% or less of its theoretical lower bound capacity (all sockets opened to the same process in the public realm) which leads to the observation that 3G deployment with RPX would not be limited by a lack of IPv4 addresses.

B. A Comparison Performance of Two Algorithms

The selection of algorithm for assigning an address and port combination is not only dependent on maximum scalability. In our previous experiments, we have found that the main contribution to system delay comes from searches to assign and delete address mappings in the RPX gateway. Therefore, it is imperative that the complexity of the two algorithms is taken into account as well.

There are two cases when a mobile terminal requests to be assigned an address and port combination for a communication end-point.

I) The terminal has no previous assignments. We define this type of terminal as a new terminal to the RPX Server (RS). Therefore, this terminal accepts any IPv4 address allocated from the RS.

II) The terminal has a previous open communication channel and thus it has already been assigned a public IPv4 address. We define this type of terminal as an in-use terminal to the RS. Therefore, for such a terminal, the RS needs to search the corresponding IPv4 address and assign a new port from the pool belonging to this IPv4 address.

In the following section, we analyse the complexities of the two algorithms in the average and worst case using Big-O notation. The assumption is that the address and port allocation process has been performed for considerable time so that there are no unused IPv4 addresses in the system and that the client may request more than one port. In the following analysis, n represents the size of the IPv4 address pool and m represents the size of the port pool for each IPv4 address.

The difference between the priority queue algorithm and round robin algorithm is the key used when searching for the IPv4 address to use for the requesting terminal. The priority queue searches for the IPv4 address in the address pool that has the minimum number of used ports. The round robin algorithm searches for the next IPv4 address that has sufficient number of free ports to allocate to the requesting terminal.

Priority queue: *Min {key (the number of ports allocated in each IP address)}*

Round robin: *Array {key (next IP address which has sufficient available ports)}*

This leads to the following complexity analysis for both round robin and priority queue algorithms:

B.1 Priority Queue Algorithm

B.1.1 An operation of RPX to assign an address/port to a mobile terminal

In case I, the requesting terminal is a new terminal. The root of the priority queue is the IPv4 address that has the minimum number of allocated ports (m). Thus, we can simply assign an available port along with the IPv4 address at the root to the mobile terminal and then reorder the priority queue since the first terminal in the priority queue has the minimum number of ports in use. It is well known that the complexity of reordering one terminal in a priority queue is $O(\log n)$ [19]. Then the complexity becomes $O(m + \log n)$.

In case II, for a mobile terminal that has previously been configured with a public IPv4 address. The IPv4 address of the mobile terminal is used as the index for searching an array. Therefore, the server only searches for an available port and then updates the priority queue. Therefore, the complexity becomes $O(m + \log n)$ as the same as case I

B.1.2 An operation of RPX to delete assigned address/port from a mobile terminal

There are also two scenarios for a mobile terminal releasing assigned address and port combinations. In the first scenario, a mobile terminal releases one or more assigned ports but maintains some open connections. In the second scenario, the mobile terminal releases all assigned ports and the assigned IPv4 address. The priority queue algorithm has the same complexity in both cases since the operation is a simple queue update and the complexity becomes *O(log n).*

B.1.3 An operation of RPX to detect blocking

When blocking occurs, there are also two possible cases since blocking occurs only in the request scenario. For a new terminal, there is a constant time required since the root of the priority queue immediately reports the maximum number of ports that is available in the pool. For the in-use terminal, since its IPv4 address is the index of the address array, there is no searching operation required. Since the priority of the corresponding IPv4 address indicates the number of ports used, this can be used to detect blocking.

In addition, in the priority queue algorithm there is always an update operation after an IPv4 address and port has been allocated or released which has the complexity of *O(log n)*. In our implementations, the server also performs the update after the IP address or ports are allocated to the requesting node. Therefore, the requesting terminal is able to establish the communication at the same time as the server updates the priority queue. Thus, the total waiting time for the requesting node is reduced by *O(log n)*.

B.2 Round Robin Algorithm

B.2.1 An operation of RPX to assign an address/port to a mobile terminal

The operations when using the Round Robin algorithm are more complex than those of the priority queue algorithm. In case I, for an un-configured mobile terminal, the RPX gateway initially searches for an IPv4 address which has sufficient number of available ports for the request. This operation is of complexity *O(n)*. The gateway then searches the port range for available ports which has complexity *O(m)*. The implementations use a variable for each IPv4 address that indicates the number of ports that are used. Thus, the complexity for the round robin algorithm in this case becomes *O(n + m).*

In case II, for an previously configured terminal, the operation becomes the same as the case above, since the server still terminals to find the correct IPv4 address entry in the table and then searches for the available port number. Therefore, the complexity becomes *O(n + m)* as the same as case I.

