THE AUSTRALIAN NATIONAL UNIVERSITY

TR-CS-97-19

Implementation Of A Portable-IP System For Mobile TCP/IP

Xun Qu, Jeffrey Xu Yu and Richard P. Brent

November 1997

Joint Computer Science Technical Report Series

Department of Computer Science
Faculty of Engineering and Information Technology

Computer Sciences Laboratory
Research School of Information Sciences and Engineering

This technical report series is published jointly by the Department of Computer Science, Faculty of Engineering and Information Technology, and the Computer Sciences Laboratory, Research School of Information Sciences and Engineering, The Australian National University.

Please direct correspondence regarding this series to:

Technical Reports
Department of Computer Science
Faculty of Engineering and Information Technology
The Australian National University
Canberra ACT 0200
Australia

or send email to:

`Technical.Reports@cs.anu.edu.au`

A list of technical reports, including some abstracts and copies of some full reports may be found at:

`http://cs.anu.edu.au/techreports/`

Recent reports in this series:

TR-CS-97-18 Richard P. Brent. *Stability of fast algorithms for structured linear systems*. September 1997.

TR-CS-97-17 Brian Murphy and Richard P. Brent. *On quadratic polynomials for the number field sieve*. August 1997.

TR-CS-97-16 M. Manzur Murshed and Richard P. Brent. *Algorithms for optimal self-simulation of some restricted reconfigurable meshes*. July 1997.

TR-CS-97-15 Peter Strazdins. *Reducing software overheads in parallel linear algebra libraries*. July 1997.

TR-CS-97-14 Michael K. Ng and William F. Trench. *Numerical solution of the eigenvalue problem for Hermitian Toeplitz-like matrices*. July 1997.

TR-CS-97-13 Michael K. Ng. *Blind channel identification and the eigenvalue problem of structured matrices*. July 1997.

Implementation of A Portable-IP System For Mobile TCP/IP

Xun Qu
Computer Sciences Laboratory, RSISE
The Australian National University
Canberra, ACT 0200 Australia
E-mail: **quxun@cslab.anu.edu.au**

Jeffrey Xu Yu
Computer Science Department
The Australian National University
Canberra, ACT 0200 Australia
E-mail: **yu@cs.anu.edu.au**

Richard P. Brent
Computer Sciences Laboratory, RSISE
The Australian National University
Canberra, ACT 0200 Australia
E-mail: **rpb@cslab.anu.edu.au**

Abstract

In the current TCP/IP-based systems, the hardness of mobile communication stems from the IP routing and addressing schemes. In this paper, we refine our mobile solution using two mappings: namely, a portable-IP mapping and a mobile mapping. We also explicitly distinguish our two mapping approach from the existing mobile-IP solutions which are of a single mapping scheme. The portable-IP mapping is supported using DHCP and DNS functions and requiring no modifications to the TCP/IP protocols. The mobile mapping is implemented in the socket layer. Our mobile solution achieves high compatibility with the current TCP/IP systems mainly because that no special functionality is required to support mobility in the network and the transport layers. In terms of performance, our mobile solution can reduce both the cost for distributing the location information of mobile hosts and the cost for forwarding IP datagrams across the Internet. In this paper, we also focus on the implementation details of the portable-IP system for supporting the portable-IP mapping. Two registration procedures and DNS updating approaches are given. We show that using the new BIND version 8.1.1 can avoid to reload the whole DNS database frequently.

Keywords: mobile communication, portable communication, TCP/IP.

1 Introduction

In the recent years, demands for supporting mobile communications in the Internet are growing rapidly. The mobile communication implies a continuous network connection and continuous communication services[1, 2]. By the continuousness, it means that the communication services will not be interrupted due to any movement of mobile hosts, and shall remain the same as the system is connected to a single fixed point in a distributed environment.

Providing mobile communications in the Internet has certain fundamental difficulties, since current TCP/IP protocols have not taken mobility into consideration. Therefore, a TCP/IP-based host system can only reside at a fixed point in the Internet. There are two non-mobile solutions to allow a host to work in a guest network environment. The first is to connect the host back to its home network via a low-bandwidth telecommunication line. The other is to reconfigure the host system in the guest network environment. The first solution does not make use of the local communication facilities efficiently. The second solution is unacceptable [2], since not all users are able to reconfigure host system manually. A system which is assigned to a new identifier in the guest network environment requires that all existing applications on this system need restarting and even reconfiguration.

Truly ubiquitous mobile computing requires that programs on mobile computers be able to process continuously, in a similar fashion to a distributed environment. There are four major mobile IP solutions [2], including Sony Virtual Internet Protocol [15], IBM I&II Mobile Host Protocol [2], Columbia Mobile IP protocol [1] and Internet Mobile Host Protocol (IMHP) [3]. IPng [11] is also supposed to support mobility.

