

# Providing Scalable and Deployable Addressing in Third Generation Cellular Networks

Björn Landfeldt\*, Sanchai Rattananon\*\* and Aruna Seneviratne\*\*

\*School of Electrical and Information Engineering and  
School of Information Technologies  
The University of Sydney  
Sydney 2006 Australia  
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

*Over the past few years, information technology has changed the way we live, work and communicate with each other. The two outstanding trends of Internet and cellular telephony growth will now enter a new phase when the two technologies converge into 2.5 and 3G networks. The success of cellular telephony is due to the mobility factor and the success of the Internet is due to its architecture, which allows a multitude of services to use the same simple infrastructure.*

*The rollout of these networks has already started around the world but there are still outstanding issues that are essential to resolve if the technology is to be a success. One crucial issue is the address space limitation of IPv4 and its impact on cellular network scalability. To date, there have been a number of proposals on how to alleviate this problem. However, as we discuss in this paper these solutions will not meet the demands of the mobile Internet in that they will either not scale satisfactorily or they will severely limit the services that can be deployed. We therefore introduce a new solution that will scale satisfactorily while preserving the Internet model of catering for a multitude of services.*

## Keywords

Cellular networks, IP migration, service enabler, 3G, NAT

## Introduction

The deployment of 2.5 and 3G networks has started around the world. The expectation is that these networks will grow in popularity to easily outnumber the current Internet users within a few years time. This rollout exacerbates the shortage of IPv4 addresses. Today, network operators that apply for IPv4 addresses for their new cellular networks are allocated far fewer addresses than the number of users they expect to support. The ratio can be as low as a few thousand addresses for an expected customer base of many millions of subscribers.

To meet the demand for addresses, telecom vendors believe that it is necessary to use IPv6 in the next generation cellular networks. IPv6, fully deployed would solve the address space problem, but unfortunately, IPv6 is not widely deployed in the Internet as yet, and it is expected that this deployment will be slow. Thus, the vast majority of services the cellular terminals are likely to access will initially be available within the IPv4 based public Internet. Since IPv6 is not deployed in the Internet, vendors will have to use migration schemes that translate addresses between IPv6 and IPv4 domains in order for the cellular hosts to use the public Internet services.

Over the past few years, a number of methods have been proposed for both extending the address space and for translating between the two Internet domains [RFC 2663]. 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 [RFC 3235].

In this paper, we will discuss the shortcomings of existing proposals and present a method for translating addresses between two different address realms that will meet the demands from operators and users until IPv6 gains enough momentum and becomes widely deployed.

### **Existing Methods for Address Realm Translation**

The existing methods for expanding the IPv4 address space are based around the concept of Network Address Translators (NATs) [RFC 1631]. Several flavours of NAT have been proposed. The common feature of these different proposals 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.

#### **Basic NAT**

A classical NAT uses a set of public IP addresses to assign to individual private nodes on a per-session basis. When a host within a private realm wants to contact a host on 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. 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 so the packet is correctly routed within the private realm as shown in figure 1.

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 does not allow a host in the public Internet to connect to a host within the private realm as data cannot be routed to the private address. This imposes serious limitations in the types of services a client can use. For example, it will exclude all push services, event based notification services and many peer to peer services.
- Secondly, although it has the benefit of being able to multiplex public addresses between private hosts on a request basis, it does not scale. 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., Thus with the expected growth in number of hosts the multiplexing gain achieved will not nearly enough cater for the demand..

- Thirdly, many applications and protocols such as ICQ, Real player and SIP embed IP address information in application layer signalling messages. Therefore this information is within the payload of the IP packet. 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. Thus the receiver will read an address that is not routable and as a result the application will malfunction. In order to overcome this problem it is necessary to deploy application layer gateways (ALGs) 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, ALGs are difficult to deploy and manage in public cellular networks, unless the ALG function is limited to supporting only a small subset of selected applications.

