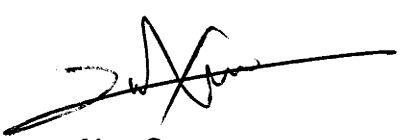
Mobile Communications In The Internet Environment

Xun Qu

A thesis submitted for the degree of
Doctor of Philosophy at
The Australian National University

November 1998

I hereby state that this thesis contains only my own original work except where otherwise explicit reference has been made to the work of others.



Xun Qu
29 November 1998

Acknowledgments

I would like to thank my supervisors and advisers for their help and encouragement during the period of my PhD study in Computer Sciences Laboratory, Research School of Information Science and Engineering. This work could not have been completed without their support. In particular I would like to thank my supervisor Dr. Jeffrey Xu Yu. He has spent a lot of time to discuss and help me to refine and formalise my raw ideas. Without his support I could not have published these papers directly related to my PhD study. I also would like to thank the current chair of my supervision panel, Professor Krishna Murthy. To help me complete my thesis in time, he even worked at the weekend to check the clarity and correctness of my thesis. Also, I benefited a lot from discussions with my adviser Dr. Bing Bing Zhou. His support and encouragement gave me more confidence.

I also would like to gratefully acknowledge support of my former supervisors Dr. Iain Macleod and Professor Richard P. Brent. My ideas for the research topic were created by talking to Dr. Macleod and he helped me publish the first two papers after I started my PhD study at the ANU. He also gave me a great deal of support with a variety of aspects of my life in Australia. Without generous support from Professor Brent, I could not have had so many chances to present my papers at international conferences. These opportunities were very important for me to access and understand recent related research.

This is the revised version of my PhD thesis. The three examiners of my PhD thesis gave me lot of help. They helped me to correct some technical details, advise me to describe some technical issues in more clear way. They also pointed many typos and gave me some suggestions to improve the type-setting of this thesis. Although they are still anonymous to me at this moment, their help and efforts are highly appreciated.

Finally, my thanks to my wife Hong Jiang. She stood by me as I took this long path

to finishing my PhD study and supported me during those hard times.

Abstract

The TCP/IP protocol suite is currently the most popular networking standard. The last decade has seen its success in both academia and industry. The world's largest network — the Internet — provides the most dramatic evidence here.

The current TCP/IP protocols have been facing a new technological challenge: mobile communications. The existing TCP/IP systems can not support mobility in the internet context due to the addressing and routing schemes. The main issue here is the so-called two-tied use of IP address: (i)Firstly, the IP address serves as an identifier to name a system, a protocol entity or an application process based on TCP/IP protocol. A system is supposed to have a unique life-time IP address for this purpose. (ii)Secondly, the IP address is also used by the routing infrastructure to locate a system. Therefore, the IP address is a location-sensitive address.

In a mobile environment there is a dilemma. For the sake of (i), the TCP/IP systems need to be assigned the constant IP address. To be mobile and satisfy (ii), however, a system can not use its original IP address in other network, but its home network.

To have an insight on mobility in the internet context we devise a functional layered model for general networking system. Any communication protocol layer can be classified into three types, namely, *static, portable* and *mobile* layer. In a full-fledged mobile system, there must be one mobile layer to hide the mobile operations to application processes so as to provide continuous mobile communication services. All layers above the mobile layer can be static layers while all layers below it must be portable layers in order to support the basic communications at different locations.

There are many proposals of mobile TCP/IP support. Based on our functional layered model, all existing mobile TCP/IP solutions can be divided into three categories according to the location of mobile layer. They are mobile data link solution, mobile network solution and mobile transport layer solution.

The main task to support mobility is to decouple the two-tie use of IP address and then do address mapping. Different solutions conduct different address mapping. However, all existing solutions have a commonplace: to provide the mobility within a single layer. Two major concerns with such one-step solutions are (i) routing efficiency and (ii) compatibility with existing TCP/IP system. As we know, the typical routing from a fixed host to a mobile host will be detoured. The IP datagrams can be forwarded to a mobile host's home network and then redirected to its current location. The detour, or *triangle routing* can waste network bandwidth and affect communication performance. Some solutions suggest that special routing devices and protocols be employed to reduce the probability of detour routing. However, the incompatibility between the special protocol and the existing routing infrastructure will limit the application of these solutions.

In order to alleviate the above two problems, we propose a two-step solution in this thesis. The first step is to support portability in network and transport layers. Our Portable-IP system makes use of existing standards DNS and DHCP and expands the function of doing IP address allocation and dynamic IP address mapping so as to support the portability in the IP layer. The second step is to implement mobile communication services in the socket layer that lies above the transport layer, but below the application layer. The enhanced socket code, or *Mobile Socket Layer*, can bridge the communication gap while systems are moving and correctly support address mapping to enable applications and users to use constant IP address or domain name to identify mobile TCP/IP systems as usual.

We also provide two other models, namely, *moving pattern model* and *cost model* to analyse system performance in a mobile environment. The *moving pattern* abstracts the moving characteristics of a mobile system, whereas the *cost model* provides the basic standard to evaluate the performance of a mobile system. Our analysis and simulation give quite encouraging results for our two-step solution.

As a future work, we are going to continue our research in a mobile TCP/IP environment in the new generation IP protocol — IPv6. Although the mobility has been taken into account in design of IPv6, its performance and compatibility still need improving.

Acronyms

ACK	Acknowledgment
AP	Access Point
API	Application Program Interface
ARP	Address Resolution Protocol
ARPANET	Advanced Research Projects Agency Network
AS	Autonomous System
AUTHP	Authentication Protocol
BAS	Base Station
BGP	Border Gateway Protocol
BOOTP	Bootstrap Protocol
CIDR	Classless Interdomain Routing
D-DNS	Dynamic Domain Name System
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DVR	Distance Vector Routing
EGP	Exterior Gateway Protocol
FA	Foreign Agent
FH	Fixed Host
FN	Foreign Network
F-PSS	Foreign Portability Support System
FREGP	Foreign Registration Protocol
GN	Guest Network
GPRS	General Packet Radio Service
HA	Home Agent

HN	Home Network
H-PSS	Home Portability Support System
HREGP	Home Registration Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
IMHP	Internet Mobile Host Protocol
IP	Internet Protocol
IPX	Internetwork Packet Exchange
ISDN	Integrated Service Digital Network
ISO	International Organisation for Standardisation
ISP	Internet Service Provider
LAN	Local Area Network
LLC	Logical Link Control
LSR	Link State Routing
LSRR	Loose Source and Record Route
MDL	Mobile Data Link
MH	Mobile Host
MR	Mobile Router
MSL	Mobility Support Layer
MSR	Mobile Support Router
NSFNET	National Science Foundation Network
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PH	Portable Host
POP	Point Of Presence
RFC	Request For Comment
RIP	Route Information Protocol
RR	Resource Record
RTT	Route Trip Time

TCP	Transmission Control Protocol
TG	Transparent Gateway
TLI	Transport Layer Interface
TTL	Time To Live
UDP	User Datagram Protocol
VP	Virtual Port
VPP	Virtual Port Protocol
WAN	Wide Area Network

Contents

Chapter 1

An Introduction to This Work

1.1 Background and Motivation

The global Internet is growing very fast. As shown in Figure 1.1 provided by Network Wizards[1], from January 1993 to July 1997 the Internet has grown more than 10 times in terms of number of hosts. Presently the total number of hosts on the Internet is more than 20 million. From the survey of NSFNET backbone usages, the top two applications are web and ftp; these consume 23.9% and 24.2% of the total bandwidth. It is obvious that more and more people access the Internet as a major information resource and retrieve data from the network at variety of purposes.

Meanwhile, more and more information services appear over and ride on the largest worldwide computer network — the Internet, which provide a large quantity of information to a variety of people. For example, web browsing is one of the fast growing application. In January 1993, there were only 130 web servers, and the number exceeded 650,000 in January 1997. This easy-to-use and powerful information retrieval means provides for many new applications on the Internet and brings the Internet into homes and all corners in our modern society. The Internet is no longer dedicated to any computer researchers and professionals as it used to be in the beginning stages[40].

Australia has a large Internet user base. According to a recent report from Connect.com.au Pty Ltd, 2.4 million of total 60 million Internet users are in Australia. The current commercial market has a 11% of monthly growing rate, and in 1998 the Inter-

[1] a California based communication company, which provides variety of Internet surveys on their web site: `http://ftp.nw.com/`

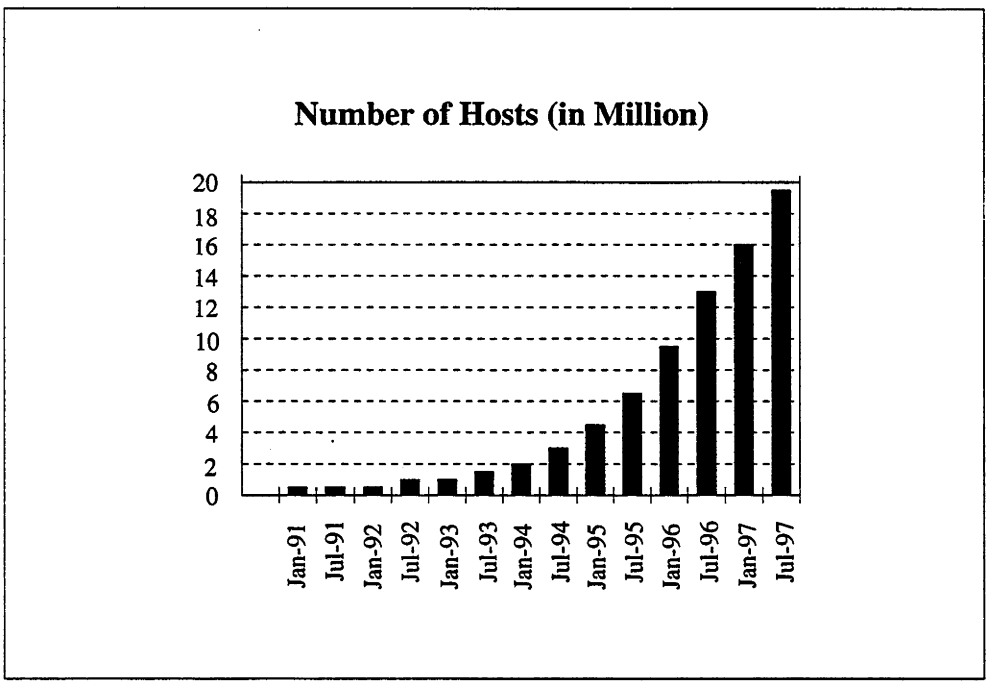


Figure 1.1: Number of Hosts (in millions) on The Internet

net traffic will surpass voice traffic. To provide easy, low cost access to the Internet, more and more local Internet Service Providers (ISPs) arise. That means people can easily obtain local internet Internet access.

Since the wireless technology is getting mature and portable computer is becoming more and more popular, there have been intense interests in portable and mobile communications[2]. It is believed that the mobile communication will become a main trend in telecommunications and data communications. The mobile communication devices, including portable PC and palm-held specific appliance(e.g. PDA) and cellular phone systems, can help users to get rid of the limit of tethered network and give them more free space to roam. An early statistics reported in [10] shows that around 74% of all workers are "mobile workers". They spend most of working time away from their desks. The mobile characteristics of today's computer users have also boosted the demand for mobile communication products and services.

However, convenient local access to the Internet does not mean access to portable or mobile services to the Internet. Usually a user subscribes the Internet access ser-

[2]we will discuss portable and mobile communications services later in details.

vices from one Internet Service Provider (ISP), which provides the Access Point (AP) of the Internet at certain locations, or called Point of Presence (POP). This user can only access Internet at assigned one or more locations where the ISP has POP there. Most of ISP companies can only provide several POPs within a small area. Although some large ISP companies can provide services to thousands of users, they allow very limited areas where their customer can obtain access.

Current mobile communications facilities, such as cellular phone systems and wireless networks, can not help to support mobile Internet communications. As is known, digital cellular phone (e.g. GSM) can provide a point-to-point data link connections. Although the mobile unit itself can obtain a local connection to a local base station, it can establish a link to the fixed Access Point of the Internet, which means the user may pay for expansive long-distance mobile connection on top of Internet access fee. Another problem is the speed. Currently GSM cellular phone unit can only provide data rate of 2400bps to 9600bps. Although the forthcoming data service for GSM network, called General Packet Radio Service (GPRS)[3], is expected to be implemented in next two years and to support data transfer at a rate of up to 100Kbps, the radio communication services are still quite slower than Local Area Network connection (LAN), ISDN connection and even normal modem (33.6Kbps to 56Kbps). The problem with wireless LAN is the roaming area, though its speed is high (from several hundred Kbps to 5Mbps). Usually, a wireless LAN covers a very small area, given the limits of its frequency usage and power. Currently, when a portable computer is carried away from home network, the practical way to connect it back to the Internet is to make a connection to home Access Point through dial-up line, either wired or wireless. Current telecommunications infrastructure is employed to link the portable systems back to their original home AP no matter how far the users are away from their office or home. Here a dedicated communication link is required for each user. In some cases, the cost of long distance telecommunication link can be prohibitively high and the low utilisation is not desired.

The principal issue with mobile communication in the Internet context is how to

[3] a introduction web page about GPRS can be found at: `http://www.pcsdata.com/paprysavy.htm`

allow a host system to obtain a fast local AP, instead of connecting to fixed AP's. Therefore, this research work focuses on: (i) the analysis and design of mobile protocols to provide portable and mobile communications services on the Internet and (ii) design of a novel mobile file system that can work efficiently in such mobile environment.

For further discussion, we briefly describe some terms we will use regarding mobile communications.

- Mobile Host, a TCP/IP-based host which can change its AP to different internetworks and access network services as a fixed host.
- Home Network, each mobile host has a home network, in which the mobile host is assigned a permanent IP address which is used by regular TCP/IP protocols to communicate with each others.
- Foreign Network, in contrast, a network in which mobile hosts temporarily stay can also be called Guest Network.

1.2 Preliminaries of Internet and Its Routing Structure

1.2.1 The TCP/IP Protocols and Internet Structure

In a further discussion, we differentiate between these two words: *the Internet* and *internet*. The former is used to mention the global computer network, whereas the latter refers to an individual network which employs TCP/IP as internetworking protocol.

The TCP/IP protocol suite is currently the most popular networking standard. The last decade has seen its success in both academia and industry. The world's largest network — the Internet provides the most dramatic evidence here. Part of the reason for its success in computer networking is the original design philosophy of the TCP/IP protocol suite: namely, the ability to internetwork computer systems in heterogeneous environments, with computer systems of different sizes, running different operating systems and supporting different application functions. From its beginnings as an experimental protocol with a four-node network in the 1960s to to-

day's Internet, the TCP/IP protocol suite has demonstrated its ability to provide communication facilities that allow almost all types of computer systems to exchange data via TCP/IP protocol networks, or so-called internets. Using TCP/IP technology, all participating computer systems are interconnected at the network layer level of the OSI Reference Model; all individual networks are connected by an abstract communication system — the internet that provides a unified, co-operative, interconnection of networks [5].

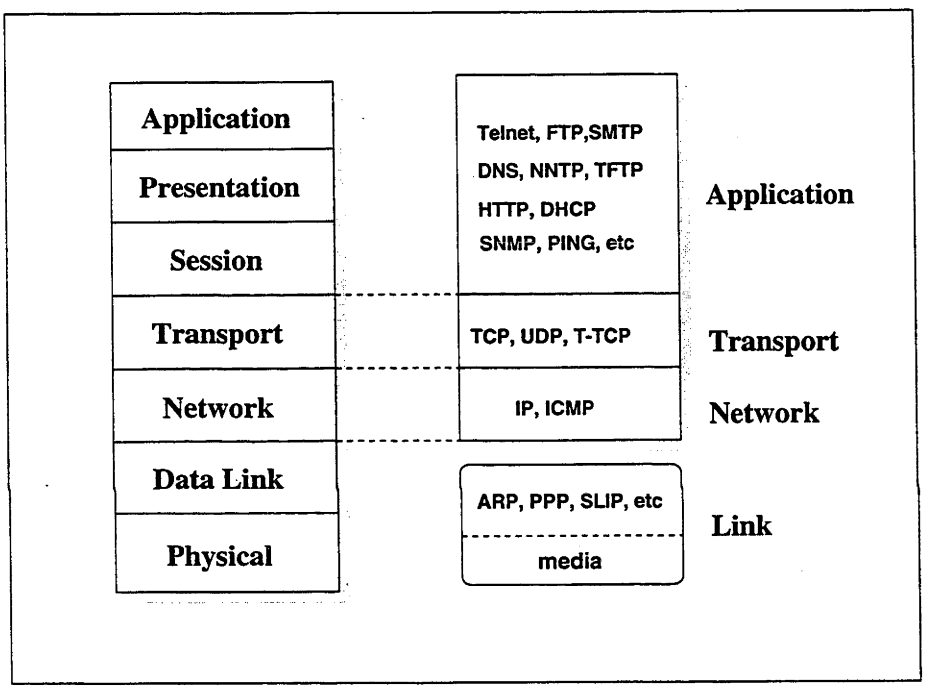


Figure 1.2: ISO/OSI and TCP/IP

As shown in Figure 1.2. The TCP/IP protocol stack has a four-layer model: the link, network, transport and application layer. Compared to seven-layer ISO/OSI model, the TCP/IP protocol suite covers the protocol standards from the network layer up to the application layer in the ISO/OSI reference model. it does not specify protocol standards at the link layer. Protocols above the network layer provide for end-to-end or process-to-process communications. The network layer protocols are primarily involved in internetworking systems at different locations and provide higher layers with services which relate to end-to-end transmissions [72]; they also form a unique logical communications network, which hides the details of underly-

ing data link and physical network technologies. For example, the TCP protocol is not aware of where its peer entity is located; the TCP entity merely needs to know the IP address of its peer entity and the IP layer isolates upper layers from the heterogeneous underlying network technologies employed. When a TCP entity sends a data segment, it provides the data with the IP address of the peer IP system. From the viewpoint of the transport layer, the networks interconnected by the IP protocol are all part of a universal logical internet; every system can reach all others in the internet. Therefore, the task of internetworking different systems and individual physical networks lies within network layer protocols.

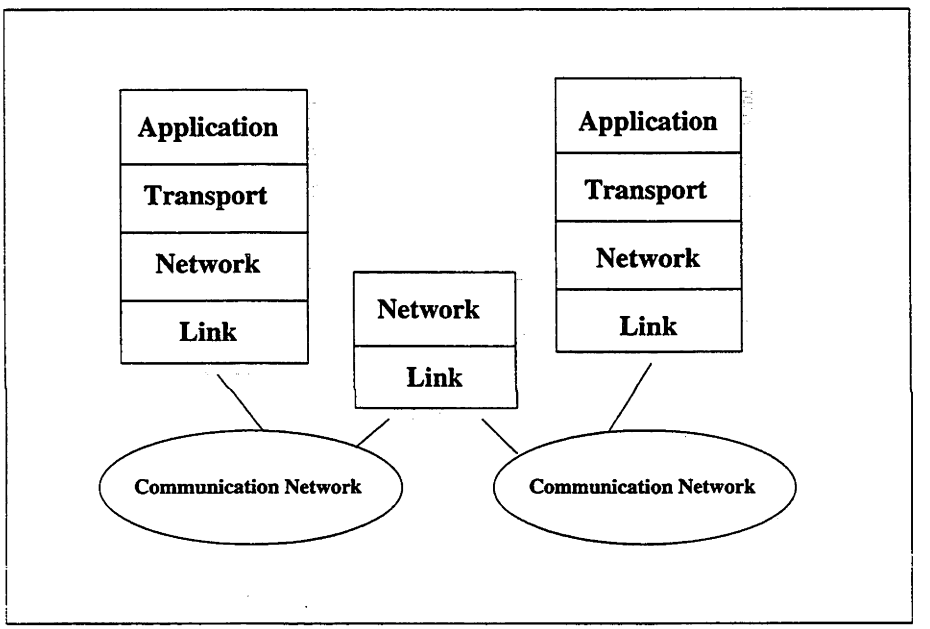


Figure 1.3: TCP/IP Systems

In internet protocol stacks, the network layer can be roughly divided into two sublayers according to different scopes of the functions performed. The lower part, the IP protocol, as well as ICMP and IGMP, aims to interconnect different host systems, so it can reside in all different types of systems in a TCP/IP-based network. In a network with a single physical layer, IP is sufficient to complete the required interconnections. As shown in Figure 1.2, the upper part of the network layer in the TCP/IP protocol suite interconnects individual networks to form a strongly connected logical network; these upper-level protocols are mainly located in internetworking devices, such as

routers or gateways. Actually as OSPF, RIP and BGP should be also put in the level according to their functions, because they are all routing protocols. But RIP and BGP use UDP and TCP as transport protocols respectively. Figure 1.3 depicts the protocol layered model of internet systems; the computer systems include all layers from the network interface up to the application layer. However, the router has the lowest three layers (end-to-end routing and transmission functions). These are sufficient for a router because it only forwards IP datagrams between the networks it connects and has no need for higher-level protocols.

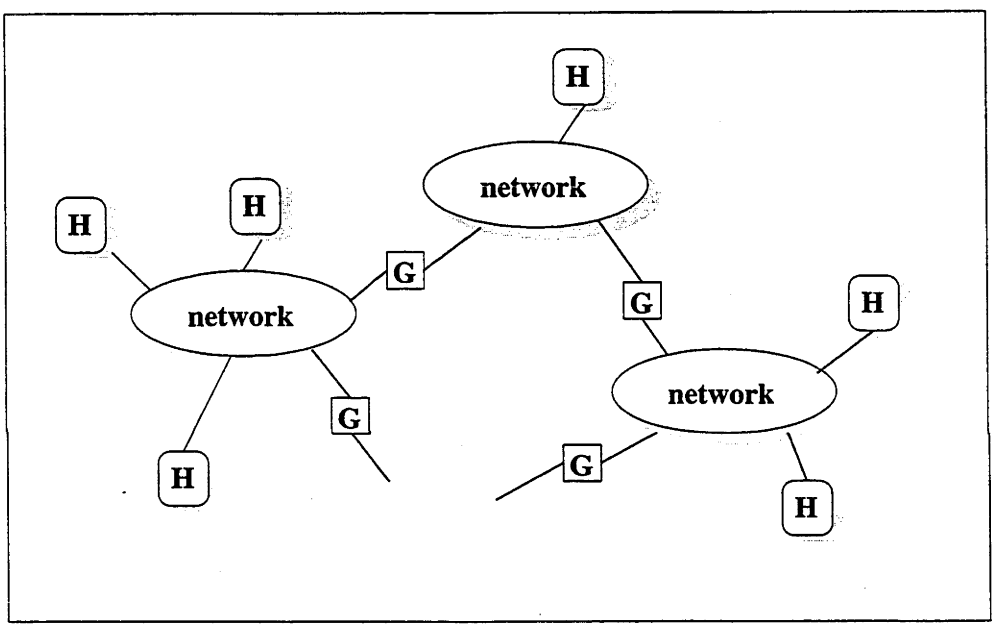


Figure 1.4: TCP/IP Systems

As shown in Figure 1.4, each host system is connected to an individual network. These networks are then interconnected by routers (known as "gateways" in internet jargon) to form an internet. The IP protocol provides a connectionless network service, which deals with addressing of IP entities, static routing between IP systems, best-effort datagram transfer between entities, etc. Other network layer protocols mainly deal with dynamic routing of IP datagrams between different networks. From the user's viewpoint, the internet is a universal virtual network to which all participating computer systems are connected. In reality, the individual networks are internetworked by routers. From the network administrator's viewpoint, one or more physi-

cal networks under a single administration authority form an administration domain, which is called an autonomous system (AS). Routers within a single AS are called intra-domain routers, and the routers between distinct ASs are called inter-domain routers. Different protocols for exchange of routing information are dedicated to these two types of routers. The basic functions of these two types of routers are, of course, the same: to route IP datagrams to their destinations between different networks, either within the same AS or not.

1.2.2 IP Addressing and Routing

The current IP address of IPv4 is 32-bit long, which consists of two parts: *netid* and *hostid*. The IP address is Internet wide meaningful, that means that with the given IP address a unique host system can be located. According to the IP addressing scheme, each individual network in an internet obtains an unique *netid*, then each network interface[4] of a host system has a network-wide unique *hostid*, with the same netid for all interfaces attached to the same network.

To reduce the number of logical networks and economise on IP address space, a single network address may be assigned to several physical networks[60]. In this case, all sub-networks appear as a single network to the outside world. For routing IP datagrams correctly to each subnet, some portion of the hostid part of the IP address is used as a subnet id, so that the netid has more bits than usual. To indicate which bits identify the subnet, including normal netid bits, a 32-bit subnet mask is used, in which ones represent the netid and zeroes the hostid.

Given such two-level addressing scheme, the routing function is mainly conducted by routers, which are the internetworking devices at the IP layer[5]. The host system will send out a IP datagram through its Access Point to the local internet. Then the local routers will find the path to a destination network and forward this datagram to next router along the path.

[4]We use the network interface as a component of the IP address instead of individual IP systems to avoid confusion in cases where multiple IP addresses belong to single IP systems. For example, a router connecting two different networks will have a different IP address for each interface.

[5]the repeater and bridge are the physical and link layer internetworking devices, respectively.

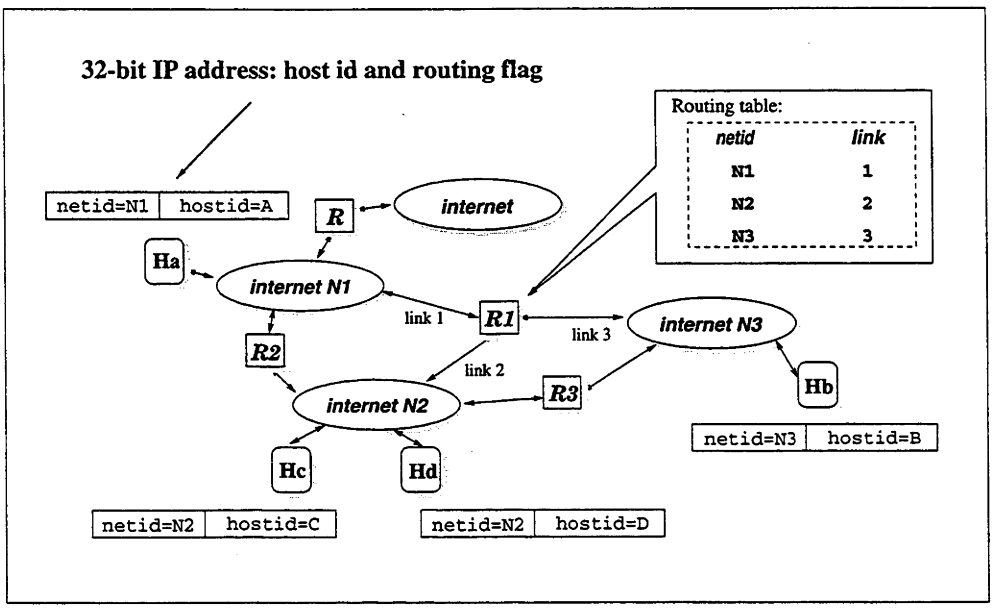


Figure 1.5: TCP/IP Systems

First, let's look at a schematic example of IP routing. As shown in Figure 1.5, there are three routers that interconnect three internets N_1, N_2 and $N3$. All hosts A, B, C and D are assigned unique IP addresses whose first part is the netids — N_1, N_2, N_2 and N_3 respectively. As indicated, router R_1 has three links, and its internal routing table is also given. If Host H_a sends an IP datagram to Host H_c, this datagram will be sent to R_1, and R_1 abstracts the netid from the destination IP address N_3/H_c. Then, the routing database on this system, called routing table, tells that the destination network N_3 is connected by link 3. Therefore, R_1 forwards this datagram to network N_3 via link 3 so that host H_b can get the IP datagram. Note that IP routing includes two basic functions. First, the host must know how to send a IP datagram to a local router. Second, the router has to be able to figure out the next hop to the destination along a "best" path in terms of variety of measures. In Figure 1.5, there are two local routers for host B and C. Furthermore, there are two potential paths from host B to D: via router R_1 or R_3. Therefore, the IP routing system is not trivial and the routing algorithms and protocols help the host and router to complete these functions.

IP Internal Routing

Actually, each IP system, including the host computer IP entity, will calculate routes before sending IP datagrams. In the kernel of each IP system, a routing table stores destination information with corresponding router and local interface details. There are two kinds of destinations: hosts and networks. When IP sends a datagram, it searches the routing table to find a matching entry for the destination IP address. If a host destination is found, the IP will send this datagram via the local interface to the router indicated in the matching entry. If only the network destination entry is matched, the datagram is also sent to the router via the interface according to the routing entry. If no match is found, the datagram is sent via a default routing entry, if it exists, The worst case is no default entry. In this situation: (i) if the datagram is generated by the current system, a "host or network unreachable" error message will be sent to the relevant application, or (ii) if the datagram is received from another system, an "ICMP host unreachable" error message is sent back to the source system. All IP entities execute this routing mechanism. Although the routing table is not defined in the protocol, it plays a very important role in routing datagrams. The table can be statically modified in two ways: user commands and ICMP redirect messages [57]. When a system starts up, certain commands (such as "route" in UNIX) are used to store initial routing entries and manipulate the routing table. Also, when any router finds the current datagram should be sent to another router, it sends an ICMP redirect message to the source system to inform it of the more-direct route.

Generally, the host system maintains the table only statically. However, routers (or host systems playing the role of routers) dynamically exchange routing information and modify their table contents from time to time. The protocols involved here are divided into two categories, as discussed below.

Intra-domain Routing Protocols

The main purpose of an intra-domain routing protocol is to allow routers in the same AS to exchange routing information and to update their routing tables dynamically, thus enabling the routing tables to correctly reflect changes of network topology over

time. Furthermore, the routers can use the exchanged information as a basis for finding the best route for each datagram, if there are multiple routes to a given destination. In order to maintain the currency of details regarding network topology, routing information is periodically exchanged between involved routers. The exchanged information should be sufficient to ensure that routers can calculate close to optimal routing for each datagram. Obviously, intra-domain routing protocols focus on these two aspects. Here the best route does not always mean the shortest path, sometimes the criterion is cheapest cost or shortest delay time. Generally the types of service defined in IP can be used to evaluate alternative routes. RIP[50], the most popular routing protocol in current use, uses UDP as the transport layer protocol to carry pairs of IP addresses and metrics as routing information. The metric used in RIP is the number of hops from this router to the destination network indicated by the IP address. After receiving routing information from other routers, the router calculates its own metric with respect to the destination, chooses the route with the fewest hops as the best route, and informs the other routers of its routes, too. The routers exchange RIP messages with neighbouring routers every 30 seconds, or on any change of a route metric. Routes which have not been updated for more than 3 minutes are deleted. This method of calculating routes is called **Distance Vector Routing**(DVR)[39]. The principle behind it is the Bellman-Ford algorithm for finding the shortest path in a directed weighted graph[6]. In some cases, where subnet addresses are applicable, RIP is not sufficient: it cannot exchange subnet masks between routers. An enhancement known as RIP-2 [7], was therefore devised. The basic protocol remains the same, but additional information such as the subnet mask is included in the routing messages. However, distance vector routing has an inherent weakness of slow convergence. If a link fails, the routing information will eventually reflect the new topology, but it can take a long time to reach a new stable state[39].

Due to problems with DVR, a new protocol known as OSPF[55] which uses IP directly to convey its messages, has been proposed as an alternative to RIP. OSPF is based on **Link State Routing** (LSR)[39], whose basic idea is that every router knows the costs and status of all its links, and can always test the status of any link. The cost associated with each link is measured by the type of service in OSPF, as defined in

IP protocol. The link costs and status (down or up) are periodically broadcast, and are also forwarded to all other routers. Therefore, all routers will eventually know about all other routers and get a complete view of the network topology. Each router can then build a complete routing table. To route a datagram, Dijkstra's shortest path algorithm is used to find the cheapest route to the destination. LSR converges faster than DVR, and fewer routing messages are exchanged between routers. But, if link costs change and do not affect the shortest path, DVR will not send any message out whereas LSR will. In practice, LSR is preferred for this reason. Detailed comparisons of LSR and DVR are given in [39].

Inter-domain Routing Protocols

Unlike intra-domain routing protocols, which always optimise a path according to the metric or cost, inter-domain routing protocols not only deal with the technical problems of routing between distinct AS's, but also with routing policies. The best route from the technological viewpoint may not be satisfactory in terms of policy. For example, competing companies may reroute their datagrams to avoid traversing each other's networks for security reasons. Another important issue in relation to Inter-domain routing protocols has to do with ownership and financing of domains. If different domains are independently funded, forwarding of datagrams for other domains (i.e. transit traffic datagrams whose source and destination lie in other domains) will consume network resources, with associated financial considerations.

A single AS normally has one or more routers that connect it with other AS's; these are called border routers. As shown in Figure 1.6, domains A, B and C are interconnected by border routers. Between them, an inter-domain routing protocol is used and global routing information is exchanged. Each domain can use its own policies to guide its outgoing traffic and to control transit traffic. For example, if domain B is not willing to forward traffic for domain C, it can advertise the routing information to other domains' border routers without reporting the link to C. This is called "domain hiding." All transit traffic through B to C is avoided. Although they may have differences in policies, all border routers have to inter-operate with each other. An early

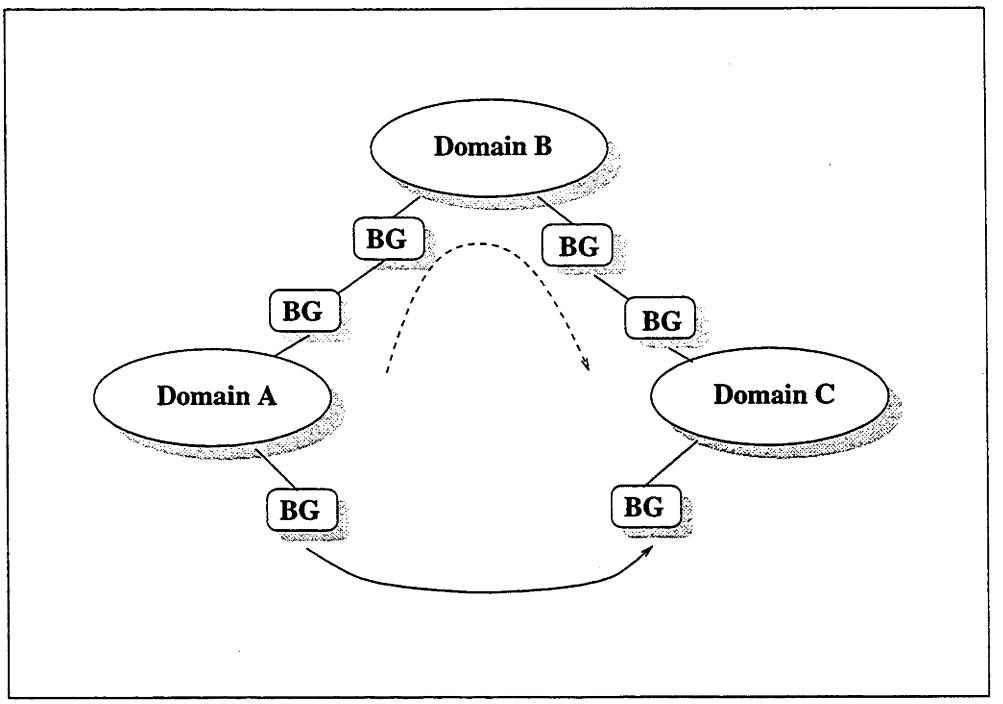


Figure 1.6: Border Gateway

version of an inter-domain routing protocol in internet is EGP. A newer protocol, BGP (Border Gateway Protocol) [51], has replaced the older EGP. BGP is a distance vector protocol; a TCP connection is established between neighbouring border routers to carry the routing information.

Another new technology currently employed for inter-domain routing is called CIDR (Classless InterDomain Routing). As we know, each network address occupies a routing table entry. With the growth in number and complexity of networks, the routing tables become larger and larger. Assignments of class C addresses make the situation even worse. In some cases, networks have more than 255 hosts and there are not sufficient class B addresses to accommodate them. An alternative here is to assign several class C addresses to a single site. However, the routing tables will keep several entries for the single network site. To reduce the size of routing tables in border routers, these addresses should be condensed into a single routing entry. For example, the following four class C addresses: 194.1.249.0, 194.1.250.0, 194.1.251.0 and 194.1.252.0 have the common prefix bit pattern 194.1.248.0, so they could be rep-

resented as a supernet with the address 194.1.248.0. This supernet address is classless. More detailed explanations here can be found in [45, 46]. This approach is very useful for some new domains, such as P. R. China which has been assigned only a small number of Class B addresses. Additional class C addresses are expected to be assigned subsequently, so CIDR can help with assignment of IP addresses and reduce the size of routing tables for border routers.

1.2.3 Various Ways to Interconnect Internet Systems

The generic structure of an internet is a logical universal network that consists of individual networks interconnected by intra- or inter-domain gateways. Another basic requirement is that the interconnection needs data link support. However, the fact that there are different environments and different low-level supporting layers means that particular consideration has to be given to certain aspects of internetworking of TCP/IP-based systems.

Transparent Gateway

In essence, each IP system has at least one IP address for internet-based communication, and every individual network should be allocated a dedicated network address. Basically, the gateways route IP datagrams with networks according to the netid, i.e. the network address. In cases where there are not sufficient IP network addresses to be allocated, the subnetting technique described above allows several subnets to share a common netid, distinguishing the subnets by means of hostid bits. In the extreme case, no subnet id is available for a given small local area network, meaning that the small net has to share the netid with another net. As shown in Figure 1.7, network A has its own netid; network B is connected to network A and uses the same netid. In other words, all the hosts in network A and network B have the same netid in their IP addresses. A normal gateway cannot distinguish such nets, but a so-called transparent gateway (TG) is capable of interconnecting them. A TG routes datagrams between network A and B not according to the netid, but according to the individual IP addresses belonging to the hosts in network B (network B is assumed to have only

a small number of hosts so the routing table in the TG can also be small). The routing algorithm is simple: the TG forwards to network B all traffic from network A destined for hosts on network B and vice versa (forwards all datagrams from network B to network A, which are sent to such destinations elsewhere except network B). In addition to separating local traffic, the TG can also reduce the usage of IP network addresses. The main limitation in using a TG is that network B needs to be reasonably small, because each host IP address occupies an entry in the TG's routing table and searching a large table is a slow process.

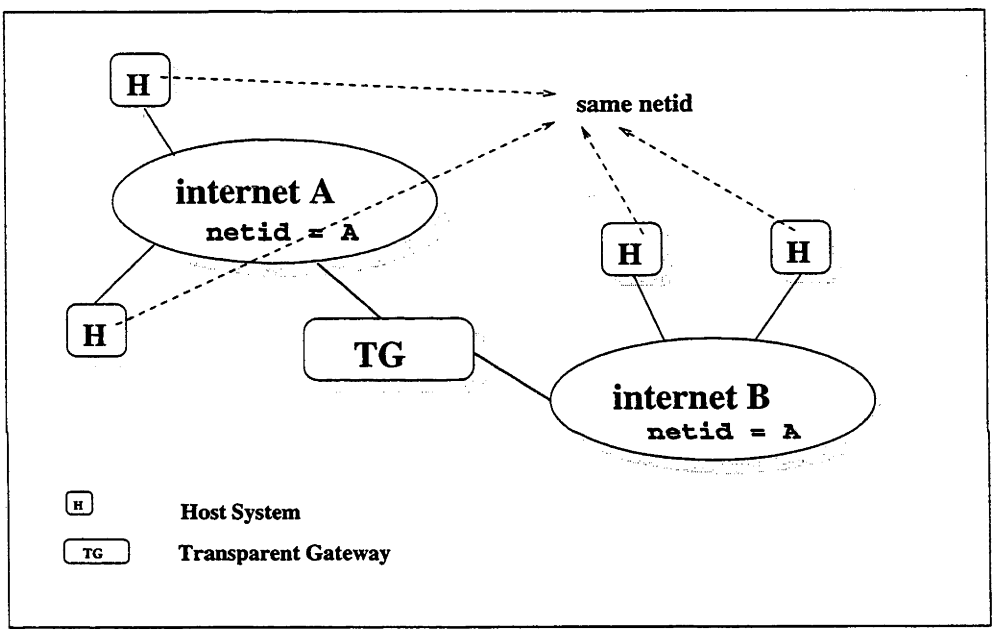


Figure 1.7: Transparent Gateway

As an example of the use of a TG, consider a small temporary network that is set up for an internet-based conference in a building served by an existing local area network. A TG could be used to connect the conference room with the existing network. When the conference is in progress, all local traffic is confined to the small network, but the hosts can communicate freely with remote hosts. When the conference concludes, the temporary network can be disabled. It will then occupy no network addresses and need have no IP addresses reserved for it. Note the distinction between a TG and a Brouter [39], which take on the roles of both a bridge and a router. The routing function of a Brouter follows the normal routing algorithm.

IP Address Reuse and Internetworking Backbone-partitioned Stubs

Actually a transparent gateway cannot save IP addresses, because each host on both sides needs to be assigned a unique IP address. As we know, IP address space is the most limited resource of the Internet where very few unallocated class A and B addresses are available. Certain methods have been proposed to save IP address space in near term, for instance, by dynamic IP address allocation. IP address translation[49] is another quite practical proposal. The central idea in NAT (Network Address Translation) is to concurrently reuse IP addresses in different networks. These IP addresses have only local significance so that no address conflicts occur at a global (Internet-wide) scale.

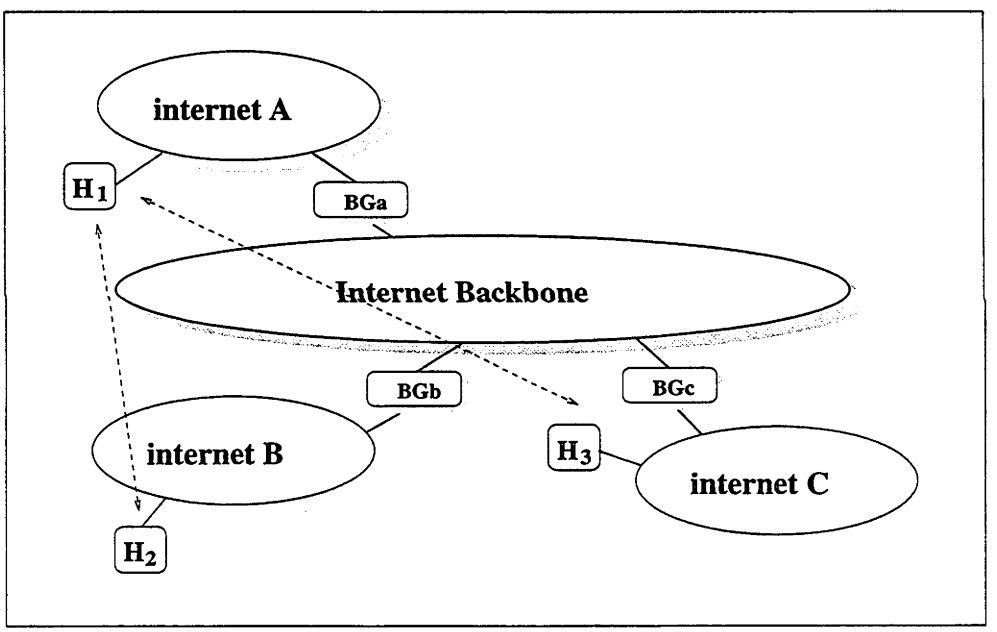


Figure 1.8: Stub Networks

Most hosts exist on a "stub" domain, which does not deal with transit traffic and hardly communicates with other domains (or possibly not at all). Such hosts can therefore be assigned addresses that are significant or meaningful only within the stub domains, meaning that these addresses are available for use by other domains. If and when hosts on a stub domain wish to communicate with external hosts, the border gateways of the stub domain will translate the local IP address to a global IP address,

so that only a small fraction of global addresses needs to be shared or reused by all the hosts in the domain. As shown in Figure 1.8, network A and B reside in two different domains and hosts H_1 and H_2 have their own local addresses respectively. The border gateways of network A and B are assigned global addresses and also have some sharable global addresses for their hosts to send or receive datagrams across the borders of the two domains. When H_1 sends a datagram to H_2, it uses its local IP address L_1 and the global address G_2 of H_2 as the source and destination addresses respectively. The datagram reaches the border gateway BG_a, and BG_a chooses an available global address, say G_1, for H_1, and makes an entry L_1 corresponding to G_1 in its translation table. BG_a then changes the source address of the datagram to G_1 and sends it to the internet backbone. Eventually, the datagram reaches BG_b, and BG_b does the reverse translation from G_2 to L_2, changing the destination address of the datagram to L_2, so that H_2 receives it in domain B.

In some cases, two domains may belong to the same administration, and they can even use a single local IP address space. Stub domains interconnected by an internet backbone are known as "backbone-partitioned stubs." All such stubs share a common IP network address space, i.e. the host IP addresses are significant among all stubs. Most outgoing traffic from these stubs is normally destined for other such stubs. In this case datagrams can even be forwarded without network address translation. For example, internets A and C in Figure 1.8 have a common IP address space, i.e. the local IP addresses of H_1 and H_3 have the same netid. If H_1 wants to send a datagram to H_3, H_1 uses L_1 and L_3 as the source and destination addresses in its datagram. BG_a will find this datagram is destined to a host in the stub connected by BG_c, and BG_a encapsulates the whole IP datagram into a special datagram with BG_a and BG_c's global IP addresses and sends it to BG_c, which then picks up the original datagram and sends it to H_3.

The most significant advantage of NAT is that it can save IP address space. For example, a global class C address can be assigned to a stub domain of the type described above and a class A address could be used for intra-domain communications. This stub can potentially accommodate 16,777,214 hosts and yet occupies only 256 global IP addresses. In this method, only border gateways are involved in address

translation (which is transparent to most other systems). Of course, NAT will affect some particular applications that need to know the global IP addresses of hosts in advance.

1.3 Issues of Mobile TCP/IP Communications

In our research work, we focus on TCP/IP internet protocol suite rather than discuss and study universal mobile computing environment. That means two things. First, when talking about mobile TCP/IP communications, the scope of network environment is confined within internet TCP/IP protocol suite, we do not study mobile communications on other network protocols here, such as Novel IPX, Microsoft NetBEUI, though they are also very popular and implemented in some general network operating systems. Second, this research work is aimed to study the mobile communication mechanisms inside TCP/IP systems, the topics of mobile communications in lower layers (physical and link layer), such as wireless LAN and cellular phone, are out of the scope of this work, given TCP/IP protocol itself does not specify special or dedicated link layer protocols and physical media.

1.3.1 Goal of Mobile Communication

As mentioned above, most of current mobile and wireless communication systems provide only mobile data communication services at the link layer. In TCP/IP internet environment, only a communication channel is provided between mobile host system and a fixed internet Access Point. In most cases, accessing an internet through a fixed AP incurs inefficient use of communication link and poor system performance. For example, a long distance link provided by telephony or Public Switched Network (e.g. X.25) is expensive and the speed is low. The poor utilisation is another drawback. The fixed AP is required by TCP/IP protocol itself, not limited by low level mobile system. Therefore, the principal goal of related research work in mobile TCP/IP communications, including this research work, is aimed to avoid fixed AP and enable the mobile system to connect to a local internet via the nearest network facility. The key issue is to allow mobile host systems to migrate and dynamically integrate into any local

internet and work as a normal host system, then to allow users continuously obtain access to the Internet.

1.3.2 Portable and Mobile Communication

For further discussion of potential mobile TCP/IP solutions, we differentiate portable and mobile communications in an internet environment.

From user's point of view, a mobile system can provide same or similar functions wherever it goes as if it was on the home network. From system viewpoint, generally speaking, a mobile system can dynamically change its attach point to the Internet, and its TCP/IP communications protocol can work properly in a foreign network. In all further discussions of mobile TCP/IP communications, we differentiate between mobile and portable communications modes.

As stated in [73, 29], portable and mobile communications in the Internet context have different meanings. The portability means that a host system can change its attach points hop by hop. At one location, or an internet, provided some standard services (such as DHCP) are supported, the host systems with built-in portability support are flexible enough to adjust its system configurations to work in a new environment that is different from its home environment or previous environment, and such communications services provided by TCP/IP can be continuous if and only if the attach point is fixed in that period. However, a portable system can only support off-line moving, all communications will be interrupted when the host system migrates from one place to another.