However, we differentiate between mobile and portable communication services [6]. A portable communication service is intermittent which means that no communication services can be provided during the period of moving. A mobile communication service can keep all current open connections alive and provide continuous communication services. In [4], we proposed the notions and concepts of the portable communication services, and outlined the ideas on how to implement them. In brief, the portable communication services mainly employ the existing protocols, such as DHCP and DNS, and do not need any enhancements in the TCP/IP layers. In [6], we proposed a mobile communication service, mobile TCP/IP socket, on top of our portable communication services. Since our mobile communication services can be implemented in the socket layer and on top of the TCP/IP layers, our mobile solution can achieve high compatibility with the current TCP/IP systems for the following three reasons. First, we adopt the same API of the socket layer. Second, we don't require any special functionality in the network and the transport layers to support mobility. Finally, we don't require all networks to support special routing mechanisms. In terms of performance, our mobile solution can reduce both the propagation cost for distributing the location information of mobile hosts and the forwarding cost for forwarding IP datagrams across the Internet.

In this paper, we refine our mobile solution using two mapping functions, namely, a portable-IP mapping and a mobile mapping, and explicitly distinguish between our mobile solution and the other mobile-IP solutions. We also focus on the implementation details of the portable-IP system to support the portable-IP mappings.

The remainder of this paper is organised as follows. In Section 2, preliminaries are given. Section 3 addresses three invariant mobile conditions. In Section 4, we discuss the existing mobile-IP solutions and the main issues we are concerned with. Section 4 refines the two mappings, namely, the portable-IP mapping and the mobile mapping. In Section 6, the implementation issues for supporting portable-IP mappings are addressed. We conclude our work in Section 7.

2 Preliminaries

In the TCP/IP systems, a "name" is used to identify an object in a network system, such as a protocol entity, a whole system, etc. Whereas an "address" is a special name which is used to locate the named object in the network system. In the Internet context, all DNS names, IP addresses and port numbers are extremely important when supporting mobile communications. Here, a DNS name is used as a literal identifier of a TCP/IP-based system. Although a DNS name has a hierarchical naming structure, it is not an address to be used to locate a host in the Internet. Whereas an IP address is the only means in the low three layers of TCP/IP protocol stack to locate a system or a protocol entity. The Internet DNS services only map a DNS name to an IP address and vice versa. Usually, a host system has only one IP address, and no two systems can share a single IP address.

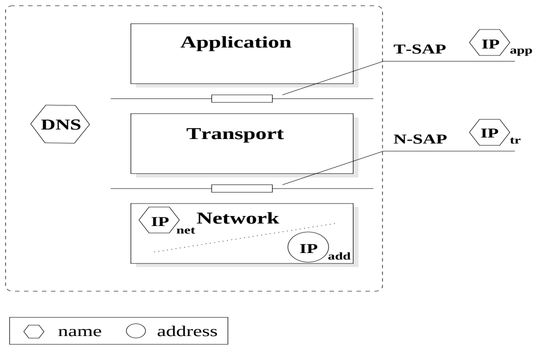

Figure 1: Functions of IP Address

In the ISO/OSI seven-layer model, both connection-oriented and connectionless communications logically occur between the two peer protocol entities in the same layer. Each entity makes use of the services provided in the underneath layers and in return provides services to its upper layer through service access points (SAP). A protocol entity is identified by the lower layers SAP address plus its internal identifier which can be either a name or an address.

As shown in Figure 1, an IP serves two functions, namely, naming an object such as a SAP, and routing IP datagrams through the Internet.[1] Applications can use a domain name to identify a TCP/IP system. Internally, an application is uniquely identified by a T-SAP (the SAP of the transport layer), which includes an IP address (denoted IP_{app}), a process id, a port number, and a transport protocol. A transport entity can be defined through its N-SAP (the SAP of the network layer), which includes an IP address (denoted IP_{tr}), and a protocol number. In the network layer, an IP address is used as both a name (denoted IP_{net}) to identify an IP protocol entity and an address (denoted IP_{add}). The IP_{add} is the IP address involved in the routing mechanism, and appears in the *destination address* field of normal IP header.

[1]Routing implies finding the location of the entity the IP represents.

3 Three invariant conditions for mobile communications

The entities in the application layer are the ultimate source, and sink of data to the underlying layers at which communication services are provided. A mobile system can be defined as a system that all applications can transparently obtain mobile communication supports and will not be aware of the motion of the system. Three basic invariant conditions have to be satisfied:

- constant names for an application entity, including a DNS name and a unique IP address [8];
- a constant T-SAP address including a constant IP address and a port number. The IP address (IP_{app}) and the port number must not be changed so as to make the communication sessions intact; and
- the same data transfer semantics. [2]

The first condition implies that applications need not reconfigure when mobile systems change their locations. The hard-coded or configurable IP addresses and domain names must be valid always. The second condition is essential which implies the first condition. The third condition sets up a guideline for the implementation issues. Two approaches have been proposed for meeting the second invariant condition:

- to support mobility in the network layer. Transparent mobile support is provide at the N-SAP level. Both transport layer and application layers can obtain mobile communication support from the network layer. The mobile operations will not affect any work conducted in the transport layer while the system is moving. That means no need to reconfigure in the transport layer and the application layer.
- to support mobility in the transport layer. Transparent mobile support is provided at the T-SAP level in conjunction with a portable-IP system as we proposed in [4, 6]. We will discuss it in detail later in this paper.