This problem is made even worse as the developers of popular applications very often does not have commercial relationships with either the NAT vendors or the network 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 and their upgrades.

## **NAPT**

Network Address Port Translation NAPT [RFC 2663], 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 and its IP address to the connection. Then the NAT translates between the assigned address and port and the original private address and port.

This post multiplexing enables the NAT to share one IP address among several private hosts. As a result, NAPT scales better than traditional NAT, but it is still impossible for hosts within the public realm to initiate sessions to hosts 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.

## **Bi-directional NAT**

Bi-directional NAT [RFC 2663] overcomes the problem of hosts within the public realm not being able to reach hosts within a private realm. This is again accomplished through the introduction of an ALG that is specific for DNS requests. The ALG in the Bi-directional NAT understands the two different address realms and is responsible for resolving Fully Qualified Domain Names FQDN, of private realm hosts to their

assigned public realm addresses. When a DNS request is made for resolving a private address the DNS-ALG assigns a public address to the private host, makes an appropriate entry in the NAT, and returns the assigned public address back to the requester. After that the operation is the same as that of a basic NAT. This is schematically shown in figure 2.

The Bi-directional NAT, similarly to the classical NAT, uses a one-to-one mapping between hosts and public IP addresses and as a result again scales poorly. Furthermore, it again requires ALGs to cater for applications which embed addressing information in the IP packet payload.

### **NAT-PT**

Another flavour of NAT, NAT-PT, adds specific protocol translation between IPv4 and IPv6 [RFC 2766]. NAT-PT operates similarly to standard NATs with the addition of a set of IPv4-IPv6 specific translation functions. NAT-PT comes in three flavours corresponding to standard NAT, NAPT and Bi-directional NAT. Thus they suffer the same drawbacks. For example, firstly, NAT-PT and Bi-directional NAT-PT uses one-to one mapping of public IP address to private realm hosts and as a result do not scale. Secondly, they require the use of ALGs to support applications that embed addressing information in the IP packet payload.

### **Realm Specific IP (RSIP)**

RSIP [RFC 2663] takes a different approach than the above-mentioned NATs to provide connectivity between different realms. RSIP uses a client server type architecture, where the client and the server, are aware of the different realms. The client first request a public IP address from the server and the server leases a public realm address to the client. The client then constructs the message using the leased public address as the source address. 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. This is schematically shown in figure 3. 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 address 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 when constructing the packets by the private clients. This mode of RSIP operation referred to as Realm Specific Address IP (RSA-IP) suffers from two of the same short comings as NATs. Firstly, as RSA-IP leases public IP addresses similar to classical NAT, it does not scale. Secondly, it does not allow public realm initiated connections as the public addresses are dynamically assigned. To overcome the address limitation, similarly to NAPT, Realm Specific Address and Port IP (RSAP-IP) has been proposed [RFC 2663]. RSAP-IP operates similar to RSA-IP except that RSAP-IP clients are assigned ports as well as a public IP address. This way, several RSAP-IP clients can share the same public IP address. The multiplexing is performed through adding both an IP tunnel and an additional transport header with the assigned port as identifier, if the public host terminates the tunnel. If the RSAP-IP server terminates the tunnel, the multiplexing is based on the combination of destination address, and port number.

## Comparison

Table 1 compares the existing methods in terms of scalability, the need for using 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.

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 1, a comparison of NAT solutions**

As can be seen from the above simple table, none of the existing proposals meets all the demands. The following section describes a solution that would meet the demands, and act as an enabler of services until IPv6 has gained sufficient momentum to overcome the address shortage.

### **Realm Based Kluge Address Heuristic-IP (REBEKAH-IP)**

Our proposal, REBEKAH-IP, illustrated in Figure 4 integrates and extends functionality provided by RSIP and Bi-directional NAT into a REBEKAH-IP server as follows. To eliminate the need for ALGs for each application, REBEKAH-IP uses the client server architecture of RSIP. REBEKAH-IP client configuration is done using the RSIP configuration signalling illustrated in Figure 3. To enable, public realm initiated communication it uses a DNS specific ALG similarly to Bi-directional NATs. As the DNS-ALG is used in conjunction with the RSIP client server architecture, REBEKAH-IP only requires this single ALG.