Mobility requires that a network system be able to provide continuous communications services at all times to the mobile hosts or (units), even when the host is moving and changing its network attach point. The ideal mobile network communication services for a mobile TCP/IP host are supposed to be able on-line moving, which is like the mobile communication services provided by a GSM cellular phone at data link layer. A GSM phone user can keep talking at all times so long as the area is covered by GSM network. In the Internet context, by the meaning of mobility, a mobile host should be able to keep all existing application sessions active at all time

within the mobile TCP/IP system. Whenever users on such mobile TCP/IP hosts want to access a remote service or wherever they are or even moving, the mobility support should be always able to provide access to an Attach Point so as to provide continuous network services.

For example, on a personal computer system with mobile TCP/IP support, a user on this system could be able to use telnet to remotely log into a workstation when he is on a moving vehicle (Figure 1.9). During the movement, this user may traverse several independent networks on his/her way and the telnet session is not interrupted. In other words, the user can continue to interact with the remote system via the same telnet session and is not aware of system movement even the actual internet Access Point has been changing all the way. Noting that this is different from using telnet over a GSM modem. The continuous service provided GSM network leaves the Access Point at network layer constant, whereas a truly mobile TCP/IP network system can support the continuous communication services and hide the change of AP.

As shown in Figure 1.9, at the beginning, the user is connected to *internet A* and the connection with the *host* is through path I; while the user moves to *internet B*, the connection is not interrupted, it just changes its access point and network path from I to II instead, there always exists only one telnet session between mobile user and the host. Logically, the telnet session seems automatically "handed over" from one attach point to another with the geometrical location of the system changing.

1.3.3 Dilemma of Mobile Communication with Current TCP/IP

Current version of TCP/IP protocol suite, particularly IPv4, was originally designed for fixed host systems. The change of AP's can incur a dilemma to TCP/IP protocols.

First, all host systems in an internet will be assigned the same netid in their unique IP addresses. If a portable computer moves away from its original "home" internet and migrates into a new location, the IP addressing and routing schemes require this system to obtain a new local IP address before it can resume internet access. The routing mechanism to a host system is "static", the IP datagrams destined to a host are always routed to the internet whose netid is given in the destination IP address.

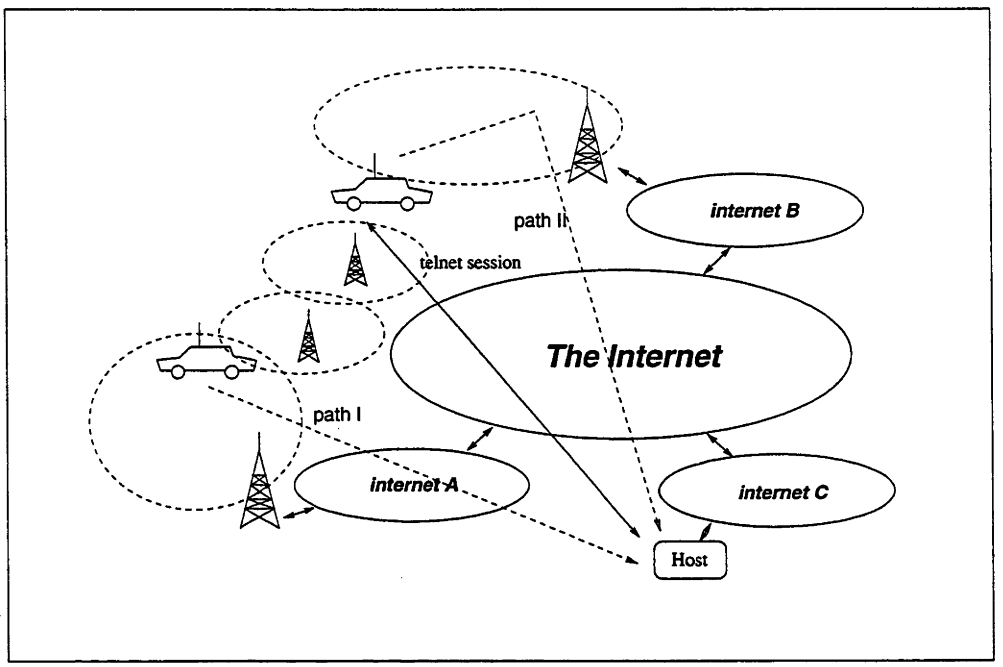


Figure 1.9: A user on a mobile PC

Although dynamic routing protocols enable routers to find a best route from source to destination in a dynamic way, there is no way to find the current location of a mobile host system and routers only know where a given internet is. An IP address uniquely belongs to a host system and this host is always supposed to be in that given internet. If a mobile host still uses its host IP address in a foreign internet, it will not receive any IP datagrams destined to it: the current routing infrastructure will forward those IP datagrams to its home network. Although the host system can move within an internet, given the limitation of covered area of a single internet, the mobility of current TCP/IP host systems are confined to a small area.

If the mobile host system simply changes its IP address when it migrates to a new location, it can use the locally assigned IP address to send and receive data in its current location. However, two problems can be caused by changing IP address. First, all the protocol connections above the network layer will be lost (this will be discussed in next chapter), no continuous mobile services are guaranteed. Second, the mobile host will lose its identifier. The IP address is a very important identifier in the internet environment. The network, transport and even application layer protocols use it to

identify local and peer protocol entities. Change of IP address means change of the important identifier, and the system can not be easily identified. In other words, other systems can not recognise it as it was in its home network. Moreover, during the changing of AP's and IP addresses, no communication services can be provided to application programs, hence the moving is not transparent to network applications.

The NAT protocol and some other address mapping methods are only practical within the same administrative domain, and some special routers are needed. In summary, to enable a host system change its AP and work in a foreign internet, current routing and addressing requires it change its IP address; whereas changing IP address means loss of current connection and identifier. All mobile TCP/IP solutions aim at solving this dilemma.

1.4 Overview of The Present Research

To achieve mobile communications in TCP/IP internet environment, we have considered various schemes. Some will be discussed in next chapter. The goals of our research include:

- providing transparent mobile TCP/IP communications services to application programs, and all mobile operations will be completed without intervention of users or application programs;
- modeling mobile TCP/IP systems to provide a means for comparing the performance of mobile TCP/IP systems;
- proposing an important mobile application — mobile file system which is flexible and adaptive to work in different mobile environments;

The major highlights of our research work are as follows:

- Defining functional model to analyse the common structure of a mobile host system.
- A novel solution of Two-Step mobile TCP to which provide portable communication services in the network layer and then mobile TCP communications to

applications. This solution is different from existing solutions and can perform better in any mobile environment.

- Two performance models are built and applied to do system performance analysis: Time Sequential Model and Traffic Model. No related or similar research work has been reported before.

- A mobile file filtering mechanism is proposed, which is aimed at reducing the traffic for synchronising duplications of files. based on this mechanism, an adaptive mobile file system is devised and presented here, which can be easily tailored for personal use in a mobile environment.

The structure of this thesis is as follows. In the next chapter we provide a survey of related research work in mobile TCP/IP and mobile file systems. In Chapter 3 we give the basic scheme of our Two-Step solution. Chapter 4 and 5 present the first and second step in Two-Step mobile TCP solution. Performance analysis and simulation work are reported in Chapter 6. Chapter 7 discusses the mobile file system. Chapter 8 concludes this thesis.

Chapter 2

Background of Mobile TCP/IP

2.1 Functional Layered Model

In a network system, the communication protocols or their implementations, which are called entities, can be collected into hierarchical layers. Each layer conducts a set of related functions to support upper layers, while making use of the services provided by underlying layers. We refer to *layer* as a set of protocols which complete the defined functions. For example, when we mention "network layer", we do not refer to only IP protocol, though it is the core protocol; we are in fact talking about all protocols in the network layer of a particular system, which can include ICMP, special protocols, etc.

2.1.1 Three Types of Layers

To discuss our functional layered model for generic mobile network system, the ISO/OSI seven layer model is employed as the basic framework. In addition, based on a given layer's behaviour and functionality in a mobile environment, we classify it into one of three categories, namely static, portable and mobile.

As given in Figure 2.2, the system can be observed in three different periods:

pre-moving: the observed system stays in a fixed location, which can be the home network or any guest internet. During this period, the system is connected to internetwork through a fixed AP which is provided by the current internet.

moving: the observed system is changing its geographical location from previous location to next. During this period, the system either changes its AP's or has no

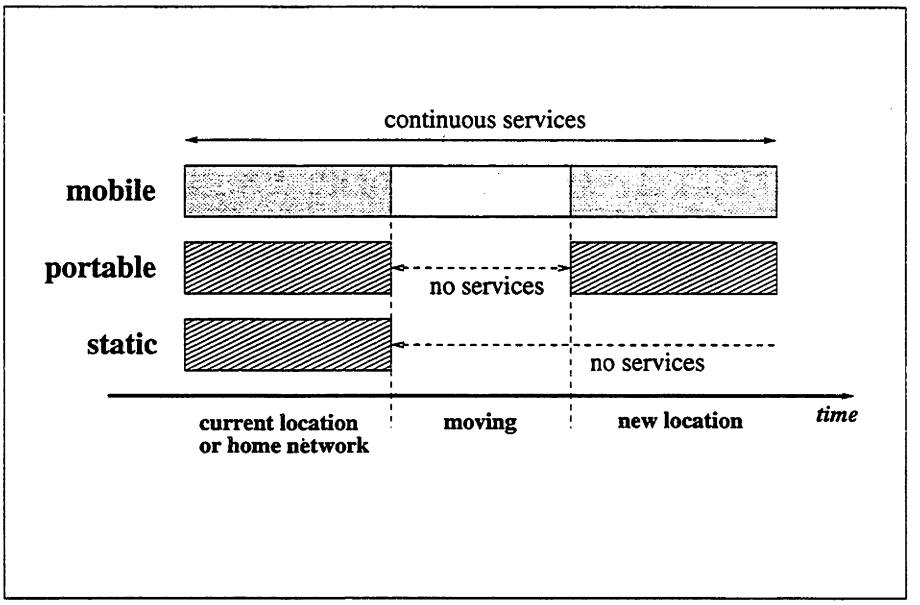


Figure 2.1: Static,Portable and Mobile Layer

connection.

post-moving: the observed system arrives at a current location, in which a local new AP is obtained. During this period, the system does not change AP like in *pre-moving* period.

A single movement consists of these three periods, and the whole observation time is composed by one or more movements. Here we assume that all protocols will take identical actions for all movements during one observation, hence we can discuss the protocol behaviour on generic movements rather than particular individual movements.

Figure 2.1 schematically shows us the functional behaviour of these three types of protocol layers:

static layer this type of protocol layer can only work in the home network environment. It requires a set of fixed system environment variables, which define an invariable environment. This type of layer is not flexible, it can not function correctly during the moving period or at other locations during *post-moving* period.

portable layer this type of protocol layer can provide the portable communication

services, which means that the protocol layer can work properly at different locations, including the home network. That means the change of AP and working environment does not affect the portable layer. However, no services are guaranteed during th e moving period.

mobile layer this type of protocol layer can provide continuous services during all kinds of three periods.

A static layer can be portable if its protocol set is enhanced to become capable of working in a new environment. Obviously, the mobile layer has portability features, because it can work at different locations. Portable layer can also be extended to a mobile layer provided the service-absent gap during moving period can be filled. Therefore, the relationship among these three types of layers is clear: static can be expanded to portable and portable can be enhanced to be mobile.

In the TCP/IP internet protocol suite, almost all existing protocols are designed for fixed host systems, no portable and mobile communication services are taken into account. For example, in an X terminal system, the BOOTP and tftp protocols can only work in a pre-configured environment. It requires local server systems to support the necessary system files to boot an X terminal up. Therefore, the application layer of a X terminal is *static* layer. Moreover, transport and network layers in TCP/IP host systems are static, because TCP, UDP and IP protocols require constant IP address, whereas change of locations or APs requires change of the IP address. Therefore, a normal TCP/IP system needs some enhancements to its protocol layers to support mobile operations. Different methods to enhance the current TCP/IP system lead to different system structure.

2.1.2 Generic Functional Model of Mobile System

In section 2.1.1, we discussed the characteristics of different types of protocol layers in a given layer. Here, we are going to discuss the relationships among these layers in one system.

To analyse different mobile systems we propose a generic functional model, as shown in Figure 2.2. Here the mobile communications are implemented in layer L_i.

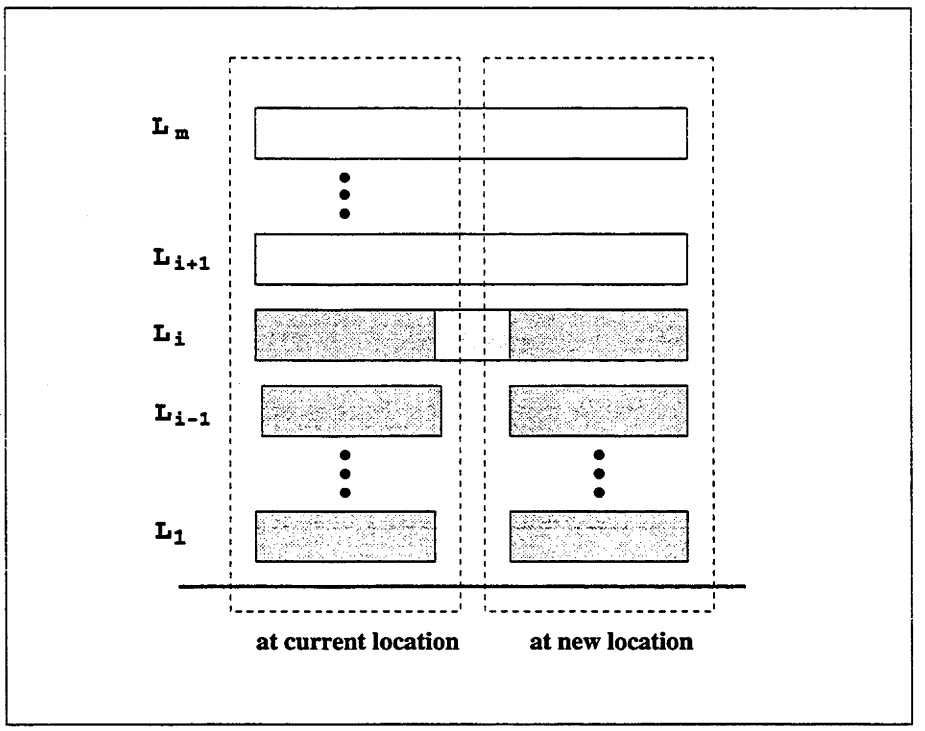


Figure 2.2: Layered Model

As we know, a mobile system is supposed to provide a harmonious working environment to application programs so as to support truly ubiquitous mobile computing. In contrast, the mobile computing system itself does not imply that all protocol layers in a mobile system must be mobile and that is not necessary. In the layered model given in Figure 2.2, the application layer (L_i)is the ultimate source and destination of data being transfered across network systems, which must be able to provide continuous communication services to application programs in turn to support a mobile network computing environment that is not affected by system movements.In order to hide all mobile operations and achieve transparent mobility, the communication gap must be bridged between in layer L_1 and L_m. The cooperation of these layers provide the mobile communications to the ultimate applications.

From Figure 2.2, it is obvious that the mobile communication services or mobility of communication systems is required only at one layer. Assuming that layer L_i supports mobile communications, in other words, it is a mobile layer and provides mobile communication services to the next layer above, L_{i+1}. Therefore, no matter

whether L_{i+1} is a mobile layer or not, it can work in any period given continuous communication services is supported by mobile layer L_i and a constant environment is maintained. Consequently, we have that all layers L_j, $j \geq i$, can work all time. To all layers below layer L_i, the basic requirement is that these layers must be portable layers otherwise the mobility can not be supported at layer L_i. Assuming that the lowest static layer is L_k, $k < i$. Further assuming $k = 1$, which is the lowest layer in the system. Because L_1 is a static protocol layer, as shown in Figure 2.1, it can not provide communication services outside of the home network. That means layer L_2 cannot obtain services from layer L_1. In turn any layer L_m, $m > 1$ can not obtain communication services from the next lower layer, hence the system can not work at other location but home network. The same situation happens If layer L_k, $i > k > 1$ is a static layer. Although all layers $L_j, j < k$ is portable, L_j can not provide communication services and neither all layers form L_k up to $L_(i-1)$. Therefore, Layer L_i can not work at a new location.

The mobile layer L_i has two basic tasks:

- bridging the communication gaps during the moving period. Because portable layers can not provide communication services during moving period, the mobile layer needs to shield the interruption from upper layer, and continuously support communication requests from upper layers;

- providing constant communication environment. When the mobile system migrates into a new network, both mobile and portable layers need to change internal system parameters to suit the new environment. However, these changes, or mobile operations need to be hidden by the mobile layer so as to provide a constant communication environment to these layers above it.

In summary, all layers in a mobile system can be static, portable or mobile layer as shown in Figure 2.1. A mobile system requires one layer to provide continuous mobile communication services in its layered model (Figure 2.2). There are no special requirements to all layers above the mobile layer, whereas all underlying layers must be portable layers to support necessary communications at visiting locations. In most networking environments, the lowest layer — link layer is portable, which provides a

basic promise to implement mobile communications in the global Internet. For example, the variety of Ethernet connections are portable, the user can unplug a Ethernet connector (either RJ45 or BNC) from current connection point and plug it in at a new location, given the Ethernet protocol and connectors are standard in all locations.

2.2 Existing Mobile TCP/IP Solutions

In the TCP/IP protocol suite, to support mobile communications the mobile layer can be placed in all four protocol layers, namely link layer, network layer, transport layer and application layer. Existing solutions can be distinguished based on the primary layer on which they provide mobility.

2.2.1 Link Layer Solution

The link layer in TCP/IP protocol suite includes the bottom two layers in ISO/OSI model: physical and data link layers. The protocols in these two layers are mainly related to the physical connection, signal transmission and data framing, sequencing and error detection/correction. The link layer provides a well-defined service interface to internet (network) layer the data frame transfer channel to a directly connected system.

Although internet protocol suite does not specify any particular link layer protocol dedicated for TCP/IP systems, link layer is still a potential place to support mobile communications in an internet environment. As we know, most link layer connections, such as Ethernet, can provide a portable communication link to provide mobile communications to TCP/IP protocol layers,

However, the connectivity of the data link is limited in certain areas. For example, any two Ethernet segment can not be connected over more than five repeaters. To support portable and mobile communications in the Internet environment, the network layer or even higher layers will be involved.

The Mobile Data Link (MDL) [33] is such a proposal that makes use of internet TCP/IP protocols to create a mobile link layer environment, and in turn to support TCP/IP mobile communications. Its layered model is shown in Figure 2.3. The MDL

layer makes use of UDP services to make several physical internet networks seamlessly integrated into one virtual mobile network.

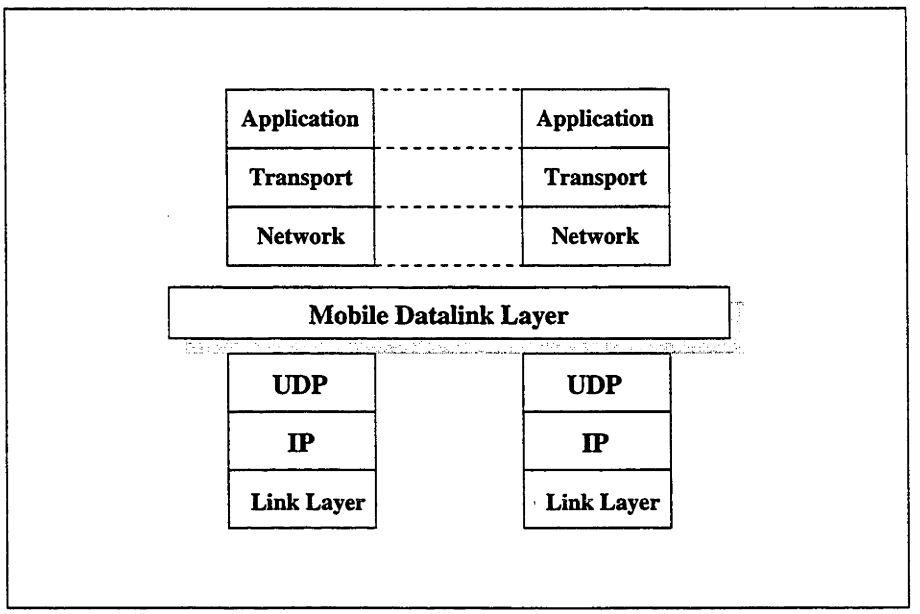


Figure 2.3: Layered Model of Mobile Data Link

MDL suggests that mobility be transparently supported at the lowest link layer above which all running protocol entities are not conscious of any mobility support. Each MDL-supported network has a Access Point (AP) for mobile hosts to connect to it. Given that MDL is basically designed for wireless host systems, AP is also a bridge to connect the wireless segment to the wired network segment.

When a mobile host is in its home network, all data link packets will be sent and received in a normal way and its home AP acts as a regular bridge. As shown in Figure 2.4, MH_2 and MH_3 connect to their home AP (AP_1 and AP_2, respectively) directly. If the mobile host moves into a guest network, as MH_1, its home access point, AP_1, and the guest access point, AP_2, will cooperate with each other to forward data link packets to/from the mobile host by encapsulating data link packets into IP datagrams.

After moving into area covered by AP_2, first MH_1 registers with AP_2, then AP_2 sends its own IP address and MH_1's link layer address back to AP_1. From now on, AP_1 will capture all link layer packets addressed to MH_1's link layer address and broadcasting packets and forward these packets to AP_2 by using UDP packets. AP_2 then forwards

these packets locally to MH_1. Similarly, all link layer packets sent by MH_1 will be forwarded back to its home network first. Therefore, to all other network hosts MH_1 seems stable at its home network.

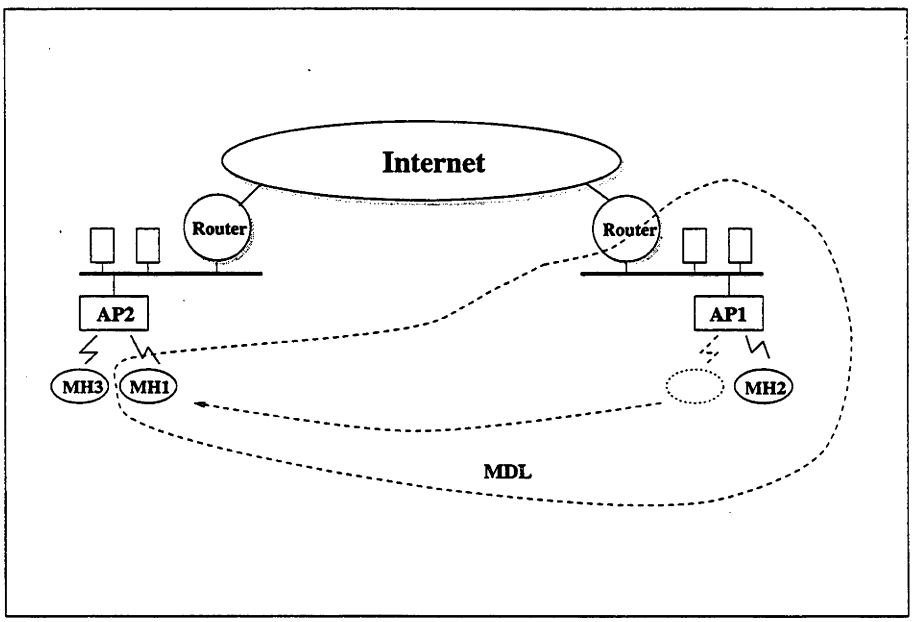


Figure 2.4: Routing of Mobile Data Link

The key issue is the association among *MH*, home *AP* and its current registered *AP*. These three kinds of component systems form the basis of the mobile data link layer. The Internet is a data carrier here, which provides connections among *AP*'s. Hence, the main advantage of MDL is that any network and transport protocols can run over MDL and obtain transparent mobile communication services, because MDL keeps all *MH*s in its home network logically.

However, main concerns with MDL are system performance and compatibility. Since every forwarded link layer packet will be encapsulated into a UDP packet, the payload of protocol units is quite low and the use of communication bandwidth is not efficient as well. Meanwhile all packets to and from a *MH* away from home are routed back to its home *AP* first, this incurs the potential much longer routing and traffic congestion at *AP*s. Furthermore, there is at least one *AP* in every participating network to support link layer packet forwarding. All access points have to know each other for correct forwarding.

2.2.2 Network Layer Solutions

The network layer is mainly responsible for interconnect individual local networks together and transfer data from source to destination. It deals with end-to-end transmission.

Most of recent research work on mobile TCP/IP has been conducted at the network layer from where the key problem stems: IP addressing and routing. The mobile network layer solutions are aimed at supporting mobile IP communications services to the transport layer and provide a seamless mobile network environment. Given mobile or portable link layer support, IP is required to provide a continuous datagram services internally to the transport layer in TCP/IP systems.

There are four major Mobile Host Protocol (MHP) proposals, namely, Sony MHP[73], Columbia MHP[14, 15], IBM MHP [36, 37][1] and IMHP[9, 34, 35]. The IMHP proposal was derived from the others, and has now been adopted by IETF as the Internet draft standard. All of these four MHPs share a common technology that is to provide mobility at the network layer and keep the IP addresses of mobile hosts constant to all the upper layers. Hence, no transport or application protocol entities will be aware of any mobile operations. All mobile functions are completed at the network layer and mobility features are hidden from the upper layers. This is a straightforward way to support mobile hosts given only that the network layer's functionality has to be enhanced.

Columbia MHP

Columbia MHP was proposed by researchers from the Department of Computer Science department at Columbia University, which was primarily designed for the campus network environment. The basic idea is that all mobile systems always use constant IP addresses in their TCP/IP protocol stacks, and additional routing support is added to handle the mobile traffic.

The functional layered model is given in Figure 2.5. All the mobile systems will cooperate with special routing systems to support a mobile network environment.

[1]David Johnson proposed similar MHP using IP LSRR[16, 17]

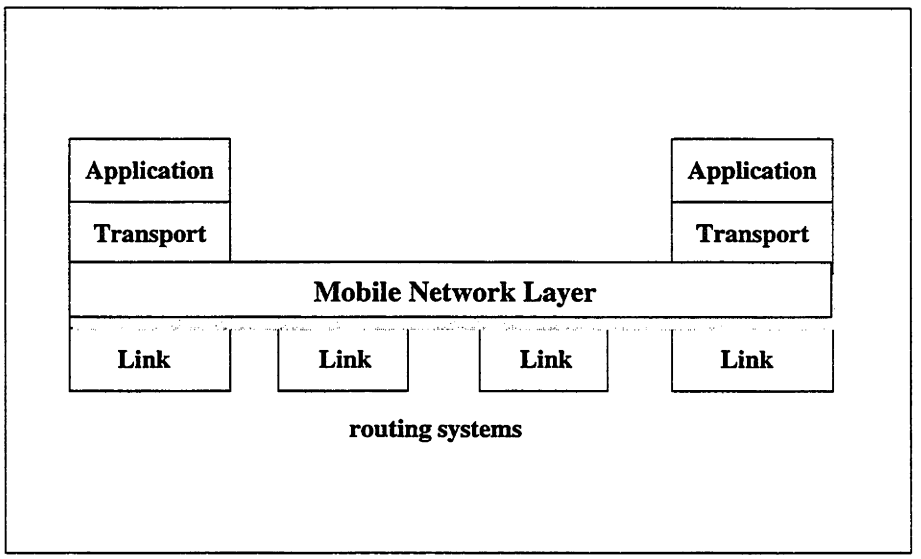


Figure 2.5: Layered Model of Columbia MHP

Typically a campus network consists of a set of subnets and a backbone which interconnects all these subnets, as shown in Figure 2.6. At each mobile-supported subnet, there is an ancillary device, called Mobile Support Router[2]. MSRs are responsible for routing IP datagrams for mobile hosts to their current subnets. The regular routers and corresponding hosts (either fixed or mobile) send IP datagrams to a nearest MSRs first, then MSRs find out the current locations of addressed MHs within the campus environment and forward the traffic to correct destinations.

There are three basic mobile operations among MHs and MSRs:

- MH-to-MSR Handshaking. When a mobile host moves into a new subnet, it has to let the local MSR know its presence, then other MSRs can forward packets for the MH;

- MSR-MSR information exchange. When a MSR gets a new MH registered with it, it sends a message to the MSR which handled the MH before, to enable it forward data to current MH's location. The previous MSR will also inform other MSRs that keep sending data to the MH through it. An MSR also can broadcast requests to all other MSR's to locate a MH;

[2]in [14] it is also called Mobile Support Station (MSS).

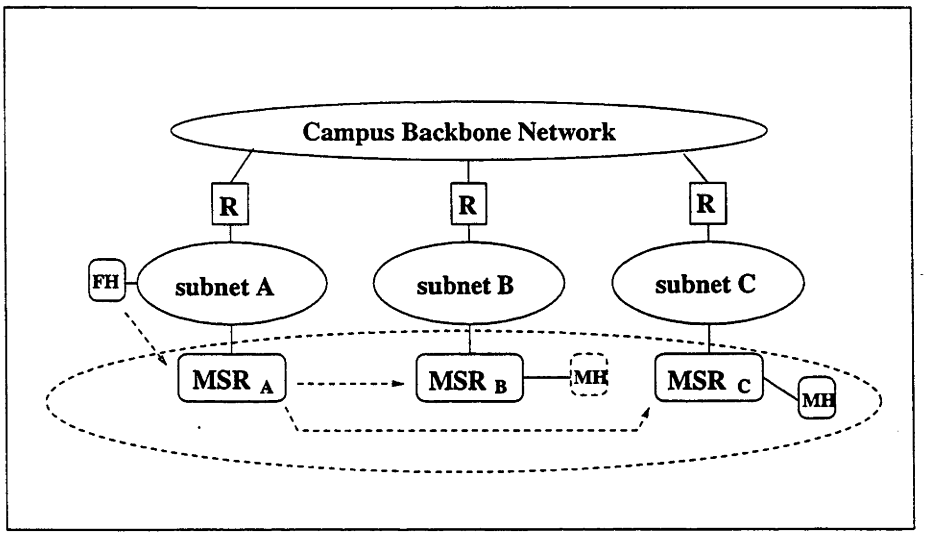


Figure 2.6: Routing of Columbia MHP

- Discovery of MSRs. Each MSR should know all IP addresses of MSRs on campus. The list of MSRs' IP addresses will be dynamically compiled and distributed.

All mobile hosts can be viewed as being located in one mobile subnet. MSRs are the routers to this subnet. If a fixed host needs to talk to an MH at subnet A, as shown in Figure 2.6 it sends packet to local MSR according to the static route to the mobile subnet. Then MSR_A locates the current location of the addressed MH within the mobile subnet by sending a request message to MSRs in subnet B and C, at the beginning, MSR_C responses the request to MSR_A, and the traffic is handled by MSR_A and MSR_C. When MH moves into subnet B, first it registers itself with MSR_B, and MSR_B will inform MSR_C. Then, MSR_C sends the redirection information to MSR_A and forwards all data destined to the MH to MSR_B. From now on, MSR_A and MSR_B continue to handle the traffic between the fixed host and the MH.

In Columbia MHP, mobility is supported by MSRs and the additional registration functions of mobile hosts. MSRs are deployed on all subnets in campus environment and the MHs can always stay on the same subnet as well. Therefore, no IP address needs changing in any MH. The data forwarded between previous and current MSRs will bridge the communication gaps while the MH is moving. This MHP also can

be expanded to suit even larger network environment The basic requirement is that there is a need for an MSR to handle the mobile traffic in certain areas. Therefore, Columbia MHP is not suitable for the global Internet environment. The first reason is that all MSRs need to know each others IP addresses and exchange location information about all mobile hosts among them. In a network environment with large number of subnets and mobile hosts, the cooperation among MSRs can become exhibitive expensive. Second, the deployment of MSRs in all subnets is almost impractical in terms of cost and administrative issues in a large network environment, especially across different administrative domains.

Sony MHP

Sony Computer Science Laboratory introduced a *virtual network* concept in [73] to support transparent migration of mobile hosts at the network layer. Basically, all existing TCP/IP internetworks are viewed as a *physical network*, in which the routing infrastructure routes the traffic based on network id and host id of destination IP address. The usual IP address is correspondingly called *physical IP*. The *virtual network* is constructed on top of the *physical network*, which is only visible to the transport layer. A mobile host always moves within a *virtual network* no matter which *physical network* it is visiting. The layered model of Sony MHP is shown in Figure 2.7. The mobile system and routing system will have two sublayers within network layer, the virtual subnet and physical subnet. Each mobile host belongs to one *virtual network* in which it obtains a constant IP address. The physical IP address of a mobile host is assigned dynamically according to its current location in the *physical network*.

Given the constant virtual IP address is used, the mobile system can be treated as a fixed host system in the *virtual network*. The transport layer and application layer use virtual IP address to identify the mobile host so as to shield all mobile operations from transport and application layers. Hence, the system can roam in the *virtual network* transparently to users and network applications. The IP datagrams to and from the mobile system are routed by the *physical network* and the physical address is employed here.

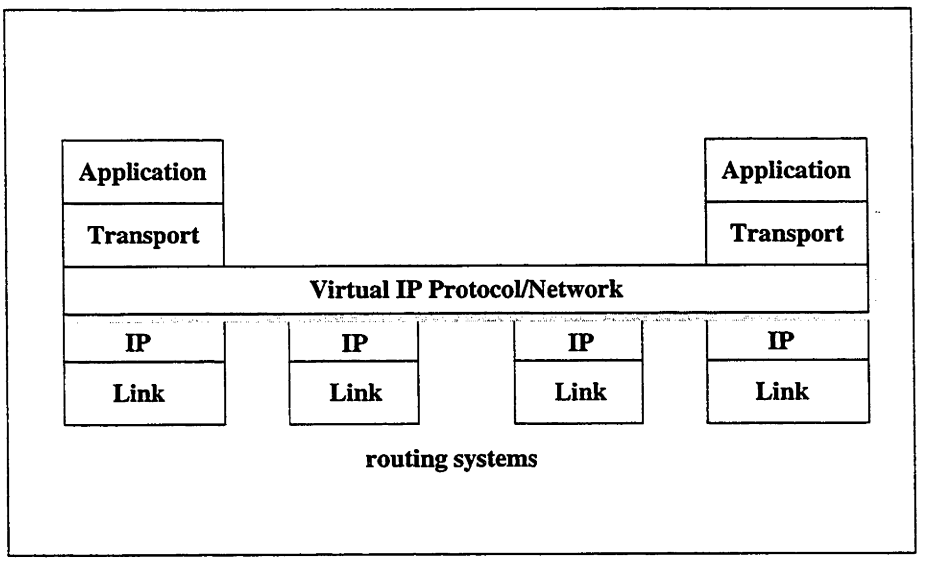


Figure 2.7: Layered Model of Sony MHP

The key issue to implement the *virtual network* is the dynamic address mapping between the virtual IP and physical IP. As shown in Figure 2.8, when the mobile host stays in its home network, also called *native network*, it has same virtual and physical IP address, the existing routing system can easily forward IP datagrams to mobile host.

When the mobile host migrates into a guest network, it is dynamically assigned a local physical IP address that is used for routing IP datagrams. However, the corresponding hosts always use constant virtual IP addresses to send data to mobile systems. Therefore, a mechanism of address mapping is required.

As shown in Figure 2.8, some special routers, denoted R^*, are deployed in network to handle mobile traffic. An IP datagram from the fixed host to mobile host will be routed by regular routers along the path from fixed host to mobile host's home network. If there is no special router on the path, this IP datagram will be forwarded to mobile host's home network, and its home R^* forwards it to mobile host's current location by using its current physical address. In fact, home special router maps the virtual IP address to the physical one. The mapping can be cached by all special routers in the network. If there is a special router in between the fixed host and the mobile host's home network, it will map the virtual address to its physical address, as

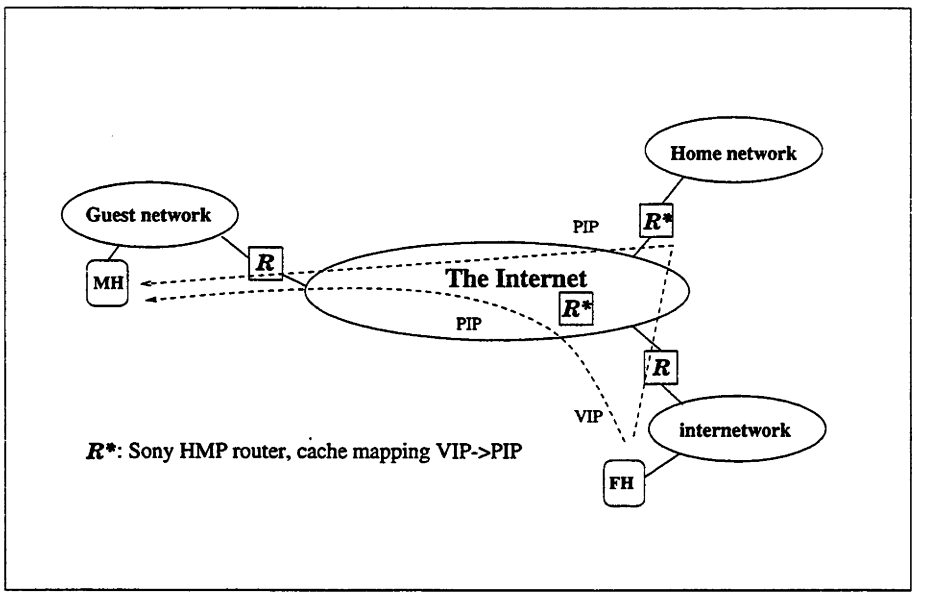


Figure 2.8: Routing of Sony MHP

what home special router does, and forward it to the mobile host.

It is obvious that the home special router must know the current location of a mobile system so that the home special router can forward traffic correctly to the current network of a mobile system. The mobile traffic may be routed in a suboptimal way. In worst case, all traffic to a mobile host will be routed to the mobile host's home network and then forwarded to its current location. However, the more special routers deployed in the network, the more likely a special router can be met between the source and destination of a packet so as to be routed to the mobile host directly from that point.

Sony virtual IP solution is different from Columbia method given that a mobile host has two IP addresses and its network layer protocol is enhanced. This method can be used in a larger network environment, because usual routing infrastructure can be used to forward datagrams to mobile hosts. However, the cost of information exchange among special routers becomes more expensive with the growth of number of special routers and mobile hosts. The protocol of this method is not compatible with current IP protocol.

IBM MHP

The IBM MHP[69] employs a standard IP option, Loose Source and Record Routing(LSRR), to forward IP datagrams to mobile hosts. Normally the routing of a packet from its source to its destination is dynamically and independently determined by all routers along the path. Whereas Source Routing enables the sender (source) to define the the routing all way to destination. In fact, LSRR option in IP packet consists of a list of IP addresses of intermediate routers. The sender will send this packet with LSRR to the first router in the list, and this router will pick up the next IP address in the LSRR list and forward it to that router, and so on. The all routers in the list will be visited. However, the route between two adjacent routers in the LSRR list can be direct or indirect. By the mean of indirect, there can be other routers in the route between two adjacent routers. This is also what "loose" in LSRR means.

The layer structure of IBM MIP is shown in Figure 2.9.

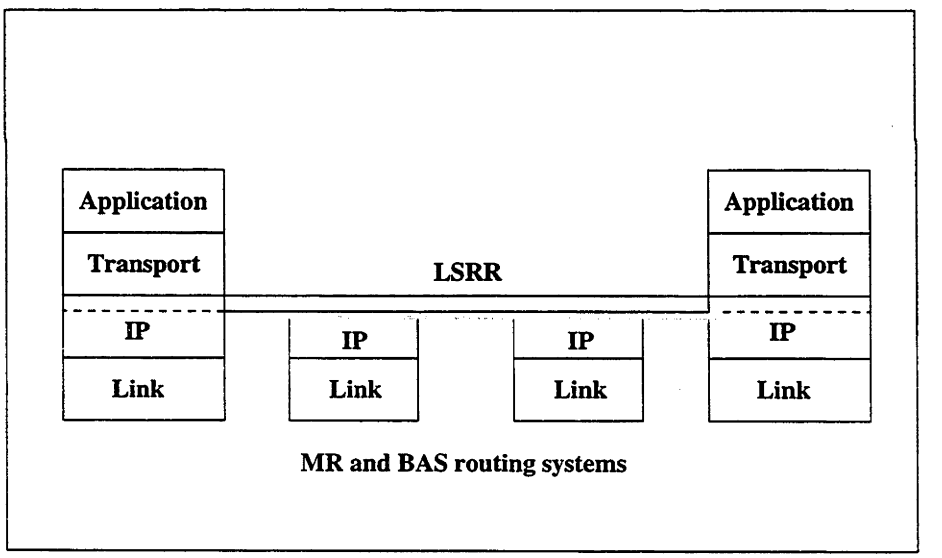


Figure 2.9: Layered Model of IBM MHP

A mobile host always uses a unique IP address to send and receive IP datagrams. If it stays in the home network, normal routing mechanism can correctly handle the IP datagrams to and from mobile hosts. When a mobile host is away from home network, there are two additional mobile support systems which are employed to support mobile communication:

Mobile Router (MR) given all IP datagrams to a MH will be forwarded to its home network, the MR at home network will catch up all these IP datagrams and forward them to MH's current location;

Base Station (BAS) normal routers can only deliver IP datagrams for local hosts directly. BAS is a special router which can send and receive IP datagrams to/from a locally connected foreign MH.

The mobile operation consists of two steps. After arriving at a foreign network, first MH registers with the local BAS, and the BAS can create a dynamic routing to the MH through the local network interface. MH uses local BAS as a default router, all outgoing IP datagrams will be sent to the BAS first, and BAS locally delivers IP datagrams destined to the MH. Second, BAS will inform MH's home MR of its own IP address so as to allow home MR to forward MH's datagrams to the current location.

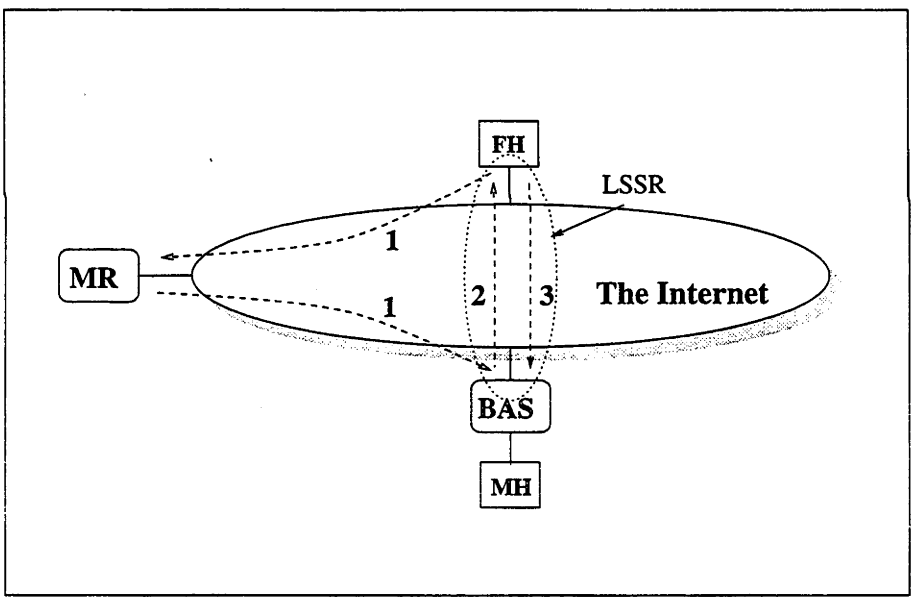


Figure 2.10: Routing of IBM MHP

Figure 2.10 shows the three routes between a fixed host and a mobile host which is currently at a foreign network. Route 1 is the initial route. At the beginning, FH has no information on whereabouts the MH is now. It sends out normal IP datagrams with MH's home IP address as destination. This IP datagram is forwarded by normal routers to MH's home MR. Provided MR knows MH's current BAS, it uses BAS's IP

address as destination and put MH's IP address in LSRR option as last hop. This IP datagram will be forwarded to BAS, and BAS gets MH's IP address from LSRR option and directly delivers it to the MH. The response from MH to FH is sent by BAS along Route 2 which is a direct path to FH. From the LSRR option in the response IP datagram, FH can figure out the directly way back to MH: it uses BAS's IP address as next router, and sends further IP datagrams directly to MH's current location along route 3. From now on, the MH and FH use LSRR options an all IP datagrams to send and receive data in a direct way.

There are no special requirements to the rest of routers along all routes. But all MR, BAS, MH and corresponding FH have to support LSRR option. Although LSRR is a standard option, most TCP/IP hosts do not support it. Like Sony and Columbia's proposals, there must be a special mobile support system at the foreign network. These two requirements limit the application of IBM's proposal.

IMHP

To avoid special requirements on the network that mobile hosts may visit and on the corresponding hosts, Internet Engineering Task Force proposed a more current-TCP/IP friendly and compatible solution: Internet Mobile Host Protocol (IMHP) [38, 67].

This protocol has three functional entities, namely Mobile Node (e.i. mobile host system), Home Agent (HA) and Foreign Agent (FA). The MN is always identified by other system by its permanent home IP address. When a MN moves to a new internet away from home, it will register with a FA if there is one at a foreign network. Then the FA's IP address is called care-of address. If there is no FA, MN can try to obtain a local IP address via other standard protocols, such as DHCP, and this newly obtained IP address becomes MN's current care-of address. In fact, the care-of address is the key to correctly route IP datagrams.

As shown in Figure 2.11, the datagrams sent by a peer system to a MN will be routed to MN's home network, because other systems only know the home IP address of a MN, then the MN's home IP address is always used as the destination IP

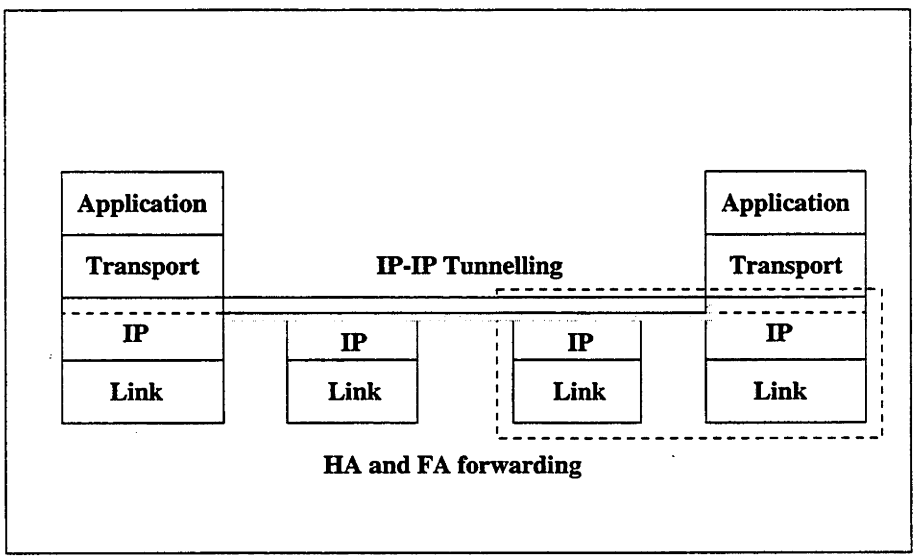


Figure 2.11: Layered Model of IMHP

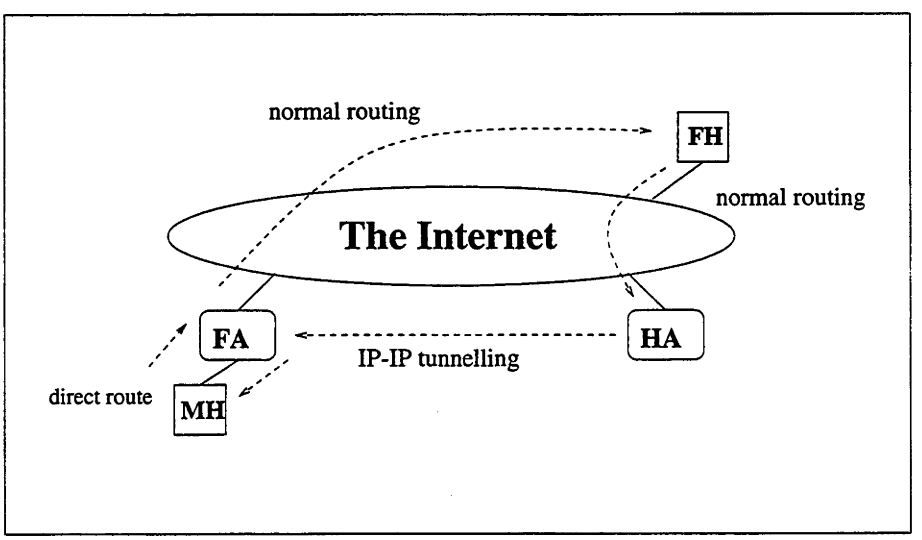


Figure 2.12: Routing of IMHP

address in these datagrams destined to the MN. The HA intercepts these datagrams, and forwards them to the care-of address of the MN, provided the MN has successfully obtained a care-of address and informed its HA. Original datagrams will be encapsulated into new IP datagrams with the care-of address as the destination address, and the whole original datagram is put in data field. This is called IP tunneling. If the FA's IP address is used as the care-of address of the MN, the FA is responsible to strip the IP-IP tunneling packets and deliver the enclosed datagrams locally to the MN. Whereas the MN has been assigned a local IP address and there is no FA, the MN has to receive tunneled data from its HA by itself. All datagrams sent by MN are routed normally by existing routing system back to the peer system.