4 Support mobility in the network layer

Most current research work has been conducted using the first approach. The main reason is that both addressing and routing are handled in the network layer. Therefore, it is practical and efficient to provide mobility in the network layer [1]. To allow all applications to run at a new location properly, the following must be satisfied.

$$IP^h_{app} = IP^h_{tr} = IP^h_{net}$$

Here, we use IP^h_x to refer to an IP at the "x" layer in the home network. In a similar fashion, we use IP^g_x to refer to an IP in the layer "x" in a guest network. The above implies that the three IPs remain the same. Moreover, in a guest network environment, this approach attempts to reuse the IPs used in its home network: $IP^h_{app} = IP^g_{app}$, $IP^h_{tr} = IP^g_{tr}$, and $IP^h_{net} = IP^g_{net}$. The main concern then is how to deal with IP_{add} which must be set up to reflect the current location of the system in order to send/receive IP datagrams. There are two methods to achieve the dynamic allocation.

The first method is to use the following IPs, $IP^h_{app} = IP^h_{tr} = IP^h_{net} = IP^h_{add}$, in the guest network. In addition, the supporting system maintains an association between the IP^h_{add} and a local *forward point* (denoted P_{fwd}). The local forward point is responsible for forwarding and routing packages to the mobile system. In general, there exists a group of forward points or a group of special mobile support routers (MSRs). The mapping of IP address can be formulated as follows:

$$IP^h_{app} = IP^h_{tr} = IP^h_{net} = IP^h_{add} \Leftrightarrow \{P_{fwd_1}, P_{fwd_2}, \cdots\}$$

The peer system always uses the same IP address, IP^h, to transfer packages to the correspondent mobile system. These packages will be interpreted by any P_{fwd} and will be routed to the correct location. Columbia Mobile IP protocol [1] and Sony Virtual Internet Protocol [15] solutions belong to this category. The MSRs can be viewed as a *virtual network* for all mobile systems. Packages to the mobile systems will be routed to one of MSRs using normal routing infrastructure, and will be handled over to the mobile system by any MSR. The packages can be routed along optimal paths. The limitations of this method are manifest: the scope of virtual network limits the possible moving area of mobile systems.

The second method is to decouple the functions of IP address. All conceptual IP address for the naming purposes are kept the same other than IP_{add} which is used for addressing and routing. A system can be dynamically associated with a local IP address for the routing purposes. The mapping is given as follows.

$$IP^h_{app} = IP^h_{tr} = IP^h_{net} \Leftrightarrow IP^g_{add}$$

[2]For example, an SAP of TCP protocol must ensure reliable data transfer regardless of system moving.

Following this scenario, the peer system sends data to a mobile system using the constant home IP address, IP^h_{add}. Hence the data will reach the home network of the mobile system at first. Then, the address mapping takes place at the home mobile support system. And data will be re-encapsulated and forwarded to IP^g_{add}. The IP^g_{add} can be managed in a special router, or even on the mobile system itself. The special router decapsulates the packages, puts the original IP address, IP^h_{add}, back to the IP header, and delivers them locally to the mobile system. The only necessary forward point in this method is the home mobile support system. However, the routing is triangle and the cost of communication is expected to be high. IMHP is such a solution.

Two issues

All the mobile solutions mentioned above aim to hide the mobile operations from the transport and application layers. The mobile communications will be supported in the network layer. All DNS domain names and IP addresses in the transport and application layers remain unchanged. However, the location information is reflected by the local IP addresses of mobile systems. If we use multiple forward points, the routes for sending IP packages to a mobile system can be reduced. However, the location information of all mobile host systems running in the networks need dispatching among all the forward points. In practice, the more accurate location information the forward points are aware of, the shorter distances the IP datagrams can be routed to the mobile systems. It is obvious that special protocols are needed to maintain location information. The first issue is the *compatibility* with the current routing facilities. It also takes system resources, such as network bandwidth and CPU time, in order to send and process location information. On the other hand, if we attempt to use only a few forward points, IP datagrams can take much long routes from the source to the mobile destination. Using IMHP as an example, if there is only one forward point, only the home forward point needs to keep location informing when mobile systems change their locations. However, the triangle routing problem becomes a key issue. If a local host needs to talk to a mobile system, which is far away from its home network, all IP datagrams from the local system will be routed to the home network of the mobile system at first, and then be forwarded to the mobile system.

In summary, all existing mobile-IP solutions have faced two problems: full compatibility with the existing routing protocols and inefficient use of existing routing facilities. The former is caused by the introduction of the special routing schemes, and the latter is caused by IP triangle forwarding [5]. Our two mapping solution attempts to alleviate both problems.