B.2.2 An operation of RPX to delete assigned address/port from a mobile terminal

For a mobile terminal releasing assigned address and port combinations, the operation follows the same rules above. Therefore, the release procedure is also *O(n + m)* of complexity.

B.2.3 An operation of RPX to detect blocking

When blocking occurs, as mentioned before, there are two cases, one is for a new terminal and another is for a previously configured terminal. For a new terminal, a search is required since the server does not have a global view of the status of the system. Thus, the server needs to search each IPv4 address to determine the available ports. Therefore, the operation of blocking detection is *O(n).*

B.3 Discussion: performance of the two algorithms

From the simulation results and the complexity analysis of the two algorithms we can draw the following conclusions regarding the two algorithms for address and port assignments in RPX and therefore conclude that the priority queue is evidently a better choice for providing this function:

Firstly, the RPX server has a global view of the system and the root of the priority queue is always the IPv4 address that has the minimum number of used ports. Secondly, it provides a lower call blocking probability and therefore a better address utilization when addresses are scarce. Thirdly, the balancing between IPv4 addresses in terms of port utilization is better which leads to a much more predictable behavior from the system and also better confidence in the call blocking behavior for individual terminals. Fourthly, the complexity using the priority queue is lower than that of round robin both for assignment for the new terminals and the blocking detection. Thus, the load on the gateway is lower which increases the scalability of the gateway.

Therefore, the comparison of the the complexities for both Round Robin and Priority Queue algorithms in terms of an operation of searching to assign/delete an address and port numbers combination and the blocking detection is shown as Table I.

TABLE I
A COMPARISON OF THE COMPLEXITIES OF THE TWO ALGORITHMS.

An operation of RPX	Round Robin Algorithm	Priority Queue Algorithm
A New Node	O(m+n)	O(m+log n)
A Configured Node	O(m+n)	O(m+log n)
Delete assigned IPv4 addresss and port	O(m+n)	O(log n)
IPv4 address and port blocking detection	O(n)	O(1)

From Table I, a new node means that a terminal has no previous assignments for both a public IPv4 address and port. A configured node means that a terminal has already been assigned a public IPv4 address. In addition, *n* is the size of the public IPv4 address pool while *m* is the size of the port pool for each public IPv4 address.

In this section, the results from our simulations allow us to conclude regarding for these reasons and their mapping to the

three main criteria namely, scalability, minimum call blocking probability and cost, the priority queue algorithm is preferable.

V. Conclusion

One of the main threats to successfully deploying 3G services and other new services is the limitation of available IPv4 addresses. Previous work has shown that IPv6 will not overcome this problem in the short to medium term and effective translation mechanisms are therefore necessary at least until IPv6 is mature enough to overtake IPv4. In previous work, REBEKAH-IP with port extension (RPX) was proposed as a candidate solution to this end. However, the scalability of RPX will depend on a set of number of publicly available IPv4 addresses (N_{IP}) to the DNS. In this paper, we have presented a mathematical model that an operator can use to determine the expected IPv4 address utilization of an RPX system. In addition, we proposed a negotiation method to manage the balancing of address allocations between several sub-networks in a system.

Furthermore, we have studied the performance of the round robin algorithm for assigning addresses and ports from the original proposal and found that it is not optimal. Next, we have proposed to use a priority queue algorithm and have shown that this algorithm is more suitable in that it has lower complexity, better predictability and exhibits better address utilization and lower call blocking probability than the round robin algorithm. In addition, our simulations give a good indication on the actual scalability RPX can achieve given an address and port range coupled with different traffic compositions.

VI. Acknowledgments

The authors would like to thank Dr.Krit Wongrujira and Dr.Stephen Herborn for their invaluable helps.

References

[1] *Minutes of the Address Lifetime Expectations working group, proceedings of 29th ETF meeting, Seattle*, April 1994.

[2] S.Deering, R.Hinden: *Internet Protocol, Version 6 (IPv6) Specification*, Internet RFC 2460, December 1998.

[3] A.Jamalipour, M.Yabusaki: *3G Mobile Network Technologies and Experiences*, IEEE Wireless Communications Magazine, special issue on 3G Networks, vol10 no.1, February 2003.

[4] P.Srisuresh, M.Holdrege: *IP Network Address Translator (NAT) Terminology and Considerations*, IETF RFC 2663, August 1999.

[5] M.Borella, J.Lo, D.Grabelsky, G.Montenegro: *Realm Specific IP: Framework*, RFC 3120, October 2001.

[6] M.Borella, D.Grabelsky, J.Lo, K.Taniguchi: *Realm Specific IP: Protocol Specification*, RFC 3103, October 2001.