The extensions address the issues associated with the RSIP servers use of a pool of public IP addresses. The address expansion in REBEKAH-IP is not achieved by assigning ports to hosts as in NAT-PT or RSAP-IP. Instead, before a public IP address is assigned to a private host a sender and receiver IP address, and a sender and receiver port number combination that is unique is found

The choice of RSIP implies changes to the terminal equipment to incorporate an RSIP client. However, in the case of 2.5 and 3G network clients, we are at a phase of introduction of new equipment that can be furnished and configured without being backwards compatible. Thus the incorporation of a RSIP client we believe does not pose any great barriers.

In the following sections we describe the operation of REBEKAH-IP by considering communication initiated from within a private realm and a public realm respectively.

### **Private realm initiated communication**

When a private host *A* wants to connect to a public host *D*, if *A* has already been assigned a public address by the REBEKAH-IP server it uses this for the communication. Otherwise, *A* sends a request for a public IP address to the REBEKAH-IP server. The server in turn queries the DNS-ALG for an address. The DNS-ALG has the pool of public addresses available for use organised into a list. When a request for a IP address is received from the REBEKAH-IP server, it simply selects an address from the list in a round robin fashion. Then DNS-ALG makes a binding of the FQDN of host *A* to the chosen assigned public address so that subsequent DNS requests from the public Internet will return this address. Once this is done, it informs the RSIP server the public IP address that has been assigned host *A*. Finally, host *A* and the REBEKAH-IP server establish an IPv6 tunnel between themselves to route the IPv4 traffic through the IPv6 domain.

With the round-robin allocation of IP addresses at the DNS-ALG, when the end of the address list is reached, it starts assigning addresses from beginning of the list. This wrapping around can potentially lead to address collision, as the same address can be assigned to two or more clients. To eliminate address collision, REBEKAH-IP uses a combination of IP addresses and port numbers to distinguish between sessions.

The REBEKAH-IP server is able to separate all traffic originating from hosts within the IPv6 domain since these hosts use different tunnels. For the return path traffic, the routing decision is made using four parameters, namely sender IP address and port number, and receiver IP and port number. As long as the combination of these four parameters is kept unique, the REBEKAH-IP server can forward the traffic to the correct tunnel.

To ensure this, when a client requests to use a public IP address from the REBEKAH-IP server the server checks the routing table to determine whether a combination of the selected source and the wanted IP addresses is in use. If it is, the server works with the DNS-ALG to find the address with the least number of previous connections to the destination address to minimise the chance of collision. Once the address is assigned and if the server detects a clash when the first packet arrives at the server and the sender and receiver ports become known to it, the server notifies the client and resets the connection, requesting the client to reopen the socket with a new port number.

### **Public realm initiated communication**

When a host on the public Internet wants to contact a host in a private realm, similarly to bi-directional NAT, a DNS request is first issued to the DNS-ALG of the

REBEKAH-IP server. The server simply has to make sure that there is no pending request between the IP address being assigned to the host in the private realm and the public IP address of the querying host. By pending request we refer to a DNS lookup resulting in an address assignment for which no data has actually reached the server yet. This is because it is not until the first data packet arrives at the server that the server will be able to determine the sender and receiver port numbers and therefore can map the combination of sender and receiver addresses and port numbers to the correct tunnel of the private host. Once the mapping has been done the combination will be unique and therefore collisions will be avoided for the following reasons:

For a host that uses a unique public IP address there is no chance of collision as long as connections to private hosts are opened sequentially. If sessions are opened concurrently, there is a very small possibility that the DNS gets requests for two private terminals with the same assigned public IP address. In this case, the DNS can avoid collisions by not replying to the second query, until the pending flow parameters from the first query (IP addresses and port numbers) have been identified. For the querying host this will be seen as a delay in the second DNS lookup.