5 Support mobility in the transport layer: our two-mapping solution

To avoid the two problems mentioned in the previous section, we attempt to use a different way to support mobility. First, the IP^h_{app} and the DNS name will be kept constant to satisfy the first two invariant conditions. Second, the usages of IP addresses are decoupled at the boundary between the application layer and the transport layer instead. All the remaining IP addresses, $IP^g_{tr} = IP^g_{net} = IP^g_{add}$, will be dynamically assigned to a new IP at a new network environment. The main reason for doing so is that using a new IP in a new guest network environment ensures that the current routing infrastructure will be efficiently used. The new local IP address is used not only for the addressing purposes in the network layer, but also for naming N-SAP and a transport entity. The dynamic mapping between IP^h_{app} and IP^g_{tr} will take place in the transport layer.

$$IP^h_{app} \Leftrightarrow IP^g_{tr} = IP^g_{net} = IP^g_{add}$$

In addition, the DNS domain name and the IP^h_{app} shall be able to be mapped onto IP^g_{tr} and vice versa. We call this mapping a *portable-IP mapping.*

In addition to the portable-IP mapping, the mobility is supported through a mobile mapping at the communication session level. Recall that an application communication session, denoted S_{app}, is presented by two peer T-SAPs:

$$\begin{aligned} S_{app} &= (\text{T-SAP}_a, \text{T-SAP}_b) \\ &= (protocol, IP_{app_a}, Port_a, IP_{app_b}, Port_b) \end{aligned}$$

This S_{app} between the pair of T-SAPs will be carried on by a underlying transport session between two transport entities, denoted S_{tr}, as follows:

$$S_{tr} = (protocol, IP_{tr_a}, Port_a, IP_{tr_b}, Port_b)$$

In traditional TCP/IP systems, all these conceptual IP addresses are identical, hence the mapping between S_{app} and S_{tr} is one-to-one as follows:

$$\begin{array}{ccc} S_{app} & = & S_{tr} \\ \updownarrow & & \updownarrow \\ IP_{app} & = & IP_{tr} \end{array}$$

However, in a mobile environment, if we keep IP_{app} unchanged, but assign a new IP to IP_{tr} when a mobile host moves into a new guest network environment, S_{tr} will be lost due to the change of IP_{tr} consequently. Our solution to this problem is to map a S_{app} onto a underlying transport session, S_{tr}, at any time, and to map a S_{app} to multiple S_{tr} during the life time of a communication session, as shown in Figure 2.

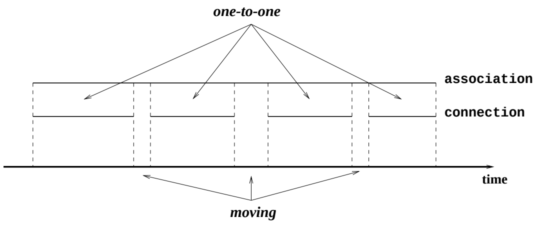

Figure 2: Mobile Session Mappings

For further discussion, the above sessions are explicitly defined as follows.

$$\begin{aligned} S_{app} &= (\text{T-SAP}_a, \text{T-SAP}_b) \\ &= (protocol, IP^h_{app_a}, Port_a, IP^h_{app_b}, Port_b) \\ S_{tr} &= (protocol, IP^g_{tr_a}, Port_a, IP^g_{tr_b}, Port_b) \end{aligned}$$

Using our mobile solution, the relationships among IP addresses and sessions are as follow:

$$\begin{array}{ccc} S_{app} & \Leftrightarrow & \{S_{tr_1}, S_{tr_2}, \cdots, S_{tr_n}\} \\ \updownarrow & & \updownarrow \\ IP^h_{app} & \Leftrightarrow & \{IP^g_{tr_1}, IP^g_{tr_2}, \cdots, IP^g_{tr_n}\} \end{array}$$

Here, $IP^g_{tr_i}$ is the same IP^g_{tr} used in the i-th S_{tr} session. The *mobile mapping*,

$$S_{app} \Leftrightarrow S_{tr}$$

takes place in the socket layer as discussed in [6] in detail. The portable-IP mapping ensures that mobile systems can properly work in any networks in the Internet, and the mobile mapping ensures that continuous communication services are provided. The advantages of the two-mapping method are obvious. First, the system will be fully compatible with the current routing systems and require no additional routing mechanisms. The second is that IP datagrams are routed along with the optimal path[3].

6 Implementation issues

6.1 System overview

The outline of our mobile system is schematically shown in Figure 3. In the socket layer, the mobile mapping is supported using a virtual port. In brief, a virtual port is introduced between S_{app} and S_{tr}. A virtual port acts as a S_{tr} from the viewpoint of S_{app}. When an application binds a socket to a S_{tr}, We create a virtual port, and bind the socket to it accordingly. All code above the virtual port is the same as used in the current socket layer, and the extensive mobile functionality is hidden from the virtual port. In order to satisfy the last invariant condition, three implementation issues are taken into consideration: how to distinguish old connection from new connection, how to keep the mobile TCP socket I/O semantics the same, and how to recognise non-mobile and mobile TCP services. The three issues are fully discussed in [6].

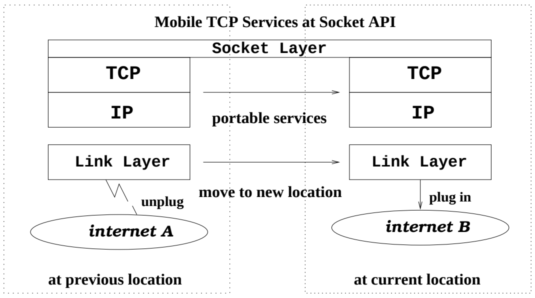

Figure 3: Two-Mapping Solution

The portable-IP system enables a host system to work through a connection at any attach point in the Internet. This portable-IP system shall communicate with its peer systems without any further limitations except for the intermittent communications when moving from a place to another. In our portable-IP system, there exist three subsystems:

- a portable host system (PH), which will move around the Internet and always obtain a data link connection locally;
- a home portable support system (H-PSS), which will trace the locations and the system settings for all PHs that are originally registered in its domain; and
- a foreign portable support system (F-PSS), which will host PHs when they move into its domain, and provide necessary system setting information to allow the PHs to gain network connections.