[7] D.Senie: *Network Address Translator (NAT)-Friendly Application Design Guideline*, RFC 3235, January 2002.

[8] F.Audet, C.Jennungs: *Network Address Translation (NAT) Behavioral Requirements for Unicast UDP*, RFC 4787, January 2007.

[9] *Work of the IETF Next Generation Transition (ngtrans) working group*, http://www.ietf.org.

[10] F.Templin, T.Gleeson, M.Lehman: *ISATAP Transition Scenario for Enterprise/Managed Networks*, IETF draft-ietf-ngtrans-isatap-scenario-01.txt.

[11] B.Landfedt, S.Rattananon, A.Seneviratne: *Providing Scalable and Deployable Addressing in the Third Generation Cellular Networks*, IEEE Wireless Communications Magazine, special issue on 3G Networks, vol10 no.1, pp.36-42, February 2003.

[12] B.Landfeldt, S.Rattananon, A.Seneviratne: *Expanding the Address space through REBEKAH-IP: An Architectural View*, ICITA 2002, November 2002.

[13] S.Rattananon, B.Landfeldt, A.Seneviratne, *Extending REBEKAH-IP with Central Port Allocations for Un-Ambiguous IPv4 Address Expansion*, ICON 2003, September 2003.

[14] http://www.iana.org/assignment/port-numbers

[15] http://www.iana.org/ipaddress/ip-addresses.htm

[16] A.Gulbrandsen, P.Vixie, L.Esibov: *A DNS RR for specifying the location of service (DNS SRV)*, IETF RFC 2782, February 2000.

[17] P.Mockapetris: *Domain Names – Concepts and Facilities*, RFC 1034, November 1987.

[18] J.Reynolds, J.Postel: *Assigned Numbers*, Internet RFC 1700, October 1994.

[19] R.Sedgewick: *Algorithms IN C (Third Edition)*, Addison-Wesley, 2001.

[20] D.Hong, S.S.Rappaport: *Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone System with Prioritized and Nonprioritized Handoff Procedures*, IEEE Transactions on Vehicular Technology, vol VT-35 no.3, August 1986.

Sanchai Rattananon received the BEng degree in electronic engineering form the University of the Thai Chamber of Commerce, Thailand in 1994, and MEng degree in Telecommunication from Swinburne University of Technology, Australia in 1997. He also received his Ph.D. at University of New South Wales, Australia in 2006. Currently he is working in the Department of Electronic and Telecommunication, School of Engineering at the University of the Thai Chamber of Commerce, Thailand. His research interests are network middleware and IP mobility management.

Zhe Guang Zhou has completed a combined degree Bachelor of Computer Science and Bachelor of Electrical Engineering with first class honours at University of New South Wales, Australia in 2000. He received his Ph.D. at University of New South Wales, Australia in 2005. His research interests are mobile location management and wireless network connectivity.

Björn Landfeldt started his studies at the Royal Institute of Technology in Sweden. After receiving a BSc equiv, he continued studying at The University of New South Wales where he received his PhD in 2000. In parallel with his studies in Sweden he was running a mobile computing consultancy company and after his studies he joined Ericsson Research in Stockholm as a Senior Researcher where he worked on mobility management and QoS issues. In 2001, Dr. Landfeldt took up a position as a CISCO Senior lecturer in Internet Technologies at the University of Sydney with the School of Electrical and Information Engineering and the School of Information Technologies. Dr Landfeldt has been awarded 8 patents in the US and globally. He has published more than 60 publications in international conferences, journals and books and been awarded many competitive grants such as ARC discovery and linkage grants. Dr. Landfeldt is also a research associate of National ICT Australia (NICTA). Currently, he is serving on the editorial boards of international journals and as a program member of many international conferences and is supervising 8 Ph.D students. Dr. Landfeldt¹s research interests include; mobility management, QoS, performance-enhancing middleware, wireless systems and service provisioning.

Aruna Seneviratne (M'94) received the B.Sc.(Hons) degree from Middlesex Polytechnic, London, U.K., and the Ph.D. degree from the University of Bath, Bath, U.K. Since graduating, he has held academic appointments at the University Bradford (U.K.), Curtin University, Australian Defense Force Academy (UNSW), and the University of Technology, Sydney. Outside academia he has worked at the Standard Telecommunication Laboratories (U.K.), Muirhead and Company (U.K.), and Telecom Australia (Telstra). He has also spent time at the MASI Laboratory, University of Pierre Marie Curie (Paris), and the Rodeo Group at INRIA (Nice) as a Visiting Professor. He is currently the Director of the Australian Technology Park Laboratory, National ICT Australia. He also holds the Mahanakorn Chair of Telecommunication in the School of Electrical Engineering and Telecommunication, University of New South Wales. His current research interests are in mobile data communication systems.