The solution does not require special routing support all over the Internet. HA is the only forward point and data forwarding can be handled by HA and MN. The FA in a visiting internet is a bonus.

IPv6

In the new version of IP protocol — IPv6 or IPng, the mobility has been already taken into account[66]. IPv6 is a new protocol which is being developed by several IETF working groups. The mobile IP working group is responsible for developing the mobility support protocol in the network layer.

The general idea of IPv6 mobility is similar to IMHP for current IP protocol. The salient difference is that the Foreign Agent is no longer part of the mobile IPv6, and there are two new features in IPv6 which are perfect for supporting mobility. The first one is automatic address configuration. All IPv6 host systems are required to be able to automatically obtain a local IP address. Therefore, without FA's support, a mobile system can easily gain a care-of address by itself. The second is security. The location information of mobile nodes can be securely updated by mobile nodes given the security communication features provided by IPv6. Moreover, a mobile node can inform correspondent hosts of its current location or its current care-of address by using IPv6's new *destination options*.

IPv6 is still being underway toward standard. In our current research work, we

focus on IPv4. However, our research work can be easily modified to work with IPv6.

Summary

These MHPs have achieved a degree of success. For example, Columbia MHP successfully provides mobile communications in a wide campus network environment with several internetworks. IBM MHP employs only Source Routing option of current IPv4, and some additional support from home network systems. Implementation of IMHP under Linux has proved it is a practical Internet wide method, which has best compatibility with current FH systems and routing facilities as well. However, unfortunately, due to the limitations of both the addressing and routing schemes in the current Internet, all MHPs sacrifice either backward compatibility or communication performance for the sake of mobility of the IP environment[3, 29]. Sony's solution[29] introduces a new option in IP header which contains mobile host's virtual IP address and requires that the routers and hosts participating in the Sony MHP cope with address cache and mapping for MHs. Columbia MHP[15] also needs support from mobile support routers that use special routing algorithms and protocols. IBM's solution employs not widely supported IP optional function LSRR. The IMHP's detour routing will affect performance.

Solution	Columbia	Sony	IBM	IMHP
Fwd Point(s)	multiple MSRs	multiple R^*	home MR	HA
Fwd Method	IP-IP tunnel	normal IP	normal IP	IP-IP tunnel
Location Info	among MSRs	among R^*	between MR and BAS	between HA and FA
Protocol	MICP	new IP option	LSRR option	new protocol

Table 2.1: Summary of Mobile IP Solutions: Note that MSR(Mobile Support Router; SR(Sony Router); MR(Mobile Router); BAS(Base Station); HA(Home Agent); FA(Foreign Agent); and LSRR(Loose Source Routing).

As a brief summary, the network layer solutions are aimed to provide a seamless mobile communication services to transport and application layer entities, hence the basic requirement is to keep the IP address constant to the upper layers. Since the existing routing infrastructure cannot properly forward IP datagrams to mobile hosts

that are away from their home internetworks, MHPs focus on two basic tasks: propagation of location information and forwarding IP datagrams through current IP routing systems in the global Internet. When a mobile host moves from one internetwork to another, the MHPs are responsible to propagate the new location information to all forwarding points. Therefore, any forwarding point can forward IP datagrams to mobile hosts. As in Table 2.1, Columbia and Sony MHPs use multiple forwarding points distributed across the Internet. The location information is exchanged within these special routers. Columbia mobile support router uses a special MICP[14] to exchange mobile hosts location information, whereas Sony router collects and caches mapping between the virtual network address and the physical network address from new IP option. IBM MHP and IMHP use a single forwarding point. The location information is exchanged between a mobile host and a single forward point. All these solution use either IP-IP encapsulation (tunneling) or source routing to forward datagrams.

2.2.3 Transport Layer Solutions

The mobile protocol can also be implemented at the transport layer. I-TCP[2] was proposed by researchers at the Department of Computer Science of Rutgers University. The research work is focused on providing mobile TCP services to a mobile system with wireless interface. All network layer solutions are focused on how to route IP datagrams to mobile hosts that move out of a home network, whereas I-TCP is aimed to support mobile TCP communications more efficiently, given that special mobile protocols can be employed at the link, network and even transport layers which are dedicated for improvement of data transfer performance in a wireless and mobile environment. The principle of I-TCP is similar to that of existing network solutions: the IP address of a mobile host will be kept constant in the Internet environment. The Figure 2.13 gives the layered model. The mobile communication is supported by special mobile routing devices and the mobile transport protocol with mobile network/link layer protocols.

As in Figure 2.14, I-TCP suggests that a single TCP connection to a mobile host be separated into two connections. The first one is the special mobile TCP connection

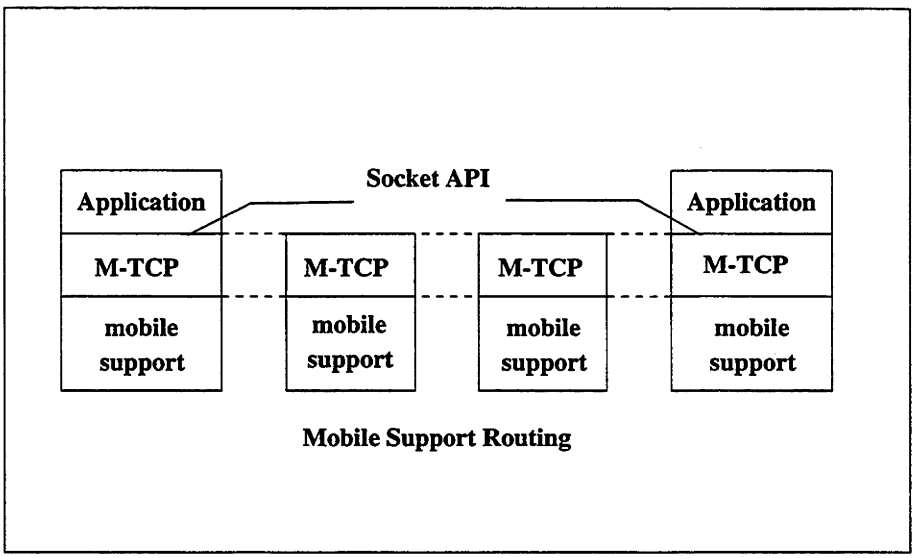


Figure 2.13: Layered Model of Indirect TCP Solution

between the mobile host and the mobile support router. The second part is a normal TCP connection between the mobile support router and the corresponding host.

The mobile TCP protocol between MH and MSR is a light-weight TCP protocol designed for a wireless link. Because the wireless link is usually unreliable and low speed, the mobile TCP can detect the data link status and provide more information to mobile-aware applications for better use of the link. For efficient use of precious wireless bandwidth, the mobile TCP/IP protocol is simplified to reduce protocol overheads, and it can also simplify the complexity of communication system in MH and shift some TCP/IP communications workload to the MSR end. For instance, the complicated TCP congestion control will be handled by MSR. When a connection is being set up between an MH and an FH, the MH first establishes a mobile TCP connection with a local MSR, Then, MSR uses HM's invariant IP address IP_m, to create a regular socket on behalf of MH, then uses this socket to set up a regular TCP connection with fixed host. As shown in Figure 2.14, the MSR is responsible for two things:

- mapping the mobile socket to the corresponding socket that represents it so that data transfer from MH to FH is transparently continued.
- When the MH moves to other area, the represented socket of MH will be automatically handed over from previous MSR to current one.

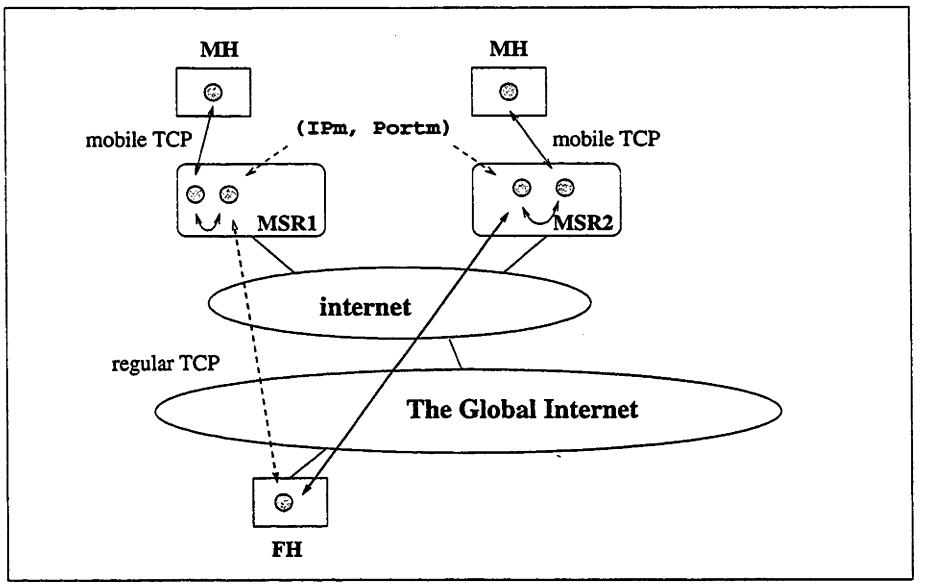


Figure 2.14: Routing of Indirect TCP Solution

The represented socket can be defined as $(IP_m, Port)$, no matter where the MH goes, this socket is always hosted by a local MSR and keeps its identifier constant. Therefore, from the viewpoint of correspondent FH, this socket looks like a normal socket and the mobile nature of an MH is hidden by MSR. On MH, the mobile TCP API is wrapped by regular socket system calls, hence normal network applications can run on an MH as if it were traditional fixed host system. For special mobile-aware applications, mobile TCP can supply more detailed information about low level communication link.

The hand-over between representing sockets occurs when the mobile host moves into another internetwork. The mobile host will first set up a mobile TCP connection with a new mobile support router and the new mobile support router will take over the normal TCP connection from the previous mobile support router by creating the same mapped socket. Note that MSR must be able to dynamically create network interfaces with the IP addresses of currently hosted MH's, because all these sockets that represent different MH's have different IP addresses.

There are limitations for I-TCP. Given that an MSR is required to use currently hosted MHs' IP addresses to create sockets and communicate with other systems, all these MH's IP addresses must be in the same internet. That limits the ability of MSR,

which can host these MH's within same internet. This also limits the moving area of MH within a given internet. Therefore, I-TCP is not able to provide a mobile solution in a large internet environment with more than one subnetwork.

Chapter 3

Two-Step Mobile TCP/IP Solution

3.1 IP Forwarding and Its Issues

All existing TCP/IP network applications and protocols are developed and designed for traditional internet environment, in which all hosts have unique IP addresses and these IP addresses are not related to mobile hosts' current location. To support continuous mobile communications in current TCP/IP internet environment, the straightforward idea is to employ one or more mobile support systems to create an illusion to all traditional TCP/IP systems that the mobile hosts would always stay where they are supposed to be — their home network, so that the IP addresses of mobile systems will be invariant to the whole internet environment and all communications can be established between mobile hosts to traditional hosts by using regular TCP/IP protocols. No matter when and where the mobile hosts move, their "shadows" in home network can continue to communicate with correspondent systems. As shown in Figure 3.1, the correspondent system can use regular TCP/IP protocols to communicate with the shadow of a mobile system[1], and only the regular routing mechanisms are involved between them. However, the mapping of mobile system to its home shadow is the responsibility of mobile support systems.

To provide a "shadow" for certain mobile host in its home network, usually a mobile support system at home network is employed which will be responsible for forwarding to map the "shadow" to its real system. This is the common mechanism in all existing solutions. As we introduced, network layer solutions use IP forwarding

[1] If the mobile system is currently in its home network, the real system and the shadow overlap.

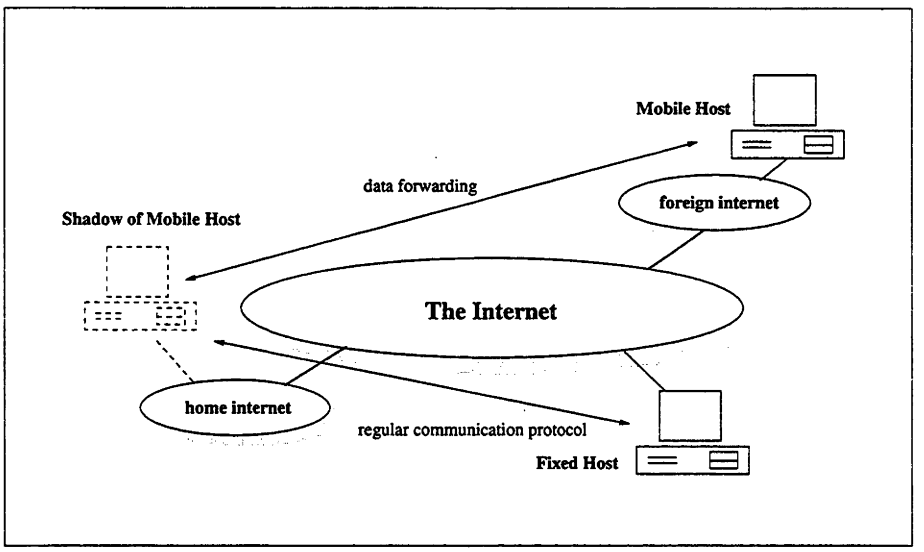


Figure 3.1: Mobile Host and Its Home Shadow

mechanisms; MDL employs even lower level forwarding: all link layer packets will be forwarded between a mobile system and its home network; I-TCP supports TCP level forwarding by mapping a mobile TCP socket to a represented regular TCP socket at MSRs. The correspondent hosts and current regular routing infrastructure is always used to route IP datagrams toward mobile hosts' home internetworks. Different solutions use different forwarding systems and forwarding methods. The commonplace is that at least one system in the home network will look after the traffic for mobile hosts.

There are two issues with IP forwarding. The first is that special routing devices are required to deploy in the network as forwarding points, because forwarding IP datagram is not supported by current TCP/IP routing infrastructure. The second is performance. As we know, current routing protocols and infrastructure can find an available "shortest" path from source to destination. However, in mobile environment, the "destination" is mobile host's home network, not current location. The traffic from the fixed host to a mobile system goes along a detour path, which means longer delay and lower reliability.

3.2 Basic Idea of Two-step Solution

Our Two-Step solution is aimed to avoid the IP datagram forwarding so as to simplify the whole mobile support systems and improve performance. In the first step, portable communication is supported at a network layer, or so-called portable IP. The mobile host firstly will be automatically reconfigured as if it were a local attached fixed system. Therefore, all traffic to and from mobile hosts can be routed by existing routing infrastructure, and there are no special mobility support systems will involve in handling mobile traffic. This can largely reduce the system overhead. Furthermore, all routings to mobile hosts are dynamically determined by existing routing protocols, hence there is no triangle or detour routing. This also can save precious network bandwidth and alleviate network congestion. In fact, our portable IP system can work independently to satisfy most mobile users' requirement of internet connectivity. Usually, mobile users only require a temporary connection in certain location. Such as a technical support staff needs consult a knowledge base during on-site service, or a professional needs to check email at a conference site. Most application sessions, such as email, web browsers or database access are short. Requirement of internet connection during the movement from place to place is not so often.

Reassign of IP addresses to mobile hosts at foreign networks will cause all existing connections are broken. Although the loss of existing communication connections and intermittent service are all tolerable in portable communications, the application layer, application programs and mobile users will not obtain continuously mobile communication services.

To provide mobile communication environment, we create a constant IP environment on top of the transport layer — the mobile TCP communication services are supported at the socket API level. The application layer can obtain mobile communication support from socket API calls. All application programs and the human users are shielded from the detail of mobile and portable communication operations. The main difference between two-step solution and I-TCP is that I-TCP uses special protocols between a mobile host and a MSR and a mobile host can only connect to a local MSR. Whereas our two-step solution has no such limitation, all mobile hosts can

directly connect to any internet.

3.3 Layered Model

Our Two-Step solution is to provide portable and mobile communications in two different layers, rather than put all mobile functions within a single layer. The mobile layer can be viewed as the portability plus continuity. The former ensures that the system can work properly at each location, whereas the later bridges the communication gaps to support continuous mobile communication services.

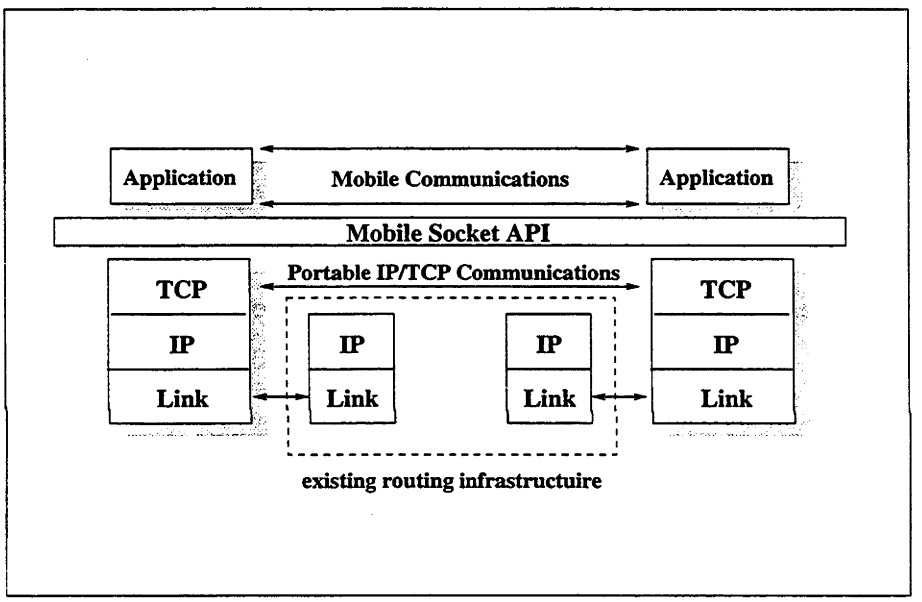


Figure 3.2: Layer Model of Two-Step Solution

As shown in Figure 3.2. assuming that the link layer connections at all foreign networks are portable, the network layer can obtain portable connection automatically. This is true to most nowadays LAN/WAN connections, such as Ethernet, Token Ring, PPP, etc. The link layer services are usually provided by standard interfaces. For example, in Unix system the Ethernet interface is supported as standard device driver, and in Windows, the link layer support is standardised as Network Device Interface Standard. These device drivers can always provide portable services given the physical connections are valid at foreign locations.

Given portable support from link layer, the first step — mobility can be easily

implemented in the network layer. As we know, if the IP address and other location-sensitive system parameters are changed to suit current network environment, the IP protocol can work without any further support. Hence, the implementation of IP protocol needs not modified at mobile system. The basic operation in portable IP is to obtain a local IP address in a foreign network as well as other parameters, then reset the IP protocol entity to use new parameters. The additional functionality in network layer will be implemented outside of IP and other network protocols so as to keep the full comparability.

Although it is hard to draw a line between IP and TCP layer, the service provided by IP is simple which is the datagram service. To make TCP work with portable IP is simple: changing the IP address used by TCP to current local IP address, and he TCP protocol becomes portable. Of course, the TCP will not provide services while moving.

On portable TCP communications, continuity is supported on top of transport layer — Socket API. Because TCP service is required to provide a reliable communication channel between a pair of TCP sockets, the mobile Socket layer is responsible to do two things: to bridge the portable communication gaps, first is to continuously accept I/O requests, either return successfully or not, to applications the socket is always there as available. The second thing is to keep the reliable data transfer at all time. As discussed before, a portable layer is responsible for filling the network service gap caused by underlying portable layers. In our two-step solution, the transport layer protocol — TCP can only provides portable service, and leaves a vacuum period while the system is moving.

The reason to choose Socket API is that Socket API is the most popular network interface on a variety of operating systems, both Unix and Windows system support kernel level API interface, which is quite standard and efficient. Of course, TLI is also a important network programming interface, our idea can easily apply on top of TLI.

From Figure 3.2, we can conclude that Two-Step does not change or enhance TCP/IP protocols in all systems, including mobile host and routing systems. The mobility can be purely implemented by software besides all TCP/IP protocol entities.

To further introduce the basic functionality of portable IP and mobile TCP socket,

we need to analyse the internal requirements of whole TCP/IP network environment for supporting portable and mobile TCP/IP communications. In next section, we discuss the identifier of TCP/IP system first.

3.4 Identifier: Name and Address

In mobile TCP/IP communications, the principal issue is how to find and locate a mobile system in the Internet. Then the peer system can make connection to specified applications or protocol entities. The key is the identifier of the host and the identifier of the application or a network entity. As we know, the IP address is the most important identifier for a system. All existing solutions adopt the basic mechanism — keeping mobile system's IP address constant, so as to maintain the unique identifier of a mobile system for other systems to find and locate the mobile host.

In fact, a TCP/IP system can be identified in many ways. There are two kinds of identifier — name and address. A "name" is to define what it is, which can be used to identify an object in a network system, such as a protocol entity, a whole system, etc. Whereas an "address" is to define where it is, in other words, it 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 important names.

Generally, a DNS name is used as a literal identifier of a TCP/IP-based system. Although a DNS name has a hierarchical naming structure [1], it only reflects the administrative structure of the named objects. DNS name is not an address which can be used to locate a host in the Internet. For example, given a DNS name under `.com`, the system can be in US, Australia or other countries, there is no location information encoded. In fact, the DNS name has nothing to do with routing systems. Whereas an IP address is the only address, which is used to locate a TCP/IP system in the internet environment.

However, usually users use DNS name to name a host. For example, to make a HTTP or telnet connection to a remote system, the DNS name is used in a URL or as a command line parameter to `telnet`. The given DNS name will be translated into an IP address for internal use, because TCP/IP protocol entities in transport and

network layers use IP addresses only. The mapping of DNS name to IP address is one of the tasks of DNS servers, another important task is to do reverse mapping from IP address back to DNS name. Some applications make use of the reverse mapping to double-check its genuineness of a TCP/IP system.

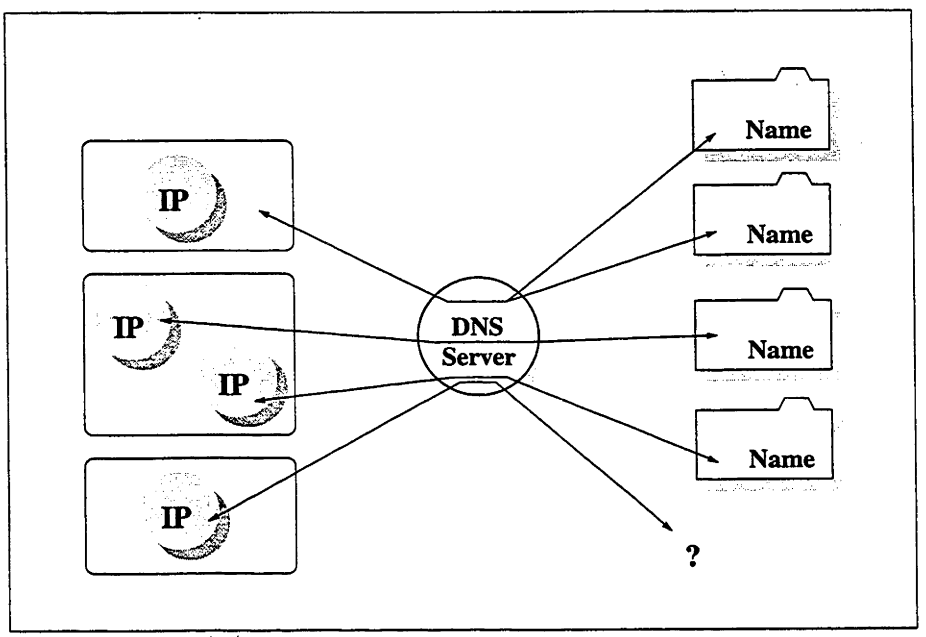


Figure 3.3: The Relationship of DNS Name, IP Address and TCP/IP System

The relationships are shown in Figure 3.3. A TCP/IP system[2] can have more than one IP addresses at the same time, which are used for different network interfaces and virtual hosting. However, one IP address can only belong to one system at any time. If one IP address had been assigned to two different system at the same time, the routing system could be confused to which it was supposed to deliver the IP datagrams with this IP address as a destination. The IP address can be reused by different systems in different times, not at the same time. Also, a system can have one or more DNS names. All these names will be mapped to the IP addresses which belong to the same system. The mapping between IP address and DNS name is done by DNS systems, which consist of thousands of DNS servers in the Internet. The basic functions of DNS server is to map DNS name to IP address or vice versa. The relationship between IP address

[2]Besides TCP/IP hosts, routers and special TCP/IP device systems can also have IP addresses.

and DNS name is one-to-many. Of course, an IP address may have no corresponding DNS name, however, a useful DNS name must be bound to an IP address.

Besides the DNS name and IP address, the port number of a transport layer protocol is also an important name, which is used accompanied to further specify a communication entities within a system identified by IP address or DNS name. Generally, in ISO/OSI layered model, both connection-oriented and connectionless communications logically occur between two peer protocol entities in the same layer. All layers make use of the services provided by the underlying layers and in return provide services to its upper layer through service access points (SAP). Each entity can concurrently have more than one communication sessions with one or more peer systems. In other words, there can be more than one active SAPs at certain layer. Therefore, there must be additional mechanisms to differentiate these concurrent sessions. In TCP/IP protocol suite, the transport SAP is assigned a port number to identify the individual SAP.

Functions of IP address

As we discussed early, the IP address takes role in identifying and locating a system. It has two-tie use in TCP/IP environment.

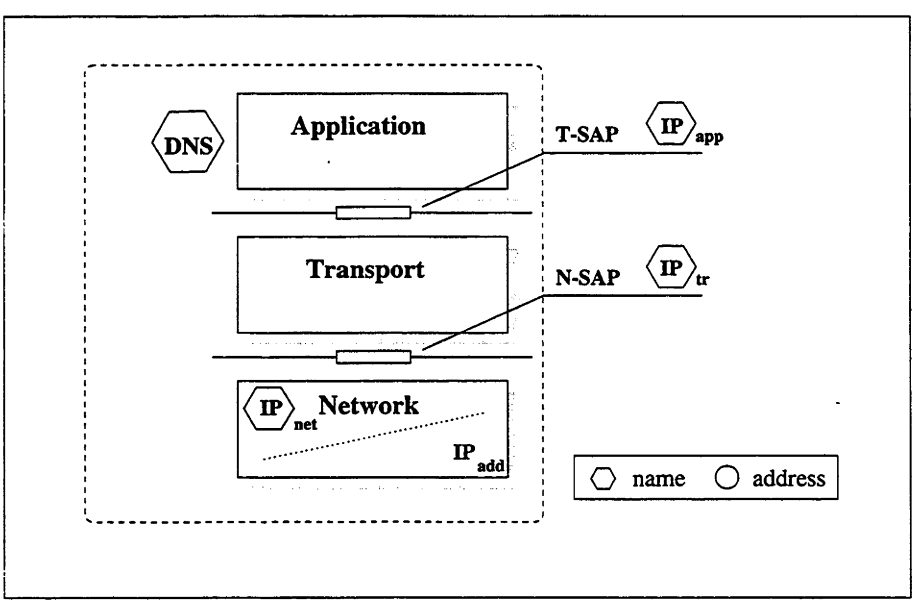


Figure 3.4: Functions of IP Address

Figure 3.4 shows the internal use of an IP in TCP/IP stack, which serves two functions, namely, naming an object such as an SAP, and routing IP datagrams to next hop towards destinations.

In the application layer, all entities have their own transport SAPs (T-SAP) to communicate with peer application entities, which are identified by T-SAPs. A T-SAP is further identified by the transport protocol (UDP or TCP), the port number and the IP address. Here, the IP address functions as a name (denoted IP_{app}). A transport entity can be defined through its network SAP (N-SAP). The name of N-SAP consists of an IP address (denoted IP_{tr}), and a protocol number[3]. Further down, 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}) for routing purpose, which is the IP address involved in the routing mechanism, and appears in the *destination address* field of normal IP header.

In mobile TCP/IP communications, if the IP address is kept constant, we do not need to consider the internal functions of the IP address. However, if you change the IP address from time to time, we need to know the impact of the IP address and how to alleviate the impact. The following section discusses the basic requirements of the IP address, which must be satisfied to provide mobile communications in the topmost layer.

3.5 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. Therefore, based on this preliminary requirement, three basic invariant conditions have to be satisfied:

- constant names for an application entity, including a DNS name and a unique IP address [44];

[3]In IP header, there is field called "protocol", which indicates the transport protocol of data in this IP datagram. TCP is assigned 6 and UDP 17

- 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. [4]

The first condition implies that applications need not be reconfigured when mobile systems change their locations. The hard-coded or configurable IP addresses and domain names must be valid always. And these names will be always mapped to same mobile systems. The second condition is essential which is derived from 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 with a portable-IP system as we proposed in this work.

In fact, the basic idea of mobile communication support in TCP/IP internet environment is to do address mapping. From application layer, the naming IP address needs keeping constant, while from network layer the addressing IP address should reflect the current location and needs changing with moving. The different methods employ different mechanisms and have different places to map naming IP and addressing IP address.

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

3.6 Address Mapping

Most current research work has been focusing on supporting mobility at the network layer. 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 [14]. 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, network solution attempts to still use the home IP in all layers, that is $IP^h_{app} = IP^g_{app}$, $IP^h_{tr} = IP^g_{tr}$, 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 addresses 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 packets to the corresponding mobile system. These packets will be interpreted by any P_{fwd} and will be routed to the correct location. Columbia Mobile IP protocol [14] and Sony Virtual Internet Protocol [73] solutions belong to this category. The MSRs can be viewed as a *virtual network* for all mobile systems. Packets 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 lim-

itations 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.

$$IP^h_{app} = IP^h_{tr} = IP^h_{net} \Leftrightarrow IP^g_{add} \tag{3.1}$$

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 packets, 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.

Our two-step solution uses different address mapping. First, the IP^h_{app} and the DNS name will be kept constant to satisfy the first two invariant conditions. Second, the usage 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.

$$IP^h_{app} \Leftrightarrow IP^g_{tr} = IP^g_{net} = IP^g_{add} \tag{3.2}$$

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*.

Chapter 4

Implementation of Portable IP Communications

4.1 Introduction

In addition to the first step of our two-step mobile TCP solution, portable IP can work independently and is quite useful to nomadic users. In most cases, the nomadic users can be satisfied by portable communication services. Moreover, the infrastructure of mobile communications which usually based on radio or wireless technologies can support mobile communication services in limited areas. Systems in limited service areas may have problems with unacceptably high bit error rates and usually do not provide sufficient bandwidth for large volume data communications — the radio spectrum is a very limited resource and available radio bands in densely-populated areas are typically quite narrow. Conversely, wireless LANs with high data rate and low BERs tend to have rather limited service areas. Therefore, the practical and reliable communications service for nomadic users is tethered connections to local Access Points, which can be either LAN or dial-up connections. Given the physical communication connections are discontinuous in hop-by-hop pattern, users can not obtain true communication services while moving. Portable communication support at the application layer to all network applications is acceptable.

The main purpose of our portable IP is to provide an Internet-wide portable communication support to normal TCP/IP systems. Concretely, portable IP system will enable a mobile host to tap into the Internet locally where it is, and once this connection is established to enable other hosts to locate this mobile host and communicate

with it via TCP/IP protocols without any help of additional protocols. Given the objective of portable IP, the link layer is not further discussed here. We assume that mobile hosts always obtains internet access through compatible interface — the link layer is portable.

4.2 System Functions

From the viewpoint of users, the portable IP system must support:

- allocation of an appropriate IP address for a mobile host when it is established at foreign network;
- provision of facilities for other hosts to find the IP address being used by the mobile host at each moment;
- automatic reconfiguration of mobile hosts for integration of a mobile system into the local internet environment.

From the internal view of naming structure, the first one is to dynamically allocate an IP^g_{add} for a visiting mobile host in certain internet, and the second is to do mapping and the third one requires that this name locating and mapping be done in a automatic way.

In foreign network, the mobile host will use locally-assigned IP address in all TCP/IP protocol layers, which means that network and transport layer will use IP^g_{add}, and application layer needs to do necessary name mappings for network application programs, as given in Form 3.2. In portable IP system, the IP_{app} is constant and equal to mobile host's home IP address,, and transport and network layers use local IP address, or IP_{add}, To simplify the discussion, we replace IP_{app} with IP^h and IP_{tr}, IP_{net} and IP_{add} with IP^g. And, the DNS name of a portable system is denoted as N^h, because it will be kept same wherever it moves. If a mobile host gets assigned a temporary DNS name, it is denoted as N^g.

There are six kinds of name/address mappings:

- M_1, home-DNS name to IP^h, $N^h \Rightarrow IP^h$;

- M_2, home-DNS name to IP^g_{add}, $N^h \Rightarrow IP^g$;
- M_3, IP^h to home-DNS, $IP^h \Rightarrow N^h$;
- M_4, IP^h to IP^g_{add}, $IP^h \Rightarrow IP^g$;
- M_5, IP^g_{add} to home-DNS name, $IP^g \Rightarrow N^g$;
- M_6, foreign DNS name to IP^g, $N^g \Rightarrow IP^g$.
- M_7, foreign DNS name to home DNS name, $N^g \Rightarrow N^h$.

These mappings are conducted by the DNS server process running at mobile host's home network and its current foreign network. The former is responsible for the first four mapping and the latter for the last three. Each mapping has its importance in the application environments.

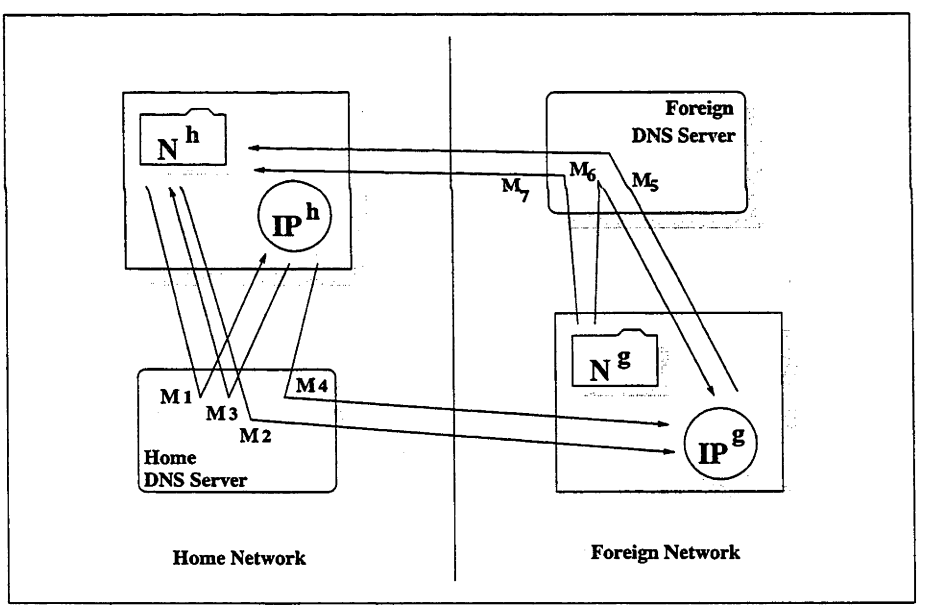


Figure 4.1: Mappings of DNS Name and IP Address

The Figure 4.1 shows all the mappings. In most cases, the literal DNS names are used by human users and most application programs to name and locate TCP/IP systems, to which a conversation is to be set up. For example, almost all web objects are located by their URL with DNS names as server names. In network programming, using DNS name is always a good idea. Because the network server programs can be

moved from server to server or from one network to another within a administrative domain, the IP address of a server system, on which a server program is running, changes with such moving. However, the network administrator can always keep the same DNS name for that server program — mapping the DNS name to current IP address. This is a quite similar case to portable IP communications. The mapping M_1 is for locate the mobile host when it is in home network and M_2 in foreign network.

M_3 and M_5 are for security and spoofing checking in some cases. For example, for securing telnet access, some telnet servers can do spoof checking. It will first get the domain name of the client by using its current IP address to see if it is a legal IP address; and then further use this domain name to get its corresponding IP address to see if the domain name is really mapped to that IP address. If these two addresses are the same, we can partially trust this client. Therefore, correctly mapping mobile host's current IP to its domain name and vice versa is very important to enable this host to pass certain security check. Because these two mappings are conducted by different servers, further operations are required to co-ordinate them. We will talk about this in following sections.

Sometimes the home IP address of a mobile host is hard-coded or manually configured into software. In these cases, some special routines are involved to transfer the IP^h into its current IP^g — the M_5 is needed. Finally, a mobile host may obtain a DNS name in foreign network, which is associated to IP^g. The M_6 is also required to correctly map.

Usually, a host can have more than one DNS name. In portable IP environment, the portable system may have two DNS names, one is home DNS name and the other is given in foreign internet. Therefore, the mapping from foreign DNS name to home DNS name, i.e. M_7, is also needed.

These three system functions are completed by our portable support modules which are installed in different systems. The next section introduces the basic system structure.

4.3 System Architecture

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 kinds of component systems:

- a mobile host with portable IP support, which will move around the Internet and always obtain a data link connection locally and call such system Portable Host (PH) here to emphasise its portability;
- 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.

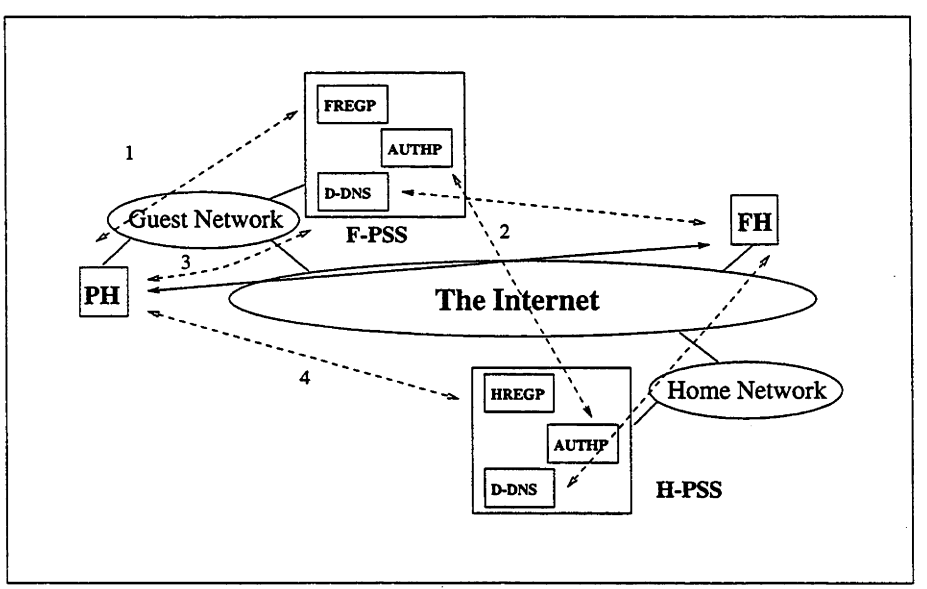


Figure 4.2: Portable-IP System

The core responsibility of the portable IP system is to manage the IP addresses and map IP address and DNS names. The three system functions can be restated as follow:

- allocation of a new IP address for the PH, and all internal addresses of the PH will be set as $IP_{tr} = IP_{net} = IP_{add}$, This enables the PH to obtain a new local IP address and communicate with other systems as a traditional local system;

- 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, in which same protocols are involved and they will be discussed in the following sections.

A typical working procedure for a PH to be dynamically set up is shown in Figure 4.2:

- 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[48]. The identifier of this PH together with its domain name and the IP address, IP^h_{add}, are signed using the PH's secret key;

- Step 2: F-PSS uses AUTHP to authenticate the PH with its H-PSS. The basic means is public-security key. The portable host supplies an encrypted message by using its security key, and the F-PSS fetches its public key from its H-PSS and then decrypt the message to see if it is correct.

- 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_{tr}^{g}, 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_{tr}^{g}.

4.4 Foreign Registration Procedure

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 [43]). 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 packets, and to receive all IP packets1 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.

After completing the registration with F-PSS, the PH will automatically configure its local TCP/IP to behaveour 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`.

4.4.1 Important System Parameters

In essence, each PH has both a permanent home IP address by which it is identified and a temporary local IP address in its current guest network. In addition to local IP address, the PH also needs to set up other system parameters to work in a local internet environment. The more detailed requirements for TCP/IP system can be found in [43, 44], which are internet standards. The following three aspects are most important to portable IP system.

subnet mask

Due to short of IP address space, many internets use subnetting to save IP addresses for single small internet. Therefore, the network address can not be derived from a given local IP address. For example, if a PH is assigned an IP address `203.29.10.200`, the netmask without subnetting is `255.255.255.0`; while it could be anyone ranging from `255.255.255.128` to `255.255.255.252`. The subnet mask is very important for PH to correctly set up routing table and determine the local broadcast address.

For example, the PH `203.29.10.10` needs to send IP packets to the host `203.29.10.10`, as shown in Figure 4.3, there are two possible routings to be chosen by IP internal routing mechanism (see 1.2.2). If a internet occupies whole this class C IP address block, in other words the network address (netid) is `203.29.10.0` and netmask is

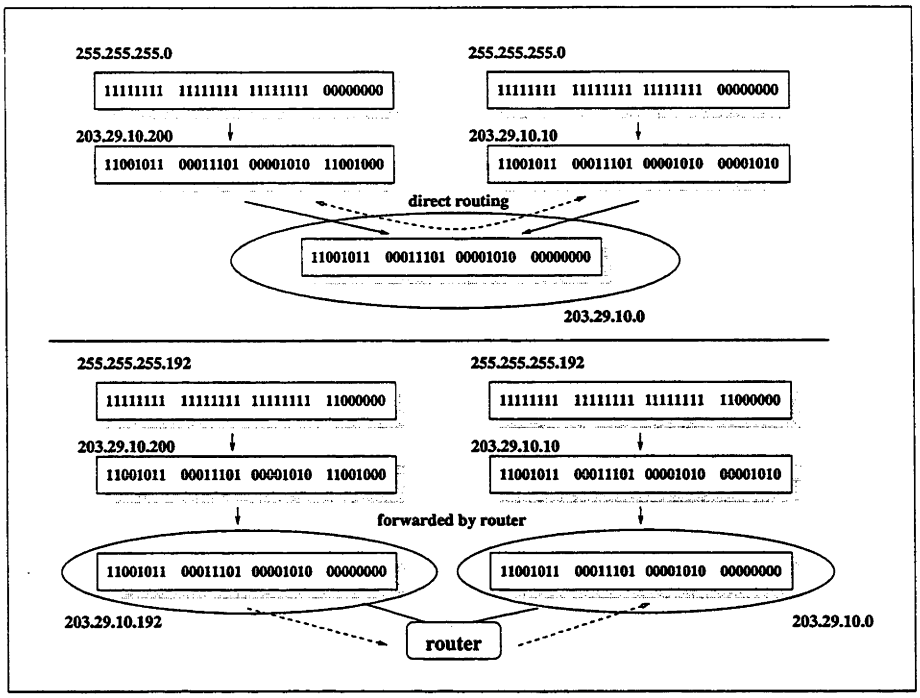


Figure 4.3: Example: Subnet Mask

`255.255.255.0`, the destination host is located in the same network,so that the IP packets can be sent out directly into the local network. While the subnet mask is `255.255.255.192`, this PH is located in the subnet `203.29.10.192`, whereas the destination host resides in a different internet: `203.29.10.0`. Therefore, all IP packets need to be forwarded by some router and there is no direct routing from the source to destination hosts in this case. Obviously, the wrong subnet mask can void correct routing algorithms and cause the PH isolated.

Local routers

At each internet, there is at least one router which connects the local internet to outside. Each machine needs to know all of routers in local internet to send IP packets to correct routers. Usually, in a small internet there is only one router which is responsible for forwarding all outgoing and incoming IP packets, whcih is called *default router*. It is very important to host systems. As we know, most desktop or portable PC systems will not run routing protocols, and the IP address of default router needs setting up manually. It is especially important to mobile hosts, which have to change

the IP address of default router at different locations.

DNS Resolver Setting

Any TCP/IP host system has a DNS *resolver*, which is responsible for querying DNS servers to do name mappings. The resolver program needs to know at least one IP address of DNS name server[1] to send DNS query to obtain required information by application programs. For efficiency, a mobile host should always use local DNS servers.

4.4.2 Foreign Registration Protocol — FREGP

Basically, our FREGP is designed for portable IP host to obtain local IP address, netmask, local router and IP address of DNS servers. As we know, BOOTP, DHCP and RARP could be helpful to assign these parameters, but they can not satisfy our requirements of portable IP.

- RARP is a link layer protocol and used for mapping a physical address to IP address. However, the mapping is static, the RARP server cannot dynamically accept a request from a unknown physical address. Moreover, the RARP request is sent as a link layer broadcast packet, it cannot be forwarded by routers. This requires all networks have to have a RARP server. Moreover, RARP can only assign IP address;

- BOOTP is designed with enhanced functions for booting up diskless workstations. It uses UDP packets to carry the information between client and server, and UDP packets can be forwarded by routers. Moreover, the server can return additional information in Vendor-Specific option field. However, BOOTP server uses the hardware address to decide what information will be sent back to client. Therefore, the system administrator can not pre-configure the server for all possible visiting mobile host. And, the option field for additional information is limited with 64 bytes of length;

[1]The IP addreses of DNS name servers are stored in `resolv.conf` on Unix system, while in Windows95/NT, these IP addresses are put in Registry file.

- DHCP is a widely-used protocol for dynamically configuring a TCP/IP host. Current Windows95/NT operating systems support DHCP. In fact, it is an extension of BOOTP and overcomes some drawbacks of RARP and BOOTP. It extends BOOTP in two aspects. First, DHCP can send back all required parameters in single message to requesting clients, no 64 bytes limit. Second, DHCP can allocate IP address in three ways: manual, static and dynamic. The first two ways are only suitable for local portable hosts, while the dynamic IP allocation is a means to support portable IP systems. However, the lack of security mechanism is a serious issue in portable IP systems.

To satisfy the security and dynamic DNS updating, we further extend DHCP to support security check. The protocol is based on client-server structure, the PH is client while F-PSS is the server. When arriving at a new location, the PH will take following steps to obtain the required system parameters:

- 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 any `FREGOFFER`s have been received, goto Step2. If no `FREGOFFER` is received after certain number of `FREGDISCOVER` have been sent out, abort.

- 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.

State Machines

The client part of FREGP has the exactly same State Machine as that of normal DHCP client (for more details refer to Figure 5 in RFC 1541). The Figure 4.4 shows a simplified client State Machine of FREGP. In the figure, each circle represents a state and the arrow line indicates the transition. A pair of event/action is associated with each transition. The event cause the transition and the action is what the client part does at that moment.