6.2 Portable-IP support

The core responsibility of the portable-IP system is to manage the dynamic mapping $IP_{app} \Leftrightarrow$

[3] The Internet routing protocols ensure all datagrams will be routed along "optimal" path in terms of the quality of communication link and routing policy.

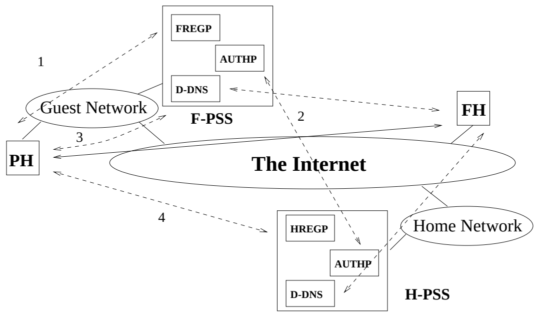

Figure 4: Portable-IP System

$IP_{tr} = IP_{net} = IP_{add}$. Here, the dynamic mapping implies the following three issues:

- allocation of a new IP address, $IP_{tr} = IP_{net} = IP_{add}$, for a PH. This enables the PH to obtain a new local IP address for network communications;

- dynamic DNS mapping in the H-PSS. This enables applications to use the original domain names and/or the constant IP addresses IP^h_{app} to identify and to locate the PHs. DNS services need correctly map the domain name and the constant IP address IP^h_{app} to the local IP address IP^g_{tr} which are dynamically assigned to by the F-PSS; and

- dynamic reverse mapping. The local IP address IP^g_{tr} should be correctly mapped back to the original domain name.

These tasks will be completed by two registration procedures, namely registration with H-PSS and registration with F-PSS, which will be discussed in the following sections.

A typical working procedure for a PH to be dynamically set up is shown in Figure 4:

- Step 1: After arriving at a guest network, a PH registers itself with F-PSS to ask for a local IP address, IP^g_{tr}, and other system parameters (such as default router, MSU etc.) using a FREGP protocol which is a modified version of DHCP[10]. The identifier of this PH together with its domain name and the IP address, IP^h_{add}, are singed using the PH's secret key;

- Step 2: F-PSS uses AUTHP to authenticate the PH with its H-PSS. In our prototype system, AUTHP is simplified. F-PSS uses DNS KEY query to ask H-PSS for the PH's public key, then decrypts and checks the signature;

- Step 3: If the result of authenticate checking is valid, F-PSS grants a local IP address, IP^g_{tr}, and sends it with all other information to the PH. The response is encrypted using the PH's public key. At the same time, F-PSS updates its DNS database to correctly respond any further queries for this PH;

- Step 4: The PH then uses HREGP to register itself with its H-PSS, which then updates its DNS database. The new location information will be distributed by the DNS systems immediately;

Afterwards, any system can obtain the local IP address, IP^g_{tr}, of the PH from the corresponding H-PSS or obtain its correct reverse DNS mapping (PTR query) from the corresponding F-PSS. Therefore, any system can then communicate with the PH using its current local IP address IP^g_{tr}.

6.3 Registration with F-PSS

The main purposes of registration with F-PSS are to authenticate a PH, and to allocate/assign a new IP address and other system parameters dynamically.

Before contacting to F-PSS, the PH has to start its TCP/IP system in a special mode, namely, the initialisation mode. The initialising procedure is given as follows. Here, we assume that the PH runs on top of Linux OS with one Ethernet NIC.

```
ifconfig eth0 down;
ifconfig eth0 0.0.0.0 mtu 1500 up;
route add -net 0.0.0.0 \
          netmask 0.0.0.0 eth0;
```

This procedure can be implemented using a shell script. The first line closes the Ethernet interface to make sure that all existing routings on the PH are cleaned. The second line brings the interface up with a special IP address, `0.0.0.0`, which is for a system to use before a valid IP address can be obtained (refer to the section 3.2 of [7]). The last line is to add a default route for sending all outgoing packages to the Ethernet.

Now the system is ready to send broadcast and unicast IP datagrams, and to receive all IP datagrams addressed to 0.0.0.0 with the broadcasting address. Then, the registration procedure can be started, during which the F-PSS registration protocol is involved for the PH to register itself to F-PSS and to obtain important system parameters, including:

- a local IP address and netmask;

- a default TCP TTL;

- local routers;

- local servers (optional), including the DNS name server, the timer server, and the line printer server (RFC 1179); and
- a local DNS name (optional).