For hosts behind existing NATs there is no chance of address collision because all versions of NATs result in unique combinations of sender address and/or port numbers

For hosts in two separate IPv6 domains behind REBEKAH-IP servers, IPv6 routing combined with tunneling across the Internet is used instead of IP address translation.

Thus, as with Bi-directional NAT, public realm initiated communication will be enabled using the DNS as key function. This can in certain cases introduce delays in DNS replies if the server waits for pending requests to be resolved before answering new requests. However, the delay will not occur if a host is making requests sequentially.

### **Experimental results**

The viability of REBEKAH-IP will depend on its scalability. To assess the scalability of REBEKAH-IP a mathematical modelling was used to estimate the call blocking probability due to un-resolvable address ambiguity.

The composition and traffic pattern of Internet connections is still a largely unexplored area and certainly the composition of data traffic to and from cellular devices is unknown. Therefore, the experiments, focused on a worst-case scenario where all connections from private hosts are made to the same public IP address using the same port number. If the scalability of the scheme is satisfactory under these conditions, it could be assumed that the scalability will be satisfactory for all cases.

In our initial work, we sought an indication of the scalability without any queuing optimizations. We therefore investigated the simplest case of call blocking probability due to at least two hosts opening sockets during an overlapping time period and with the same ephemeral (sender) port number. We therefore used the following simple function:

$$P_{(col,N,N_{IP})} = \left( \frac{\lambda t}{m^2} \right) \cdot \frac{N}{N_{IP}}$$

$P_{(Col,N,N_{IP})}$  is the call blocking probability,  $\lambda$  is the average number of sockets opened per host and unit time,  $t$  is the average time each socket remains open,  $m$  is the number of available ports,  $N$  is the number of private realm hosts and  $N_{IP}$  is the number of public IP addresses.

The assumption is that the number of ephemeral ports is  $2^{16}$  - the first 1024 ports that are reserved ports [IANA]. It was also assumed that ephemeral ports are randomly selected. Different operating systems use different schemes for selecting ephemeral ports, so with existing operating systems this does not hold true. However, for future cellular terminals it is easy to implement random ephemeral port allocation to obtain a better spread of the port space, hence minimising the risk of collision. The  $m^2$  relation is the probability of two hosts randomly selecting the same port number.

### **Input parameters**

In order to obtain realistic input values to the model, we examined Internet traffic traces from University of California at Berkeley [Berkeley]. The data contains traces of web traffic activity from the university during a period in 1996. From these traces we extracted the number of hosts in the network, the average number of opened sockets per minute, and host and the average duration of each connection during the busiest period in the data set. We selected web traffic traces since this type of traffic is the most likely candidate from cellular terminals. It is unlikely that this type of terminal will largely be used for large file transfers such as FTP sessions or file sharing services.

### **Results**

From the traces we found that the number of active hosts during the period was 360, the average number of opened sockets per second was 0.015 and that the average duration of each connection was 17 seconds. The duration reflects a faster network connection than that of 2.5G and initial 3G services. However, it is likely that users will use the cellular connections more conservatively and that the content will be smaller and more suited to cellular terminals, both lowering the utilisation of the network.

Using the formula above and scaling the input parameters to the range of one to ten million hosts we obtain the call blocking probability for REBEKAH-IP. Figure 5 shows the call blocking probability depending on the number of hosts in the private network with 5000 and 10000 public IP addresses respectively.

### **Discussion**

When moving to a public cellular environment, users will want to be able to use the same services they use with their fixed connections. Even if applications behave differently, it is vital not to prevent users from using them in cellular networks. REBEKAH-IP was designed to meet this demand whilst ensuring the support of three

main criteria, namely scalability, public realm Initiated communication and application friendliness.

### **Scalability**

The limitations of REBEKAH-IP are hardware capacity and call blocking probability due to irresolvable address ambiguities.