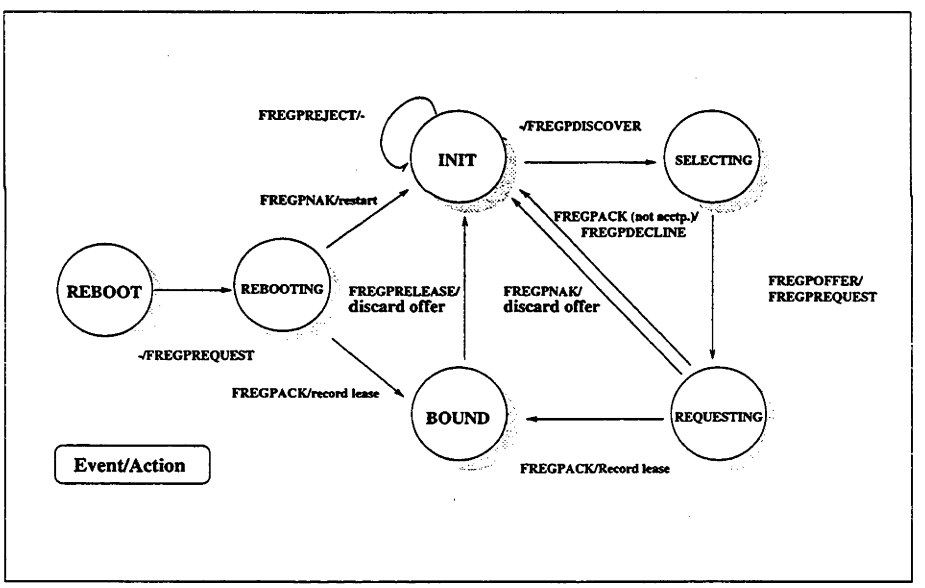


Figure 4.4: State Machine of FREGP Client

First, the client starts from state INIT, it broadcasts FREGDISCOVER message, this is the first step. If the server can not successfully authenticate the client, a FREGPREJECT is sent back then client will stay at INIT. If the authenticaiton is successful or there are other DHCP servers, the client may receive more than one response FREGPOFFER from different F-PSS servers in state SELECTING; then it chooses one of them and

sends a FREGPREQUEST to solicitate the parameters and goes into state REQUESTING. The server may or may not allow what the client requests. If FREGPACK is received and the parameters are acceptable, the client records what it obtains and enters into state BOUND, which means the client has sucessfully obtained the system parameters and started to work. Otherwise, if the client receives a FREGPNAK or some parameters are not acceptable, the client comes back to state INIT and restarts the procedure. There are another two states worth mentioning, the REBOOT is a state in which PH system has successfully registered with F-PSS and jsut rebooted at same interent. What the PH or client needs to do is that to send out a request FREGPREQUEST to that F-PSS to make sure that the previously allocated parameters are still available. If the server acknowledges the request, client can continue to use them and enters into state BOUND; otherwise it needs restart the registration procedure. However, all the parameters are only available within certain time period. Before they expire, the PH needs to send again FREGPREQUEST to renew. Before leaving the current network, the PH is supposed to release the IP address, IP^g_{tr} by sending a FREGPRELEASE message to F-PSS, which has the same values as given in FREGPREQUEST, in addition to an additional option, *MSGTYPE*, to differenciate from FREGPREQUEST.

The state machine of server — F-PSS is different from DHCP server. The server system will keep individual thread for each client and a new state AUTH is introduced. The simplified SM of F-PSS server for a client thread is given in Figure 4.5.

In the INIT state, server is waiting for the request from clients. When a FREGPDISCOVER message is coming, the server thread for this client enters into state AUTH. In this state, server sends request to client's H-PSS to inquire its security identifier to authenticate the client. If the authentication is successful, server sends back FREGPOFFER message with necessary system parameters to client and goes into state OFFERING. Server will response the client a FREGPREJECT if the authentication fails. In state OFFERING, the server may receive a FREGPREQUEST to request thees system parameters, and server sends back FREGPACK to confirm the allocation and enters into state BOUND. Of course, if client rejects the system parameters, server will receive a FREGPDECLINE that brings server back to state INIT.

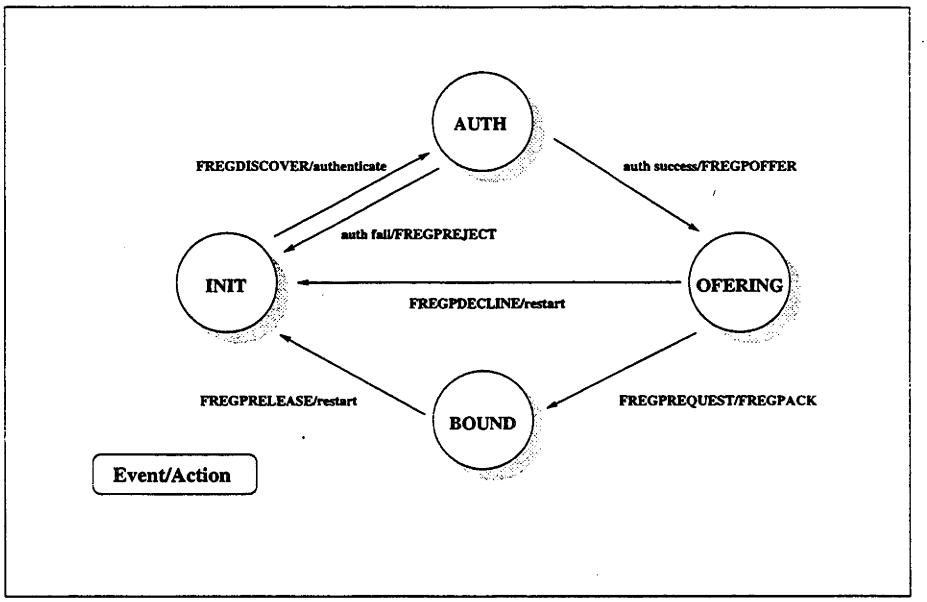


Figure 4.5: State Machine of FREGP Client

PDU Formats and Compatability

As mentioned, the FREGP is an extension to DHCP and makes use of the same PDU formats of standard DHCP. There are four messages in FREGP, i.e. `FREGPDISCOVER`, `FREGPREQUEST`, `FREGPDECLINE`, `FREGPRELEASE`; and four server messages as well, `FREGPOFFER`, `FREGPACK`, `FREGPNAK`, `FREGPREJECT`. The basic frame is shown in Figure 4.6.

1 byte	1 byte	1 byte	1 byte
op	htype	hlen	hops
xid (4)			
ciaddr (4)			
yiaddr (4)			
siaddr (4)			
giaddr (4)			
chaddr (16)			
sname (64)			
file (128)			
options (312)			

Figure 4.6: DHCP Message Format

The one byte OP field indicates the type of this message in BOOTP protocol, because the mesage format is borrowed from BOOTP and the OP field is a BOOTP field. There are only two types of BOOTP messages: client messages are associated with BOOTREQUEST and server messages use BOOTREPLY. The DHCP messages are differetiated by a special option, called *DHCP message type*. In our FREGP all messages except the FREGPREJECT use the original types of corresponding DHCP messages. Due to no corresponding message in DHCP, FREGPREJECT has been asssigned a new value. Table 4.1 lists all these codes. The other fields in PDU formats have same meanings as in DHCP and BOOTP protocol.

FREGP Msg	DHCP Msg Type	OP
FREGPDISCOVER	DHCPDISCOVER	BOOTREQUEST
FREGPREQUEST	DHCPREQUEST	BOOTREQUEST
FREGPDECLINE	DHCPDECLINE	BOOTREQUEST
FREGPRELEASE	DHCPRELEASE	BOOTREQUEST
FREGPOFFER	DHCPOFFER	BOOTREPLY
FREGPACK	DHCPACK	BOOTREPLY
FREGPNAK	DHCPNAK	BOOTREPLY
FREGPREJECT	-	BOOTREPLY

Table 4.1: OP code and Option of DHCP Message Type

In FREGP there are two important messages or PDU's, FREGPDISCOVER and FREGPOFFER. The former will carry additional authentication information to server, and the latter is the response from server to offer the required parameters. We are going to have a look at these two messages now. In the FREGPDISCIOVER message, the PH fills in all the fields given in Table 4.2. 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 4.7. In this option, the domain name in the home network, the

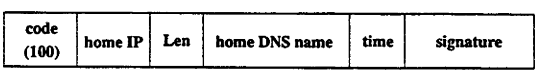


Figure 4.7: Hostid Option

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. In addition, there is a signature used

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

Table 4.2: Fields and Values in FREGPDISCOVER

to avoid bogus request, which is the encrypted MD5 result.

After receiving a FREGPDISCOVER, F-PSS will abstract the domain name and the home IP address of the PH from the option field, and then use them to query PH's H-PSS for its public key via authentication protocol. The public key is then 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 FREGPOFFER. Its fields and values are listed in Table 4.3.

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

Table 4.3: Fields and Values in FREGPOFFER

Note that all the fields specified in the option will be encrypted. It is worth noting that all the options defined in [47] are legal. FREGPREQUEST is used to confirm the acceptance of the offered parameters with F-PSS. Additional parameters can be required in the option field in FREGPREQUEST. FREGPACK from the server is to en-

sure that the registration is committed. FREGPNAK declines the request form the PH. These two messages have the similar field values as given in FREGPDISCOVER and FREGPOFFER.

4.4.3 Authentication Procedure

When a network is ready to accept PH's from the "outside world", it is very important to avoid damage or disruption from malicious or incompetent "guests". The first step should be taken with respect to network security is to identify the PH, which applies for temporary connection to the network, then decide whether it is granted to perform certain classes of communications according to clearly stated rules. FREGP is able to collect the identifier information from PH's, and the authentication procedure will perform the check.

What the FREGP collects from a PH is all in the hostid option (Figure 4.7). The time field aims to avoid replay of the same message. In PH system, 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.

There are two steps for F-PSS to check the signiture. First, it uses the same MD5 function to get a digest; then F-PSS makes use of the same public encrypt function but

with the PH's public key to get the original MD5 digest. If theese two MD5 digests are the same, that means the PH is what it claims to be.

Nevertheless, the another key issue here is how to obtain the public key of PH. In our prototype system, the DNS servers are employed to be the trusted servers for distributing public keys. As we know, although the hierarchical DNS server structure is quite loose, it is not easy to set up a bogus DNS server to dispatch the fake public key.

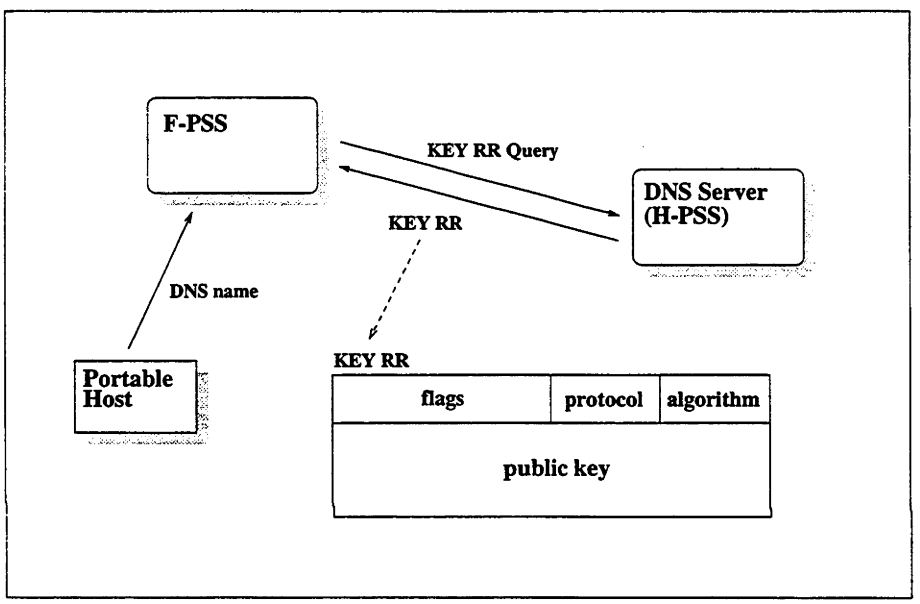


Figure 4.8: DNS KEY RR

Fortunately, in RFC 2065, a new type of Resource Record is desfined — KEY RR [52], which is used to associate a public key with a Domain Name. Therefore, the AUTHP can be simplified as a DNS request and response of KEY RR. Based on the given domain name the F-PSS sends a query to the DNS server for the public key; then the DNS server which delegates the domain will response the query with a valid public key. As shown in Figure 4.8, the RR of KEY has four fields:

flags indicates variety of features of the key. For our authenticaiton purpose, we only consider the bit 0 an 1, if these two bits are all zero, that means no key; otherwise a valid key is included;

protocol indicates the protocol for which this key is intended; we choose the value of

198 as for our portable IP system[2];

algorithm is always 1 for MD5/RSA.

It is worhy noting that the PH's H-PSS is the DNS server as well, which is responsible for the additional DNS mapping functions.

4.5 Home Registration Procedure

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[3] and SSL protocol to provide 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 4.4. The sructure of PR is as the same as the structure

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 4.4: Fields and Values in HREGREQUEST

[2]The values between 1 and 191 are under control of IANA, and values from 192 to 254 are for experiment.

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

of the options defined in [47]. After checking the signature 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 4.5. The *rid* protects the server from replay-attack and

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 4.5: Fields and Values in HREGACK

the *signature* avoids bogus messages to cause any updates.

4.6 DNS Services For Portable IP System

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 as constant identifier. 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 correctly complete the six mappings, M_1 to M_7.

In our portable-IP system, the two PSS systems, H-PSS and F-PSS, act as the DNS sevrers. The DNS system is based on a client-server structure. All mappings are performed in three steps:

- any TCP/IP system may have a set of resolver functions and interact with DNS server systems. The resolver function can send a query to a DNS server for certain information of the given identified system;
- DNS server will consultate its database files or to other DNS servers for answering the queries;

- the responsible DNS server sends back the result.

The client starts a mapping to inquire about what it needs, and the server needs to maintain a updated database to correctly response the client, hence, the query and database is two key issues here.

4.6.1 DNS Query and Database

In DNS, all information is associated with a DNS name or domain name. When a client needs some information, it sends a query to a server with the given domain name and query type, or Resource Record (RR) type. Then server responses the query by sending back an answer. The format of DNS mesage is shown in Figure 4.9.

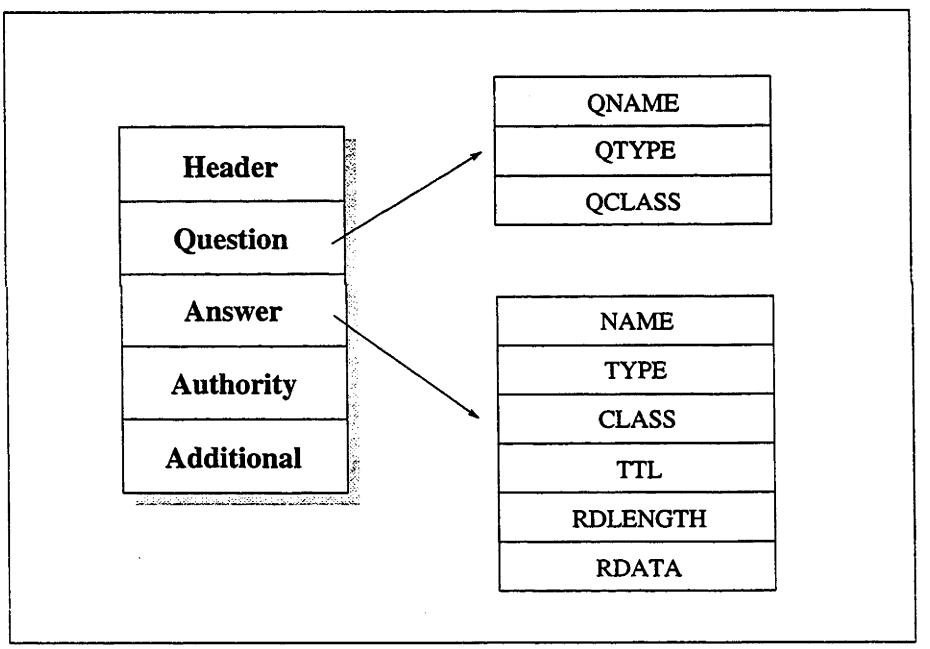


Figure 4.9: DNS Message Format

The field "Question" is filled by client or resolver program. There can be more than one records in this field. Each record can be further devided into three fields. `QNAME` is a domain name of the given object. In fact, there are two kinds of domain name. The most common-seen is the literial DNS name, such as `www.microsoft.com`. The other type is reverse IP domain name. As we know, DNS can map an given IP address to its domain name. However, the DNS structure limits that an IP address can not be a

QNAME. Hence, there is a mechanism to construct a DNS name for a given IP address, the details can be found in [1].

QCLASS specifies the class of query. In the Internet context, it is IN for internet protocols. QTYPE indicates what kind of query or what information is requested. There are more than twenty types in current DNS standard, three of them are related to our mappings:

type *A* the client requrest for the IP address of a given DNS name;

type *PTR* the client asks for the domain name of a system specidied by its IP address;

type *CNAME* given a DNS name, the client inquires the canonical name[4] of that system.

$M_{1,2,6}$ belong to type *A* query, and $M_{3,5}$ are type *PTR* type. In fact the M_7 is *CNAME* query. The only missing is M_4, from IP address to IP address. The standard DNS does not support it. Here we define a new type, called IPIP, its type code is assigned number 50, because this number has not yet been occupied by any internet RFC's.

In response, the "Answer" field is filled by DNS server program. The NAME, TYPE and CLASS are same as that in question field. The RDATA, or so-called Resource Data is the result of the query. The RDLENGTH indicates the length of RDATA. Note that TTL field indicates the valid time for this answer. Usually, server program will cache a result from another server, and uses the cached information to answer the consequent queries for the same domain name and class/type. The TTL affects the time for caching.

On server side, all the information is stored in DNS database, which can be a DBMS or just plain text files. The current popular DNS implementation – BIND uses latter format. In the database file, our new record has similar representation to PTR record as follow:

```
<owner> <class> <ttl>  IPIP  <address>
```

[4] A TCP/IP system can have more than one DNS names, and one of them is chosen as canonical name, which is also the result of reverse mapping, or PTR query.

For example, a PH has been assigned a home IP address IP^h: `150.203.126.233` and its current local IP address IP^g is `203.63.100.192`, then in database file the RR is:

```
150.203.126.233  IN  IPIP   203.63.126.233
```

The TTL value can be omitted and default TTL is used in DNS "Answer" message. However, the value of TTL in a answer DNS message is especially important in portable IP environment.

As we know, the normal TCP/IP systems rarely change IP addresses and domain names, therefore, any answer can be cached for a relatively long time to reduce the DNS traffic, which is usually set to more than one days. However, the portable hosts are expected to change locations or AP quite often. A PH even can visit several internets in one day. In fact, the DNS queries and answers related to PH's current IP address imply the location information of PHs. Therefore, caching of such information is a suttle issue [5]. To simplify our prototype system, we use TTL of zero to respond any query of portable host's current IP address or local DNS name, when this PH is away from home network. The value of zero means no caching for this answer. In section 6.1.2.1 of RFC 1123, it is clearly stated that all DNS name servers and resolvers must properly handle RR's with a zero TTL. Such RR's will be returned to client programs and will not be cached. Therefore, client programs will not get a obsolete RR's related to PH's from cache.

4.6.2 Dynamic Update of Database

In our portable-IP system, the two PSS system act as the DNS sevrers, which are responsible for distribution of PH's location information by correctly responding DNS queries. In the seven mappings related to a PH, some mappings are dynamic and others are static. The dynamic mapping means that the same DNS query can have different answers at different times due to the PH's moving from place to place and

[5] Mitchell P. Tasman submitted a PhD thesis at Computer Science Department, University of Wisconsin in 1994, which was entilted "Protocols and Caching Strategies in Support Internetwork Mobility" and discussed the caching issue.

change of location identifier (both local DNS name and IP address). In Table 4.6 we summarise all the mappings. As we discussed in last section, all dynamic mappings will have assigned zero TTL to avoid caching. The key issue here is that the DNS databases have to be updated dynamically. Two methods are avaliable.

Server	Mapping	Query (RR)	Dynamic	TTL
H-PSS	M_1 or M_2	A	Y	0
	M_3	PTR	N	-
	M_4	IPIP	Y	0
	-	RR	N	-
F-PSS	M_5	PTR	Y	0
	M_6	A	N	-
	M_7	CNAME	Y	0

Table 4.6: DNS Mappings For Portable Host

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 databases 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 [53, 54]. We use the new BIND version 8.1.1 in our prototype system.

To support dynamic updating, the new DNS message, UPDATE, is newly defined. The fields are listed in Table 4.7. 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 *);
```

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 4.7: Fields and Values in HREGACK

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.

4.7 Implementation Issues Of Portable IP System

4.7.1 Additional Functions of H-PSS

When a PH is away from its home network, its home IP address becomes unreachable. It means that other systems can not connect to the PH via this IP address. However, if the IP address is hard-coded in an application program, this program will wait for a long time, which is about several minutes for a TCP/IP system gives up[6]. This long timeout is highly unexpected. To avoid it, there are two solutions, first, H-PSS sends back a message to inform of the unreachability of this IP address; second, the H-PSS acta as a proxy server fro certain services on behalf of the PH's which are homed by this H-PSS.

To support either solution, the H-PSS needs to take some additional ARP[59] actions. There are two points. First, after a particular PH moves out home, the H-PSS needs to fresh the ARP cache of the IP address-hardware address mapping on all TCP/IP systems at local internet, because the hardware address of the PH is not available. The second is to be a proxy ARP for the PH. If any local system, especially local routers, inquires about the hardware address of given PH's home IP address, the H-PSS will response it with its own hardware address, hence it can act as a proxy of certain services to communicate with other systems.

Figure 4.10 schematically shows the solutions to reduce the waiting time. The H-PSS can know whether or not a PH is in its home internet, if the PH registers its new location with its H-PSS. Then, the first action taken by the H-PSS is to broadcast an ARP response with its own hardware address, *HA*, and the home IP address of that PH, say, `203.29.10.201`. This causes all local systems refresh ARP cache with such new pair, including the router. Assuming a host on other internet starts to initialise a TCP connection with the PH via its home IP address. This request packet is fowarded to the local router at PH's home internet. Given the newly cached ARP entry, the router delivers the request locally to H-PSS's hardware address. In the first solution, the H-PSS simple sends back an ICMP `host unreachable` message back

[6]The example given in Section 21.2 of [69] shows that the TCP gives up after about 9 minutes.

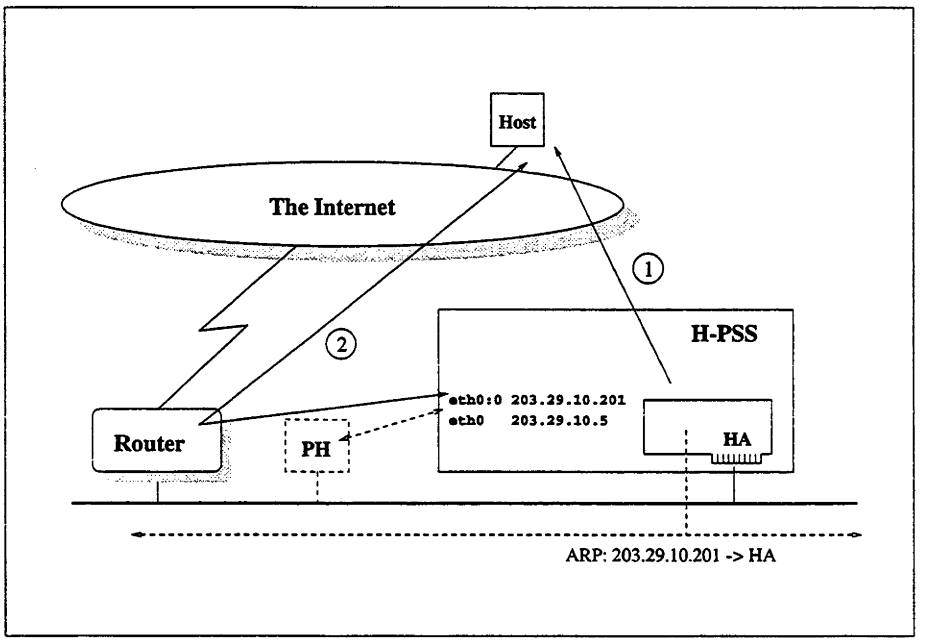


Figure 4.10: Example: Proxy ARP

to the host. This host will immediately abort the connection and return a error message to corresponding application. The second method is to act as a proxy server on this services. In fact, H-PSS can selectively respond the request based on the destination port number. The first thing for H-PSS to do is to create a IP alias interface for the PH's IP address on local network interface, then run the proxy programs on the top of this virtual interface on behalf of the PH. For exapmple, in Figure 4.10, the H-PSS has got an Ethernet interface with IP address `203.29.10.5`, and this interface is defined as device `eth0` in the Unix system. To support proxy, a new logical interface, say, eth0:0 is created and associated with the PH's IP address `203.29.10.201`. Then communications can be set up via regular TCP/IP protocols.

Here, the system administrator is responsible for configuring the H-PSS to act as a proxy for each MH that is registered with it. There are two things to be defined: when to act as proxy for MHs and what services to support. We think that when a PH is in disconnection or it is away from home network, certain services, such as file sharing, bulk data transfer services should be supported by the H-PSS on behalf of MHs, that can avoid the services being interrupted and large number of data being transfered over long distance network.

4.7.2 Enhancements to Network Library and Socket API

As we know, in some cases, the IP addresses of remote systems are hard-coded into the program or predetermined by configuration files. In portable IP environment, this kind of programs may fail to connect to a PH which is away from its home network, either an ICMP error message comes back or the H-PSS does not support the service as a proxy. Hence, the only solution is to require the application programs to correctly map PH's home IP address to its current local IP address. Although H-PSS supports the new RR IPIP query, there is no existing mechanism to use the new type of query. There are two ways to solve this problem.

New Function Call in Network Library

A new function call is added into standard network service library to support the mapping from home IP to current IP. The application program has to be coded to use the new function to get the correct IP address along with `gethostbyname()` and `gethostbyaddr()`, which is put into resolver library and defined as:

```
struct hostent  *gethostbyhaddr(char *addr,
                    int len, int type);
```

All the parameters and the definition of return values of this function call are the same as function call `gethostbyaddr`. The *addr* in fact points to a structure of `in_addr`, in which application program provides the home IP address of this host; the second parameter *len* is the length of the structure of `in_addr`; and the *type* must be `AF_INET` only for internet IP address. The PH's current IP address is returned in structure `hostent`:

```
struct hostent {
  char  *h_name;
  char  **h_aliases;
  int   h_addrtype;
  int   h_length;
  char  **h_addr_list;
```

```
}
```

As we can see, the stucture has an element called `*h_addr_list`, and the addresses associated with this host are given in `*h_addr_list[0]`, `h_addr_list[1]` and so on. In our DNS system, the home IP address is always in the first place and the current IP address in the second. If the PH is in its home network, or this function call is for a fixed host, only the host IP address is returned. Please note that these two pointers, `*h_addr_list[0]` and `*h_addr_list[1]` do point to the structure `in_addr` although they are defined as pointers to `char`.

As other functions with name `gethostbyXXX`, `gethostbyhaddr()` will look into the local database file first to find the required IP address. If no information is found locally, this function then calls the functions in resolver library as other related functions do. Primarily, `res_query()` is the most important function to carry out the DNS name solving. The prototype of this resolver function is as follows:

```
res_query(const char *name, int class, int type,
          u_char *answer, int anslen);
```

`Name` is the domain name for the corresponding IP address. For example, IP address `150.203.126.185` has domain name `185.126.203.150.IN-ADDR.ARPA.`. The `class` must be `IN`(1) for the Internet, and `type` is `IPIP`(50) for our new type of query. The query result will be pointed by `*answer`. The caller is responsible for checking to see if it is a valid answer. This function is also used in following section.

Enhanced Socket Code

Obviously, all existing programs, which use IP address directly to make a connection, do not use this function call to covert a home IP address to its current IP address. In this case, we can enhance the socket layer to play this trick for portable IP system. In fact, there are four socket API system calls by which an application indicates the partner host:

- `connect(int sockid, struct sockaddr *servaddr, int addrlen)`
- `accept(sockfd, struct sockaddr *peer, int *addrlen)`

- `sendto(int sockid, char *buff, int nbytes, int flags, struct sockaddr *to, int addrlen)`
- `sendto(int sockid, char *buff, int nbytes, int flags, struct sockaddr *from, int addrlen)`

Both TCP and UDP sockets can use these four function calls to make and acept connection, and send or receive data. The `connect` system call makes an association between two sockets and the address of remote system will be stored in a *socket* structure locally. The `accept` is dedicated for TCP server, which waits for a incoming TCP connection request. For any given socket, the IP address of remote system needs to be set only once, say, by `connect` or `accept`. In some cases, a UDP session can be set up without first calling `connect` given that UDP is connectionless protocol. Therefore, the remote IP address is required by all consequent `sendto` or `recvfrom` calls. In all these system calls, the IP address of peer system is given in the structure `sockaddr` by application programs. For normal applications, they supply only the permanent IP address of peer system. Internally an IPIP query is required to obtain current IP address for remote system's DNS server. To avoid sending many IPIP queries over the network, the socket code will cache the mapping for only current session.

Here we present an example implementation of portable enhanced socket code for CSRG's[7] 4.4BSD-Lite distribution. First, we put a cache buffer into structure `socket` which is associated with every open socket. This cache buffer is actually a list of mapping of home IP address and current IP address. For TCP socket, only one entry is needed, because TCP is a connection-oriented protocol and at any time only one remote system is connected by one socket. Whereas UDP socket can send data to one or more remote locations. The revised structure `socket` is as follow:

```
struct socket {
    short   so_type;
    short   so_options;
    short   so_linger;
```

[7]Computer Systems Research Group at the University of California at Berkeley.

```
        short   so_state;
        caddr_t so_pcb;
        ...
        caddr_t so_upcallarg;
        struct ipcache {
            struct in_addr   h_addr;
            struct in_addr   c_addr;
        } **so_addrcache;
        short   so_cachelen;
    }
```

For normal socket, the local and remote IP addreses are stored in a structure, `inpcb`, which is pointed by `so_pcb`. The network I/O codes use these IP addresses to prepare the PDU's. In our portable socket code, these addresses in structure `inpcb` are home IP addresses, whereas the mappings are given in list `so_ipaddrs`. Each pair of IP addresses is stored in `so_addrcache[0]`, `so_addrcache[1]`, and so on. The actual number of entries is indicated by `so_cachelen`.

That four system calls listed above are modified by adding a small piece of code to check the destination's IP address. If the destination IP address belongs to a portable host and there is a corresponding IPIP RR, the mapping will be fetched and the current IP address is used. The psuedo-code is given below:

```
H-IP = so_pcb.inp_faddr;
if (H-IP not in so_addrcache)  {
    so_addrcache[so_cachelen-1].h_addr = H-IP;
    answer = call res_query(); /* Send a IPIP query for C-IP */
    if (valid answer)
        so_addrcache[so_cachelen-1].c_addr = C-IP;
    else   /* timeout or invalid answer */
        so_addrcache[so_cachelen-1].c_addr = H-IP;
    so_cachelen++;
}
```

Note that the element `inp_faddr` in structure `so_pcb` is the foreign IP address.

The salient benifit of this method is that the current IP address is transparently obtained by socket code if the peer system is a portable host. However, it requires that the socket code is patched up for this built-in function. It is not very difficult to implement it on Unix system, but the network code under Microsoft Windows is not open now.

4.8 Simple Performance Estimations

Now we are going to compare the system performances of our portable IP solution and current internet starndard IMHP. As we know, using basic IMHP protocol between a mobile host and a fixed host may cause triangle routing problem. There might be two extreme cases. The worst case is MH and FH are located in same network which is not MH's home network. Therefore, all traffics from FH to MH will go a round-trip detour back to MH's home network first. Hence, the triangle routing will cause unnecessary traffics from current network to MH's home network. As shown in Figure 4.11, in contrast, the best case could be that the home network or MH's Home Agent is located in the way from FH to MH. There might be no additional traffics caused by triangle routing.

If we analyse the network topology in real environments, we can find in most cases the situation is neither worst and best case. To compare the systerm performance, we choose the network traffic as a measure and two popular applicaions: ftp and rlogin[8] as the applicaiton protocol. For analysis purpose, we assume that our portable system will set up TCP connections for the applicaiton programs and the intermittent portable services will not affect applications.

Rlogin

Rlogin is a network terminal emulation protocol designed for Unix systems. It will not cause large quantity of data flow between client and server system, but large number

[8]It is a simple version of Telnet, which is implemented for Unix system.

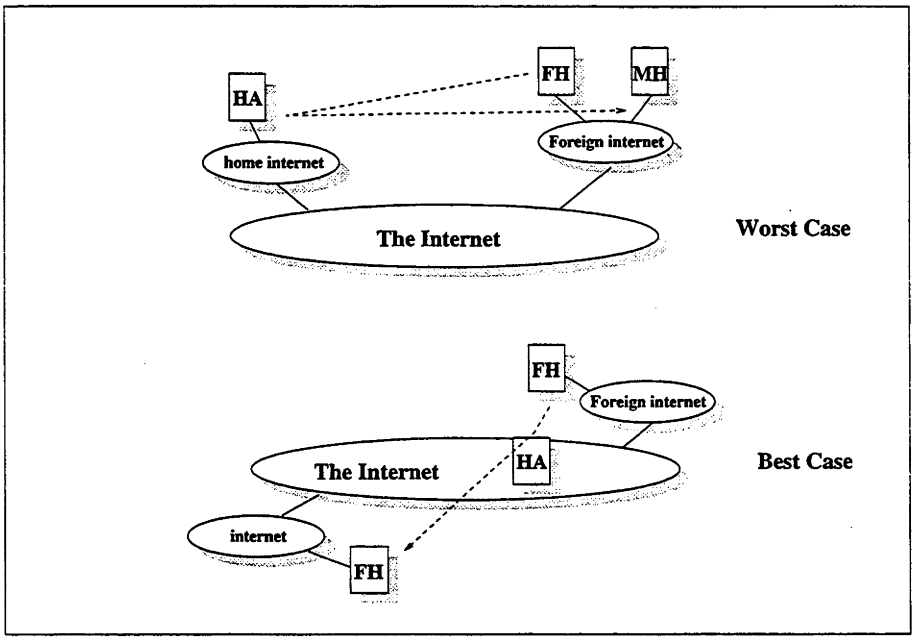


Figure 4.11: Triangle Routing: Best and Worst Case

of small packets. Typical useage of Rlogin is to interact with a shell program at remote system. If we ignore the time for reconnection of mobile TCP connection, our solution will have better performance in term of response time, given corrent routing schemes always try to route packets in direct and fast way. In fact, the time for maintanece of mobile TCP connection is normally quite short comparing with the user's typing speed, thinking time and server system response time. Because only once for each MH's moving, it is not a major contribution compared with time of movement of the mobile system, so it can be absorted by the time of movement. Therefore, here we just analyse the network workload, which includes all packets sent by mobile system, server system and forwarded packets by Home Agent of mobile IP system.

In table 4.8, all the important parameters and their meanings are given.

According to *TCP/IP Illustracted, Volume 1*, α is about 20, and each character typed by user will cause 3 short packets without Nagle algorithm enabled in TCP, one from MH, an echo with ack back from FH and then an ack from MH. The size of these IP packets are 41, 41 and 40. Given the meaning of α, the average size of command response L_{rsp} is $L_{cmd} \cdot \alpha$. Table 4.9 gives the actual sizes of different packets and their descriptions. Here we also assume that mobile IP protocol employs minimal encap-

Name	Description
R_{cmd}	average command entering rate (commands/second)
L_{cmd}	average length of command (in characters)
R_{moving}	average moving rate of mobile host (/second)
α	the ratio of bytes sent by client and server
L_{rsp}	average size of response (in characters)
N_{link}	header and trailer of link layer protocol

Table 4.8: System Parameters (Rlogin)

sulation between HA and MH, hence additional 12 bytes are added to those encapsulated packets.

Name	Description	Average Length
N_1	length of packet with single character from MH to FH or a packet with a echo character and ack from FH to HA	$N_{link}+41$
N_1'	length of encapsulated packet with length of N_1 from HA to MH	$N_{link}+12+41$
N_2	average length of packet with a response from FH to HA	$N_{link}+40+\alpha \cdot L_{cmd}$
N_2'	average length of encapsulated packet with length of N_2 from HA to MH	$N_{link}+12+40+\alpha \cdot L_{cmd}$
N_3	length of ack for response from MH to FH and normal TCP connection establish and close packets	$N_{link}+40$
N_3'	length of encapsulated TCP connection packets	$N_{link}+12+40$
N_4	length of INI_REQ	$N_{link}+40+64$
N_5	length of INI_RSP	$N_{link}+40+24$
N_6	length of REC_REQ and REC_RSP	$N_{link}+40+56$

Table 4.9: Packet Lengths

For mobile IP protocol, the movement of MH will not incur more network traffic, and the total network workload can be formulated as a function of time period t: $W_{mip}(t) = W_1(t)+W_2(t)+W_3(t)+W_4$, here $W_1(t), W_2(t)$ and $W_3(t)$ are the workload within time period t on links from FH to HA, HA to MH and MH to FH respectively, and W_4 is the workload for making and closing a TCP connection. W_4 is a constant with value of $5N_3+2N_3'$, there are three packets for *threeway handshake* and the other two for closing.

In the time period t, the total number of commands sent by user is $R_{cmd} \cdot t$, it is also the number of responses. The total number of characters in commands sent from MH to server in individual packets is $R_{cmd} \cdot t \cdot L_{cmd}$, ans same number of packets are sent as well as response packets from server. Therefore, we get:

$$W_1(t) = R_{cmd} \cdot t \cdot L_{cmd} \cdot N_1 + R_{cmd} \cdot t \cdot N_2;$$
$$W_2(t) = R_{cmd} \cdot t \cdot L_{cmd} \cdot N_1' + R_{cmd} \cdot t \cdot N_2';$$
$$W_3(t) = R_{cmd} \cdot t \cdot L_{cmd} \cdot N_1 + R_{cmd} \cdot t \cdot N_3.$$

For portable solution, all data traffic is directly routed between FH and MH by normal routing mechanism. Additional traffic is introduced by maintaining TCP connections before and after the MH moves to a new location. Therefore, the total network workload of our solution is: $Wpip(t) = W_5(t) + W_6(t)$, here, W_5 is the normal data trsnfer workload and W_6 is the overhead of maintanence of TCP connection, including normal TCP connection setup and tear-down as well as mobile connection overhead. We have got:

$$W_5(t) = R_{cmd} \cdot t \cdot L_{cmd} \cdot (2N_1 + N_3) + R_{cmd} \cdot t \cdot (N_2 + N_3)$$
$$W_6(t) = (5N_3 + N_4 + N_5) + R_{moving} \cdot t \cdot (4N_3 + 2N_6)$$

Note that an active movement will cause more traffic than passive movement, hence we assume mobile system always does active movement. In Table 4.10, we fix some system parameters for evaluation the performance of both systems. Then, we can get the result formulas for network workload as following:

$$W_{mip}(t) = a_1 \cdot t + c_1$$
$$W_{pip}(t) = a_2 \cdot t + a_3 \cdot R_{moving} \cdot t + c_2$$

We give the values of a_1, a_2, a_3, c_1 and c_2 in Table 4.11.

Given these direct results, we can easily get that the network workload percentage saved by our solution is about 18% in all four cases. We also can get the critical point of R_{moving}. If the actual moving rate is less than the critical point, our solution has better performance. Whereas faster moving rate will cause worse performance in our solution. The value is given in Table 4.12.

Name	Heavy work	Easy work
R_{cmd}	0.25	0.10
L_{cmd}	10	5
α	20	20
L_{rsp}	200	100
N_{link}	10/18	10/18

Table 4.10: Settings of System Parameters (Rlogin)

N_{link}	Heavy Work		Easy Work	
	10	18	10	18
a_1	553.0	629.0	118.7	133.1
a_2	455.0	519.0	96.0	109.6
a_3	248.0	560.0	412.0	460.0
c_1	374.0	430.0	374.0	430.0
c_2	448.0	504.0	448.0	504.0

Table 4.11: Values of Parameters

Obviously, at the worst case, mobile system is required to stay at a network averagely longer than 20 seconds, our solution will gain better performance. Figure 4.12 shows this conclusion. The negative saving means that our solution has worse performance gaining. Note that the figure shows quite wide range of MH's movement rate, from a stationary situation to a very fast case ($R_{moving} = 0.3$ means that MH stays at one location only for 3.33 seconds). Moreover, in general situations, we believe, a MH will stay in a network for several minutes or longer, say 1 minute at least, then the R_{moving} is less than 0.0167 and we can obtain about 12% performance improvement at least in four cases above.

Link Layer overload(N_{link})	Heavy Work		Easy Work	
	10	18	10	18
R_{moving}	0.40	0.20	0.055	0.051
stay time	2.5	5.0	18.18	19.61

Table 4.12: Critical Point

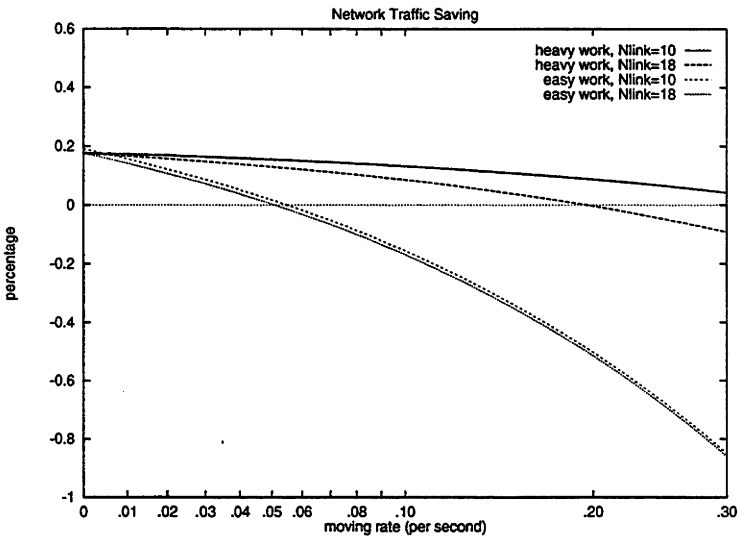


Figure 4.12: Network Traffic Saving (Rlogin)

ftp

Ftp protocol is generally for all kinds of systems, which transfers files between two systems and the network traffic is largely dependent on the size of files transfered. For sake of simplicity, we only analyse the data channel of ftp. In term of network workload, triangle routing of mobile IP solution will cause much more network traffic. Whereas our solution uses normal direct route, and the overhead of mobile TCP connection is only about two hundred bytes for each movement. We still assume the MH is client system and FH is the server in a ftp session.

Name	Description
MTU	maximum transmission unit
S_{file}	size of file being transfered
β	the ratio of acks to data packets
R_{moving}	average moving rate of MH
T	average data transfer throughput

Table 4.13: System Parameters (Ftp)

In Table 4.13 we give some parameters for analysis of ftp. T is a process-to-process data throughput in normal condition. To transfer a file with size of S_{file}, transfer time is S_{file}/T, and $N_m = (R_{moving} \cdot S_{file})/T$. The packet lengths are listed in Table 4.14, N_3 to

N_6 have same definitions as in Table 4.9. The total number of data segments sent from server to client is about $N_{data} = \lceil S_{size}/MTU \rceil$. Normally, one ack packet is responsible for 1.5 data packets, then we fix β as 2/3.

Name	Description	Average Length
N_1	length of packet with maximum data payload	$N_{link} + MTU$
N_1'	length of encapsulated N_1 packet	$N_{link} + 12 + MTU$
N_2	length of ack packet from MH to FH	$N_{link} + 40$

Table 4.14: Packet Lengths

Now, we can get the network workloads for both mobile IP and our solution as following:

$$W_{mip} = N_{data} \cdot (N_1 + N_1' + \beta N_2) + 5N_3 + 2N_3';$$
$$W_{pip} = N_{data} \cdot (N_1 + \beta N_2) + 5N_3 + N_4 + N_5 + bN_m \cdot (4N_3 + 2N_6);$$

We envision three typical network environments, a short distance interconneected LAN, a long distance WAN environment and a serial line connection. In LAN, the MTU is large, and it provides high throughput. Whereas WAN and serial line environment has a small MTU and low bandwidth. We fix some system parameters which are given in Table 4.15. We assume that two routes have same MTU, for simplicity of calculation, we further assume the encapsulated packets do not exceed the maximum packet limits as well.

Name	LAN (ethernet)	WAN(X.25)	Serial line (PPP)
MTU	1500	576	296
N_{link}	18	10	10
T	500Kbps	50Kbps	1.5Kbps
β	2/3	2/3	2/3

Table 4.15: Settings of System Parameters (Ftp)

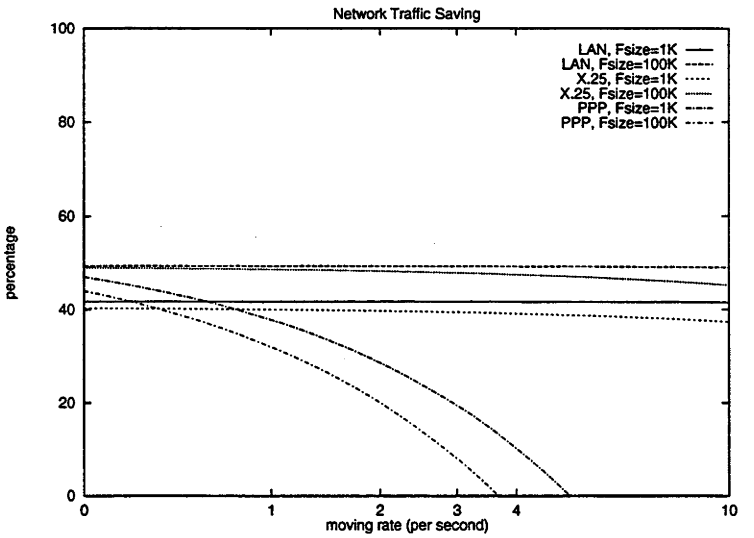


Figure 4.13: Network Traffic Saving (Ftp)

4.9 Conclusion

In this chapter, we introduced the first step of our mobile solution: portable-IP. The main advantage of our solution is that it can significantly reduce the cost for forwarding IP packets 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. We will attempt to conduct a performance study on the two costs mentioned above.

Chapter 5

Implementation of Mobile TCP Communications

5.1 Introduction

In this chapter, we describe a mobile TCP socket which is the second step of our two step approach to support mobility. The first step of our approach is to support portability as reported in last chapter. In our mobile TCP socket, a mobile mapping is introduced which maps IP addresses and maps TCP associations to TCP connections. The mobile mapping can be implemented in the socket and on top of TCP/IP layers. Our approach achieves high compatibility with current TCP/IP protocols, due to the fact that both of the two steps in our approach do not impose any changes on TCP/IP and the underneath layers. At the same time, our approach 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. The concept of mobile mapping and its implementation details are given in this chapter.

As we discussed before, truly ubiquitous mobile computing requires that programs on mobile computers be able to process continuously, similar to in a distributed environment. However, the current TCP/IP protocols are designed for hosts which have an Internet-wide location-sensitive IP address and are not able to move from one internet to another while network applications are running. In TCP/IP systems, the hardness of mobile communication stems from two-tie use of IP address. Most existing mobile IP solutions[14, 15, 73, 36, 37, 35, 9] support mobility in the network layer from which the problem stems. Therefore, the current IPv4[56] needs to be en-

hanced by some special routing mechanisms which have to be supported by mobile hosts and/or routing facilities. This will be the major barrier for quick, wide and easy deployment of the existing mobile IP solutions.

Our previous work on portable IP communication employs the existing protocols and extends their services to support portable communication[42]. The Portable-IP solution employs mainly on the existing protocols, such as DHCP and DNS, and doesn't need any enhancement for TCP/IP and its underneath layers. Here we further extend our portable IP system to provide a mobile solution for TCP/IP systems.

The mobile solution is aimed at providing mobile TCP socket which can be implemented in the socket layer and on top of the TCP/IP layers. This approach can achieve high compatibility with 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 or the transport layers to support mobility. Finally, we don't require all networks, where mobile hosts may visit, 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.

As claimed in [14], the network layer seems to be a reasonable place to provide mobility. We focus on the network layer solutions. Although these network layer solutions mentioned above have achieved a degree of success, all of them face one or two of the problems, namely, the performance and the backward compatibility[29].

As for the backward compatibility, all of the existing mobile solutions are not fully compatible with existing Internet protocols and infrastructure at the network layer where they intended to solve the mobility. The Columbia and IBM MHP limit mobile hosts to visit such internetwork where mobile support stations exist. Sony MHP needs more special routers in Internet forwarding IP datagrams. Although IMHP claims that no special requirements will be imposed on the internetworks where mobile hosts will visit, it assumes that a mobile host can communicate with its corresponding host using its home IP address. Unfortunately, most of the current routings devices do not route local IP datagrams with a foreign IP address as source address. Some networks perform source address filtering for security reasons. Therefore, a foreign address can

not get through the routers or firewall[4]. Hence, IMHP still needs to impose some special support on the visiting internetwork.

The performance issues rely on the cost to propagate location information, $\mathcal{P}$, and the cost to forward datagrams, C. Obviously, more forwarding points and wider propagation of location information will achieve suboptimal routing, i.e. lower C, while this will result in less backward compatibility and high $\mathcal{P}$. On the other hand, if we employ fewer forward points, the *mathcalP* will be very low but at the same time *mathcalC* will be higher. In such a case, all traffic from fixed hosts to mobile hosts has to be forwarded by the home networks of the mobile hosts. This indirect and triangle routing has significant impact on performance. The detailed performance analysis will be discussed later.

Our portable IP support system does not require special support from any foreign networks and routers except the standard protocols, such as DHCP, BOOTP etc. The main advantages of portable system are as follows.

- **Direct routing:** It can save communication costs compared to indirect and triangle routing. Since an portable system does not require special routing mechanisms, a portable system and its correspondent system communicate through existing routing infrastructure. Unlike IMHP which requires IP-IP encapsulation, our portable system consumes less network bandwidth and CPU time at both home support systems and either special routers at foreign network or at mobile hosts.

- **Few requirements are imposed:** Only standard DHCP or BOOTP services are required at foreign networks. Special protocols are implemented only on the portable system itself and its home portable support system. Then it is easy to deploy portable systems in the Internet and give large roaming area for mobile hosts in Internet.

- **Ease to be implemented:** Because a portable system does not modify network and transport layer protocols, not only can it talk to normal TCP/IP system, but it satisfies the most likely compatibility with any internet environment without any portable support at a foreign network. The users of our portable system can

access server systems as normal using telnet, ftp, email, etc.

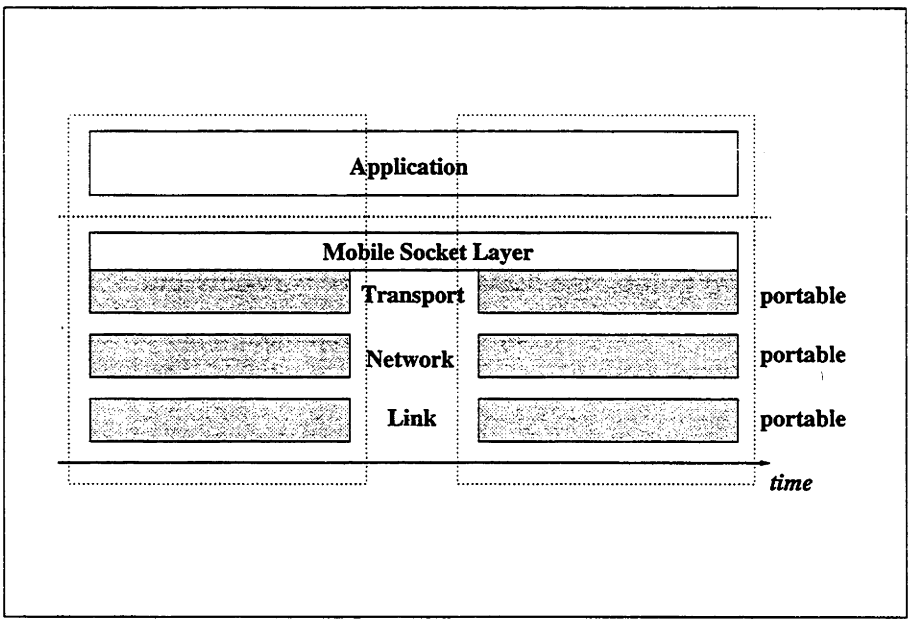


Figure 5.1: Two-step Solution

For consistency, we use IP^g to represent the local IP address of mobile system, which can be equal to home IP address IP^h.

5.2 Mobile TCP Solution

Our portable support system plays an important role in our mobile TCP solution. In our mobile TCP solution, we further exploit the advantages of our portable system, and enhance it with mobile TCP communications service. Figure 5.1 portraits a layered structure of our mobile system. To provide application layer with mobile TCP services, the mobility in our mobile solution is provided in two separate steps.

- **(Step-1) Portability:** Up to transport layer, the portability is supported by unmodified TCP/IP protocols with additional portable support module. The Link layer in TCP/IP protocol suite is supposed to be portable since various types of Ethernet, Token Ring or serial line offer portability in the sense that users can easily plug into and unplug from the network connector at the physical and link layer. Our portable system is adopted as given in the previous section.

- **(Step-2) Continuity:** In our approach, TCP/IP protocols themselves are intact which guarantees that our mobile system can communicate with all TCP/IP systems. The continuity is provided at the socket API layer. The socket API is not a protocol layer in TCP/IP protocol stack, but it is an access interface to transport services in all BSD-clone TCP/IP implementations as well as in Windows environment (Winsock 1 and 2). We call our enhanced socket layer Mobile Socket Layer(MSL).

With our mobile TCP solution, mobility is equal to portability plus continuity which bridges the vacuum period of communication services provided by Portable-IP communications. Our mobile solution is twofold. First, the mobile TCP support is at socket layer, all underlying protocols need not to be modified to support mobile TCP communications. Secondly, the deployment of our mobile system is much easier than existing MHP solutions in terms of employing current routing infrastructure. The implementation details will be discussed in the later sections.

To clarify the system functions of mobile TCP Socket, envisage that an application process is communicating with its partner through TCP. This application, denoted as App_a is on a mobile host MH, which is currently located at network Net_a and has IP address IP_a, whereas its partner App_b is on a fixed host FH of IP address IP_b. As we know TCP/IP protocols use the IP address to identify an communication party, the TCP association is bound to this pair of IP address (IP_a, IP_b). Now we assume that MH_a needs to move to another network Net'_a. After MH left Net_a, its original IP address IP_a is no longer valid, and underlying IP layer can not use it to communicate with other systems. In other words, the TCP association is broken. After arriving at Net'_a, the MH can obtain a new IP address IP'_a by Portable-IP system. The original TCP association can not be recovered due to the change of MH's IP address, although the FH keeps the same IP address.

Therefore, the continuity provided by MSL has to support two main functions, namely, the continuous TCP association and consistent I/O support. First, the active TCP associations between a pair of TCP sockets must be kept open and uninterrupted by change of IP address. Portable-IP can only provide portable operations, such as

obtaining IP address, are transparent to applications, while MSL needs to hide the change of IP address from application as well to provide transparent continuous communications services. Second, during the moving period, the underlying communication links are not available, and data sent to mobile hosts may be lost and local I/O requests can fail abnormally. This will change the I/O semantics and affect the applications, which will take the moving action as a serious error condition instead of taking it as a acceptable working condition. Hence, MSL needs to remedy this situation to keep a harmonious working environment for applications all the time.

5.3 Preliminaries

5.3.1 TCP Association and TCP Connection

The BSD socket API is a set of system calls which provides applications with accesses to communication services. The core concept is *socket*, as first introduced in TCP protocol [58], which was defined as an endpoint of TCP communication and identified by an IP address and a TCP port number. All communications occur between a pair of such endpoints. The original definition is only for TCP protocol. In BSD Unix socket API, *socket* is further extended to a protocol-independent communication endpoint at transport layer, that means the *socket* can be created on top of different network/transport protocols which carry on the real data transfer. Given there are several popular protocol suites (e.g. OSI, XNS, SNA and Internet TCP/IP), a *socket* needs clearly to be bound to one protocol stack, which is called *Domain* of *socket*. Each *domain* can have more than one candidate transport or network layer protocols to be further assigned to a open socket. As shown in Figure 5.2, in the Internet domain, three types of sockets are available including datagram (UDP), data stream (TCP) and raw socket.

In Internet domain, a socket has three important components and can be presented as a triple:

```
(protocol, IP-address, port-number)
```

The *protocol* indicates which transport protocol is used to convert data from and to this

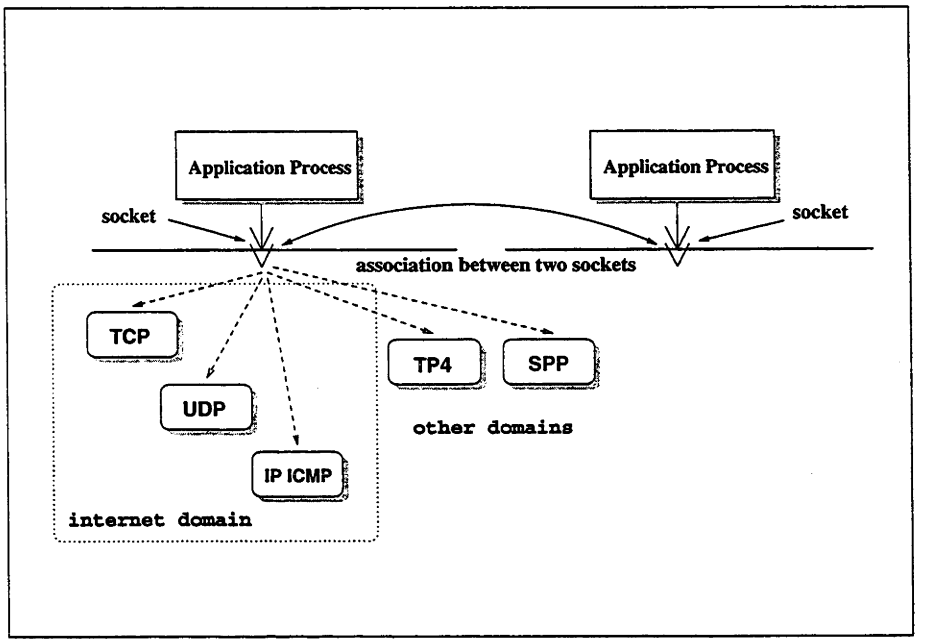


Figure 5.2: Socket and Protocols

socket: either TCP or UDP; the IP address identifies the system on which the socket is located and the port number is used to differentiate among local sockets.

The data transfer usually takes place between a pair of sockets. For connectionless transport protocol, the association is not required. Because a single endpoint of connectionless transport protocol can communicate with more than one remote endpoints, and the destination is given in individual output requests. Of course, an association can be virtually set up by one end for connectionless transport protocol, and then no destination address is requested for consequent send requests. To connection-oriented protocol, association is an essence. Such binding between two sockets is call an association which is defined as a 5-tuple[68]:

```
(protocol, IP-address-1, port-number-1,
           IP-address-2, port-number-2)
```

Importantly, an association is Internet wide unique, that means an association between two sockets can be set up once at a time, no more than one associations can exist between any two sockets simultaneously.

In this research work, we restrict ourselves to TCP given our objective is to provide

mobile TCP communications services. The association of TCP is a logical binding between two sockets, a TCP connection is required to set up for every associations between TCP sockets.

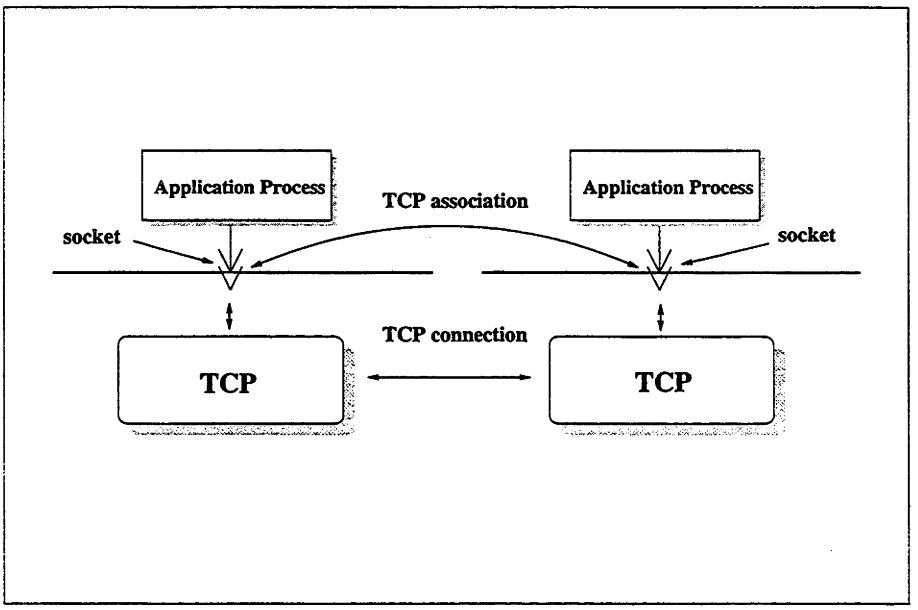


Figure 5.3: TCP Association and Connection

As shown in Figure 5.3, the relationship between TCP association and connection is one-to-one. Given the association is unique internet-wide, the connection is also unique in the same scope. Internally, a TCP connection is identified by the IP addresses and port numbers of both ends. It can be represented as:

```
(IP-address-1, port-number-1, IP-address-2, port-number-2)
```

The socket will keep all information regarding both association and connection. To application processes, the socket API provides a unique and simple I/O interface to access network services, whereas socket code itself is responsible for mapping association to corresponding connection. Socket code will interact with TCP protocol entity to set up and tear down TCP connections and perform I/O operations.

5.3.2 Socket System Calls

A TCP association will be established between an active open socket and a passive one. The application which actively open a socket and starts a TCP connection is

called client, whereas the application passively open a socket and waits for incoming connection requests is called server.

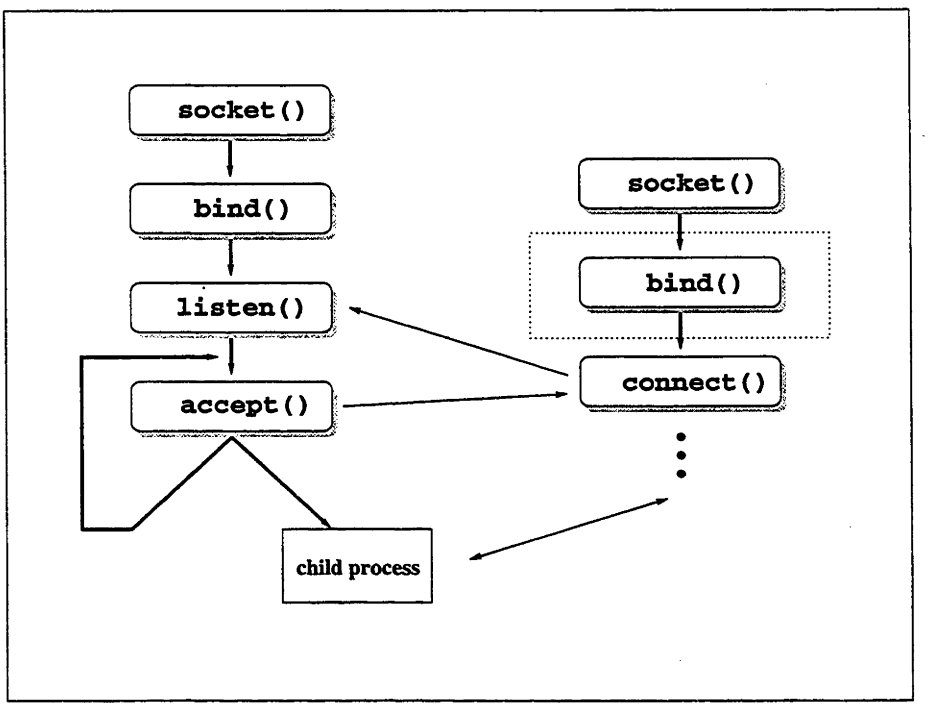


Figure 5.4: Sequence of System Calls

A typical sequence of socket system calls is shown in Figure 5.4. The server first creates a socket by calling `socket()` system call to reserve internal structure for particular transport protocol which is specified via parameters. Then the application process explicitly call `bind()` to bind the socket to a local address, namely a local IP address and a port number. Now the socket is a named socket with specified local address. The `listen()` is called to inform TCP entity of the opened endpoint for waiting for incoming connection request. This system call only indicates that the application process is willing to accept incoming connection. The actual waiting is executed by calling `accept()`.

All client programs must know that server's local address: server IP address and port number before initiating a connection. The client first creates a local socket and names it by calling `socket()` and `bind()`. The next step for client is to start a connection by calling `connect()`, in which The remote server address is supplied. Then

socket code requests TCP entity to start a three-way handshake procedure to make a TCP connection.

The incoming request is accepted and completed by `accept()`. Usually, the server program will go back to wait for incoming connection and fork out a child process to handle the connection. `Accept()` will return a new socket for the completed connection, This is very important for a server program to establish more connections simultaneously. The server program will always wait for incoming request on the original socket, and use the new sockets to communicate with different partners.

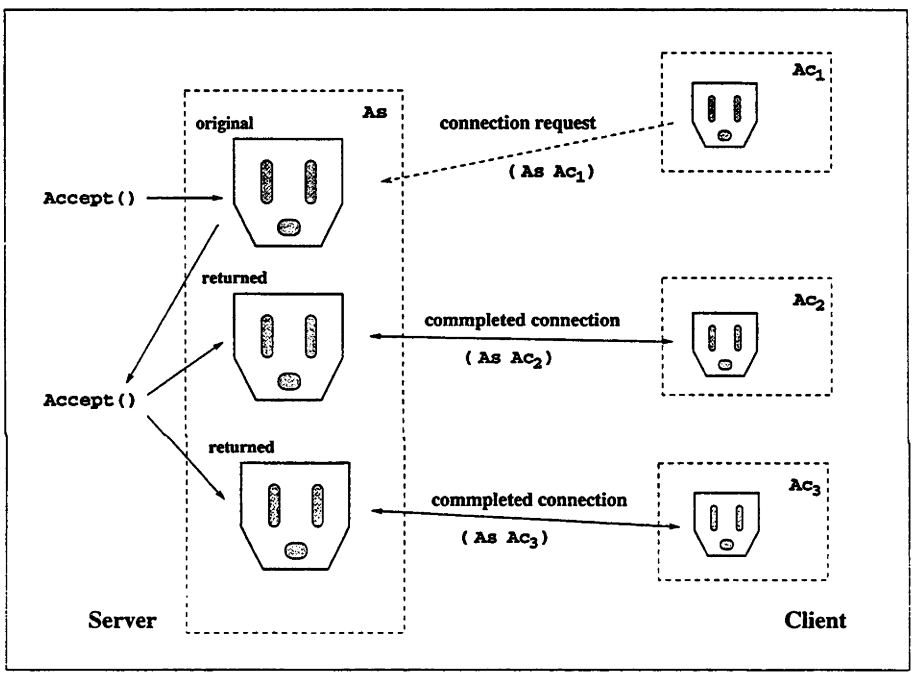


Figure 5.5: Original and New Sockets

Note that at server side the original socket and consequent sockets returned by `accept()` share the same local address, As shown in Figure 5.5, all sockets on the server side have exactly same IP address and port number, denoted as A_s. The differences among them are different partners: the different remote sockets, say, A_{c1}, A_{c2} and A_{c3}. As stated above, the combination of a pair of sockets must be unique Internet-wide. Although more than one sockets can share local address, these sockets can not have same peer sockets. Hence, the TCP association, i.e. the combination, is still unique. In the figure, the associations are (A_s,A_{c1}, (A_s,A_{c2} and (A_s,A_{c3}, respectively.

Not only can server ports share the local IP address, the client programs also can use same local IP address to bind sockets to. This technique is called *port reuse.*

5.3.3 The TCP I/O Semantics

The TCP I/O semantics is mainly affected by its internal buffer and the I/O system calls which manipulate the internal buffers. Using 4.4BSD-Lite as an example[74], there are two groups of socket I/O system calls that manipulate the internal buffer, namely, the *write system calls* and the *read system calls.* They include `write`, `writev`, `sendto`, `sendmsg` and `read`, `readv`, `recvfrom`, `recmsg`, respectively.

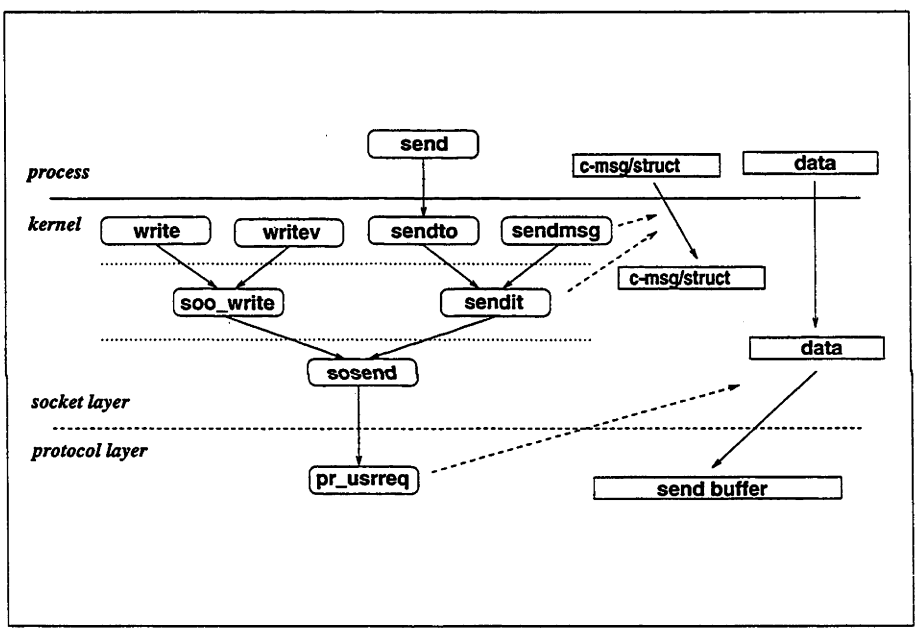


Figure 5.6: Write System calls and Buffering

Figure 5.6 shows both the control and data follows for the *write system calls.* Application processes pass to the socket layer data and control messages including the destination IP address by calling one of these four write system calls. Two socket layer calls `soo_write` and `sendit` keep such requests in an internal data structure. The socket layer `sosend` first tries to obtain enough buffer space to hold all data if possible, and then copy them to the sending buffer by calling the TCP layer subroutine — `pr_usrreq`. Figure 5.7 shows both control and data flows for *read system calls.* The two socket layer calls, `soo_read` and `recvit`, keep user requests in an internal struc-

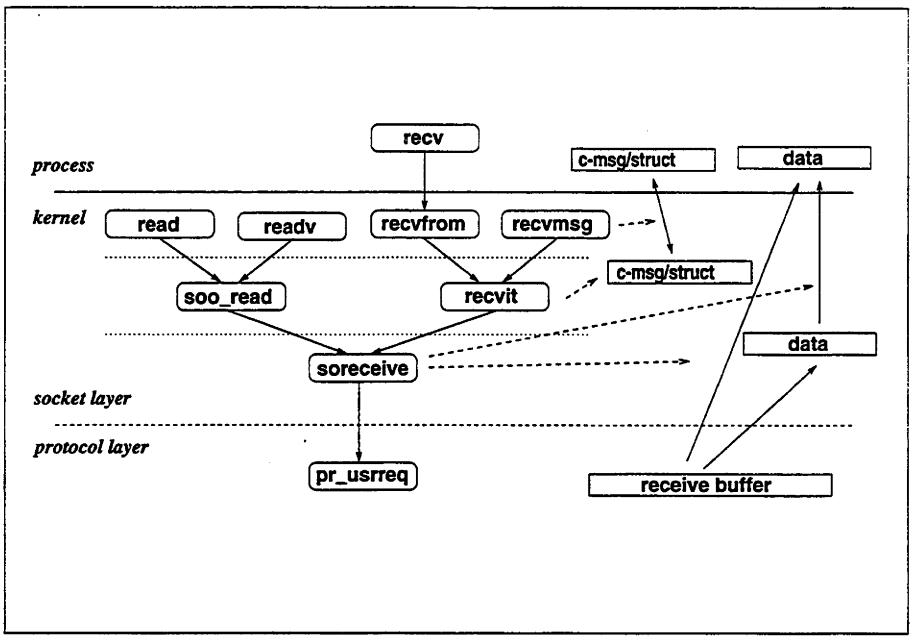


Figure 5.7: Read System calls and Buffering

ture with which a socket layer call `soreceive` performs real data transfer from the TCP receiving buffer to the application process buffer.

As for write calls, locally the data being sent are first copied into the socket layer buffer, and then to the TCP layer's sending buffer. For synchronous I/O socket, write system calls return after data have been copied into the TCP sending buffer. For asynchronous I/O socket, though write calls return immediately, the signal of I/O completion is sent to application processes after data are properly put into the TCP sending buffer. Therefore, there are no any chances of loss of data when transferring data to the TCP sending buffer. However, no error occurring in copying data locally does not ensure that data are successfully transfered to the remote site. Closing socket will cause loss of data in the TCP sending buffer. Usually `close` closes TCP socket. At default, `close` call will return immediately and all data in the TCP sending buffer are sent out. When tearing down the TCP connection, application processes can discard pending data to be sent by set the socket option `SO_LINGER`. Another system call `shutdown` performs similar closing function.

As for read calls, TCP protocol will not discard any data correctly received. If there are no buffers in the receiving buffer, TCP will close its receive window. All data out

of the receiving window are discarded and will not be acknowledged. If processes provide a small read buffer, the socket layer will only transfer up to the size of that small read buffer. The remainder will be supplied for next read. Similarly with the sending buffer, the receiving buffer will be cleared by `close` and `shutdown`.

A socket can be closed in half way — *read half* and *write half*. When closing *write half*, application can further indicate whether the pending data are to be discarded or not.

In summary, TCP socket guarantees that data will be correctly deliver to remote process without duplication and loss in normal data transfer period. Socket I/O system calls will not introduce additional possibility of loss of data. When sockets are closing, the data remaining in the sending and receiving buffers may be discarded by TCP entity.

5.4 The Continuity of TCP is Mobile Mapping: The Concept and Issues

In this section, we first introduce the concept of mobile mapping and discuss the issues on how to implement such mobile mapping.

5.4.1 Mobile Mapping

In conventional TCP/IP host systems, the relationship between a TCP association and a TCP connection is 1:1. When application processes request to open or close an TCP association, in response, the socket layer underneath will request the transport layer to setup or tear down a corresponding TCP connection correspondingly. A pair of association and connection can be considered as to have the same life time.

Our strategy is briefly given as follows in three items.

- Keeping TCP associations unchanged. Since the home IP address, IP^h, is a unique identifier of a mobile host, IP^h is always used to set up TCP associations even when the mobile host is in a guest network. Such TCP association is

independent of the locations of the mobile hosts, or so-called Mobile Association.

- Underlying TCP connections are always established according to the change of local IP addresses, when mobile hosts are in guest networks. The current IP addresses, or IP^g is used to make TCP connections, because Portable-IP system uses IP^g in network and transport layers, that is $IP_{tr} = IP_{net} = IP_{add} = IP^g$. Also, in order to keep use of IP^h in mobile association, the address mapping $IP^h \leftrightarrow IP^g$ is required and it must be transparent to application process.

- Hiding any loss of TCP connections from applications by introducing mobile mapping mechanisms between TCP associations and TCP connections.

Our Portable-IP system can provide the basic services for mobile host systems to obtain local IP address and automatically start network services up to IP layer; as well as provide new DNS services to map IP^h to IP^g. The key role of mobile mapping is to manage the mapping from mobile association to regular TCP connection.

The relationship between TCP association and connection in mobile environment is of most importance here. As the first item claims, MSL is required to provide consistent and uninterrupted mobile TCP associations no matter where the mobile hosts migrate to. While as we know, in traditional fixed-location environment, the TCP association is bound to a single TCP connection, which can be broken due to change of IP address incurred by moving. Therefore, the underlying TCP connection needs to be set up according to the current IP address or location of mobile hosts. The Figure 5.8 schematically depicts the life time of mobile TCP association and TCP connection. When mobile host stays with an Access Point, or within a internet, the association has single underlying TCP connection. When the mobile host is moving, there is no TCP connection can be set up due to that Portable-IP does not support communications in moving environment. When the MH arrives at a new location, as the second item states, the MSL's at both side need to re-establish a new TCP connection for the given association. Therefore, a one-to-many relationship between TCP association and connection is introduced.

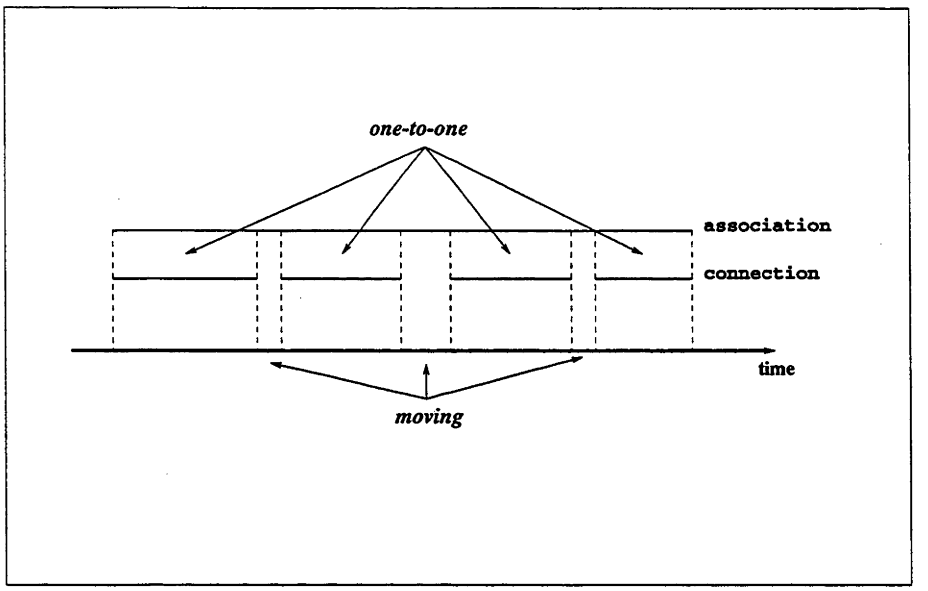


Figure 5.8: Life time of Association and Connection

The third item requires MSL to bridge the vacuum period of communication services during moving. Both sides of an association, or MSLs, need continuously to support socket I/O operations as long as send buffer is not full, and make the TCP associations alive to application processes. When the mobile host arrives at a new location, and a TCP connection is automatically reconnected by two MSLs, the queued I/O requests and new I/O requests are processed through the new TCP connection. In such

The core of MSL resolves the one-to-many relationship between TCP association and connection, and the key point is socket. Figure 5.9 shows the internal mapping scheme between mobile TCP association and its underlying TCP connection(s). At socket level, the application processes can obtain network communication services through standard socket API, that means the processes on mobile or fixed host systems have same TCP communication services and same interface, as well as same I/O semantics. This kind of consistent mobile TCP association is supported by the MSL code between socket and TCP protocol implementation entity. All mobile operations are hidden beneath socket API.

As we discussed above, the data transfer along an TCP association is carried by underlying TCP connection. At this level, for single association more than one TCP

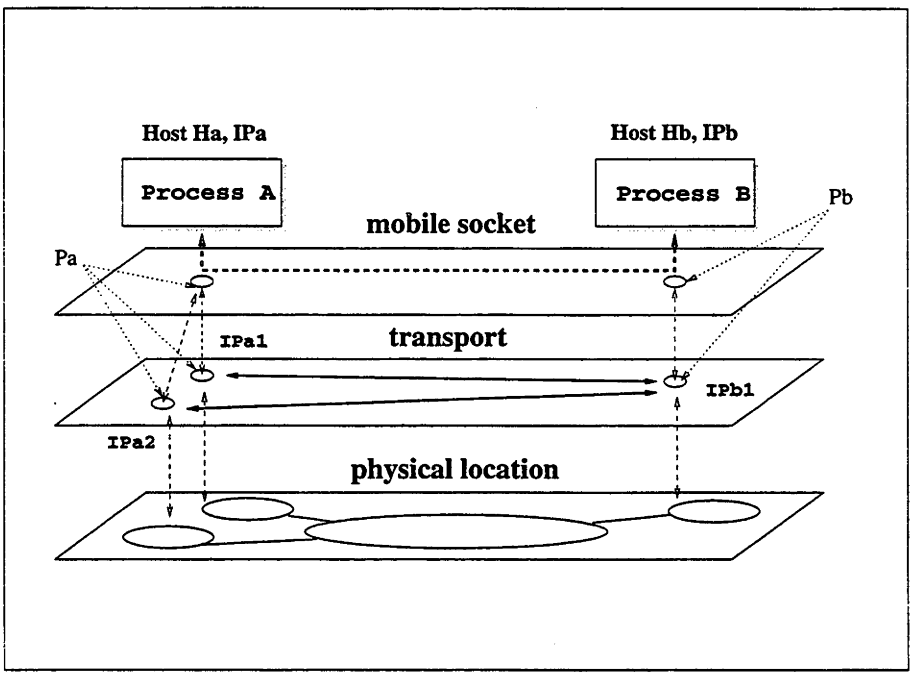


Figure 5.9: Mobile TCP Association and Connection

connections may be set up to support real data transfer between two systems. MSL code is responsible for creating a TCP connection to connect the two endpoints. Note that both the endpoint in socket level and in TCP level are called socket, and the relationship is one-to-one. The MSL will pass data between the endpoint at socket level and the corresponding one at TCP level. Usually, if the underlying TCP connection is broken, the endpoint at TCP level becomes invalid and the association will not exist as well. To avoid this, MSL must keep a virtual map there to continue serve the I/O during moving period, the detail will be introduced in next section.

As shown in Figure 5.9, we assume an application process A on mobile host H_a needs to talk to the application process B at host H_b via TCP protocol. A TCP association can be established between two sockets which are created by H_a and H_b as $(TCP, IP_a^h, P_a, IP_b^h, P_b)$. Where both IP_a^h and IP_b^h are home IP addresses and P_a and P_b are TCP port numbers. As in traditional environment, the underlying TCP connection can be first set up between this two addresses as $(IP_{a1}, P_a, IP_{b1}, P_b)$. Here IP_{b1} is equal to IP_b^h given host B is always at a fixed location. Now we further assume the mobile host geometrically moves into a new location after a while, it is assigned a new local

IP address IP_{a2}. The previous IP address is invalid to host A so that the original TCP connection is no longer valid. The MSL at both hosts are establishing a new TCP connection as soon as host A obtains the new IP address, $(IP_{a2}, P_a, IP_b, P_b)$. Obviously, a new endpoint is created at host A, whose local address is (IP_{a2}, P_a). Again, MSL will map this new endpoint at TCP level to the same socket and use the newly created TCP connection to continue the data transfer for the mobile association. Note that the the new endpoint and previous one have the same port number but IP address.

The mapping from socket to a endpoint at TCP level is what mobile mapping does and MSL code needs achieve. The address mapping given in formula 3.2 (Section 3.6) is implied by creation of new TCP connection at each location.

5.4.2 Issue 1: Distinguish Old Connection From New Connection

Whenever a mobile host changes its location and therefore its current IP address, the opening TCP socket association must be mapped on to the new TCP connection using the new current IP address accordingly. However, implementation of such a mapping and re-mapping an open association to a new connection is challenging. The major contribution to the complicity is the reuse of port numbers. Both server and client systems may reuse a local port number to make different connections with different peer sockets.

The mechanisms of reusing port numbers make it difficult to map a mobile TCP association to a new TCP connection. For example, as in Figure 5.10, a server program is running on a fixed host. The port number P_s is used to connect with peer client processes which can be on mobile systems. Suppose that MH_1 and MH_2 are talking to socket S_1 and S_2 on the fixed host, which have the same local address and share the same port number. The server is also waiting for new incoming requests. Suppose that MH_1 moves away from current network. Then its TCP connection will be forced to terminate, but its corresponding TCP association will be kept alive. However, the server has no way to determine whether a new connection request is either a new request or a reconnection from MH_1, because the incoming request will have a new source IP address.

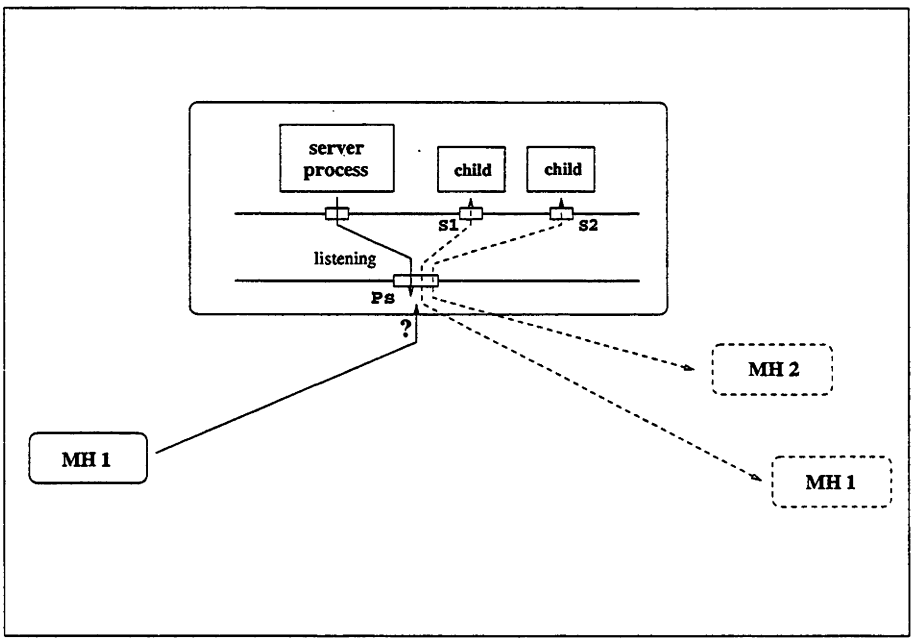


Figure 5.10: Reuse of Port Number

Another potential problem is from the reuse of the temporary IP addresses. For example, a server has a TCP connection with a mobile host, MH_i, using a temporary current IP address IP_{curr}. After a while, MH_i moves away and leaves the server in the state of waiting for reconnection from MH_i. The IP_{curr} may be allocated to another mobile host, MH_j, which may try to set up a connection with exactly the same IP address and port number at server as used by MH_i. In spite of the same IP address and port number, the two mobile hosts shall be treated differently.

5.4.3 Issue 2: Keeping the Mobile TCP Socket I/O Semantics the Same

To keep the same TCP socket I/O semantics in our mobile socket environment is not simple. At first glance, a TCP association is mapped to a TCP connection and the semantics should be maintained by TCP entities which shall remain unchanged in mobile environments. However, a potential problem occurs when closing of connection. In a non-mobile network environment, a TCP association is always mapped to a single TCP connection. The only possible loss of data without further warning is at the end of communication. Whereas a mobile TCP association might be mapped to

many TCP connections in different time periods (referring to Figure 5.8). At all ends of these connections, data may be lost.

There two possible cases. First, a mobile host tries to close TCP connection just before moving. The TCP entities at both ends will try to send out all data kept in the sending buffers before closing. But this process will take time. If a mobile host moves before completion of active/passive close, both sides have no idea whether the data remaining in TCP send buffer have been successfully sent. There are two further possible cases: a) the data have already been sent to the correspondent site and the only loss is the acknowledge which may be lost or has not been sent out; and b) the data have lost on the way toward the remote site. Therefore, the data cannot be simply resent when a new TCP connection is setup. Second, the network link of mobile host gets lost in a non-voluntary way, so mobile hosts have no time to close their current TCP connections and are forced to abort it. For example, a user unplugs the network cable, or inadvertently goes beyond the covered area of current wireless LAN. In such cases, old TCP connections are aborted at both sides which detect loss of network connection or any network changes. Both systems do not know whether data in the sending buffer are correctly received by the remote side.

To mobile TCP association, any kind of loss of data or any suspicious situation might occur in the middle of life cycle. This could change the TCP socket I/O semantics. That is data might be lost when system moves.

5.4.4 Issue 3: Recognition Between Non-mobile and Mobile TCP Services

The MSL-supported systems are expected to communicate with both conventional and peer MSL-supported TCP systems so as to allow all applications to communicate without hassles of dealing with different systems and different services. Two solutions are possible. One is not to change the conventional socket API interface that supports both conventional and mobile TCP services to applications. The other is to find ways to detect whether a request is from a conventional system or a MSL-enabled system. The former request to change the socket API syntax and therefore the semantics. Consequently, existing applications can not be able to access our mobile TCP ser-

vices without recompilations. The latter incurs unnecessary system overhead when applications do not need mobile service.

5.5 Virtual Port

Our solutions to the first two issues mentioned in the previous section are introducing a virtual port which is an additional port between socket and TCP port as shown in Figure 5.11. A virtual port acts as a TCP port from the viewpoint of a socket. When application binds a socket to a TCP port, MSL creates a virtual port, and binds the socket to it accordingly. The mobile TCP association is established between two such virtual ports. All code above the virtual port is the same as used in the current socket layer, and extensive mobile functionality is hidden by a virtual port.

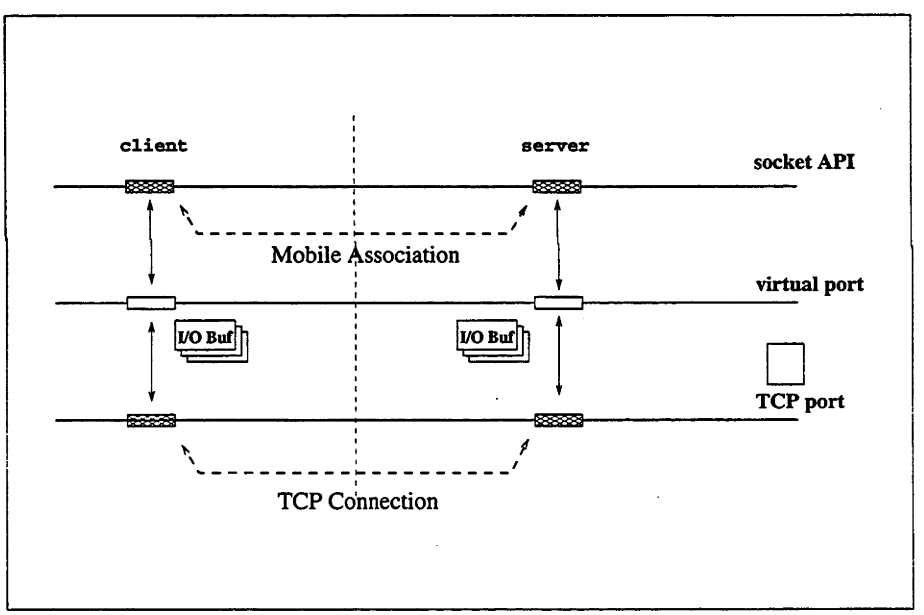


Figure 5.11: Virtual Port

When created, a virtual port is assigned to a home-IP and a local port number. Therefore, what the socket sees is a TCP port with consistent permanent address. MSL maps a virtual port associated with a home-IP and a port number, to a real TCP port associated with current-IP and a port number, dynamically[1]. The TCP connection is

[1]Note that the port number at server side might be changed, we will discuss this later.

set up between two real ports by conventional TCP/IP protocol stacks. Furthermore, like TCP ports in the current TCP systems, a virtual port can be shared by many sockets.

The following information must be managed in MSL system, namely, a six tuple $(A, V_{id}, P, B, B_n, F)$. A is a TCP association, represented as $(TCP, IP_1^h, P_1, IP_2^h, P_2)$.

As mentioned before, it is difficult to distinguish reconnection requests from new connection requests due to port reuse at the server side. Hence, an identifier, V_{id}, is added to identify a unique virtual association from all associations through the same virtual port. V_{id} itself does not need to be unique in Internet wide. The V_{id} is assigned by the server virtual port when a client requests to initial connection. The V_{id} will be passed to the client as an identifier. Later, the clients use the V_{id} as an additional information to reconnect to the server virtual port.

P is a regular TCP connection between two addresses, namely, $(IP_1^g, P_1, IP_2^g, P_2)$ The MSL first maps IP^h used in A to the corresponding local IP addresses IP^g used in P. The port number in general can be different but in our system, they will be kept same in both sides. F is used to indicate the type of local socket, either client or server. B is a sending buffer being managed inside of the MSL system. B_n is a serial number to indicate the serial number of data transfer. It starts from 0 at the beginning of mobile association.

5.5.1 Procedure for Connection

The procedure for connection between two sockets is given in this section. We illustrate the steps in Figure 5.12, assuming that the server system is a fixed host.

- **Step 1:** A server application creates a socket and binds it to a local address, (IP_1^h, P_1). The MSL system at the server site creates a virtual port using the same address, accordingly.

- **Step 2:** The server application passively opens the socket. Correspondingly, the MSL system at the server site opens a passive TCP port, (IP_1^h, P_1), and waits for incoming request. Here IP_1^h is the same as IP_1^g.

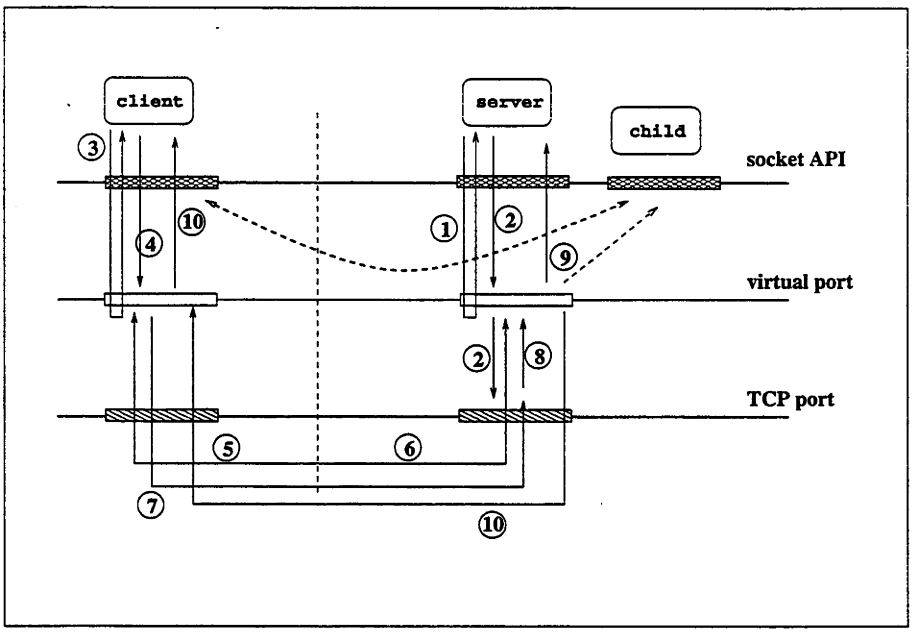


Figure 5.12: A Simple Example

- **Step 3:** A client application completes creating and binding its local socket, (IP_2^h, P_2). The MSL system at the client site creates a virtual port using the same address as server does in Step 1.

- **Step 4:** The client process then initialises a connection procedure to establish a TCP association with the server socket. The address of the server socket given by the client process is a permanent address.

- **Step 5:** The MSL system at the client site maps both local and server permanent addresses to corresponding current TCP socket addresses, and starts a regular TCP three-way handshake to set up a real TCP connection between two virtual ports at both the client and the server sites at first. This procedure is not visible to application layers at both sides.

- **Step 6:** The MSL system at the server site will then accept the TCP connection from (IP_2^h, P_2) to (IP_1^g, P_1) for further exchange of additional MSL control information. Noting that the TCP association between two sockets has yet to exist at this stage.

- **Step 7:** By the real TCP connection, the MSL system at the client site sends its virtual port address including additional information to its peer MSL system at the server site indicating that it is an initial connection request.

- **Step 8:** Based on the information provided by the MSL at the client site, the MSL system at the server site decides whether or not to accept this request according to the parameters passed from the upper layer at the server site.

- **Step 9:** If this request is accepted, a new socket is created at the server site, but not a new virtual port. The MSL system at the server site will assign a V_{id} to this connection and also inform the MSL system at the client side of the acceptance.

- **Step 10-a:** The MSL system at the client side keeps the V_{id} internally and informs the client application of the successful connection.

- **Step 10-b:** If the server application refuses the connection at Step 9, the MSL system at the server site sends a deny information back to the MSL system at the client site. Then client MSL returns a error code to client.

Obviously, the major difference of the connection procedure between the MSL system and the conventional systems is that a real TCP connection is set up before the socket association and before the connection between virtual port is accepted.

Next, we briefly describe how to implement the above steps using the API for the standard socket system calls. At the server side, for Step 1, a `socket()` creates a socket structure and a `bind()` assigns a local permanent address to the socket. For Step 2, the server calls both `listen()` and `accept()` to passively open the socket. The `accept()` will create a virtual port locally, and the MSL maps the virtual port address to a real TCP address. This `accept()` will return when a virtual port is accepted at Step 9. At the client site, for Step 3, the client process calls `socket()` to create its own local socket. The client can call `bind()` to explicitly assign an address to local socket. If the client does not call `bind()`, `connect()` will implicitly assign such an address, at Step 4. To both the server and the client processes, all the steps between Step 5 to Step 8 is transparent. The additional mobile operations are completed

in `accept()` and `connect()` system calls. When these two system calls return successfully, a mobile TCP association is created.

The reconnection procedure will be the similar. The server virtual port is always waiting for incoming request from the underlying TCP port. The MSL system at the client site will send different messages with a V_{id} to the MSL system at the server site. Hence the MSL system at the server site understands whether or not it is a reconnection, and is able to map it to correct destination socket.

The server application itself may be ceased before its child processes and its socket is closed. Normally, there is no alive socket listening on the corresponding TCP port. In other words, all sockets associated with this TCP port are created for child processes and are not in a passive open mode. Any incoming requests to this TCP port will be refused by the TCP layer. However, as for MSL, the processing is different in a certain circumstance. If there is a socket, which has active association with a client process which is not currently connecting to the TCP connection due to its movement, the MSL system at the server site will passive open the underlying TCP port, and wait for reconnection requests.

The MSL systems have to distinguish the close of last TCP connection from all previous ones. At the last connection, the virtual ports and sockets at both sides need to be deleted. Otherwise, the virtual ports must be kept alive. When the client or server applications need to close an association, the MSL systems will explicitly close the TCP connection and will delete such virtual ports and associations.

5.5.2 I/O Semantics

The sending buffer, B, is aimed at two targets: providing continuous I/O services and remaining the TCP I/O semantics unchanged.

As for providing continuous I/O services, the local data buffering scheme is given as follows. If the underlying TCP connection between the two real TCP ports function, the virtual ports at both sides will not buffer any data. All these data will be put into the TCP sending buffer directly. During the movement, there is no real TCP connection, and no I/O requests can be performed. The MSL system is expected to

buffer data in its own sending buffer. These buffered data will be sent out when a new TCP connection is created later on. A virtual port can reject data when its buffer is full. The rejection is not an exceptional case by all means. For instance, in normal socket communications, network congestion can occur. In this case TCP cannot send more data when the sending window is full and the consequent sending requests will fail. The loss of underlying TCP connection looks the same as loss of network bandwidth to application processes in terms of I/O failures.

Regarding TCP I/O semantics, TCP uses both sliding window and acknowledgment mechanism to ensure correct delivery of all data being transfered between two TCP ports over unreliable networks. When a TCP connection is being closed, either side may discard data in its send buffer. Hence, TCP associations are reliable during the data transfer period but there is no guarantee during the closing time. In mobile association environments, there might be more that one TCP connections to convey data between a pair of sockets, and exist no mechanisms to ensure that these TCP connections can be closed in a complete manner.

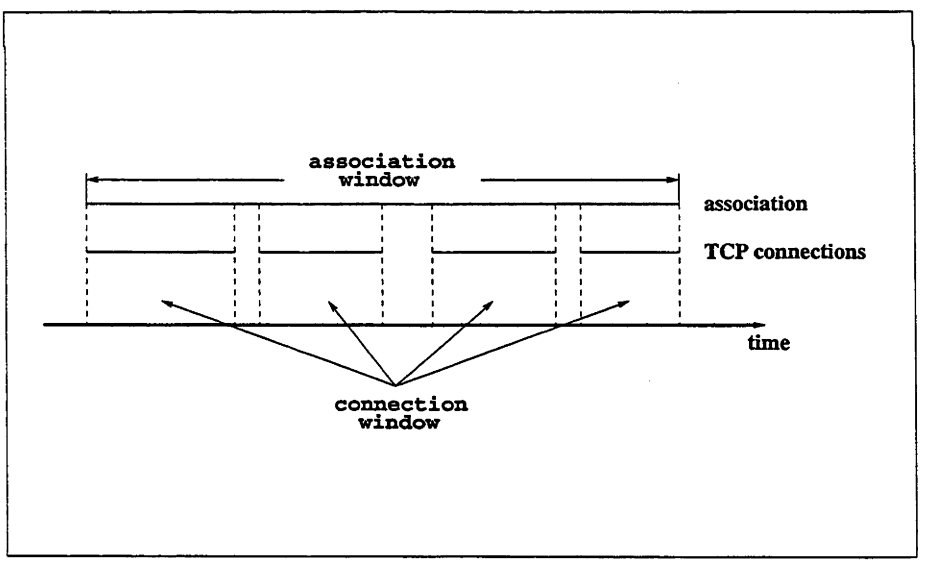


Figure 5.13: Association Window and TCP Window

To keep the I/O semantics unchanged, besides the buffer, B, we introduce a new mechanism, called association window, which spans over entire association lifetime and is independent of TCP window mechanism as shown in Figure 5.13. TCP win-

dows make sure that data can be correctly transfered in single TCP connection, whereas association windows are aimed at the correct data transfer during entire lifetime of an association. Association window gives all data being transfered a sequential number, B_n, like TCP window does. The initial number is always from zero. At each beginning of re-establishing underlying TCP connection, both virtual ports will exchange the sequential numbers in association window to indicate how many bytes in incoming data stream have been correctly received, hence the peer systems can resume transfer from correct starting points. Virtual ports will keep records of the sequential numbers of both last byte sent to the TCP layer and the last byte received, these numbers are updated when each time virtual port sends data to TCP sending buffer or receives data from TCP receiving buffer. Given the loss of data in TCP communications can only occur at the end of each TCP connection, only these data in TCP sending buffer might be lost. Therefore, MSL fetches back the remainder of data in TCP sending buffer into virtual port buffer, B, at the end of a TCP connection. On reconnecting, virtual ports will re-send all or partial portion of data in the buffer, B, according to the requested sequential number B_i sent by remote side. Both B and B_i are recorded in the aforementioned 6-tuple of virtual port.

5.5.3 Virtual Port Protocol

The Virtual Port Protocol(VPP) is a protocol used between a pair of virtual ports. The basic tasks of VPP are demultiplexing and mapping the incoming connection requests to proper associations and resynchronising data transfer after reconnection.

There are four types of Protocol Data Units (PDU), as given in Figure 5.14. After setting up the underlying TCP connection between the pair of virtual ports, VPP PDUs are exchanged before any data transfer of application processes.

First, the MSL at the client site sends a `INI_REQ` with the definition of virtual connection *(local home-IP, local-Port, peer-home-IP, peer-port)*. The MSL at the server site then finds that the incoming request is for a new connection. According to the condition specified in the server application's `accept()`, the MSL system at the server site decides whether or not to accept this request. The result is sent back to the client in

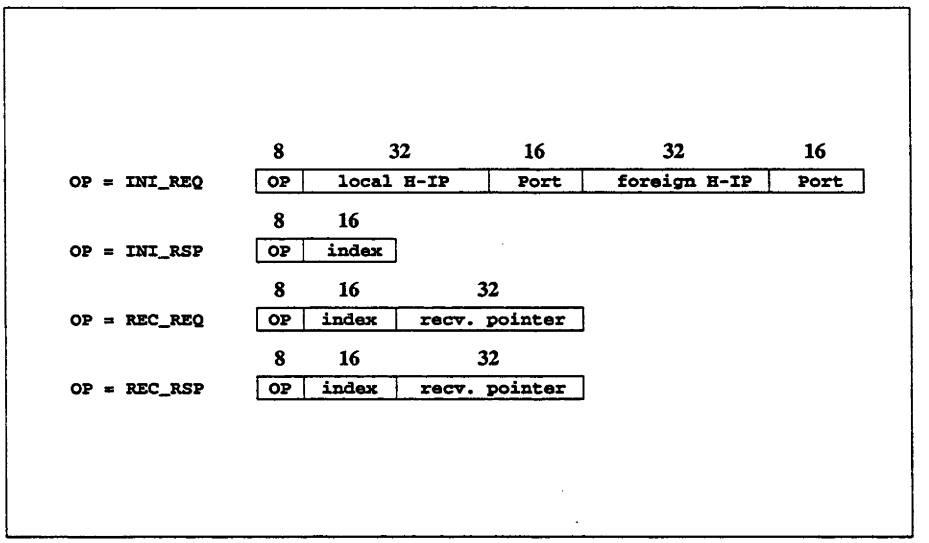


Figure 5.14: Protocol Data Units

INI_RSP PDU. If the connection is refused, the value of V_{id} is assigned to a negative number, and underlying TCP will then be closed by server system immediately as well. If it is accepted, V_{id} will be a positive number as the valid and unique association identifier among all associations of this virtual port at the server site. Afterwards, all incoming data will be directly passed to the socket by virtual port, and not further control information is embedded into the normal outgoing data stream as well.

When a client moves into a different location, the MSL system a the client site uses the V_{id} to reconnect. This request is sent in REC_REQ UDP. And the MSL system at the server site will recognise the reconnection request and find the corresponding association followed by replying a REC_RSP message. REC_RSP indicates that the current underlying TCP connection is successfully mapped to the destination socket. Like INI_REQ, if the MSL system at the server site cannot find a V_{id}, a response with an invalid V_{id} will be sent back. During reconnection period, both sides need to exchange the B_ns to inform the other side of the position from which data transfer should start.

5.5.4 The API of MSL

MSL provides two types of TCP services, namely, regular and mobile services at both server and client sides. As for the API of MSL, there are two solutions. One is to define

a new API for a mobile protocol supported in the internet domain as `IPPROTO_MTCP`. The other is to use the existing internet domains by which MSL must be able to handle both regular and mobile inside the MSL layer. The advantage of the later approach is the transparency at the expense of extra cost for handling regular TCP services. Due to the compatibility issues, we adopt the latter approach for our prototype system.

Suppose that we use the current internet domain. Then, the next issue is how to make a regular/mobile connection. For example, the MSL at the client site always attempts to make a mobile connection with a remote virtual port following the VPP protocol, which defines the PDU's and exchange procedures. However, the server system may not support VPP. Therefore, the establishment of underlying TCP connection may have different meanings to the client and the server. The client will use VPP, whereas the server will consider the connection between two sockets has been completed and bypass PDUs up to the server application processes. Two solutions are available. The first solution is to use an additional TCP option. The client virtual port sends `INI_REQ` to normal well-known port with new TCP options which inform the server that the client system wants to set up a mobile connection. If the server system is MSL-enabled, it will correctly understands and responds such request with new mobile TCP options. If the server system does not support mobile service, it must discard it and complete normal TCP connection according to the requirement to TCP/IP host system[43]. The second solution is to use the so called magic number. After establishing a underlying TCP connection, the MSL system at the client site sends the server a magic number at the beginning of data stream which is followed by the VPP `INI_REQ` or `REC_REQ` PDU's. From the magic number, the server will understand that the client system needs mobile services. Otherwise, a regular association is set up. As for the first approach, it requires to enhance the lower layers. We adopt the magic number approach for our prototype system.

The disconnection of a TCP connection between a pair of VPs is quite similar to the close procedure of the TCP connection between normal TCP ports. It is worth noting the waiting period for reuse of a local VP port. In order to prevent a TCP connection from being interfered by data of previous TCP connections that arrive too late, a state, called `TIME_WAIT`, is introduced in TCP state transition diagram (pp.241,

[69]). When a local TCP port is in that state, all data arriving at the TCP port will be discarded and it can not be used by a new connection. This waiting time period is 2 times of Maximum Segment Lifetime (MSL). Accordingly, a VP can be reused after the same amount of waiting time. In RFC 793, the MSL is specified as 2 minutes. In our implementation of VP, the common value of 30 seconds waiting time is chosen.

5.6 An Example

In this section, we demonstrate the functions of the virtual port protocol using an example.

Initial Connection of Association and Underlying TCP

Figure 5.15 shows the simplified procedure of initial connection between two sockets, namely, a server and a client.

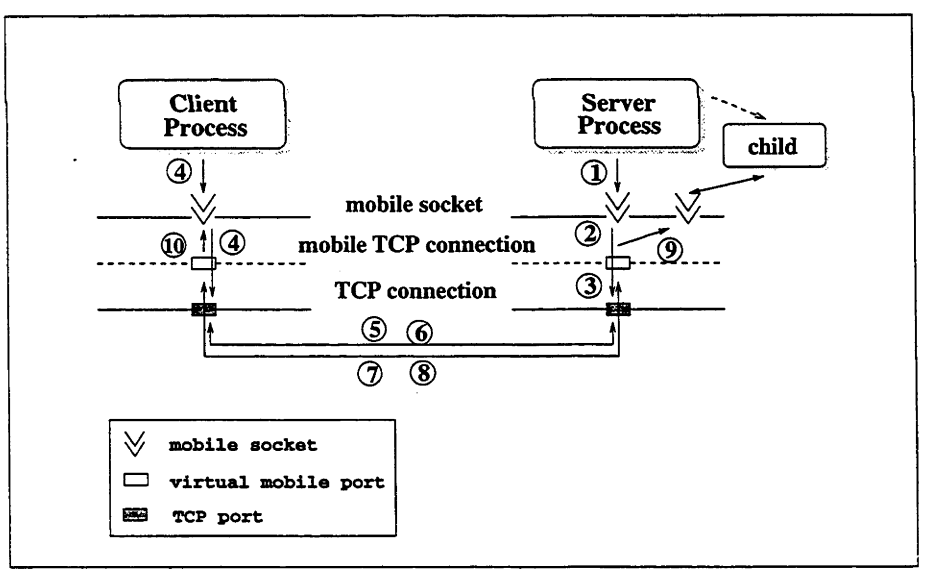


Figure 5.15: Initial Mobile Connection

In the first step, the server process creates a socket and binds it to a local permanent address (home-IP_S, $Port_S$). In Step 2, the MSL system at the server site creates a virtual port for this socket and translates the permanent address into a local current address, (IP_S^h, P_S). Then server process listens on the socket in Step 3. Inside the MSL system,

the virtual port listens to the real local TCP port on behave of the socket. In Step 4, the client process creates and binds an active socket to an assigned local address. The MSL system at the client site creates a underlying local virtual port, (IP_C^h, P_C). Then, the client process initiates a connection with the server process using its permanent address in Step 4. The MSL system at the client site maps both permanent addresses, for the client and the server hosts, into current IP addresses, and then attempts to set up a regular TCP connection between these two current IP addresses, $(IP_C^g, P_C, IP_S^g, P_S)$, in Step 5. In Step 6, the server virtual port accepts the connection request so that a regular TCP connection can be set up. The virtual port protocol entities exchange information between the pair of virtual ports in Step 6, 7 and 8, as depicted in Figure 5.16.

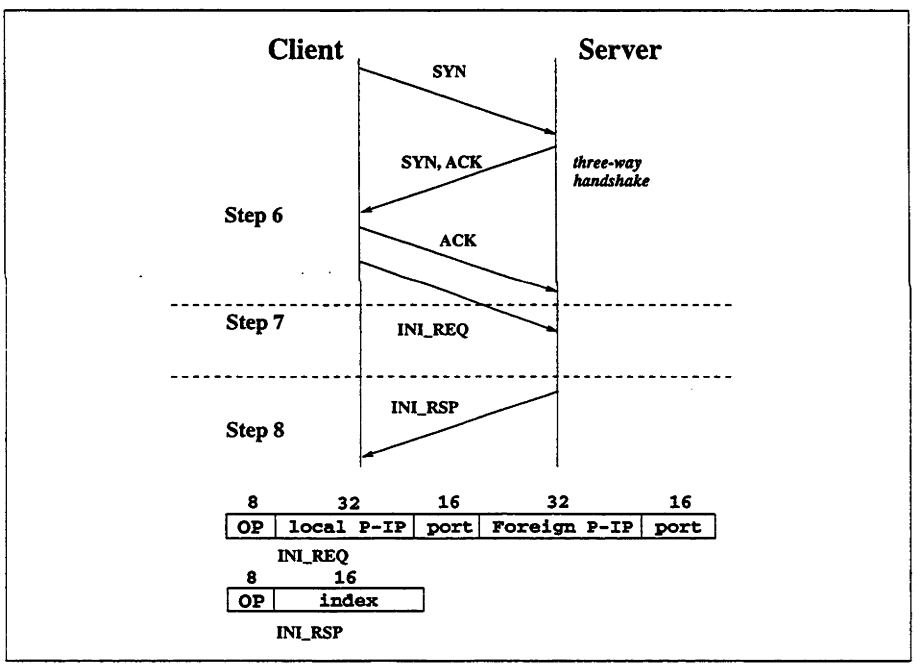


Figure 5.16: Initial Connection of Virtual Ports

As shown in Figure 5.16, the client virtual port first sends an initial request (`INI_REQ` message) to the server virtual port, which contains both local and remote permanent addresses. The server will check to see if the request is destined to a correct system and if there is a running server. Suppose that the server MSL positively replies the request with `INI_RSP`. The 16-bit index V_{id} in `INI_RSP` is assigned for later reconnection.

Typically five TCP segments are exchanged to establish a virtual connection. But we can combine `INI_REQ` with `SYN`, since it is legal under TCP protocol with most TCP implementation[70] The `INI_REQ` and `INI_RSP` messages are used internally in MSL and will not be delivered to the upper layers. If the server system rejects this virtual connection, it simply closes the real TCP connection. The client system will know the request is rejected and all internal data structures will be freed at both systems. Note that unlike TCP, no initial sequential number for association window needs to be exchanged, because it always starts from zero. Each of the two virtual ports maintains two pointers for receiving and sending association windows. The values of pointers indicate the sequential number of the last byte received and sent form and to TCP port. Initial values of these pointers are zero.

In Step 9 and 10, both the client and the server virtual port inform their corresponding sockets of the successful connection. Then, the underlying TCP connection is switched to normal data transfer status.

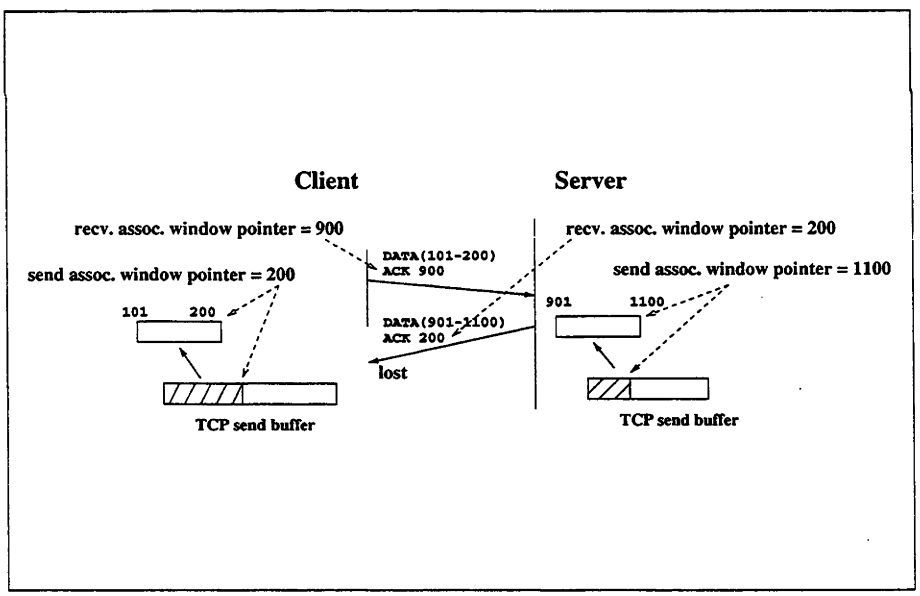


Figure 5.17: Loss of TCP Connection

Suppose that at later stage the client system starts to move and the TCP connection is lost between these virtual ports, as shown in Figure 5.17. Further assuming that the client received the first 900 bytes from the server, and sent the first 200 bytes into TCP

port. We also suppose that the server has received all the 200 bytes, and sent 1100 bytes to the client. Here, the 901st byte to 1100th byte of data being sent by the server are lost. After the TCP connection is broken, both sides fetch the remaining data in the sending buffers of the TCP ports to their virtual port buffer. In the client, there are 100 bytes left in the TCP sending buffer due to the fact that the last IP datagram carrying the ack information from the server was lost. In the server, 200 bytes remain in the TCP sending buffer, and these data are totally lost.

Reconnection of Underlying TCP

When the client arrives at its new location, new TCP connection will be re-established to resume the communication between these two virtual ports. In our VPP, the client system always actively tries to connect to the server system. Envision that client system moves from network N_1 to a new network N_2, and current IP address is IP_{C_2}.

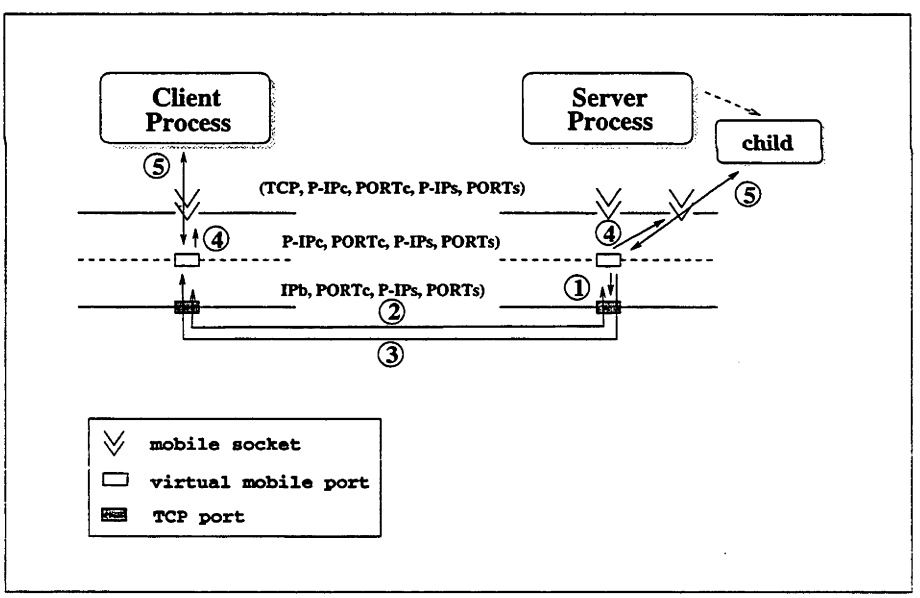


Figure 5.18: Reconnection of Virtual Port

As shown in Figure 5.18, the server always listens on P_S if there are any alive virtual ports associated with it (Step 1). After the client is ready to communicate at its new location, it starts a *three-way handshake* to establish a new TCP connection with the server virtual port at first. The new TCP connection between these two virtual port is

$(IP^g_{C_2}, P_C, IP^g_S, P_S)$. The server will accept this connection first at the real TCP port level which is done in Step 2. In Step 3, the client virtual port sends reconnection request to the server virtual port with V_{id} and its receiving association window pointer. The server virtual port then looks up its association list, and checks to see if there is a pending mobile association to wait for reconnection. If successful, a response is sent back to the client to inform the success of a reconnection (Step 3). Then, in step 4, both virtual ports will resend the data in the buffer of virtual port at first, and then continue the I/O services provided to sockets (in Step 5).

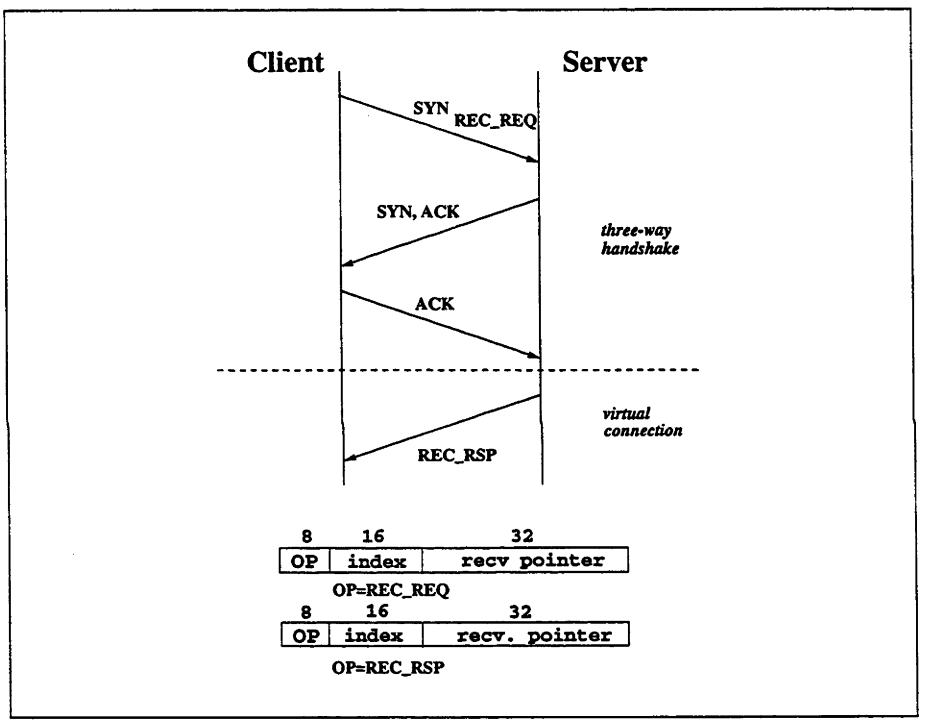


Figure 5.19: Reconnection of Virtual Port

During the reconnection, the messages changed between virtual ports are depicted in Figure 5.19. Firstly, the *three-way handshake* establishes a TCP connection between the two virtual ports. Then, the client virtual port sends a `REC_RES` to the server virtual port which contains the V_{id} and the receiving pointer of its receiving association window of 900. The server responses a `REC_RSP` with a pointer of 200 from which client virtual port understands that there is no need to resend the data. The server virtual port has to resend the data to the client starting from 901.

If the server system moves, or even both systems move during the same period, the reconnection procedure will be similar to the case when only the client system moves. The server system always passively listens on a certain TCP port. When the client system arrives at a new location, it will attempt to connect the server system at server's previous IP address. If it fails after timeout or ICMP error message goes back, the client system will try to fetch server system's new current IP address and try again until the re-connection is successfully completed or the association between mobile sockets is broken due to failure of connection.

System calls `getpeername` and `getsockname` will be used if a process enquires the IP address from DNS name. In a mobile environment, we need emphasize that an IP address returned by these two system calls is a permanent IP address. application processes are shielded from current IP address at all.

5.7 Conclusion

In this chapter, we discussed a two-step solution to support mobility, in particular the second step, namely, the mobile TCP socket. In our mobile TCP socket, a mobile mapping is introduced which maps TCP associations to TCP connections. The mobile mapping an be implemented in the socket and on top of TCP/IP layers. We show the implementation details in this paper as well. Our approach achieves high compatibilities with the current TCP/IP protocols due to the following reasons. First, we adopt the same API of the socket layer. Second, we don't require any special functionality in the network or the transport layers to support mobility. Finally, we don't require all networks, where mobile hosts may visit, 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. As a future work, we attempt to conduct a performance study on the two costs mentioned above.

Chapter 6

Performance Studies

6.1 Introduction

In this chaper, we present some performance results on mobile TCP/IP solutions. Several simulation experiments have been conducted to analyse the two-step solution and compare with the other existing mobile TCP/IP solutions. The results demonstrate the effectiveness of our two-step mobile solution.

We choose MDL and IMHP as the typical data link and network layer solutions to be compared with the two-step solution. A summary of these three solutions are given in Table 6.1

Solution	The Layer	Supported Protocols
MDL	Link	TCP/IP, Novell, etc
IMHP	Network	TCP, UDP
Two-Step	Socket/Portable-IP	TCP

Table 6.1: Three mobile solutions.

Obviously, the MDL is powerful in terms of support of diverse protocols, It can support most existing network systems, given the implementation of today's communication protocol stacks are usually compatible with common interface at data link level. However, the functionality is at the price of sacrificing system performance. The IMHP is a current internet draft standard for mobile communications, it can support both TCP and UDP in mobile environment, because IMHP provides standard interface at network layer. However, it still has got some performace problem. Our Two-Step solution support portable communicaiton in IP and mobile for connection-

oriented transport protocol — TCP only in this research work. It seems less powerful, however, the better performance and flexible system structure can compansate it.

To compare the performance, first, we propose a cost model and the results are analysed from simulation programs.

6.2 Cost models

The two major costs for supporting mobility are: the propagation cost to propagate the location information of mobile hosts, and the forwarding cost to forward packets to mobile hosts. The propagation cost and the forwarding cost for reconnection can vary depending on the primary layer on which the mobility is supported.

A generic moving pattern model

The generic moving pattern model at the layer L_j can be represented as a sequence of pairs:

$$S^j = \langle (T^j_{m_1}, T^j_{w_1}), (T^j_{m_2}, T^j_{w_2}), \cdots, (T^j_{m_n}, T^j_{w_n}) \rangle$$