The FREGP protocol we use is an extention of DHCP[10, 9]. The standard DHCP message format is employed and the procedure will follow the standard DHCP protocol. The additional functions are carried on through the options field. The protocol on the PH is given below:

- Step 1: Broadcast the `FREGDISCOVER` message to ask for a local F-PSS or a normal DHCP server, and wait for the `FREGOFFER` message from the server(s). If time-out, send `FREGDISCOVER` again. If too many re-sending have issued then abort. Otherwise go to Step 2.
- Step 2: If `FREGOFFER` messages have been received from multiple severs, pick up an arbitrary server, and send `FREGREQUEST` to it. Then wait for a message to be replied. If time-out and no response, try the next server until all the servers are attempted. If no server responds further, then abort. Otherwise go to Step 3.
- Step 3: When receiving `FREGACK`, the PH has to be reset according to the assigned system and environment parameters. If either `FREGNAK` or `FREGDECLINE` is returned from the selected server, go back to Step 2 to contact the next possible server.

F-PSS takes the following steps to respond requests from PHs:

- Step 1: Check the identifier of the PH specified in `FREGDISCOVER`, and authenticate it with the corresponding H-PSS. Send back all the assigned parameters in the encrypted `FREGOFFER`. If no `FREGREQUEST` response from the PH, then abort. Otherwise go to the next step.
- Step 2: Check all requested parameters. If approved, send back `FREGACK` and commit the dynamic registration. Otherwise, send `FREGNAK` to abort.

As can be seen above, four FREGDHCP messages are used. The basic format of these messages is as the same as the standard DHCP message format. If there is only one standard DHCP server, a PH will use DHCP to obtain the system settings. The basic structure of the FREG messages is given in Figure 5.

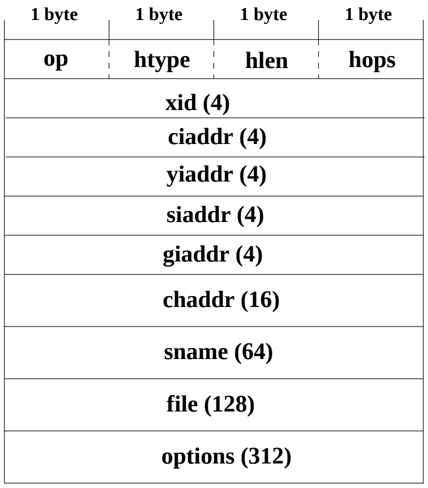

Figure 5: DHCP Message Format

Field	Value
op	1: BOOTREQUEST
htype	1: 10MB Ethernet
hlen	6
hops	0
xid	random number
secs	>= 0
flag	8000(hex): broadcasting

Table 1: Fields and Values in `FREGDISCOVER`

In the `FREGDISCIOVER` message, the PH fills in all the fields given in Table 1. The other fields will be set to null values except the options, in which a new *Hostid Option* must be provided. The format is shown in Figure 6. In this option, the domain

Figure 6: Hostid Option

name in the home network, the constant home IP address, IP^h_{app}, and the current time are given to F-PSS in order to check its sanity with the corresponding H-PSS via the standard DNS queries. In addition, there is a signature used to avoid bogus request, which is the encrypted MD5 result. We make use of the encryption library `libcrypto.a` which is distributed within *SSLeay* package. The following function:

```
unsigned char *
MD5(unsigned char *d, unsigned long n,
    unsigned char *md);
```

performs the calculation of the digest of the input message given in `*d` and puts 16 bytes output in `*md`. Then, the function:

```
int
RSA_public_encrypt(int from_len;
                   unsigned char *from,
                   unsigned char *to,
                   RSA *rsa);
```

is called to encrypt and decrypt the 16-byte digest. The PH will pass its secret key in `*rsa` to encrypt, and F-PSS will use the PH's public key to decrypt the digest.

After receiving a `DHCPDISCOVER`, F-PSS will abstract the domain name and the home IP address of the PH from the option field, and then use them to get the PH's public key from the PH's domain name server, which usually is its H-PSS and is requested to support `KEY` RR [12]. The public key is used to decipher the signature and check the MD5 digest. This checking cannot protect from the replaying recent `DISCOVER` messages. Therefore, F-PSS sends encrypted information in `FREGOFFER`. Its fields and values are listed in Table 2.

Field	Value
op	2: BOOTREPLY
htype	1: 10MB Ethernet
hlen	6
hops	0
xid	same as in `FREGDISCOVER`
secs	>= 0
flag	0: unicasting to PH
sname	server host name
options	Ph's local IP address, DNS name and others

Table 2: Fields and Values in `FREGOFFER`

Note that all the fields specified in the option will be encrypted. It is worth noting that all the options defined in [9] are legal. `FREGREQUEST` is used to confirm the acceptance of the offered parameters with F-PSS. Additional parameters can be required in the option field in `FREGREQUEST`. `FREGACK` from the server is to ensure that the registration is committed. `FREGNAK` declines the request form the PH. These two messages have the similar field values as given in `FREGDISCOVER` and `FREGOFFER`.