How computation-intensive the RSIP function really is remains an open issue. However, REBEKAH-IP is more scalable than NAT, since the server escapes re-writing headers and re-calculating checksums.

In addition, from the results above, we can see that even in the worst-case scenario, the REBEKAH-IP's scalability is very promising. The call-blocking probability due to irresolvable address ambiguity is far less than the call-blocking probability of both cellular networks [ETSI] and that of servers in the Internet [Arlitt96]. Therefore, for short-lived connections, such as retrieving web pages and polling mail servers, REBEKAH-IP will not degrade the level of service to end-users.

### **Public Realm Initiated Communication**

REBEKAH-IP allows public realm initiated communication. The only limitation is that the method builds on the mapping of FQDN to public addresses. Therefore, it is impossible to make a connection directly to an IP address through the RSIP server. This limitation is of minor significance since the communication between a mobile host and any host on the public Internet, mobile or fixed is unlikely to require this type of connection.

### **Application Friendliness**

Unlike NATs, the method does not require the rewriting of any headers. Similarly to RSIP, there is no need for ALGs to be deployed to deal with application IP address signalling. This enables REBEKAH to support a much larger set of applications.

The major drawback with ALGs is that they need to be deployed for each application. Even if application developers could be trusted to provide ALGs for their applications the operators would have to test the code thoroughly before installing it into their systems, making sure they comply with their environment. Furthermore, it would not be possible for a single party to take on the role as total system integrator. Thus distributing and maintaining ALGs will at best become extremely cumbersome for the operators.

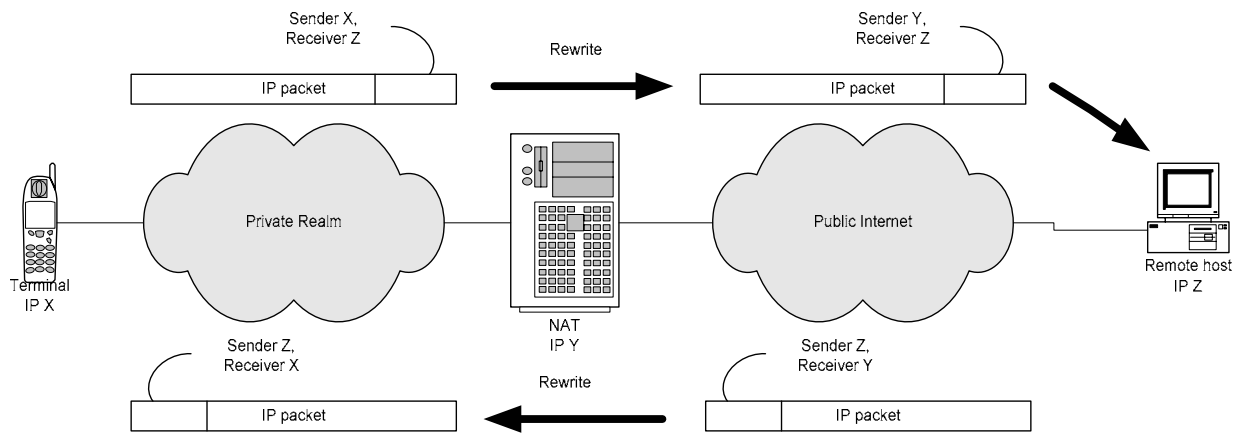
### **Conclusion**

In this article, we have discussed the problem with the limited IPv4 address space for wide deployment of Internet services in public cellular networks, and gave an overview of the existing solutions to this problem. We argued that the existing solutions do not fully meet the requirements of these networks. To overcome these limitations, we proposed a new method, REBEKAH-IP, which combines RSIP, and Bi-directional NATs with extensions to the address allocation mechanisms. Finally, using a simple analytical model, it was shown that REBEKAH-IP, will scale sufficiently to satisfy the needs of the emerging cellular networks.