where $T^j_{w_k}$ and $T^k_{m_k}$ are the k-th working period and moving period, respectively. It models that a mobile host will physically be moving for T^j_m time units in which no network services are available at the layer L_j, and will then be connected at a new attach point for T^j_w time units of services at the layer L_j. We use S^0 to refer to the geographical movement of a mobile host regardless of the communication protocol stacks.[1]

Furthermore, as depicted in Figure 6.1, a T^j_w is a triplet (T_d, T_s, T_c), where T_s is the service time during which the layer L_j can provide mobile services to the upper layers. T_d and T_c are the times for disconnection and reconnection, respectively.

Suppose that the layer L_i supports mobility. For each layer $j, 1 \leq j \leq i$, we have:

$$T^j_{w_k} = T^{j-1}_{s_k}$$

[1] We assume that the MAC layer connection/handover times can be absorbed by the moving time $T^0_{m_k}$ of S^0.

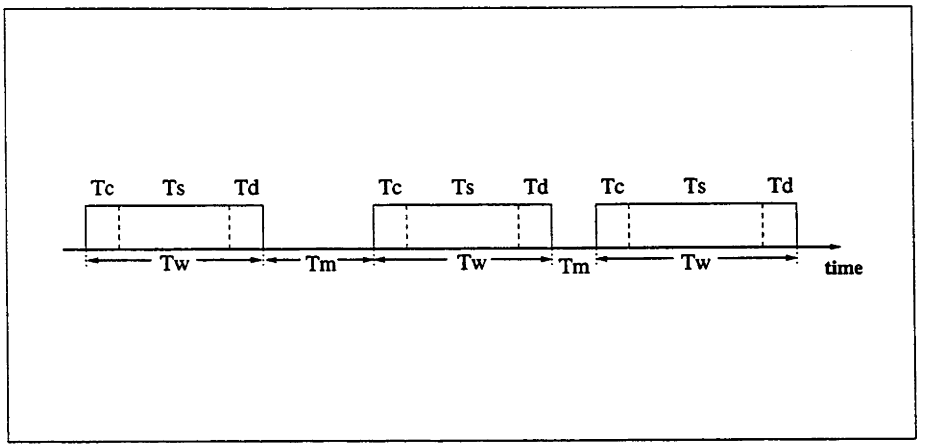


Figure 6.1: A generic moving pattern model.

$$T^j_{w_k} + T^j_{m_k} = T^{j-1}_{w_k} + T^{j-1}_{m_k}$$

Following the above properties, the moving pattern S^j at the layer L_j can be determined based on S^0. Therefore, it is important to note that S^0 provides working patterns based on which we can analyse different mobile solutions.

The throughput

The throughput at the layer L_j where mobility is supported, is defined as:

$$\mathcal{T}^j = \frac{V^j}{\sum_{k \in S^j}(T^j_{w_k} + T^j_{m_k})}$$

Here, V^j is the total amount of data being transfered at the layer L_j. Obviously, the throughput, $\mathcal{T}^j$, is affected by the mobile protocols and the moving patterns S^0. The longer the average of moving periods is, the lower the throughput of $\mathcal{T}^j$ is. Even though T_m will tremendously affect the throughput given above, it will be the same for all different mobile protocols in our simulation studies. To get rid of T_m in the throughput formula, We define the ratio of the working time and the moving time of given moving pattern S^j as:

$$\beta_{S^j} = \frac{\sum_{k \in S^j} T^j_{m_k}}{\sum_{k \in S^j} T^j_{w_k}} \tag{6.1}$$

Then, we have the throughput formula as follows:

$$\mathcal{T}^j = \frac{V^j}{\sum_{k \in S^j}(1+\beta_{Sj})T^j_{w_k}}$$

To more precisely compute the throughput, the affects of T_m should be totally avoided. Therefore, we ignore the T_m, and define the effective throughput as follows.

$$\mathcal{T}^j_e = \frac{V^j}{\sum_{k \in S^j} T^j_{w_k}} \tag{6.2}$$

The maximum throughput at the application layer

Two main factors will affect the throughput $\mathcal{T}^{App}(= \mathcal{T}^4)$: the bandwidths and the behaviour of the communication protocols. However, the dominate factor is the bandwidth which limits the maximum $\mathcal{T}^{App}$.

The maximum $\mathcal{T}^{App}$ can be obtained as follows. Since the data being transfered will be encapsulated into the Protocol Data Units (PDU) for transmission which consists of the Protocol Control Information (PCI) and the Service Data Unit (SDU) used in its immediate upper layer PDU[40], the ratio of the payload at Layer L_j is

$$\gamma_j = \frac{|SDU^j|}{|PCI^j| + |SDU^j|} = \frac{|SDU^j|}{|PDU^j|}$$

where $\gamma_j < 1$. Because $SDU^j = PDU^{j+1}$,

$$|PDU^{j+1}| = \gamma_j |PDU^j|$$

Therefore, given a bandwidth B, the maximum amount of application data can be transferred is:

$$\begin{aligned} V^{App}_{max} &= \sum_{i \in S^4} |PDU^4_i| \\ &= \sum_{i \in S^1} (\prod_{j=1}^{4} \gamma_j) |PDU^1_i| \\ &\leq (\prod_{j=1}^{4} \gamma_j) B \sum_{i \in S^0} T^0_{w_i} \end{aligned} \tag{6.3}$$

where $|PDU_i|$ is the total amount of PDU being transferred during the i-th working period. By combining (6.2) and (6.3), the upper bound of the effective throughput $\mathcal{T}_e^{App}$ is:

$$\mathcal{T}_e^{App} \leq (\prod_{i=1}^{4} \gamma_i) B \tag{6.4}$$

The utilisation of mobile services

Since the times for disconnection and reconnection are so important in mobile communications, the utilisation of mobile services is defined as a ratio as follows.

$$U = \frac{\sum_{i \in S} T_{s_i}}{\sum_{i \in S} T_{w_i}} \tag{6.5}$$

The communication cost and the propagation cost

In general, a mobile solution can employ multiple forwarding points in the Internet. If all the forwarding points know where mobile hosts currently stay, then any of the forwarding points can forward datagrams to mobile hosts. Therefore, it is possible to find a suboptimal route from any host to a mobile host via one of the forwarding points. Furthermore, if the number of forwarding points increases, then the actual cost of sending datagrams to mobile hosts can be reduced to minimum. It is simply because that datagrams do not need to be forwarded to the home network of the mobile host and then be forwarded to the mobile host. However, in order to dynamically find optimal routes, location information needs to be frequently exchanged among all forwarding points. Let C^i and P^i denote the communication cost for regular data and the propagation cost of exchanging location information for a mobile solution supported at the layer L_i as follows.

$$P^i = \mathcal{F}(m, \mathcal{P}, S) \tag{6.6}$$

$$C^i = \sum_{j}^{d} (c_{1_j} \cdot V^4 + c_{2_j}) \tag{6.7}$$

Here, P^i is determined by three parameters: the number of forwarding points m, the behaviours of a mobile protocol $\mathcal{P}$, and a moving pattern S. Obviously, P^i is inde-

pendent of data volume, whereas C^i is related to data volume being transferred. As given in equation (6.7), d is the number of hops for transferring data between two ends. Both c_{1_j} and c_{2_j} are the costs of sending and processing a unit of data at the j-th hop, respectively. As can be seen above, these two costs, P^i and C^i, are correlated with each other. If m increases, then P^i will be high. However, the possibility of finding suboptimal routes to mobile hosts will be high as well. Therefore, C^i can be reduced, since d can be minimised. An ideal mobile solution is to keep both P^i and C^i as low as possible.

All the costs, c_1 and c_2, can be estimated based on bandwidths, delays, and consumption of the system resources, as what OSPF supports in its routing algorithm[69].

6.3 A performance study

6.3.1 Three mobile systems and their generic layered models

We use *C++SIM* to simulate three mobile systems, namely, MDL, IMHP and our two-step solution.

- **MDL:** The MDL solution provides mobility at Link layer, L_1. There is an Access Point (AP) at each participating network, which basically is a Link layer bridge to connect wireless mobile hosts to a wired network. After arriving at a new location, a mobile host needs to register with its home Access Point (AP) and the local AP. It will take T_c time units to resume mobile communication services to TCP/IP protocol suite (Figure 6.2). During T_c, four packets need to be exchanged between the mobile host and its home and local APs. Since the delay for registering with the local AP can be ignored, T_c will approximately be a RTT between the mobile host and its home AP.

- **IMHP:** IMHP is a layer 2 solution. Its layered model is shown in Figure 6.3. A mobile host needs to register with its Home Agent (HP). Therefore, T_c is approximately one RTT.

- **Two-Step Mobility:** Our two-step mobile TCP socket is a solution at the layer 2 and 3, because it provides portable and mobile supports on both network and

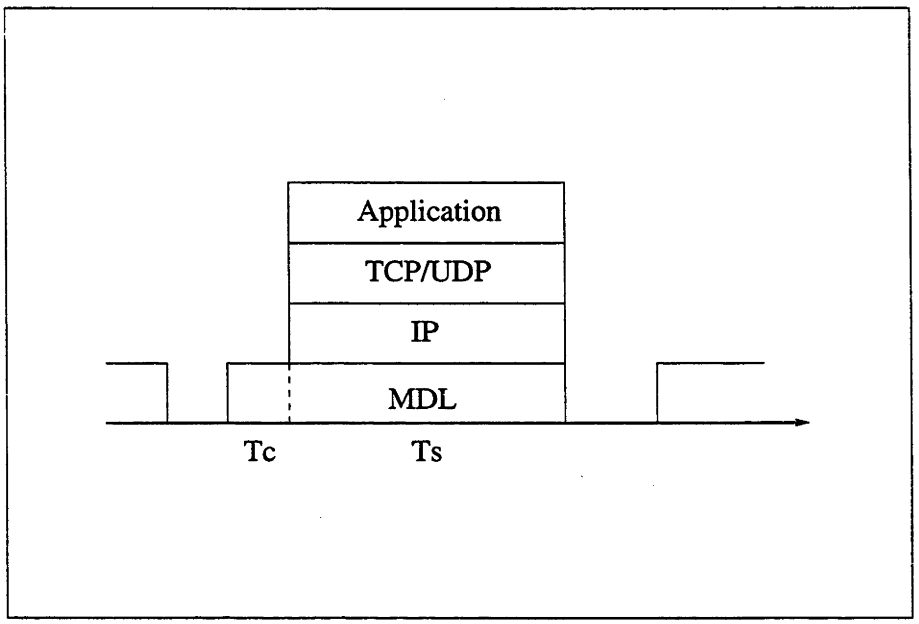


Figure 6.2: The layered model of MDL.

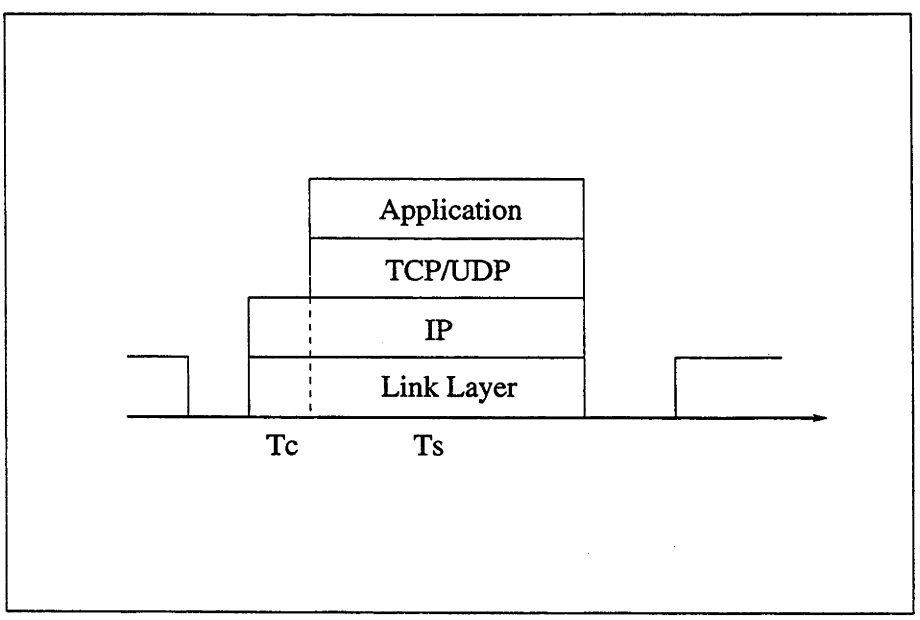


Figure 6.3: The layered model of IMHP.

transport layers. In a similar fashion, a mobile host needs to register with its home portable support system, and the mobile socket connection will take time to make a new TCP connection for an active TCP association. As depicted in Figure 6.4, the T_c is the sum of that one RTT from MH to its home portable support system for registration, and 1.5 RTT from MH to FH for TCP reconnection as for TCP three-way handshake[69].

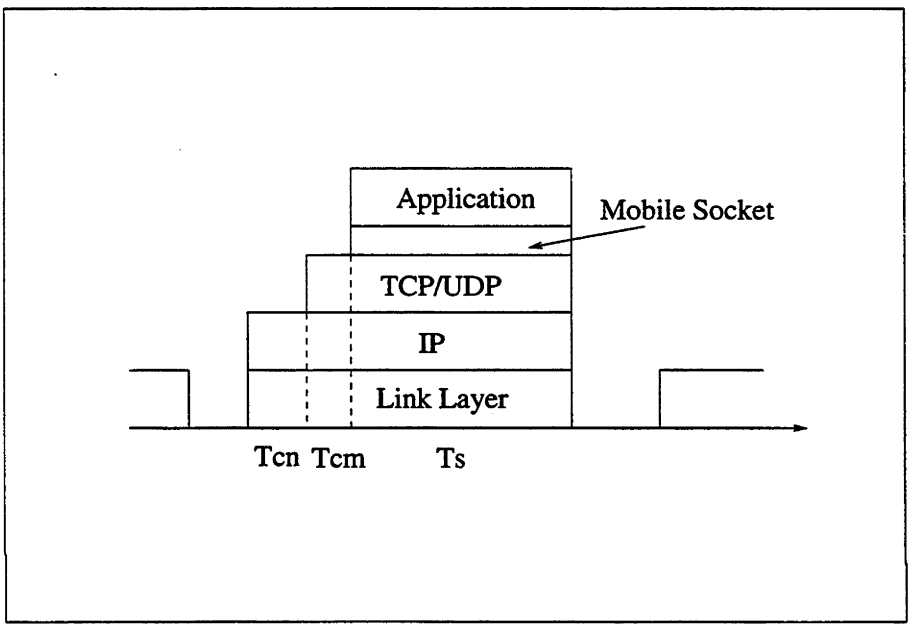


Figure 6.4: The layered model of the two-step solution.

Generally, T_c over the Internet varies from several milliseconds to several hundreds. However, T_s is considerably long, and varies from seconds to hours.

6.3.2 A simulation environment

As shown in Figure 6.5, there are three components in our simulation environment, namely, a mobile host (MH), its home support system (HA, for either a Home Access Point or a Home Agent), and a fixed server host system (FH). The moving area will be specified by ranges of RTT(Round Trip Time). For simplicity, we assume that the length of link-1 is fixed.

The data being transferred between MH and FH will go through different paths and will use different encapsulation methods. As shown in Figure 6.6(a), MDL uses

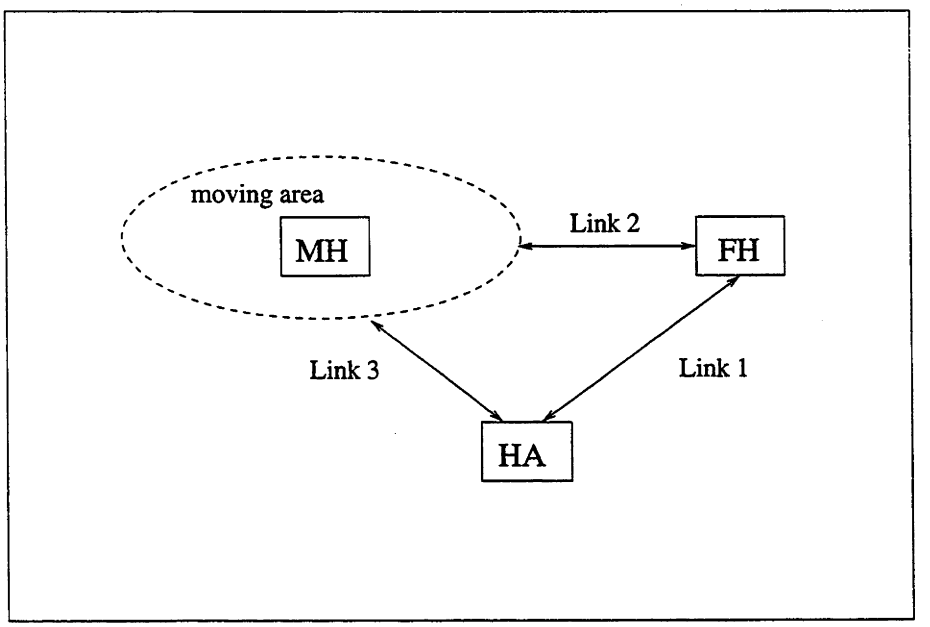


Figure 6.5: A simulation environment.

HA to forward data in both directions when MH is away from home network. All Link Layer packets will be encapsulated into a UDP datagram and then sent out with IP header and Link Layer header. IMHP uses triangle routes (Figure 6.6(b)). The packets from MH to FH and from FH to HA are regular IP datagrams with Link Layer header, whereas all IP datagrams from HA to MH will be re-encapsulated by additional IP header according to IP-IP tunnelling protocol. As shown in Figure 6.6(c), our two-step solution employs direct routes between MH and FH. In addition, all packets are sent using the current IP datagrams.

To describe the differences between a direct route and a triangle route (detour), we use a Detour Ratio factor which is denoted α.

$$\alpha = \frac{R_{dir}}{R_{det}} \approx \frac{B_{dir}}{B_{det}}$$

where R_{dir} and R_{det} are the number of hops used in the direct route and the triangle route, respectively. B_{dir} and B_{det} are the bandwidths used in the direct route and the triangle route, respectively. Since a longer path in the Internet usually causes low throughput, we assume that the ratio of B_{dir}/B_{det} is approximately the same as

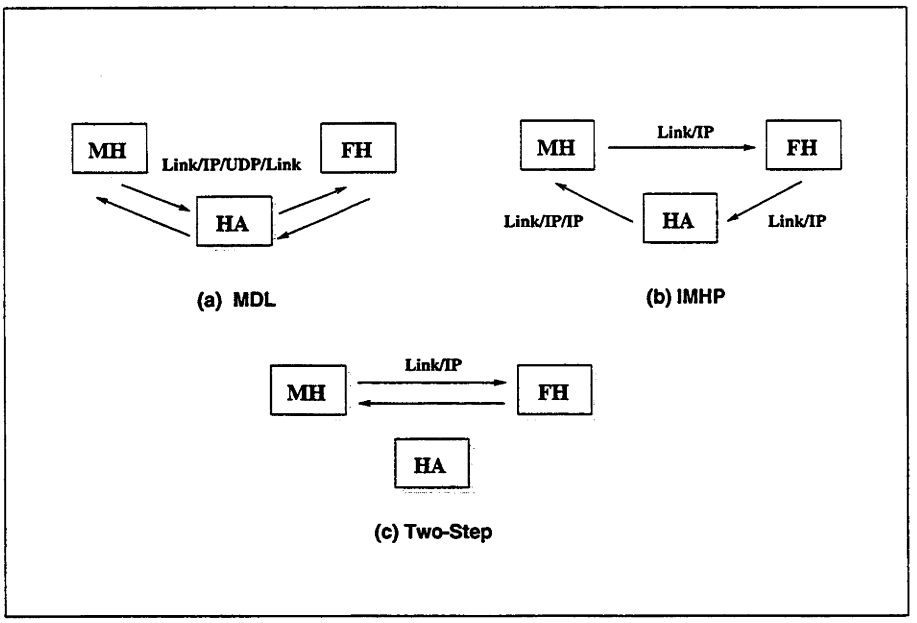


Figure 6.6: The routes used in the three mobile systems.

R_{dir}/R_{det}.

6.3.3 The parameters

The system parameters used in our simulation are given in in Table 6.2.

Given the amount of application data being transfered, V^{App}, and a moving pattern, S^0. We simulate how many working periods, T_{w_i}, are needed to transfer all V^{App}. Since the effective throughput given in the equation (6.2) is as follows:

$$\mathcal{T}_e^{App} = \frac{V^{App}}{\sum_{k \in S^{App}} T_{w_k}^{App}}$$

we need to estimate both the service time of a mobile solution and the additional control information to be added. The j-th service time $T_{s_j}^{App} = T_s^i$ (the service time at the layer L_i where mobility is supported). Therefore, $T_{s_j}^i = T_{w_j}^i - T_{c_j}^i$, since we simply assume that $T_{d_j}^i = 0$. The different T_c values are given in Table 6.3.

The total amount V^{App} is needed to be transferred in several working periods, n, as follows.

$$V^{App} = \sum_{i=1}^{n} V_{w_i}^{app} = \sum_{i=1}^{n} \Gamma \cdot V_{w_i}^1 = \sum_{i=1}^{n} \Gamma \cdot B \cdot (T_{w_i} - T_{c_i})$$

Name	Description
V^{App}	the total amount of application data
$\overline{T_w}$	mean working time
RTT_1	average round trip delay between HA and FH
RTT_{3l}	low boundary of RTT between HA and MH
RTT_{3h}	high boundary of RTT between HA and MH
α_l	low boundary of detour ratio
α_h	high boundary of detour ratio
B_l	low boundary of bandwidth of link 2 between FH and MH
B_h	high boundary of bandwidth of link 2 between FH and MH
$\overline{L}$	mean length of packets
Link	(PPP/HDLC/Ethernet)

Table 6.2: System parameters.

System	T_{c_i} Value
MDL	$1 \times RTT_1$
IMHP	$1 \times RTT_1$
2-Step	$1 \times RTT_1 + 1.5 \times RTT_3$

Table 6.3: Connection times.

Here,

$$\begin{aligned}\Gamma &= \prod_{i=1}^{4}\gamma_i = \frac{|v^{app}|}{|PDU^4|}\cdot\frac{|PDU^4|}{|PDU^3|}\cdot\frac{|PDU^3|}{|PDU^2|}\cdot\frac{|PDU^2|}{|PDU^1|} \\ &= \frac{|v^{app}|}{|PDU^1|} < 1\end{aligned}$$

where v^{app} is the amount of application data in a packet, and

$$|PDU^1| = (((\underbrace{|v^{app}| + |PCI^4|}_{|PDU^4|}) + |PCI^3|) + |PCI^2|) + |PCI^1|$$

$|PDU^3|$ and $|PDU^2|$ as nested underbraces.

In Table 6.4, we list the sizes of headers used in different protocols. Table 6.5 lists the PCI^i for the three mobile systems. Note that usually the whole application PDU is considered as the application data. But MDL method will put whole original link layer packet as the application PDU into a UDP datagram. Therefore, it has a large PCI^4 value. As for IMHP, we use the IP-IP encapsulation to calculate PCI^2. Here, the link from HA to MH is the bottleneck compared with the same bandwidth of link from FH to HA.

Given both a protocol and an encapsulation method, the $\sum_{i=1}^{4}|PCI^i|$ is a constant. In fact, $|PCI^i|$ is the sum of header and trailer of i-th layer protocol. Although the length of IP and transport protocol (TCP) headers may vary according to the number of options, for simplicity, we assume no options included in all those headers. The Γ value we will use in our simulation is calculated as follows.

$$\Gamma = \frac{|v^{app}|}{|\overline{L}|} = \frac{|\overline{L}| - \sum_{i=1}^{4}|PCI^i|}{|\overline{L}|} < 1$$

Layer	Protocol	Length (bytes)
Link	PPP	8
	HDLC	10
	Ethernet	14
Network	IP	20
Transport	TCP	20

Table 6.4: The sizes of the protocol headers. Note that in most statistics, 7 bytes "preamble", 1 byte "start delimiter" and 4 bytes CRC in Ethernet Frame are excluded.

System	PCI^1	PCI^2	PCI^3	PCI^4
MDL	H_L	H_{IP}	H_{Tr}	$H_L + H_{IP} + H_{Tr}$
IMHP	H_L	$2H_{IP}$	H_{Tr}	0
2-Step	H_L	H_{IP}	H_{Tr}	0

Table 6.5: PCI's of the three mobile systems.

where $\overline{L}$ is the mean length of a link layer packet. We choose three different communication links: namely, the serial point-to-point *PPP* link, the HDLC long distance link[69], and Ethernet. We assume that $\overline{L}$ are, 296 and 576 which are the Maximum Transmission Units for PPP and HDLC links, respectively. But $\overline{L}$ is chosen as 285 on average for Ethernet based on our observations of a 3COM Ethernet Switching Hub (refer to the Appendix). All the values of Γ are given in Table 6.6.

Parameter	PPP	HDLC	Ethernet
MDL Γ	0.68	0.83	0.62
IMHP Γ	0.77	0.88	0.74
2-Step Γ	0.84	0.91	0.81

Table 6.6: Γ Values.

As discussed above, the effective throughput can be simulated based on data volume V^{App}, a bandwidth B, the mean length of a link layer packet $\overline{L}$, RTT's and a moving pattern S^0.

As for the propagation costs, since we assume all the three schemes only use a single forward point in this simulation study, the three mobile solutions will have

similar propagation cost as given in the equation (6.7).

However, it is obvious that the communication costs along the route from FH to MH may vary widely when different encapsulation methods and routes are used. Usually, Internet Service Providers (ISP) charge fee based on the amount of link layer data traffic and the classes of data traffics (local/domestic/international traffic). In order to estimate the communication costs, we use a modified version of the equation (6.7) in which c_{1_j} is the cost to transfer a unit of application data, and c_{2_j} is ignored because our major concern is the amount of data being transferred. The actual communication cost is calculated as follows,

$$\begin{aligned} C &= \sum_{j}^{d}(c_1' \cdot V^{Link}) \\ &= \alpha \cdot c_1' \cdot V^{Link} \\ &= \alpha \cdot c_1' \cdot (p \cdot \overline{L} + V_{rec}) \end{aligned} \tag{6.8}$$

where α is a detour factor. If data can be transferred from FH to MH directly, $\alpha = 1$. V^{Link} is the total amount of link layer traffic for transferring V^{App}, and p is the number of packets we need to send.

$$\begin{aligned} p &= \lceil \frac{|V^{App}|}{|V^{app}|} \rceil \\ &= \lceil \frac{|V^{App}|}{|\overline{L}| - \sum_{i=1}^{4} |PCI^i|} \rceil \end{aligned}$$

It is worth of noting that p varies based on a mobile solution. Hence, C varies accordingly. V_{rec} is the amount of data for reconnection which is only needed in the two-step solution.

6.4 Simulation results

We first analyse the various system parameters which will have impacts on the throughput.

The effectiveness of RTT

In this study, we fixed all parameters but RTT's to see the effectiveness of RTT. The RTT of link-1 and link-3 are the same. Table 6.7 lists the values of the system parameters being used. The selection of this set of parameters is based on a typical ISDN link. Two $\overline{T_w}$ values are chosen as 3 and 30 seconds to simulate fast/slow movements of MH, respectively. The results are shown in Figure 6.7 and 6.8. From these figures, we can see that all mobile systems are affected by RTT values, in particular, our 2-Step solution. In Figure 6.7, 2-Step outperforms MDL and IMHP when RTT is less than 400ms. If RTT is longer, MDL and IMHP will outperform 2-Step. However, if $T_s = 30$sec, as shown in Figure 6.8, 2-Step will outperform MDL and IMHP in terms of throughput, even though the performance of the 2-Step solution drops by 50% while RTT is tuned from 500ms to 800ms. Since 30 second mean working time is not really long, we expect that 2-Step outperforms MDL and IMHP in most cases.

Name	Value
V^{App}	2 Gbyte
α_l	1.3
α_h	1.8
B_l, B_h	512Kbps
Link	PPP

Table 6.7: Values of the system parameters.

The effectiveness of Γ

In this experiment, we use $\overline{T_w} = 60$ seconds, and investigate different throughputs by fixing all the system parameters except Γ. Two sets of Γ values, for Link PPP and HDLC, are used as given in Table 6.6. The results are shown in Figure 6.9. It is obvious that the more efficient encapsulation will gain better throughput. And, the more fast the link speed is, the better performance can be obtained. In Figure 6.9, the highest two lines are of the 2-Step system, which means 2-Step outperforms MDL and IMHP, no matter whether Link Layer protocol (PPP or HDLS) is employed.

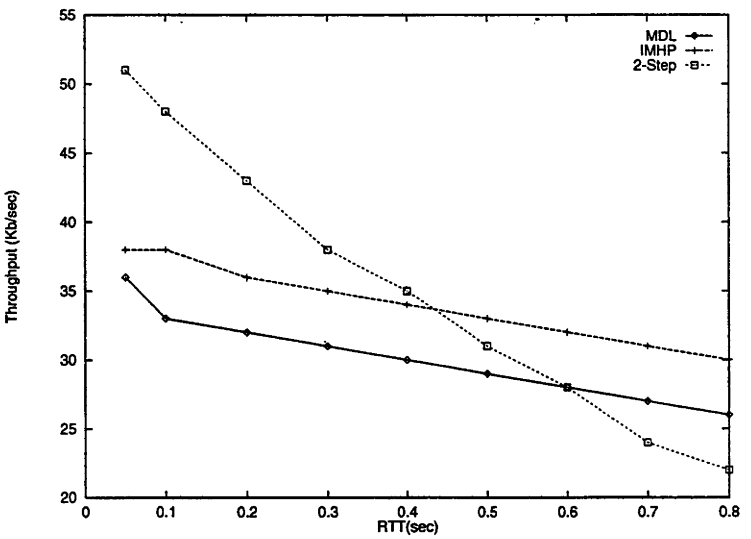


Figure 6.7: Fast movements of MHs, where $\overline{T_w} = 3$ seconds.

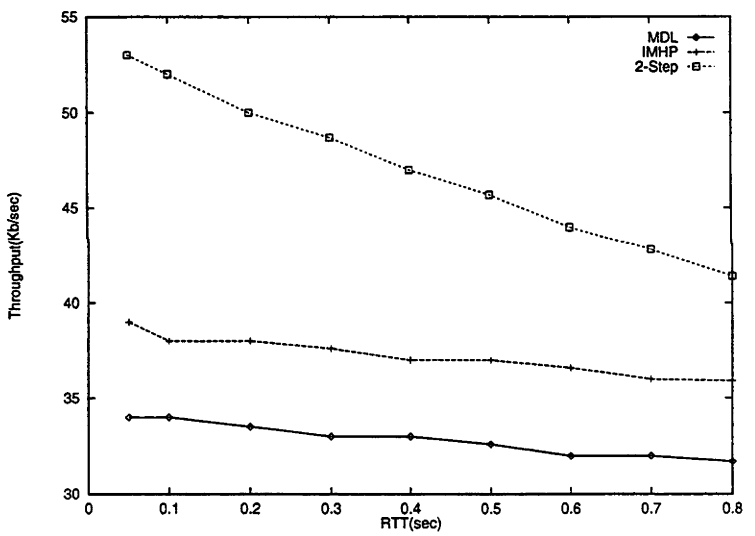


Figure 6.8: Slow movements of MHs, where $\overline{T_w} = 30$ seconds.

Name	Value
V^{App}	2 Gbyte
$\overline{T_w}$	60 seconds
α_l	1.3
α_h	1.8
Link	PPP/HDLC

Table 6.8: Values of the system parameters.

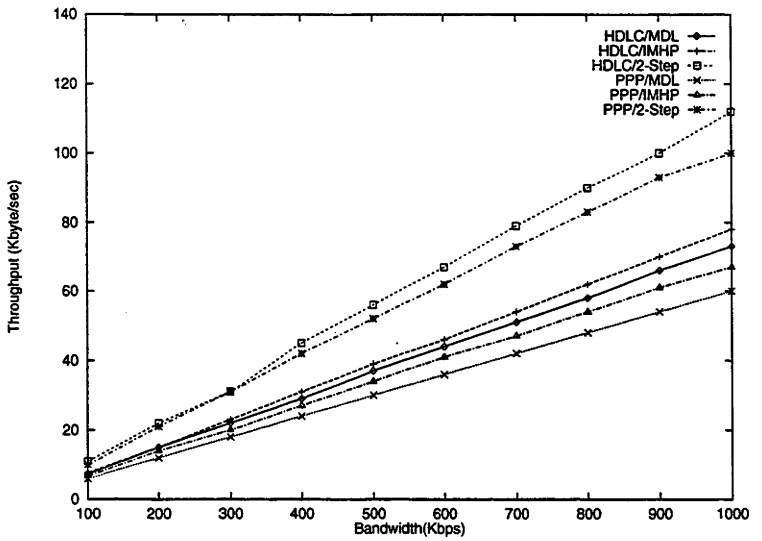


Figure 6.9: The effectiveness of Γ.

The effectiveness of α

This experiment shows how the detour rate, α, will affect throughput. The values of the system parameters for this study are given in Table 6.9. As shown in Figure 6.10, our 2-Step solution will not be affected by detour ratio, since direct route is always used. However, the higher detour ratio will affect MDL and IMHP, and waste bandwidths of higher speed links.

Name	Value
V^{App}	2 Gbyte
$\overline{T_w}$	60 seconds
α_1	between (1.2, 1.5)
α_2	between (1.5, 1.8)
Link	HDLC

Table 6.9: Values of the system parameters.

Simulation of an WAN environment

In this experiment, an WAN environment is simulated. All values of the system parameters are listed in Table 6.10. The communication link between a fixed host and a forward point is very close ($RTT_1 = 80ms$). The link between a mobile host and its

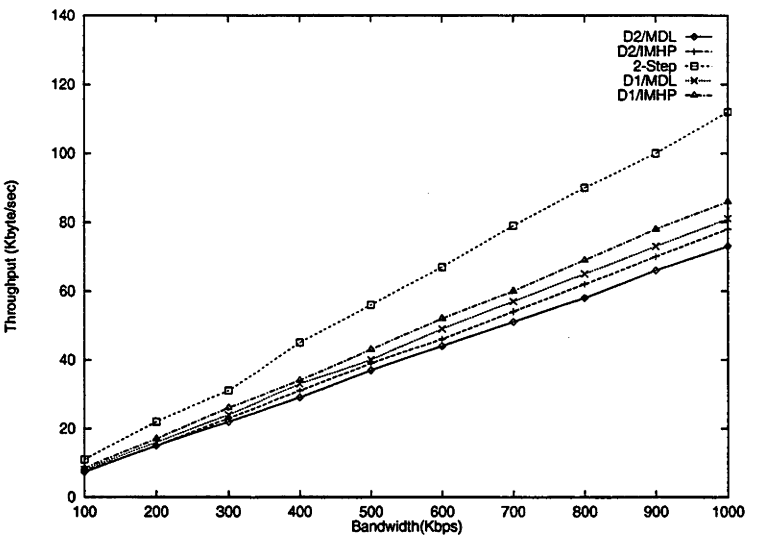


Figure 6.10: The effectiveness of α, where $D1 = \alpha_1$ and $D2 = \alpha_2$.

Parameter	Value
V^{App}	2 Gbyte
RTT_1	80ms
RTT_{3l}	200ms
RTT_{3h}	400ms
B_l	100Kbps
B_h	200Kbps
α_1	between (1.1, 1.3)
α_2	between(1.4, 1.6)
Link	PPP

Table 6.10: Values of the system parameters.

home forward point is long and slow (RTT_3 is between 200*ms* and 400*ms*). The range of RTT_3 implies the possible moving area of mobile hosts. The bandwidth of link-2 is low between 100*Kbps* and 200*Kbps*. We use two detour ratios to simulate an interstate internetwork environment. One is a small detour where α_1 is between 1.1 and 1.3. The other is a medium detour where α_2 is between 1.4 and 1.6.

The results are shown in Figure 6.11 and 6.12. 2-Step outperforms the other two solutions if $\overline{T_w}$ is equal to or greater than 1.5 seconds. If a mobile system stays in a network longer than 4 seconds, all the three systems become stable. The detour has almost no effects on the 2-Step system, since direct routes are used. In contrast, a large detour factor causes low throughput to the other two solutions. Given the same detour ratio, IMHP outperforms MDL, because the double UDP/IP and link layer encapsulation waste communication bandwidths.

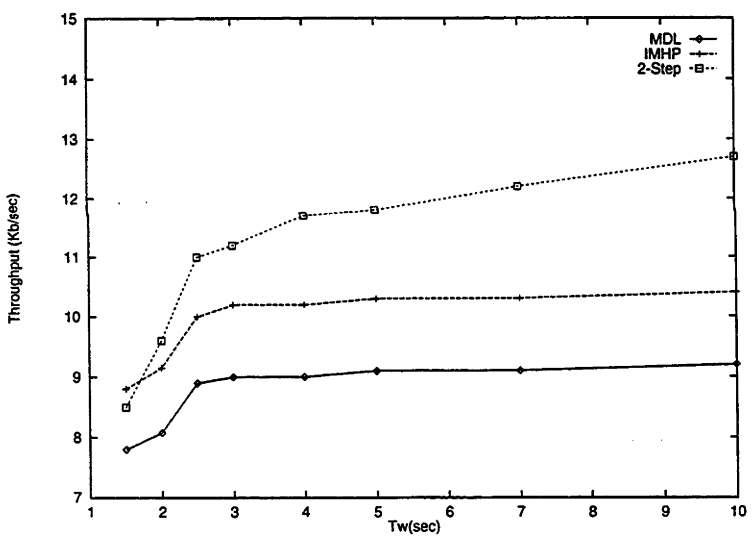


Figure 6.11: The effectiveness of α_1 in an WAN environment.

Simulation of a moderate communication bandwidth and RTT

In this experiment, we change both RTT_3 and the bandwidth to simulate a network with both moderate communication bandwidth and RTT delay. The values we use are based on a metropolitan network as shown in Table 6.11. Since the smaller area all mobile hosts move around can cause a larger detour, we set the detour ratios to be

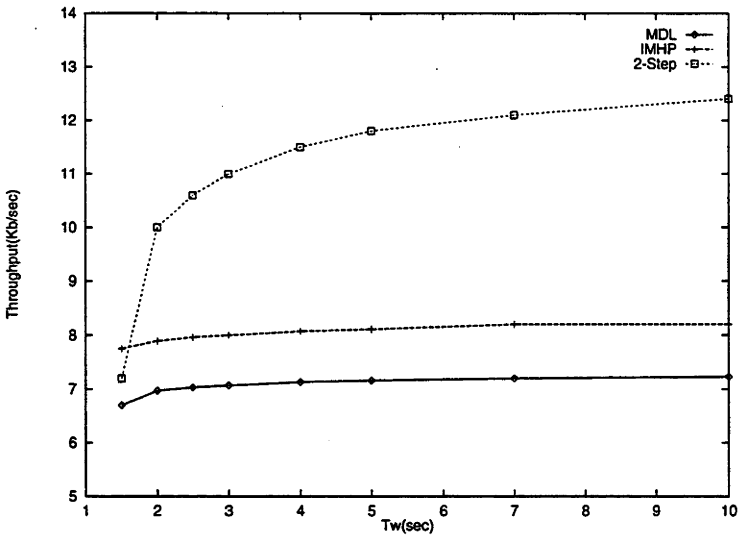


Figure 6.12: The effectiveness of α_2 in an WAN environment.

larger than that used in our simulation for WAN environments.

Parameter	Value
V^{App}	2 Gbyte
RTT_1	80ms
RTT_{3l}	100ms
RTT_{3h}	250ms
B_l	500Kbps
B_h	800Kbps
α_1	between (1.3, 1.5)
α_2	between (1.6, 1.8)
Link	HDLC

Table 6.11: Values of the system parameters.

From Figure 6.13 and 6.14, we understand that this environment shares the similar characters of the WAN environment. However, the gap between 2-Step and the others is reduced, which is caused by the additional connection time for using portable-IP. Although the absolute value of reconnection time is small, the waste of bandwidth is evaluated as $T_c \times B$. The higher speed a mobile host moves, the more bandwidth will be wasted. The stable time is delayed until 10 seconds. If a mobile system moves fast, the performance of 2-Step can be even worse. However, it is not the case that the mean stay time of a mobile host in a network is less than 4 seconds.

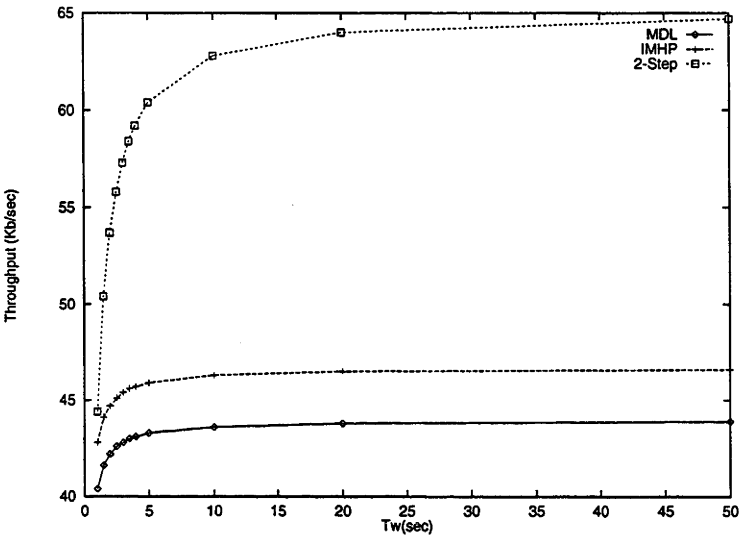


Figure 6.13: A moderate environment using α_1.

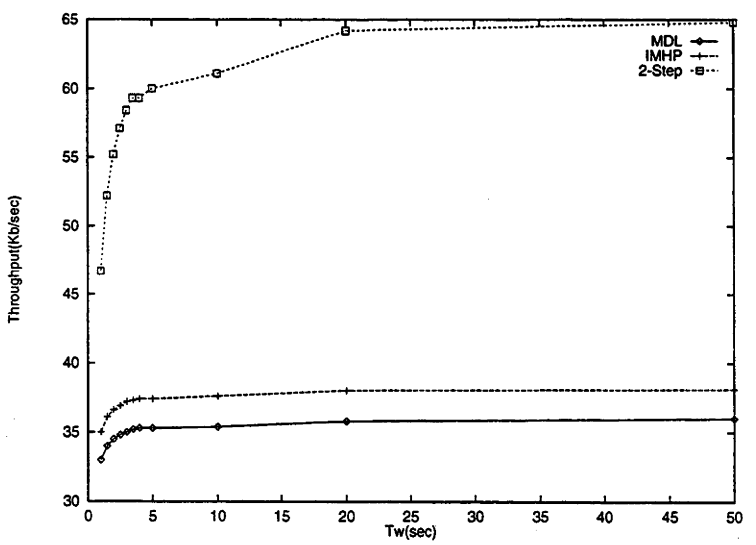


Figure 6.14: A moderate environment using α_2.

Simulation of a high speed network with low RTT

In this experiment, we simulate a high speed network with low RTT delay in which Ethernet segments are interconnected by routers within a campus. The values of the system parameters are listed in Table 6.12.

Parameter	Value
V^{App}	20 Gbyte
RTT_1	60ms
RTT_{3l}	60ms
RTT_{3h}	120ms
B_l	1000Kbps
B_h	2000Kbps
α	between (1.5, 1.8)
Link	Ethernet

Table 6.12: Values of the system parameters.

The results are shown in Figure 6.15. The performance of the 2-Step solution is much better than the other two, even when mobile hosts migrate from a place to another very fast. Obviously, the wide bandwidth is the main factor for our solution to outperform the others. In comparison with the previous simulation, the performance of 2-Step increases about 50% if the mean stay time is one second. However, MDL and IMHP remain unchanged.

Simulation of a high speed network with moderate RTT

In order to investigate the effectiveness of moving area, we adopt a moderate RTT delay time, but use a fast communication link like the previous simulation. All values of the system parameters are listed in Table 6.13.

In Figure 6.16, we find that the performance of our 2-Step solution is largely affected by RTT when mobile hosts move fast. In addition, comparing with the previous experiment, the mean performance of all solutions is affected by bandwidths. A larger bandwidth may cause worse performance which means that more bandwidth may be wasted. When mobile hosts move fast ($\overline{T_w}$ is about 1.2 seconds), our solution is even with the others. It is expected since a worse performance of 2-Step occurs, if

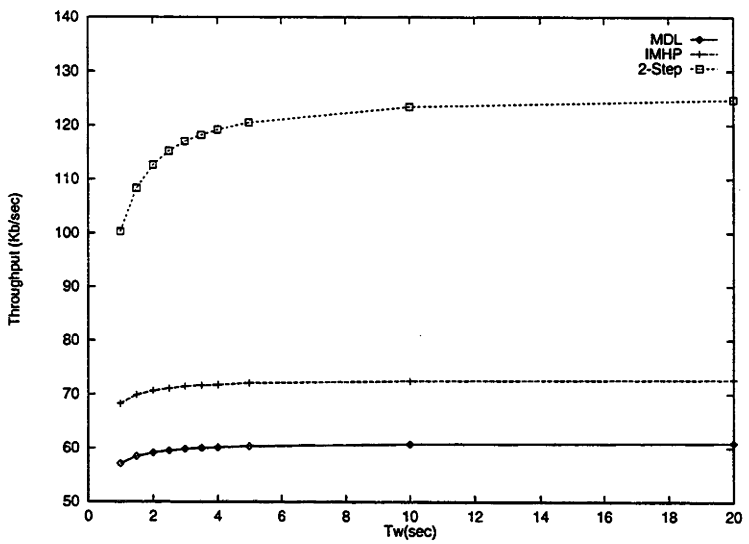


Figure 6.15: A high speed network with a low RTT.

Parameter	Value
V^{App}	2 Gbyte
RTT_1	80ms
RTT_{3l}	150ms
RTT_{3h}	350ms
B_l	800Kbps
B_h	1500Kbps
α	between (1.5, 1.8)
Link	Ethernet

Table 6.13: Values of the system parameters.

mobile hosts move fast.

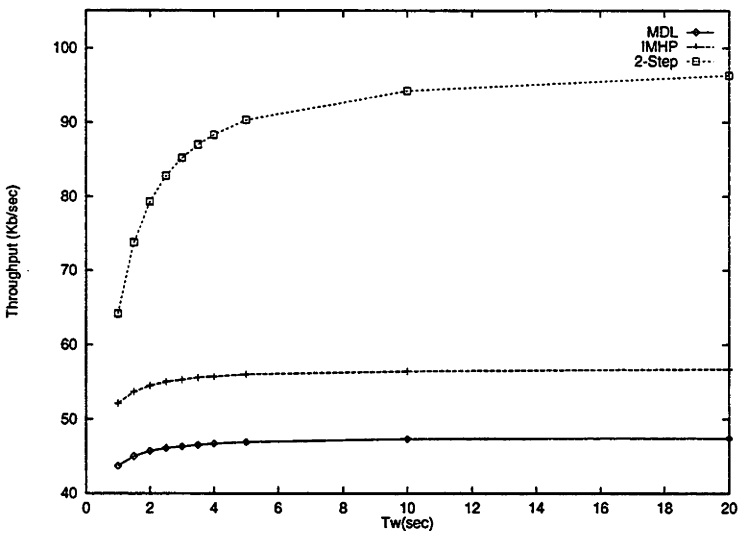


Figure 6.16: A high speed network with a moderate RTT.

In summary, our two-step solution outperforms the other two in both wide area and local network environments. Only when the mobile system moves very fast, the two-step solution can be worse than IMHP or MDL. However that might be a rare case. The detour and bandwidth factors are the other two main factors affecting the performance, but seem to be less important.

Communication costs

In this simulation, the communication costs are investigated based on equation 6.8 where c_1' is chosen to be 1.

First, we investigate the relationship between communication cost and data volume. The environment is a 128K HDLC link and all values of the parameters are given in Table 6.14. The results are given in Figure 6.17. The communication costs of all the three solutions are a linear function to data volume. That means even in a very fast moving environment, the reconnection traffic is only a very small portion and does not affect the performance very much in terms of cost.

Second, we investigate the relationship between moving speed and the communication cost. The environment is described in Table 6.15, and results are given in Figure

Parameter	Value
$\overline{T_w}$	10sec
RTT_1	100ms
RTT_{3l}	150ms
RTT_{3h}	250ms
B_l, B_h	128Kbps
α	between (1.5, 1.8)
Link	HDLC

Table 6.14: Values of the system parameters.

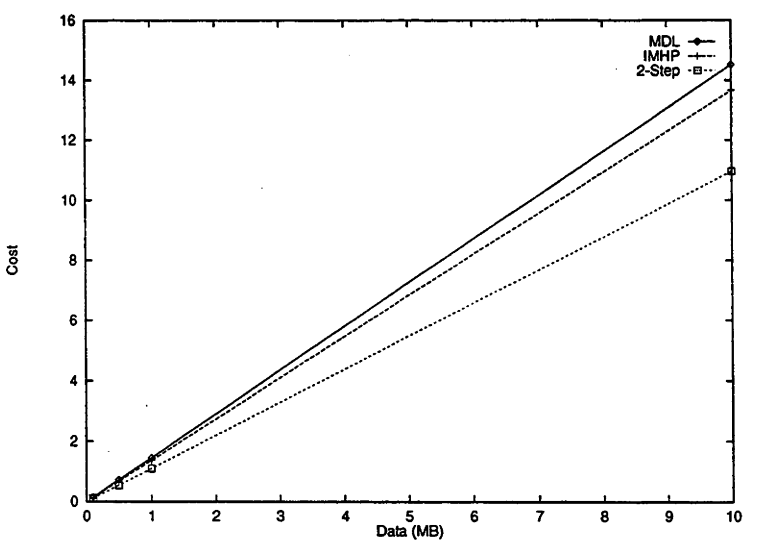


Figure 6.17: Costs v.s. amount of data.

6.18. We can find that the communication cost of MDL and IMHP remain constant, but 2-Step will be affected by moving speed. The faster the MH moves, the worse the system performance is. However, only when mobile hosts move fast, the communication cost will be raised by frequent reconnection. The cost will become stable after the mean working time is larger than one second. Therefore, the communication cost of 2-Step is much less than that of the other two systems and is very stable in most cases.

Parameter	Value
V^{App}	10M bytes
$\overline{T_w}$	10 sec
RTT_1	100ms
RTT_{3l}	150ms
RTT_{3h}	250ms
B_l, B_h	128Kbps
α	between (1.1, 1.3)
Link	HDLC

Table 6.15: Values of the system parameters.

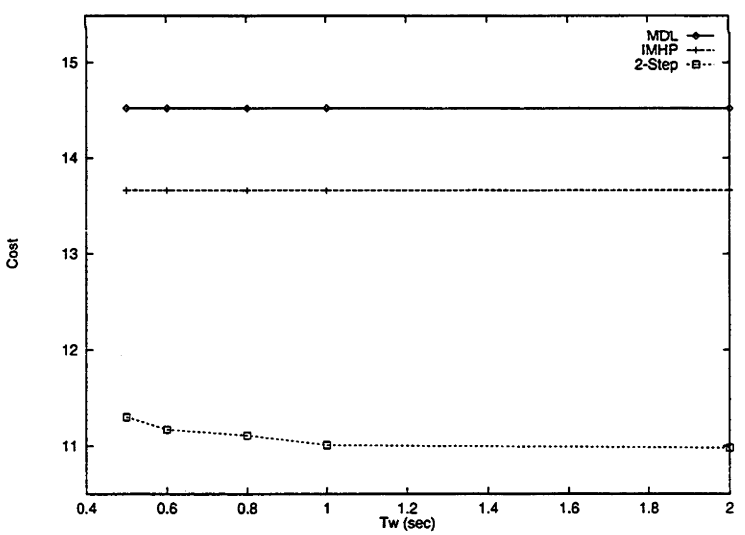


Figure 6.18: Costs v.s. $\overline{T_w}$.

Finally, we investigate different costs when different Link Layer protocols are used. All values of the parameters are listed in Table 6.16. A 10MB data will be transfered on a 128K link, all three Link Layer protocols are simulated. The Table

6.17 shows the results. Again, more efficient encapsulation can save communication costs, and 2-Step outperforms the other two in all cases.

Parameter	Value
V^{App}	10M bytes
$\overline{T_w}$	20 sec
RTT_l	100ms
RTT_{3l}	150ms
RTT_{3h}	250ms
B_l, B_h	128Kbps
α	between (1.1, 1.3)

Table 6.16: Values of the system parameters.

System	PPP	HDLC	Ethernet
MDL	17.80	14.52	19.32
IMHP	12.00	13.66	16.21
2-Step	11.93	10.96	12.34

Table 6.17: Costs v.s. the link Protocols.

6.5 Conclusion

As a mobile solution supported at the highest layer, our two-steps solution provides a seamless mobile TCP communication environment. The results of our simulations indicate that in most cases the two-steps solution outperforms MDL and IMHP in terms of throughputs. The two-step solution also achieves high compatibility, since it doesn't impose any changes on the TCP/IP layers. In this paper, we show that the throughput can be affected by the behaviours of communication protocols. Avoiding triangle routing can tremendously save network traffic. Although two-step solution seems to have slightly low utilisation of communication channels, its optimal routing without extra protocol overhead can obtain much lower transmission delay for individual packets so as to improve the whole system performance.

Appendix: statistics of the Ethernet packet lengths

In Table 6.18, the statistics of a 3COM Switch 1000 switching hub is given. The table shows the distributions of packet lengths based on different sections. Note that the minimum length of an Ethernet packet is 60, 62 is used as the average length of category "less 63".

	less 63	64 - 127	128 - 255	256 - 511	512 - 1023	1024 - 1518	mean length
average	62	95	191	383	767	1271	
Port 1	54%	19%	8%	5%	4%	10%	243.7
Port 2	54%	0%	0%	45%	0%	0%	205.8
Port 3	46%	26%	3%	5%	3%	17%	317.2
Port 4	50%	15%	5%	4%	6%	22%	193.9
Port 5	48%	18%	6%	4%	5%	19%	353.5
Port 6	46%	22%	3%	4%	5%	19%	363.0
Port 7	49%	15%	5%	3%	8%	19%	368.5
Port 8	48%	26%	9%	3%	4%	10%	240.9

Table 6.18: Statistics of the packet lengths.

Chapter 7

Mobile File Filtering

7.1 Introduction

The marriage of portable computer systems and mobile communication devices permit increased number of users to employ mobile systems to perform their daily work. One of the key requirements of this new computing environment is access to critical data regardless of user location. Truly ubiquitous computing requires that programs on mobile computers be able to process work continuously much as they do in a LAN/WAN environment. Appropriate file system support for mobile workers is essential if collaborative workgroup activity is to continue unimpeded. However, mobile file systems have to be able to work under a variety of network conditions ranging from reliable high speed fibre-optic links to the intermittent low bandwidth links associated with mobile phones in marginal service areas. Therefore, mobile file systems are required to provide solutions to problems arising from weak data links and intermittent connectivity [41], namely, involuntary and voluntary disconnections. Involuntary disconnection is caused by communication link failure, whereas voluntary disconnection results from nomadic users switching mobile systems into a partially or fully disconnected communication mode [12] when network performance is so poor that file access delays are unacceptable. Consequently, it becomes important to manage shared files in a way which makes best use of the limited bandwidths of wireless networks and reduces the probability of voluntary disconnection by allowing work to continue using local copies of shared files which are subsequently brought into a consistent state.

Data replication is a basic technique widely used in distributed and mobile file sys-

tems to provide high data availability and improve system performance [21, 28, 71]. If data are replicated on mobile systems, file access requests can be processed locally so as to avoid communications over networks. However, file systems have to manage data consistency if data is replicated across multiple sites. Research work on data replication on distributed/mobile file systems focuses on data consistency. In this paper, we restrict ourselves to the issue of effective utilisation of bandwidths of wireless networks. This work is motivated by the following two observations. First, a replicated workgroup file system will be shared by multiple users but will typically have only a single user for each replica on a personal mobile computer. This differs from the situation where multiple users share the file system on a workgroup file server. Second, the communication cost varies according to the volume of volatile data replicated at multiple sites. Therefore, minimising the data replicated on a mobile computer for a single mobile user is a critical step towards making best use of wireless network bandwidths. Within such an approach, we allow mobile users to define flexible "filters" which specify the set of data files under replication control necessary to support their work while away from the office. Through filtering, we provide a minimal set of shared files for mobile users to carry out nominated common tasks on their mobile computers. It is worth noting that this minimal set can include subtrees of files from multiple volumes and will usually contain only a small subset of the files within each included directory hierarchy. This minimal subtree shares the same name space with the file system from which it is replicated. In addition, consistency will be managed between the minimal subtree on a mobile computer and the main central file system. The filtering mechanisms we propose here lead to *user-level dynamic partitions*, which differ from the system-level static partitions [27, 64, 11] employed in existing distributed file systems.

7.2 Data Replication

Previous studies of distributed file systems [65, 11, 11, 19, 30, 64, 20, 31, 13, 21] have shown that data replication can improve data availability and system performance. In such systems, information about the location and content of each replica is avail-

able to all participating sites. Data in replicas is shared by all the participating sites and is under consistency control. Two units of replication have been proposed: *file* replication in Roe [7] and RNFS [26] and *volume* replication in AFS [27], Coda [64] and Ficus [11]. Since file-based replication requires a system to maintain a large database that keeps details of all replicated individual files, most existing distributed/mobile file systems adopt volume replication using a system-level static partition approach. By system-level static partition, we mean that a system designer or an administrator organises directories and files into non-overlapping volumes and then merges them to form a seamless unique file hierarchy. Users can then work on these volumes at different sites. Since a volume is a whole subtree of a file system, data replication is forced to follow all-or-nothing semantics — to either include all files in a volume or none when a replica of a volume is created on a mobile system. In this approach, such a system-level static partition is needed to manage file sharing among multiple users of workstations connected on high speed networks. However, as a result of the two observations we made above, a policy of system-level static partitioning can cause unnecessary data transfers and consistency management overheads for mobile users, leading to performance degradation with wireless networks.

Coda [65] and Ficus [21] are file systems which maintain data replicas on mobile systems and preserve data consistency even with low bandwidths and intermittent network connections. The Ficus file system employs a single level data replication in which all replicated files and directories are organised into volumes. A volume is a subtree which contains all files rooted at the root of such a subtree, and is the unit distributed across hosts [20]. Replicas of a volume have peer-to-peer relationships to maintain *one copy availability* sharing semantics, which means that clients can access any available replica. The update propagation mechanism ensures that all replicas of the same volume will converge to the same status. Volumes can be merged to form a unique file hierarchy with the notion of a *graft point*, which is a mount point of a volume and contains information pertaining to the location of all replicas of that volume. In order to provide for mobile users, volumes that contain the files under request have to be replicated on the mobile system [21].

As a derivative of AFS, Coda adopts a two-level data replication scheme. Similar

to Ficus, a unique file hierarchy is composed of a set of volumes, each of which is replicated among server systems located on a WAN. A volume storage group (VSG) is a group of server systems that maintains all replicas of a single volume. Such information is managed by the system control machine. Along with volume replications among servers, a local cache at each mobile computer is also employed [13, 63].

In an exception to the above, the *Rumor* project [61] at UCLA adopts a user-level static partition mechanism which aims to provide a user-level data replication and therefore assists mobile users to manage their replicated data on different systems. The data replication is, however, predetermined as a volume which is a subtree containing all the files under the root of that subtree.

7.3 Motivation

The following examples illustrate the considerations underlying our proposals for mobile file filtering.

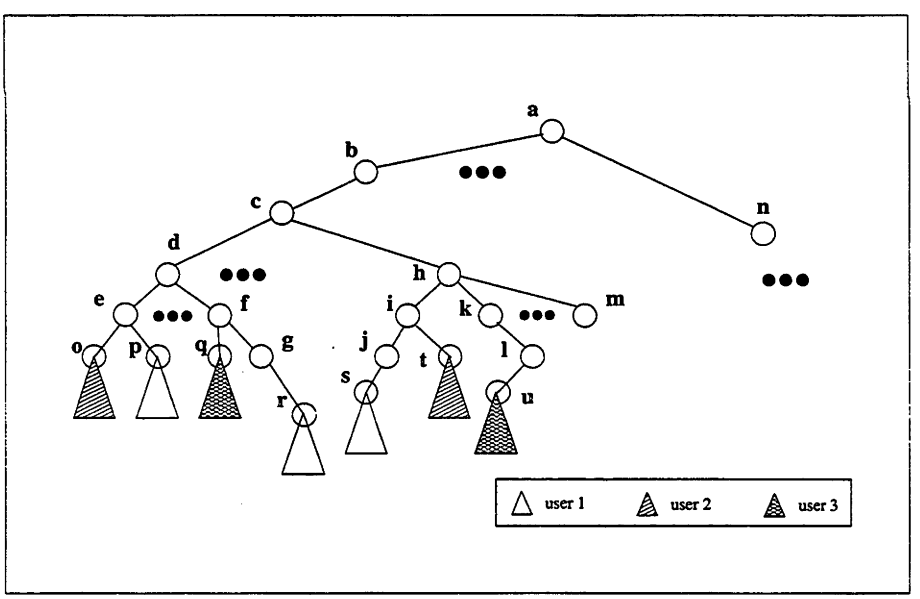


Figure 7.1: Shared Single Volume

Example 1 *A volume is schematically depicted in Figure 7.1. Several mobile users work exclusively on some small subtrees in a volume which is shared by other users on workstations connected to a LAN/WAN.*

Suppose that a system-level static volume is adopted. In the first example, each of the mobile users has to replicate the whole volume on their mobile computer. Further suppose that there are n mobile users. Then the number of replicas of this volume is $n+m$ if m replicas are used across the WAN. Therefore, data consistency has to be managed among the $n+m$ replicas, and in particular among the n replicas in the wireless network. When a user modifies a file in either a mobile computer or a workstation, messages then have to be exchanged among the $n+m$ replicas. Even in the case where mobile users are interested in only disjoint subsets of files, messages still have to be exchanged. As is illustrated in Figure 7.1, where three mobile users access disjoint subsets of files, the file systems still have to exchange messages to maintain consistent complete volumes on individual systems, given that these files all lie within a single volume.

Example 2 *A mobile user has to access several files in two volumes which are shared by users on workstations connected to a WAN.*

In the second example, although the user needs to access only a few of the files from these volumes, each volume needs to be fully replicated on the mobile system. Consequently, the number of messages exchanged for consistency management of replicated files increases proportionally to the total volume and activity of volatile data, rather than in accordance with the volatile data relevant to their needs.

In the above two examples, one might argue that messaging overheads associated with consistency management can be managed by changing the volume sizes. In practice, the volume size and content is largely predetermined. Based on the system-level static partition, there is only one possible solution — that a system administrator divides a disk's storage into several volumes. A user can then have exclusive access to a single volume and organise their data in this volume. However, the optimal sizes of such volumes cannot be determined beforehand. If the volume is large, users have to replicate unnecessary files in that volume. If the volume is small, users have to replicate multiple small volumes. Even though the volume size can possibly be chosen to be close to optimal at a particular point in time, it still is unlikely to satisfy the user's requirements in the long term. Moreover, the benefits of a collaborative workgroup

environment arise as a consequence of users sharing data with other members of their workgroup.

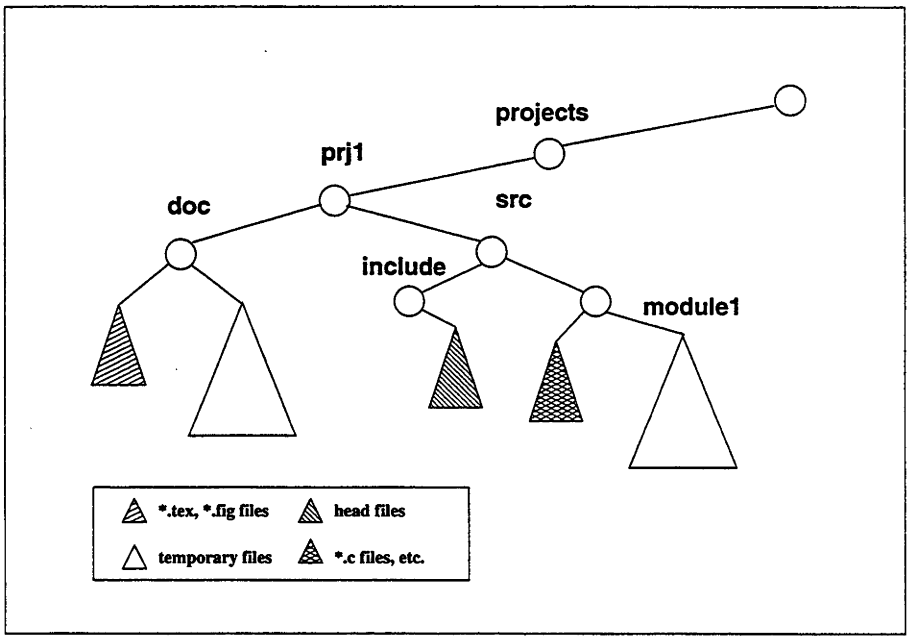


Figure 7.2: Software development project

Example 3 *As shown in Figure 7.2, a group of users collaborates on a software development project. Each member in the project works within their own subdirectory for editing and compiling, and draws on material in other users' subdirectories. Some members of the project team work with mobile computers.*

In this example, we assume that all source files, header files and documents remain consistent. Suppose that each member uses a C compiler to compile files and uses LaTeX to produce accompanying documentation. The C compiler, for example `cc`, generates a number of temporary files during compilation and finally generates binary files and possibly an executable file. The question here is whether or not we need to manage consistency for all such files — including temporary files, binary files and executable files — in a mobile environment. To explain further, suppose a user has several C source files and header files to compile. The total size of these files is B bytes. After running the `make` facility, there will be several object files and an executable file newly created. In addition, many temporary files will be created and deleted during compilation — we assume that the total size of all such files is S bytes.

The proportion of B/S can be very small, as little as 20-30%. This implies that 70% of data transfers between the mobile computer on which a user runs `make` and other replicated file systems can be avoided. If there are n replicas, then the potential saving here is $0.7 \times n \times S$. In addition, `core` dump files used during debugging do not need consistency management. The situation for use of LaTeX is similar: only new or volatile files required as LaTeX inputs need consistency control.

We introduce a user-level replication unit, called an *adaptive volume,* which is a minimal subtree containing only the minimum number of directories from the root and the data files users want to replicate. Such an adaptive volume, which can be built on top of existing system-level partition volumes, can be regarded as a filtering since it selects the files a user has to replicate. We call the first component of filtering *inclusive* filtering. The specification of such filtering for the first example is given below. We show a simple specification for a user called `user1` in Figure 7.1. This specification consists of three pairs, tentatively written as `((/a/b/c/d/e/p *) (/a/b/c/d/f/g/r *) (/a/b/c/h/i/j/s *))`. Each pair specifies a path from the root and files under that path. The wildcard `*` specifies all the files under a given path. Given the above specification, a minimal subtree is created as an adaptive volume on `user1`'s mobile computer. As can be seen, the minimal subtree contains only the minimum set of tree structures and files. Figure 7.3 shows three adaptive volumes on three different mobile computers for the three users.

The major advantage of an adaptive volume is that users have the flexibility to replicate a minimal set of files with the structure of a minimal subtree. As a result, file systems can use this information to eliminate unnecessary communication which would otherwise be required in order to maintain consistency for files that are not relevant to the particular user.

In addition to inclusive filtering, exclusive filtering is also proposed which simply indicates the files users don't want to be managed under consistency control. This implies (i) that the mobile user doesn't need to inform other sites of any updates on the files given in the exclusive filtering and (ii) that the user doesn't need to be aware of any updates to these files made at other sites. A simple specification of exclusive filtering is given by `(*.o)`. This filter will exclude from consistency control all files

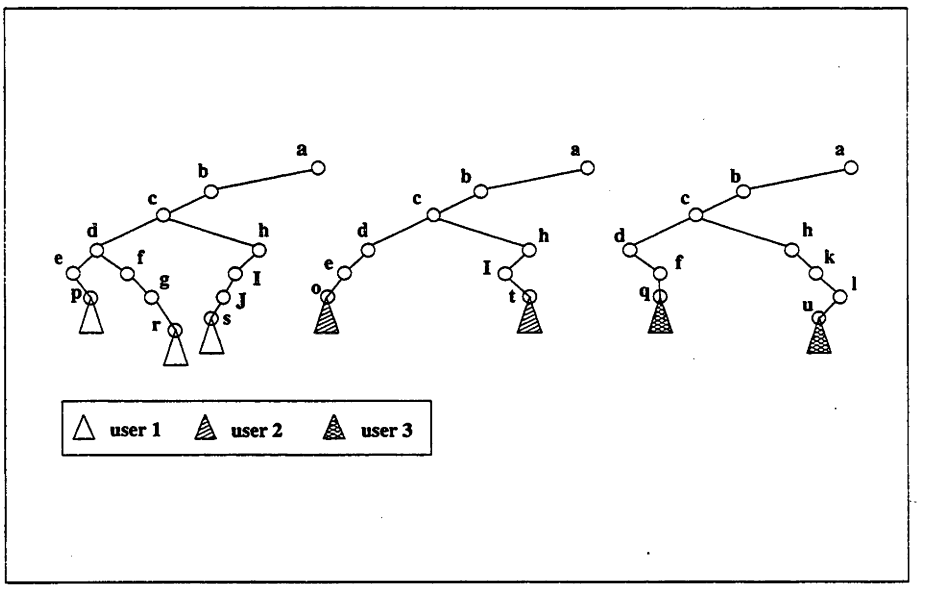


Figure 7.3: Inclusive Filtering

with a name matching * . o under certain directories.

Our proposed filtering has the following advantages. First, from the users' viewpoint, they can replicate the minimal set of files to process their tasks on a mobile computer. The minimal file set can be specified with filtering mechanisms without having to be aware of the system-level partitions. Since it is a filtering, users will share the same name space used in the original file systems. The file system service will be responsible for maintaining consistency only for files whose changing content is relevant to the mobile users. Users can work on one adaptive volume and then switch to a second adaptive volume if needed. As an extreme case, a user can specify an adaptive volume which includes the whole file system. Second, from the system's point of view, the filtering mechanisms enforce locality by which the file system servers can understand which files should be kept in consistency. These filtering mechanisms can save a large amount of data transfer and therefore improve utilisation of network bandwidths.

It is worth noting that the adaptive volumes proposed here differ from caching in several important respects. First, a cache stores data at the level of files and is based on past accesses. Therefore, mobile users cannot access files they haven't yet accessed on a mobile computer if the network is disconnected. Second, caches are designed

so that they do not need to know anything about the semantics of files under their control. The situation is similar to data prefetching using either a profile [63] or a predictive mechanism [23]. Any changes in data in a local cache have to be passed to and processed at the server. On the contrary, adaptive volumes replicate all necessary files including both data and directory files. The filtering mechanism eliminates unnecessary data transfers.

7.4 Filtering

In this section, we first give definitions of a file system hierarchy. For simplicity, we assume that a file system is a rooted tree defined as follows[1].

Definition 1 *A file system is defined as a rooted directed tree $G = (V,E)$ where V is a set of vertices and $E \subseteq V \times V$. V can be divided into two subsets V_d and V_f where $V_d \cap V_f = \phi$ and $V_d \cup V_f = V$. We refer to the vertices in V_d as d-vertices for directories and the vertices in V_f as f-vertices for data files.*

Either d-vertices or f-vertices can be leaf vertices in G. Only d-vertices can be non-leaf vertices in G. In the following, we use v_r to refer to the root vertex of G and call v_r the root of the file system. In Unix file systems, for example, a file name can have a suffix such as `.c` for a `C` source file and `.o` for a binary file. A file descriptor keeps information for the corresponding file. For example, a file can have a type which can be categorised as b, c, d, l, p or f for block special file, character special file, directory, symbolic link, fifo (named pipe) or plain file, respectively. A file belongs to the user who is the owner of the file and also belongs to a group. In addition, a file has permission flags to control read/write/execution permissions.

Definition 2 *A filter, f, is a set of pairs (n,a) where n is a file name using a simplified regular expression and a is a list of options specifying file attributes kept in file descriptors.*

A specification of inclusive filtering is given below to include only LaTeX text files, figure files and style files belonging to a user named `mark`.