After completing the registration with F-PSS, the PH will automatically configure its local TCP/IP to behaviour as a local host from the viewpoint of the network and the transport layers. The assigned IP address, netmask, local router will be stored in the shell environment variables and will be set up by the following commands:

```
ifconfig eth0 down;
ifconfig eth0 $IPADDR netmask \
            $NETMASK mtu $MTU up;
route add default $ROUTER
            netmask $NETMASK eth0;
```

if there are additional routers, more `route add` command will be executed for each of the routers. Other application-related parameters need to be set up properly. For example, the local DNS name and the IP address need to be put in the file `/etc/hosts`.

However, all the parameters are only available within certain time period. Before they expire, the PH needs to send again `FREGREQUEST` to renew. Before leaving the current network, the PH is supposed to release the IP address, IP^g_{tr} by sending a `FREGRELEASE` message to F-PSS, which has the same values as given in `FREGREQUEST`, in addition to an additional option, $MSGTYPE$, to differenciate from `FREGREQUEST`.

6.4 Registration with H-PSS

During the first registration period, F-PSS contacts H-PSS, in order to check the domain name and the home IP address of the PH, IP^h_{app}, and to get the PH's public key. H-PSS is not aware of any new location information of the PH until this registration takes place. The main purpose of this registration is to inform its H-PSS of the system parameters used by the PH.

The communication of H-PSS takes place on top of TCP[4] and SSL protocol, which is for secure and reliable data transfer. The SSL connection is first established using the PH's public key. Then, the PH sends an `HREGREQUEST` message, which consists of a header and a number of *Parameter Records* (PR). All the PRs will be encrypted using a secret key, which is statically set up on H-PSS and the PH. The fields and values in the header are given in Table 3. The sructure

Field	Length	Value
rid	2	request id
op	1	1: REQUEST
flag	1	reserved
hip	4	constant IP address
nlen	2	length of home DNS name
hname	var.	home DNS name
npr	2	number of Parameter Records
signature	16	signed by PS's secret key

Table 3: Fields and Values in `HREGREQUEST`

of PR is as the same as the structure of the options defined in [9]. After checking the signature

[4] we use TCP port number 4000 in our prototype system.

of the header, the server deciphers all PRs, updates its PH database, and sends back a `HREGACK`. The fields of `HREGACK` are explained in Table 4. The *rid* protests replay-attack and the *signature* avoids bogus messages.

Field	Length	Value
rid	2	request id
op	1	2: ACK 3: NACK
flag	1	reserved
signature	16	signed by server's secret key

Table 4: Fields and Values in `HREGACK`

6.5 DNS updating

The registration procedures are aimed to set up a new location for a PH, and to inform H-PSS of the location information of the PH. Another important task of the portable-IP system is to update the DNS databases, in order to distribute accurate location information of all PH's in its domain. The DNS databases are managed by the DNS servers in the guest network and the home network,

At any location, a PH always uses its constant home IP, IP^h_{app}, and its original domain name. In addition, it may be assigned to a new local IP address, IP^g_{tr}, and a new local DNS name. In order to uniquely identify the PH, DNS must be able to do the following things for any PHs:

- to answer the DNS-A query when the corresponding home domain name of the local IP address, IP^g_{tr}, is requested;
- to answer the DNS-PTR query when the corresponding home IP address, IP^h_{app}, of the home domain name is requested;
- to support the DNS-CNAME query for the local domain name and the home domain name;
- to answer the DNS-A query when the corresponding local domain name of the local IP address, IP^g_{tr}, is requested; and
- to answer the DNS-PTR query when the corresponding local IP address, IP^g_{tr}, of the local domain name is requested.

In our portable-IP system, the two PSS system act as the DNS sevrers. The F-PSS is responsible for the first two types of queries, whereas H-PSS is responsible for the last two types of queries. The key issue here is that the DNS databases have to be updated dynamically. Two methods are available.

Reload DNS servers

The first method to provide updating is to reload the DNS server. For example, a PH has already had a home domain name *hph1.cslab.anu.edu.au* and a home IP address `150.203.126.173`. In addition, in the guest network, it has been assigned to a new local domain name: *fph10.tsinghua.edu.cn* and a new local IP address `202.112.10.63`. Assume that both name servers are *named* using the BIND version 4.9 running on a Unix platform. The DNS databses at both sites must be updated. The reverse domain name in the home network needs no updating because the home IP address is always mapped onto the home domain name. In the home network, the domain name database will be revised as follows:

```
-   hph1.cslab.anu.edu.au. \
      IN   A    150.203.126.173
+   hph1.cslab.anu.edu.au. \
      IN   A    202.112.10.63
```

In the guest network, the following lines will be added in the domain name database, and will be deleted when this PH moves out:

```
+   fph10.tsinghua.edu.cn. \
      IN   A       202.112.10.63
+   fph10.tsinghua.edu.cn. \
      IN   CNAME   hph1.cslab.anu.edu.au.
```

In addtion, in the guest network, the reverse mapping file needs to update:

```
+   fph10.tsinghua.edu.cn. \
      IN   A       202.112.10.63
+   fph10.tsinghua.edu.cn. \
      IN   CNAME   hph1.cslab.anu.edu.au.
```

The second line above is important to the email systems. If a host is sending an email message to a person on the PH, the canonical name will be used. Then, email messages addressed to the PH can always reach the mail exchanger (MX record provides this infomation). After updating the databases, an HUP signal is used to reload the databases:

```
kill -HUP ‘cat /var/run/named.pid‘
```

Obviously, this method is not practical in a heavy-loaded environment, since it requests to reload the databases frequently.