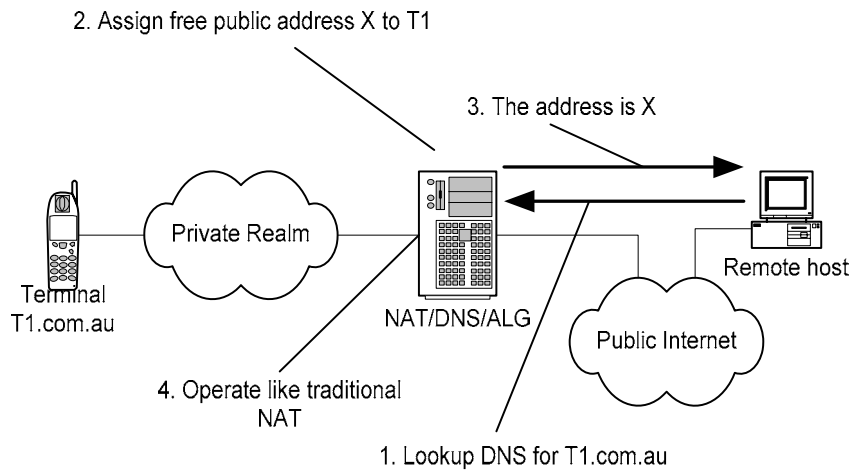
Many issues that apply to NATs [RFC 2993] also apply to REBEKAH-IP. Even though the application support is better than for previous suggestions, a full understanding of the implications of introducing large-scale NAT networks will not be achieved until they are deployed. We believe that a revised version of IP, such as IPv6, will be a better method for solving the address space shortage but in the interim period, REBEKAH-IP will meet the crucial demands for operators, users and applications.

## References

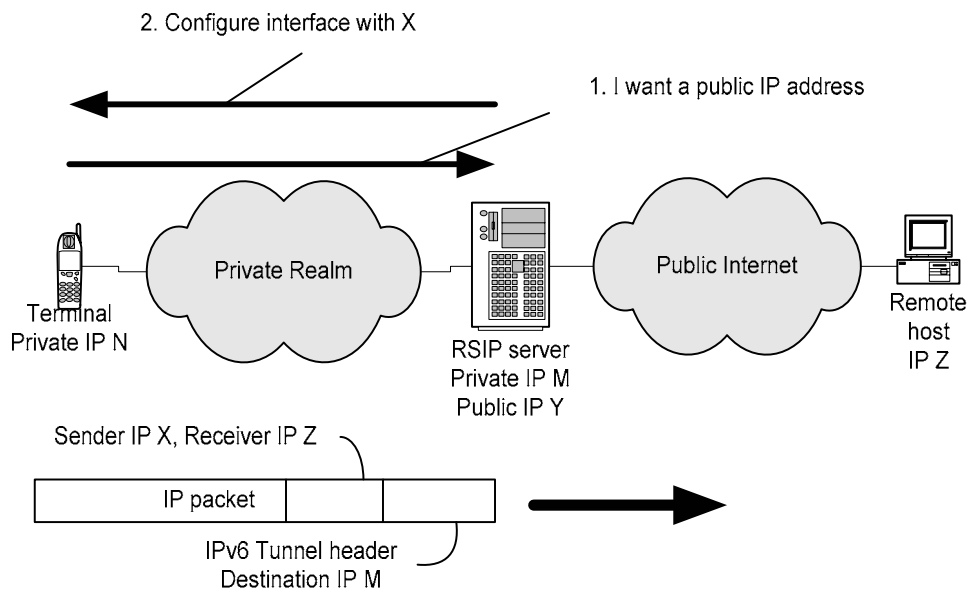
- [Berkeley] <http://ita.ee.lbl.gov/html/contrib/UCB.home-IP-HTTP.html>
- [IANA] <http://www.iana.org/assignments/port-numbers>
- [RFC 1631] “The IP Network Address Translator (NAT)”, K. Egevang, P. Francis May 1994
- [RFC 1918] “Address Allocation for Private Internets”. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear.
- [RFC 2663] “IP Network Address Translator (NAT) Terminology and Considerations”. P. Srisuresh, M. Holdrege. August 1999.
- [RFC 2766] “Network Address Translation - Protocol Translation (NAT-PT)”. G. Tsirtsis, P. Srisuresh. February 2000.
- [RFC 2993] “Architectural Implications of NAT”. T. Hain. November 2000.
- [RFC 3235] “Network Address Translator (NAT)-Friendly Application Design Guidelines”. D. Senie. January 2002.
- [Arlitt96] M. Arlitt and C. Williamson, “Web Server Workload Characterization: The Search for Invariants”, In Proc. Of ACM SIGMETRICS, Pages 126-137, Philadelphia, PA, USA, April 1996
- [ETSI] “Universal Mobile Telecommunications System (UMTS) : Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.1.0)”, TR 101 112 V3.1.0 (1997-11)



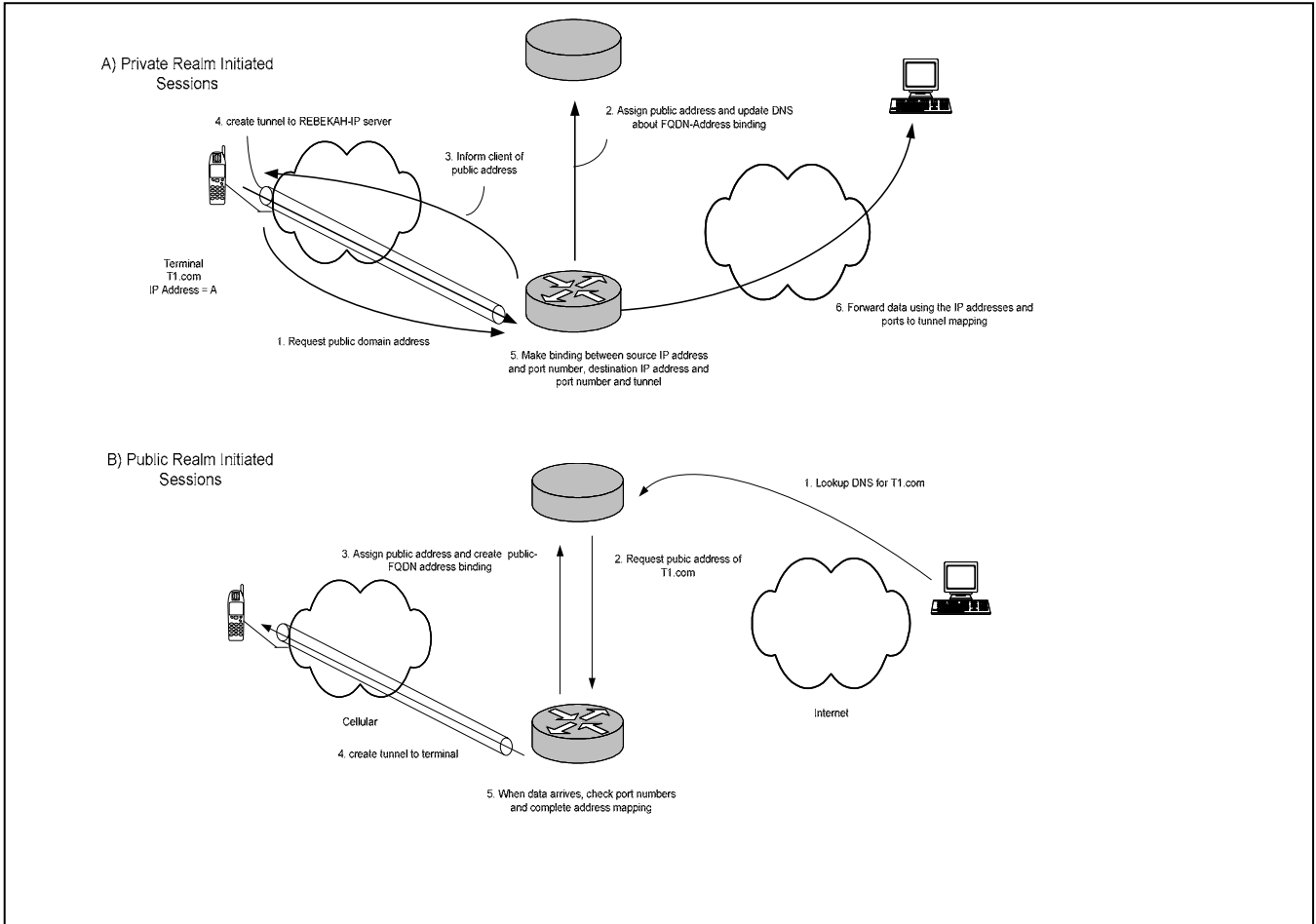
**Figure 1. Operation of Basic NATs**



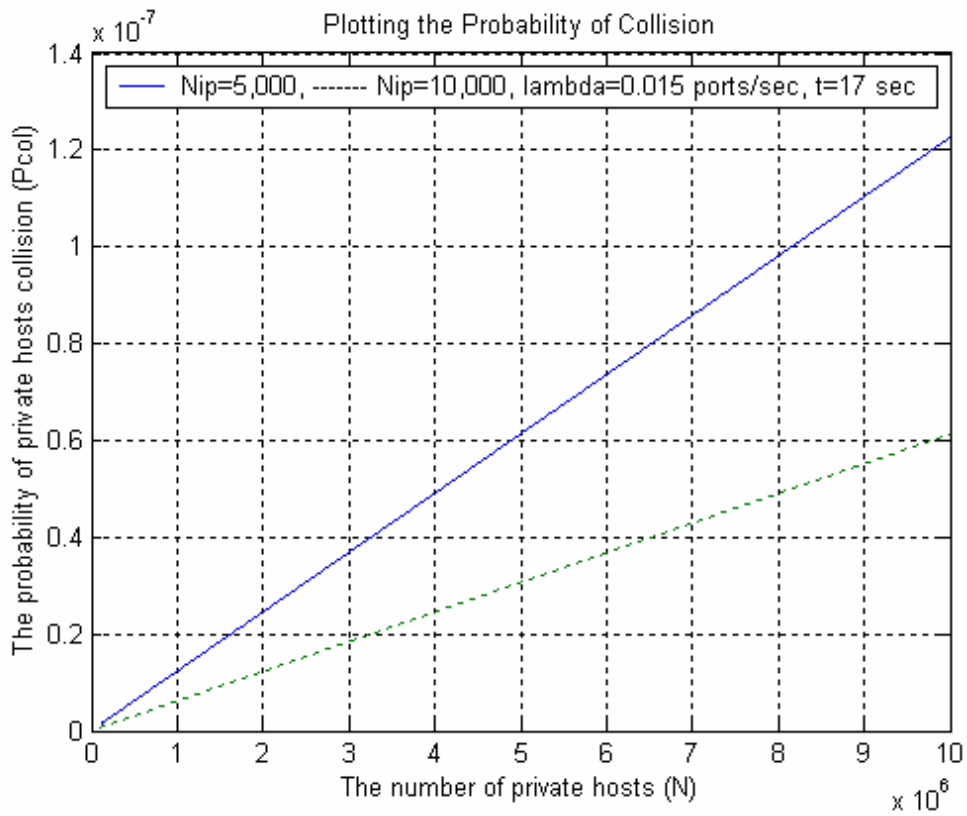
**Figure 2. Operation of Bi-directional NATs**



**Figure 3. Operation of RSIP**



**Figure 4. REBEKAH-IP components**



**Figure 5. Call blocking probability using REBEKAH-IP**