Dynamic DNS updating

The new internet standard provides an alternative solution, the dynamic DNS updating, which is defined in [13, 14]. We use the new BIND version 8.1.1 in our prototype system.

Field	Explanation
Header	modified header
Zone	specifies the zone to be updated
Prerequisite	RRs or RR sets which must (not) preexist
Update	RRs or RRsets to be added or deleted
Additional Date	additional data

Table 5: Fields and Values in HREGACK

To support dynamic updating, the new DNS message, UPDATE, is newly defined. The fields are listed in Table 5. Any system can send this UPDATE message to request the DNS server to add or delete one or more RRs in its datatbase. Additionaly, in the BIND version 8.8.1 package, there is a new function avaliable in the *resolver* library:

```
int res_update(ns_updrec *);
```

This function takes one parameter the pointer of the structure `ns_updrec`, which is defined in the file *nameserv.h*. Some of the important fields are given below.

```
struct ns_updrec {
  char* r_dname;  /* owner of the RR */
  u_char* r_data; /* rdata fields
                   * as text string
                   */
  int r_opcode;   /* type of operation */
      ...
};
typedef struct ns_updrec ns_updrec;
```

The domain name of the PH will be given in the `r_dname` field, and the `r_opcode` will be assigned to either 0 for delete or 1 for add operation. The RRs being added or deleted will be put in the `r_data`.

Using the new DNS server, we can avoid to reload the whole database frequently. Additionally, the H-PSS/F-PSS and the DNS name server can run on separated machines. It will assist us to integrate our portable-IP system into the existing network frameworks.

7 Conclusion

In this paper, we refine our mobile solution using two mappings, the portable-IP mapping and the mobile mapping. We also focus on the implementation issues for supporting the portable-IP mapping. Our portable-IP system has three subsystems: a portable host system (PH) on a mobile host, a home portable support system (H-PSS) in the home network where the mobile host originally resides, and a foreign portable support system (F-PSS) in the guest network where the mobile host may visit. The details of the two registration procedures with F-PSS and H-PSS are given in this paper. The H-PSS and F-PSS can server as the DNS servers. Two DNS updating approaches are also discussed. One is to reload the DNS databases when a mobile host moves in or out. The other is to use the UPDATE function provided in the new BIND version 8.8.1. We attempt to use the latter approach since it can reduce the cost to reload the whole DNS database frequently. In terms of performance, our mobile solution, built on top of portable-IP system, can reduce both the cost for distributing the location information of mobile hosts. It is because that only the F-PSS and H-PSS have to work together for DNS services. The main advantage of our mobile solution is that it can significantly reduce the cost for forwarding IP datagrams across the Internet. Here, we mean that there is no triangle effort when a host is sending a large amount of data to the mobile host. IP diagrams are not needed to be sent to the home network and then be forwarded to the guest network where the mobile host resides. As future work, we will attempt to conduct a performance study on the two costs mentioned above.

References

[1] John Ioannidis, Dan Duchamp, and Gerald Q. Maguire Jr. IP-based protocols for mobile internetworking. *ACM Computer Communications Review*, 21(5):235–245, September 1991.

[2] Andrew Myles and David Skellern. Comparing four IP based mobile host protocols. *Computer Networks and ISDN Systems*, 26:349–355, 1993.

[3] Charles Perkins, Andrew Myles, and David B. Johnson. The internet mobile host protocol (IMHP). In *Proceedings of INET'94/JENC5*, 1994.

[4] Xun Qu, Iain Macleod, and Hong Jiang. A practical method to achieve portable communication in internet context. In *Proceedings of GlobeCom'95*, pages 1512–1516, Singapore, November 1995. IEEE.

[5] Xun Qu and Jeffrey Xu Yu. An analysis of mobile TCP/IP solutions. accepted by

First International Conference on Information, Communications and Signal Processing (ICICS'97), September 1997.

[6] Xun Qu, Jeffrey Xu Yu, and Richard P. Brent. A mobile TCP socket. Technical Report TR-CS-97-08, Computer Sciences Laboratory, RSISE, The Australian National University, Canberra, ACT 0200, Australia, April 1997.

[7] Requirements for internet hosts — communication layers, RFC 1122, October 1989.

[8] Requirements for internet hosts — application and support RFC1123, October 1989.

[9] DHCP Options and BOOTP Vendor Extensions: RFC1533, October 1993.

[10] Dynamic Host Configuration Protocol: RFC1541, October 1993.

[11] The Recommendation for the IP Next Generation Protocol, January 1995.

[12] Domain Name System Security Extensions: RFC2065, January 1997.

[13] Dynamic Updates in the Domain Name System (DNS UPDATE): RFC2136, April 1997.

[14] Secure Domain Name System Dynamic Update: RFC2137, April 1997.

[15] Fumio Teraoka, Yasuhiko Yokote, and Mario Tokoro. A network architecture providing host migration transparency. *ACM Computer Communications Review*, 21(1):209–220, January 1991.