```
*.{tex,sty,fig} -user mark
```

[1] We will extend this to DAG structures in future work.

Similarly, a three line specification of exclusive filtering is given below to exclude unnecessary files.

```
core
*.o
* -type f -perm 0111
```

The first line excludes the dump file `core`, the second line excludes all binary files with a suffix of `*.o` and the last line excludes all executable files other than directory files.

Definition 3 *A subtree, S, is specified with a triplet (p, f_i, f_e) where p is an absolute path from the root, v_r. f_i and f_e are inclusive and exclusive filters, respectively, according to Definition 2.*

Here, a path p indicates the root of a subtree. The filter f_i specifies the files to be included in the subtree, whereas the filter f_e specifies files to be excluded. The files to be included in the subtree are those files satisfying filter f_i but not filter f_e. It is worth noting that if a directory satisfies filter f_i but not f_e, all files under that directory will automatically be included in the subtree. Also, filter f_e applies to all the files under the path p.

Definition 4 *An adaptive volume, $\overline{V}$, is defined using a set of subtrees, $\{s_1, s_2, ..., s_n\}$. $\overline{V}$ is a minimal subtree in the sense that it cannot be a connected tree including all the required files if any vertex is removed from this subtree. The graft point is defined as the common path for all paths p_i given in a subtree $s_i = (p_i, f_i, f_e)$.*

Several comments can be made with respect to adaptive volumes:

- Since an absolute path is used, an adaptive volume is rooted at the root of the file system. The directory structure of a volume constructs a basic framework to provide a limited but efficient working space for mobile users.
- The graft point is a common path which is not allowed to be removed if any adaptive volume is in use.
- Changes to a path p_i of $s_i = (p_i, f_i, f_e)$ in an adaptive volume should be managed under consistency control. If a directory in the path p_i changes its name, it has to

inform all replicas that use this path. If a directory in the path p_i is removed, it also has to inform all replicas. An exception is that adding any new directories in p_i does not necessarily require all replicas to be informed.

- Changes to any file satisfying f_i but not f_e of $s_i = (p_i, f_i, f_e)$ in an adaptive volume should be managed under consistency control. This applies to both data files and directory files.

7.5 Implementation Issues

7.5.1 Client-Server Model

The goal of an adaptive volume is to allow mobile users to make a local copy of a working file set for a mobile computer in order to be able to continue their work while away from their office. Different users can define their own adaptive volumes and use them concurrently.

Several servers in the wired networks will function as servers for adaptive volume replicas. If there are multiple replicas of the same file system in the wired networks, these servers will cooperatively manage consistency using existing approaches. A replica on a mobile computer needs to communicate only with a server in the wired networks and not with other mobile computers.

7.5.2 Initialisation

When first setting up an adaptive volume for any system, a user can specify an adaptive volume for the files they want to work on by defining inclusive and exclusive filters. Then, a minimal subtree will be replicated on the mobile computer as requested. This initialisation process, which replicates all the directories and the files on demand, is quite straightforward. While some entries in directory files may not be necessary, for simplicity we will transfer entire directories to the mobile computer and remove unnecessary entries subsequently.

7.5.3 Runtime Support

To keep the server system simple, the graft point of any adaptive volume cannot be changed after replicas of this volume are constructed. An adaptive volume defines the scope of files and directories. Following this scope, both client and server will work cooperatively to keep the files within this scope consistent. Interestingly, a mobile user can create any files locally under the adaptive volumes. There are four kinds of changes possible: file addition/deletion and directory addition/deletion, on both server and on client respectively. Server and client systems have to abide by the following rules to maintain adaptive volumes in the same state at both sites:

- The client and the server cannot delete/change the directories at graft points.
- Users can add a new directory in any path p_i of $s_i = (p_i, f_i, f_e)$ in an adaptive volume on either client or server. This action requires only local operations.
- Users can add a directory/file in an adaptive volume. If the newly added directory/file satisfies the inclusive adaptive filter and does not satisfy the exclusive adaptive filter, this change has to be passed to the other sites. Otherwise, this newly added directory/file can be handled locally.
- Users can delete a directory/file on either client or server. This operation will remove the directory/file locally, and will inform the other sites if this directory/file satisfies the inclusive adaptive filter and does not satisfy the corresponding exclusive adaptive filter. The other sites will remove the directory/file accordingly.

7.5.4 Sharing Semantics

An adaptive volume service system is responsible for maintaining consistency among replicas of shared files, thus keeps the sharing semantics. First, we address issues that are introduced by use of adaptive volumes. In existing distributed file systems, a data file in any two replicas must be the same. This part of the semantics is also supported in our system. In order to handle directories, existing distributed file systems require

that two replicas have the same number of directory entries and have the same attributes (except for `inode` number) for every entry. However, our filter mechanisms exclude some directory entries. Therefore, two replicas are not identical in terms of directories in our system; the term "same" implies that the common portion of entries defined by the adaptive volume is the same. For example, a directory contains three file entries: `a`, `b` and `c`. If only `a` is included in user A's adaptive volume, any changes to the files under this directory except `a` are outside user A's interest. Adding a file `d` at the other sites will not affect the consistency under this directory that user A is accessing. Updating a directory will have impact on users whose filters specify that the changes are relevant to them.

Strong consistency requires strong network connectivity. As a result of the limited bandwidths available in mobile networks, the trade-offs between consistency and network bandwidths have to be reexamined. For example, [71] proposes a variable consistency scheme for mobile environments, and [12] discusses different ways to keep consistency by efficiently exploiting different network connectivities. In our system, we adopt two different sharing semantics. When a satisfactory network connection is available, the strict CTO [8] semantics is supported. If the client system disconnects, it is impossible to keep CTO semantics with the server. Therefore, all file access operations will be executed locally in an uncommitted status. If the client system reconnects to the server, the adaptive volume will immediately be synchronised with the server using existing approaches to resolve conflicts [65, 18, 24].

In the following, we address only the situation where a client can connect to a server in the wired LAN/WAN network. In our system, we use different strategies to keep data files and directories consistent. We adopt CTO semantics [8] to keep data files consistent in adaptive volumes. The CTO semantics is also used in NFS [62] and Sprite [31]. Following CTO semantics, consistency checking will be triggered only by *read* and *write* operations on data files. For directories, we assume that the structure of the file hierarchy is changed less often than with data files and employ the callback mechanism used in Coda [63, 28].

Parameters	Description	Values
thinkTime	thinking interval time	60
workTime	working interval time	10
openFiles	number of open files	5
ctrlMsg	length of control message	1 unit (200 bytes)
updtMsg	length of update message	100 units (20K)
wfsSize	size of working file set	30 60 100
writProb	percentage of write access	20%
updtTime1	system updating interval time 1	180
updtTime2	system updating interval time 2	60

Table 7.1: Simulation Parameters

7.6 Performance Study

7.6.1 Simulation

In this section we report on a simulation study of the filtering mechanisms, based on the Unix operating system, file system and applications. The parameters and default values used in the performance study are given in Table 7.1. An event-driven simulator was developed using the *C++SIM* simulation package [25]. While the figures given below concerning observed percentages and average sizes of temporary files will vary according to the applications used, most results of the study are relevant across platforms and operating systems.

7.6.1.1 Model

Our example mobile environment comprises mobile computers in the wireless network and workstations in the wired networks. In the simulation experiment, we consider only file sharing in a single system-level volume. There are n replicas in the wireless network, and m replicas in the wired network for the system-level volume. On top of the system-level volume, a mobile user can have a single adaptive volume as a replica on their mobile computer. The mobile user will share files with other users. The client file system on mobile computers in the wireless network and file servers on workstations in the wired network are denoted as C_m and S_w, respectively. The system C_{m_i} needs to communicate with only one server S_{w_j}. The server S_{w_j} will communicate

with all other S_{w_i} if any updates occur.

The mobile user operates with a set of working files. We use the average size of working file set and the average size of files given in [22]. Combining the data for productivity and programming environments, we use 30, 60 and 100 Kbytes as the sizes of working files including data files and directories. This is controlled by the parameter *wfsSize*. The average length of data files is 27 Kbytes. We assume that in a working file set 30% are directory files and the size of a directory file is 128 bytes. On average, the size of a file is approximately 20 Kbytes. The working pattern is modelled using two alternating periods, namely, a working period, *workTime*, and a thinking period, *thinkTime*. In the working period, a user opens several files for read or write access. According to [32], 90% of files will be closed in 10 seconds, so we use 10 seconds as the average for the working period. The server S_{w_j} will periodically update files since multiple users work on the same volume in the wired network. Such updating can affect the mobile user's working file set. The average interval time for the server S_{w_j} to update a file in the working file set is controlled by *updtTime1*. The average interval time for the server S_{w_j} to update a file outside the working file set is controlled by *updtTime2*. The write access rate of system C_{m_i} is governed by *writProb*, which is assigned a value of 20%. The results of our simulation indicate that the parameters *updtTime1*, *updtTime2* and *writProb* are reasonable. For example, file write update conflicts occur about 0.07% of the time, which is close to the figure reported in [22].

The CTO sharing semantics is chosen in our simulation. At the S_{w_j} site, when a file/directory is updated in a volume, the server S_{w_j} will send a control message to all the C_{m_i} that communicate with this S_{w_j}. If a system-level volume is used, all updates against directories have to be sent to C_{m_i}. If an adaptive volume is used, it is not necessary to send all such messages to C_{m_i}. Upon receiving such a control message, the mobile system C_{m_i} will send back a message to confirm that it has performed the necessary local updates. At the C_{m_i} site, when opening a file, the C_{m_i} first checks if it is updated. If any updates occur, the local copy will be updated. Before closing a file, the C_{m_i} will send back a new copy to the server S_{w_j} if the file is locally updated. We do not consider update conflicts because this likelihood is small and resolution of conflicts

will not affect our filtering gain. We also assume that the protocol control message is 200 bytes long. Two methods are simulated to update files. One is data-based, in which a whole file will be transferred over the network. The other is a log-based, in which only audit log information, assumed to be 10% of a file, is transferred. In other words, 10 and 100 units of data are transferred over the network respectively.

7.6.1.2 Experimental results

In this simulation, we analyse the network load between two servers, namely, C_{m_1} and S_{w_1}. Updates on S_{w_1} also imply update propagation to other servers S_{w_i} for any $i \neq 1$.

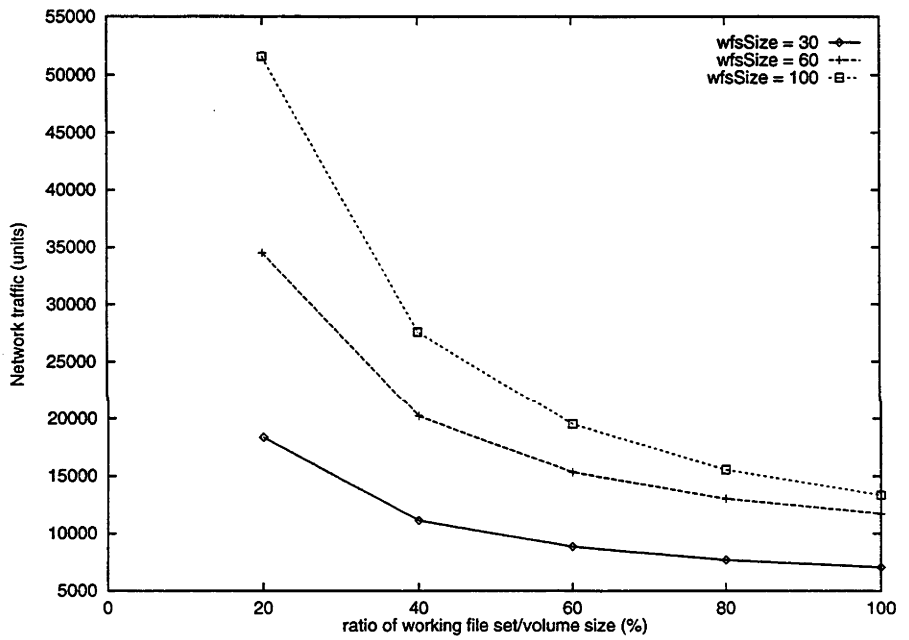


Figure 7.4: network traffic (log-based updating)

In Figures 7.4, 7.5 and 7.6, the abscissa is the ratio between the size of the working file set and the size of the volume to be replicated. A figure of 100% means that the adaptive volume contains only those files a user really needs. In this case, the size of the adaptive volume is the smallest possible. A value of 40%, for example, implies that 60% of the files in the adaptive volume are not files that a mobile user requires to be under consistency control (because they are not accessed, not volatile, or not relevant to other users).

In Figures 7.4 and 7.5, there are three curves plotted for wftSizes of 30, 60 and 100.

The plotted values decrease as the ratio increases. The values for wfsSize = 30 decrease faster than for the other two cases. This indicates that a large volume will cause a large amount of network traffic in order to keep files consistent. In these two figures, the value of 100% on the abscissa is the point that the adaptive volume is smallest. At this point, network traffic reaches its minimum value on each plotted curve. In Figure 7.4, all three curves decrease more rapidly than those in Figure 7.5. This means that the saving via filtering is greater if log-based updating is used. Since most of the network traffic is caused by file updating, our filtering mechanisms reduce the control messages, and thereby reduces unnecessary network traffic otherwise incurred when updating files outside the working file set.

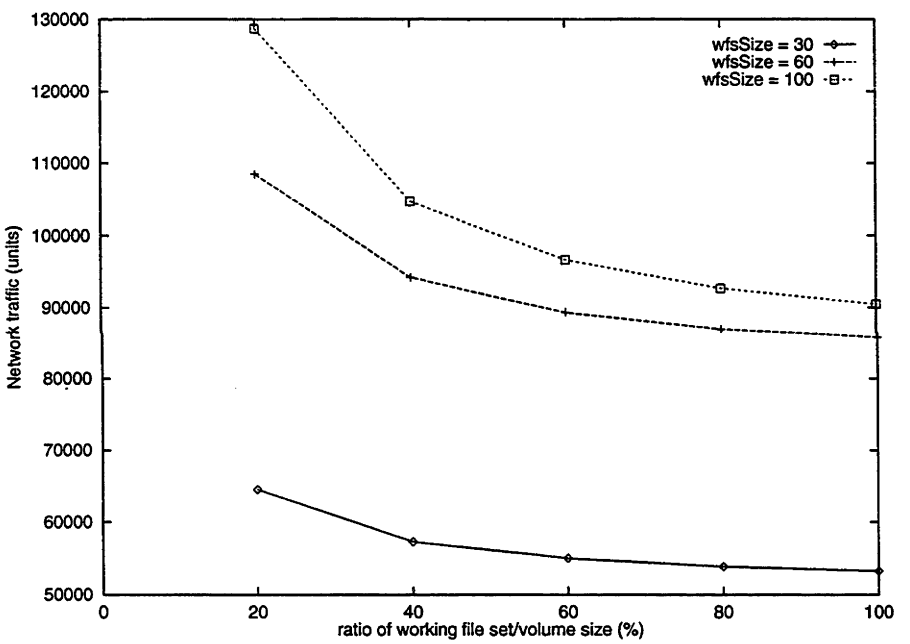


Figure 7.5: network traffic (data-based updating)

In Figure 7.6, the potential performance gains are given for both data-based and log-based updating. With the log-based approach, the potential performance improvement is large if the working file set is correspondingly large. A large working file set with a ratio smaller than 100% implies that there are unnecessary files being replicated. Our filtering mechanism aims to move this ratio as close to 100% as possible. The zero potential performance improvement in Figure 7.6 means that this system has reached its optimal situation.

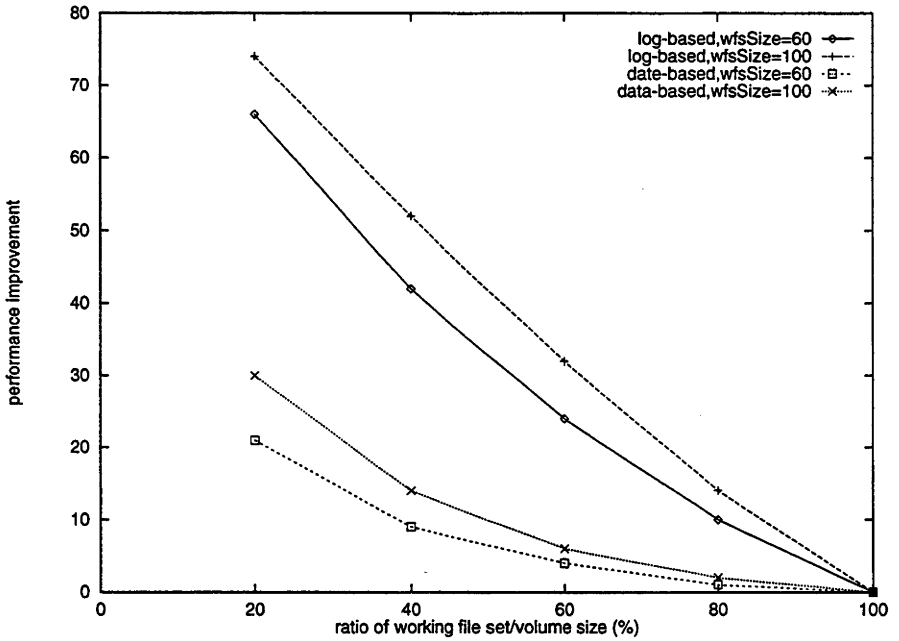


Figure 7.6: Performance improvement

Utilities	Temporary files	Non-temporary files
LaTeX	`*.aux *.log *.dvi *.ps`	
text editor (`emacs` or `vi`)	`*~ #*# *.bak`	`*.tex *.bib`
xfig	`*.fig.bak`	`*.fig *.eps *.ps`
bibtex	`*.blg`	`*.bst *.bib *.bbl`

Table 7.2: Two file groups.

7.6.2 Exclusion of Unnecessary Files

In this section, we report our findings regarding the benefits arising from exclusion of unnecessary files. We choose LaTeX utilities as an initial example since mobile users will frequently work on preparing documents. We assume that a mobile user uses a text editor, such as `emacs`, to edit text files and the LaTeX facility to turn the source text into display form. The text editor and LaTeX facility will create many temporary files that need not be managed under consistency control. These temporary files include backup files, log files and some intermediate output files. In Table 7.2, we list temporary files and non-temporary files for this style of document processing. We include `*.ps` in both groups since users may want to have postscript files exported by utili-

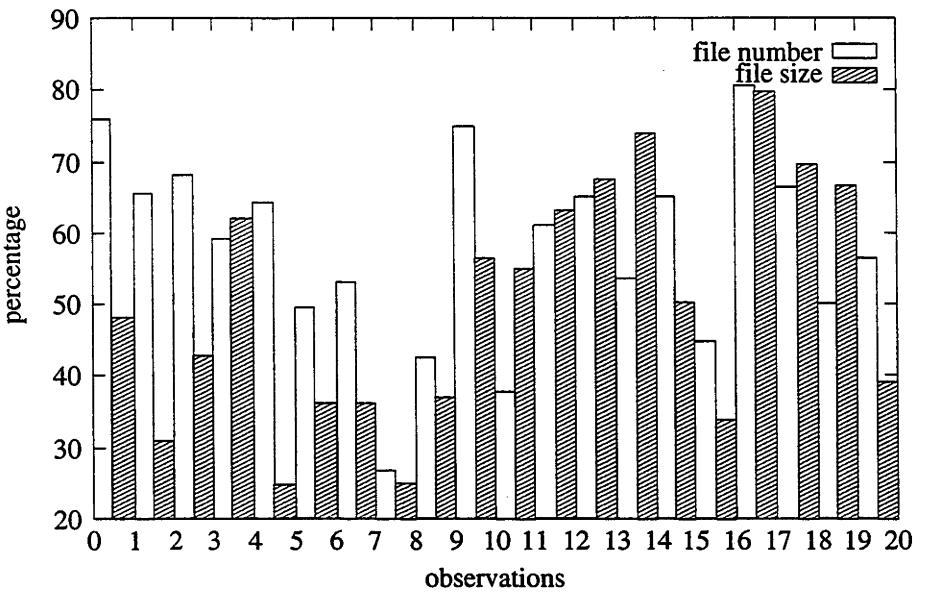


Figure 7.7: Temporary files

ties such as `xfig` managed consistently between the client and the server. However, the final postscript file need not be managed under consistency control since it can be reproduced at either client or server site.

We have investigated many users' document directories in Unix systems and found that the number and the size of temporary files cannot be ignored. Figure 7.7 shows some results of our investigation for 20 such document directories. The average number and the average size of temporary files are 49.8% and 56.65%, respectively. Therefore, our adaptive volume can reduce up to 50% of the total number of files to be replicated and kept consistent. The total saving can be up to 50%.

7.7 Conclusion

We have proposed an adaptive volume for data replication using two filter mechanisms, namely, inclusive and exclusive filters. Our performance study shows that a significant proportion of communication costs associated with replicated data can potentially be avoided. Our adaptive volume can be simply extended to support file facilities in other distributed/mobile environments.

Chapter 8

Conclusion

8.1 Conclusion

8.2 Summary

To discuss mobile communication services, we differentiate the portable and mobile communications. The mobile communications is continuous and a mobile system can support constant services at all times. Whereas portable communication is intermittent and can not provide services during moving; also it can not guarantee communication connections to be alive when the system moves from place to place.

The major problem with mobile communication in current TCP/IP internet environment stems from the two-tie use of IP address. Firstly, the IP address is used as an identifier. In the application and transport layer IP address is used to define the communication end points so as to name communication entities. Secondly, IP address is employed as an address by network layer and existing routing infrastructure to route IP datagrams. IP address becomes location-sensitive. The main concern with all mobile solutions is how to solve the two-tie use of IP address.

The research has been aimed to provide a more efficient solution to support mobile communications in the TCP/IP internet environment. We have two initial goals. First, the system needs to be compatible to existing TCP/IP to the most possible extension and its stucture must be flexible and easy to be deployed. Second, the performance must be improved in comparision with existing solutions. The first goal permits that this solution can be applied in almost all internet environment, while the second goal is the key to achieve the success of this research.

In this thesis we first investigated the existing mobile TCP/IP communication solutions in the Internet context. To analyse the systems and performances of different mobile TCP/IP solutions, we developed two models — layered model and cost model.

Based on the layered model, we defined three different kinds of layers in terms of the communication functionality in mobile environment: *static, portable* and *mobile* layer. In a mobile host system, the ultimate source and sink of data being transfered over network is application processes or application entities. Therefore, in a mobile system there must be at least one mobile layer beneath the application layer, which can support transparent mobile communication services to upper layers. All layers under the mobile layer must be portable layers. The portability is the basis of mobility. All current solutions aim to provide mobility in a single layer. According to the location of mobile layer, we classify existing solutions into three groups: data link solution, network solution and transport layer solution.

The cost model was proposed for analysis of system performance. First, we abstract the movement of a mobile system as *moving pattern model*, which is a sequence S and consists of one or more pairs of moving time T_m and working time T_w. Given the moving pattern and the relationships of T_m and T_w, we can estimate the throughput, communication cost and propagation cost of location information for any solution.

As the name indicates, our two-step solution provides mobility in two steps. First, portable-IP system supports portability in the network layer. Then, the Mobile Socket Layer (MSL) bridges the communication gaps of a portable system to provide mobile services at socket API to applications. These two steps can be also abstracted as two address mappings to correctly break the two-tie use of IP address in two steps. The first step is to map the addressing-purpose IP address in network layer to identifier-purpose IP address in the network and transport layer(as given in formula 3.1). Then, the second step maps the IP address in network and transport layer to the constant IP address used in application layer(as given in formula 3.2). The constant IP address in application layer permits MSL to provide consistent and continuous communications mobile services in a transparent way. This solution amortises the cost of TCP reconnection and DNS address mapping into consequent direct routing, hence provides a

better system performance compared with other solutions.

We also conducted research in a typical mobile application field: mobile file system. We believe that all existing network file systems were designed for traditional fixed-host environment. We proposed an *adaptive volume* concept to tail the networked file system dedicated for mobile personal use. Users can define what they need and the mobile file system is responsible for keeping a small set of files and directories updated with other systems so as to reduce the network traffic and support disconnection operation.

Our novel two-step mobile TCP/IP solution is based on existing internet standards to expend the static TCP/IP communication mode to portable and then mobile. The pure software implementation promises a flexible implementation structure and easy deployment of this system in the Internet wide. From our performance studies, the two-step solution can provide 30% to 90% performance improvement in terms of throughput and communication cost. This large improvement encourages us to carry out further research on this system solution.

8.3 Future Research

In the near future, MSL can be enhanced to support UDP. Given the UDP is a connectionless protocol, there can be no predefined association between a pair of UDP sockets. Although a UDP socket must be bound to a local address, the address of peer system can be left open until the output system calls are invoked. In other words, the destination address can be different in individual output requests. The address mapping has to be supported in a more dynamic way unlike what we proposed for TCP association.

The second aspect of extending this research is concerned with the mobility support in future TCP/IP structure. Nowadays, IP protocol faces a serious problem: running short of 32-bit addresses because the Internet expands in a so fast way that original designers could not expect. Therefore, IPng or IPv6 is proposed to provide 128-bit large IP address space to satisfy the requirement for a long term. Also, mobility is an important aspect in design of IPng. The basic mechanism of mobility in IPng is bor-

rowed from the IMHP, say, home network and home agent, and care-of address. The mobile host is always reachable to send IP datagrams to its home network. Hence, the triangle routing still exists there.

New and Modified Network/DNS Library Functions

A.1 Gethostbyhaddr()

This function call provides the new DNS mapping from home IP address to current IP address, if any. The source code employs two other functions. The first one is getanswer, which is a standard routine in network library; the other one is res_query, which is included in DNS source code.

```
/* related definition and static variables */
#define T_IPIP   50
static struct hostent   host;
static char *h_addr_ptrs[MAXADDRS + 1];
static struct in_addr host_addr, h_host_addr;


static struct host *
gethostbyhname(const char *addr, int len, int type)
{

        int rc;
        querybuf buf;
        struct hostent *hp;
        char qbuf[256];       /* the maximum length
                                 of domain name */

        /* only support Internet Address */
        if (type != AF_INET)
                return (struct hostent *) NULL);

        /* construct the query buffer */
```

```
        sprintf(qbuf, "%u.%u.%u.%u.IN-ADDR.ARPA",
                ((unsigned)addr[3] & 0xff),
                ((unsigned)addr[2] & 0xff),
                ((unsigned)addr[1] & 0xff),
                ((unsigned)addr[0] & 0xff));

        rc = res_query(qbuf, C_IN, T_IPIP,
                       (char *)&buf, sizeof(buf));

        /* res_query() failed */
        if (n<0)
                return ((struct hostent *) NULL);

        hp = getanswer(&buf, rc, 1);

        /* if no IPIP result returned,
           both home IP and current have same value. */
        if (hp==NULL)  {
                host_addr = (struct in_addr)addr;
                h_host_addr = (struct in_addr) addr;
                hp = host;
        }

        hp->h_addrtype = type;
        hp->h_length = len;
        h_addr_ptrs[0] = (char *)&h_host_addr;
        h_addr_ptrs[1] = (char *)&host_addr;
        hp->h_addr_list = h_addr_ptrs;
        return hp;
}
```

A.2 Getanswer()

This function analyses the DNS query result returned from `res_query()`. Basically, it is responsible for picking up answers and expending DNS abbreviation. The amend of this function is little: just to enable it accept the new type of query result and process it as *PTR* query result. Here we just list modified portion of the source code.

```
static struct hostent *
getanswer(querybuf *answer, int anslen, int iquery)
{

        ...
```

```
        if ( (iquery && type == T_PTR) ||
             (iquery && type == T_IPIP) )  {
                if ((n = dn_expand((u_char *)answer->buf,
                     (u_char *)eom, (u_char *)cp, (u_char *)bp,
                     buflen)) < 0)
                         break;
                cp += n;
                host.h_name = bp;
                return(&host);
        }

        ...
}
```

Appendix B

Simulation Program

B.1 Introduction

To simulate the real world of mobile communication environment, we use four types of compenets to abstract the Internet:

- MHost simulates the mobile host system, which can talk to a fixed host system in either same network or another network. Each MHost has two important properties: the current location and home network. Different solutions use different routing algorithms, to which these two properties are the main parameters.
- LAN simulates the local internet, it has two roles. First, LAN will deal with the local traffics, which have the local LAN as both source and destination address. Second, it also simulates the fixed host,which will communicate with MHost's to complete different types of jobs. There can be many LAN's in a real system and each LAN in turn can host more than one MHost's.
- Backbone is the global Internet, which provides the connections among all LAN's. All traffics which go from one LAN to another will pass through this component. In the simulation system, there is only one Backbone component.
- Link connects LAN to Backbone, which is a two direction channel between a LAN and the Backbone.

Figure B.1 shows the basic structure of the simulated system. From view point of queuing, Backbone is a $M/M/\infty$ queue. The service time of individual jobs is determined by its routing, and as soon as it arrives at the Backbone, it gets service immediately. The LAN and Link are $M/M/1$ queue, the service time is determined by the length of the job in byte and all jobs will be queued in waiting room only the current job or the first job in the queue is served.

The MHost's will create jobs to simulate different types of traffics. Here, we simulate five types of jobs, namely, multimedia, distributed database, telnet, ftp and general-purpose job.

We use *C++SIM* simulation package, which is a public-domain software. We run our simulation program on a Sun Sparc workstation under SunOS 5.6. In next section

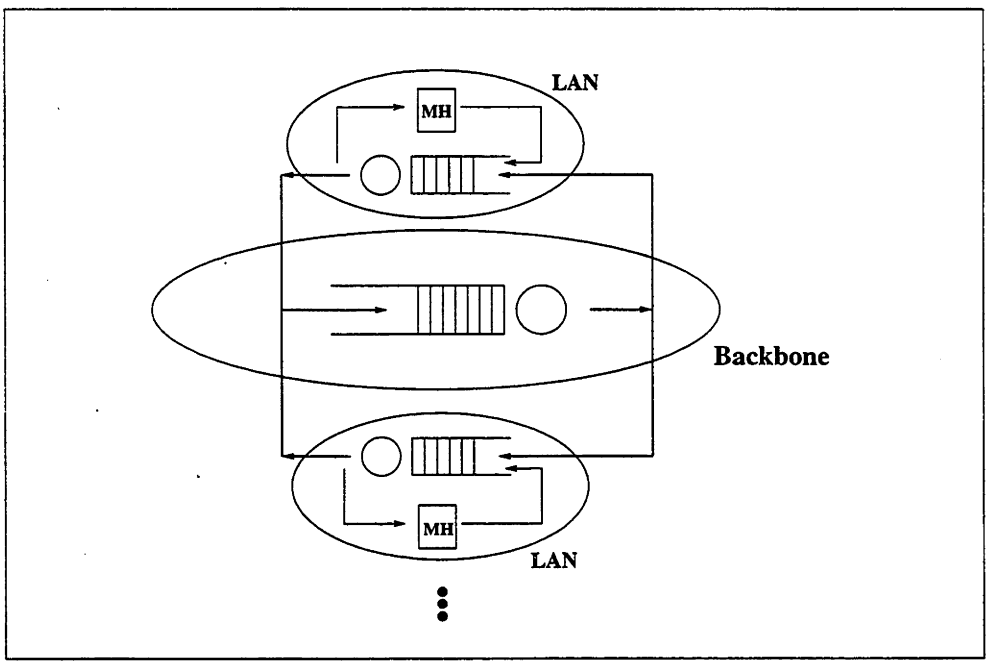


Figure B.1: Structure of Simulation Program

are listed the important header files, in which the constant variables and classes are defined.

B.2 Important Header Files

B.2.1 Job.h

```
#ifndef _JOB_H_
#define _JOB_H_

//
// job types
//
const int           MD_I   = 0x21;
const int           MD_II  = 0x22;
const int           DDBMS  = 0x23;
const int          TELNET  = 0x31;
const int             FTP  = 0x32;
const int             GRL  = 0x33;
const int          RECONN  = 0x34;
const int            REGO  = 0x35;
```

```
const int         MD_I_REQ = 100;
const int         MD_I_RSP = 1500;
const int        MD_II_REQ = 1500;
const int        MD_II_RSP = 1500;
const int        DDBMS_REQ = 200;
const int        DDBMS_RSP = 800;
const int       TELNET_REQ = 44;
const int       TELNET_RSP = 100;
const int          FTP_REQ = 40;
const int          FTP_RSP = 1500;
const int          GRL_REQ = 80;
const int          GRL_RSP = 400;
const int       RECONN_REQ = 60;
const int       RECONN_RSP = 60;
const int         REGO_REQ = 100;
const int         REGO_RSP = 60;

//
//  directions
//
const int      ROUTE_MH_FH = 1;
const int      ROUTE_FH_HA = 2;
const int      ROUTE_FH_MH = 3;
const int      ROUTE_HA_MH = 4;

//--------------
// Job class
//--------------
class Job {
  private:
    int               type;       // set by Job::EncXXX()
    int              route;       //
    int            host_id;       // set by Job::Job()
    int           home_lan;       //
    int           dest_lan;       // set by Job::SetUp()
    int           curr_lan;

    int            req_num;
    int           req_size;
    int           rsp_size;
    int           pkt_size;
    float        life_time;
    float       start_time;
```

```
    int         remain_data;
    int         resent_flag;

    Job             *prev;
    Job             *succ;

  public:
    int         trans_data;
    float       ss_time;
    float       work_time;
    float       lb_time;   // time to leave backbone
                Job (int hid, int lanid);
               ~Job ();
    int         DestLanId();
    int         SetUp(int t, int dlan, int clan,
                      float stime, int datavol);
    int         SetUp(int t, int dlan, int clan,
                      float stime, float lftime);
    int         GetRoute();
    void        SetRoute(int rt);
    int         EncReq(int rt);
    int         EncRsp(int rt);
    int         EncFwd(int rt);
    int         GetPktSize();
    int         GetHostId();
    int         GetHLanId();
    int         GetCLanId();
    int         GetDLanId();
    int         GetType();
    int         SetLTime(float lt);
    int         IsFinished();
    void        SetResent();
    float       GetLTime();

    void        dump();
    void        AssertValid();

    friend      class  JobQueue;
    friend      class  JobOrderList;
};


//-----------------
```

```
// JobQueue class
//-----------------
class JobQueue {

 private:
     Job                    *head;
     Job                    *tail;
     int                      len;

 public:
                         JobQueue();
                        ~JobQueue();
     int          Enqueue(Job *job);
     Job                 *Dequeue();
     int                 QueueLen();
     void           AssertValid();
};


//---------------------------------
//  JobOrderList based on lb_time
//---------------------------------

class JobOrderList {

 private:
    Job      *head;
    Job      *tail;
    int       len;

  public:

             JobOrderList();
            ~JobOrderList();
    int      ListLen();
    int      Insert(Job *job);
    Job     *GetFirst();
    Job     *Delete();

};


#endif
```

B.2.2 Lan.h

```
#ifndef _LAN_H_
#define _LAN_H_

#ifndef BOOLEAN_H_
#  include <Common/Boolean.h>
#endif

#ifndef PROCESS_H_
#  include <ClassLib/Process.h>
#endif

#ifndef RANDOM_H_
#  include <ClassLib/Random.h>
#endif

#include "job.h"

const int NUM_MHOST = 2;

class Backbone;
class JobQueue;
class MHost;
class Link;

class Lan : public Process {

    private:
       int          lan_id;
       MHost       *mhost[NUM_MHOST];
       Link        *link;
       int          state;
       JobQueue     jobq;
       Lan        **lan_list;

    public:
                    Lan(int id, Backbone *backb, Lan **list);
                    ~Lan();
       void         Body();
       void         ActivateHost();
       void         DeactivateHost();
       int          Submit(Job *job);
```

```
        int          Arrival(Job *job);
        Link        *LocalLink();
        MHost       *GetHost(int id);
        Lan         *GetLan(int id);
};

#endif
```

B.2.3 Link.h

```
#ifndef LINK_H_
#define LINK_H_

#ifndef PROCESS_H_
#include <ClassLib/Process.h>
#endif
#ifndef RANDOM_H_
#include <ClassLib/Random.h>
#endif
#include "job.h"

class Lan;
class Backbone;

class Link : public Process {

    private:
        int          link_id;
        Lan         *lan;
        JobQueue     jobq;
        int          state;
        Backbone    *backb;
        float        speed;     // link speed, bytes/sec
        int          num_job;
    public:
                     Link(int id, Lan *pl, Backbone *pb);
                     ~Link();
        void         Body();
        int          Arrival(Job *job);
};

#endif
```

B.2.4 Backbone.h

```
#ifndef BACKBONE_H_
#define BACKBONE_H_

#ifndef BOOLEAN_H_
#  include <Common/Boolean.h>
#endif

#ifndef PROCESS_H_
#  include <ClassLib/Process.h>
#endif

#ifndef RANDOM_H_
#  include <ClassLib/Random.h>
#endif

#include "job.h"
class Link;
const int NUM_LINK = 6;


//
// class definition
//
class Backbone : public Process {

   private:
      Link           *link_list[NUM_LINK];
      JobOrderList    jobList;
      float           delay_time(Job *job);
      int             state;
      UniformStream  *job_delay;
      UniformStream  *DelayTime[NUM_LINK][NUM_LINK];

   public:
      Backbone();
      ~Backbone();
      void     Body();
      int      ConnectLink(int lid, Link *link);
```

```
    int       Submit(Job *job);
};

#endif
```

B.2.5 MHost.h

```
#ifndef  _HOST_H_
#define  _HOST_H_

#ifndef PROCESS_H_
#include <ClassLib/Process.h>
#endif
#ifndef RANDOM_H_
#include <ClassLib/Random.h>
#endif

#include "job.h"

//
// Mobile Host
//

const char   MH_WORK = 1;
const char   MH_MOVE = 2;
const char   MH_RCON = 3;

const int   NUM_JOBS   =  2;


class Lan;


class MHost : public Process  {
  private:
     int                  host_id;
     int                  home_lan;     // home network id;
     int                  curr_lan;     // current network id;
     int                  state;        // running state: W/M/R
     Job                 *job1;         // normal job
     Job                 *job2;         // for reconnection
     Job                 *job3;         // for registration
```

```
    int                  pend_job;
    int                  first_run;
    ExponentialStream   *stay_time;
    UniformStream       *move_time;
    UniformStream       *dest_lan;
    Lan                 *phome_lan;
    Lan                 *pcurr_lan;

    // statistics data
    long                 total_ftpjob;
    long                 total_ftpdata;
    float                total_ftpwt;

    int                  nextLanId();
    int                  process_job(Job *job);


  public:
    MHost(int hid, int home, float mean_stay,
          float mean_move_lo, float mean_move_hi,
          Lan *hlan);
    ~MHost();
    void Body();
    int  ReceiveJob(Job *);
    int  next_lan();
    void dump();

};

#endif
```

Appendix C

Relevant Publications of the Author

[1] Xun Qu, Iain Macleod and Hong Jiang. Internetworking of TCP/IP-based Systems: The Gateways and Network Support, in Proceedings of The fourth International Conference for Young Computer Scientist, Beijing, P.R.China.

[2] Xun Qu, Iain Macleod. A Practical Method for Achieving Portable Communications in the Internet Context, in Proceedings of GLOBECOM'95 (IEEE Catalog: 95CH35756, ISBN:0-7803-2509-5), 1512-1516. February 1995, Singapore.

[3] Xun Qu, Jeffrey X. Yu. And Iain Macleod. Mobile File Filtering, in Proceedings of IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN'97, ISBN:0-88986-238-9), 239-244. August 1997, Singapore.

[4] Xun Qu, Jeffrey X. Yu. Analysis of Mobile IP Solutions, in Proceedings of The First International Conference on Information, Communications and Signal Processing (ICICS'97, IEEE Catalogue Number: 97 TH8237, ISBN: 0-7803-3676-3), 235-240. September 1997, Singapore.

[5] Xun Qu, Jeffrey X. Yu and Bing B. Zhou. Performance Studies of Mobile TCP/IP Protocols, in Proceedings of Mobile Computing & Database & Applications (MCDA'97), 14-25. November 1997, Melbourne, Australia.

[6] Xun Qu, Jeffrey X. Yu and R. Brent. Mobile TCP Socket, in Proceedings of IASTED International Conference on Software Engineering (SE'97, ISBN: 0-88986-244-3), 248-253. November 1997, San Francisco, USA.

[7] Xun Qu, Jeffrey X. Yu and Richard P. Brent. Implementation of a Portable-IP System For Mobile TCP/IP, Australian Computer Communications Volume 20, Number 1. Springer (ISBN: 981-3083-90-5), 499-510. February 1998.

[8] Xun Qu and Jeffrey X. Yu. A Performance Study of Mobile TCP/IP Solutions. The Australian Computer Journal, pp.53-64, Vol. 30, No. 2, May 1998.

Bibliography

1. ALBITZ, P., AND LIU, C. *DNS and BIND*. O'Reilly and Associates, Inc., 1997.

2. BAKRE, A., AND BADRINATH, B. I-TCP: Indirect TCP for mobile hosts. Tech. Rep. DCS-TR-314, Department of Computer Science, Rutger University, Piscataway, NJ 08855, USA, October 1994.

3. BHAGWAT, P., PERKINS, C., AND TRIPATHI, S. Network layer mobility: An architecture and survey. *IEEE Personal Communications* (June 1996), 54–64.

4. CHESHIRE, S., AND BAKER, M. Internet mobility 4x4. *ACM Computer Communications Review 26*, 8 (August 1996).

5. COMER, D. E. *Internetworking with TCP/IP Vol I: Principles, Protocols, and Architecture*, thrid ed. Prentice Hall, Upper Saddle River, New Jersey, 1995.

6. CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. *Introduction to Algorithms*. MIT Press, 1990.

7. C.S.ELLIS, AND R.A.FLOYD. The Roe file system. In *Proceedings of The Third Symposium on Reliability in Distributed Software and Database System* (October 1983), IEEE, pp. 175–181.

8. DUCHAMP, D., AND TAIT, C. D. An interface to support lazy replicated file service. In *Proceedings of The Second Workshop on Management of Replicated Data* (November 1992), IEEE Monterey CA, pp. 6–8.

9. (EDITOR), C. P. IP mobility support, version 17. IETF Internet Draft, May 1996.

10. FITZGERALD, M., BAKOSS, M., AND TOMES, C. Wireless ways. *Network World* (October 1994), 52–57.

11. GUY, R. G., HEIDEMANN, J. S., MAK, W., ET AL. Implementation of the Ficus replicated file system. In *USENIX Conference Proceedings* (June 1990), USENIX, pp. 63–71.

12. HONEYMAN, P., AND HUSTON, L. B. Communications and consistency in mobile file systems. Tech. Rep. 95-11, CITI, University of Michigan, Ann Arbor, USA, 1995.

13. HUSTON, L. B. Disconnected operation for AFS. Tech. Rep. 93-3, CITI, University of Michigan, Ann Arbor, USA, 1993.

14. IOANNIDIS, J., DUCHAMP, D., AND GERALD Q. MAGUIRE, J. IP-based protocols for mobile internetworking. *ACM Computer Communications Review 21*, 5 (September 1991), 235–245.

15. IOANNIDIS, J., AND GERALD Q. MAGUIRE, J. The design and implementation of a mobile internetworking architecture. In *Proceedings of 1993 Winter USENIX Conference* (San Diego, CA, USA., January 1993), USENIX.

16. JOHNSON, D. B. Mobile host internetworking using loose source routing. Tech. Rep. CMU-CS-93-128, School of Computer Science, Carnegie Mellon University, 1993.

17. JOHNSON, D. B. Scalable and robust internetwork routing for mobile hosts. In *Proceedings of the 14th International Conference on Distributed Computing Systems* (June 1994), IEEE Computer Society.

18. JOSEPH, A., DELESPINASSE, A., TAUBER, J., ET AL. Rover: A toolkit for mobile information access. In *Proceddings of 15th Symposium on Operating Systems Principles* (Copper Mountain Resort, USA, December 1995).

19. JR., L. K., BECKHARDT, S., HALVORSEN, T., ET AL. Replicated document management in a group communication system. In *Proceedings of Conference on Computer-Supported Cooperative Work* (Porland, Oregon, September 1988).

20. JR., T. P., GUY, R., HEIDEMANN, J., ET AL. Management of replicated volume location data in the Ficus replicated file system. In *USENIX Proceedings* (June 1991), USENIX.

21. J.S. HEIDEMANN, T.W. PAGE, R. G., AND POPEK, G. Primarily disconnected operation: Experiences with Ficus. In *Proceedings of The Second Workshop on Management of Replicated Data* (November 1992), IEEE.

22. KUENNING, G., POPEK, G., AND REIHER, P. An analysis of trace data for predictive file caching in mobile computing. In *Proceedings of the 1994 Summer Usenix Conference* (1994).

23. KUENNING, G. H. The design of the Seer predictive caching system. In *proceedings of Mobile Computing Systems and Applications 1994* (Santa Cruz, CA, December 1994).

24. KUMAR, P., AND SATYANARAYANAN, M. Flexible and safe resolution of file conflicts. In *Proceedings of the USENIX Winter 1995 Technical Conference* (New Orleans, USA, January 1995).

25. LITTLE, M., AND MCCUE, D. *Construction and Use of a Simulation Package in C++.*

26. MARZULLO, K., AND SCHMUCK, F. Supplying high availability with a standard network file system. In *Proceedings of Eighth International Conference on Distributed Computing Systems* (June 1988), IEEE.

27. MORRIS, J. H., SATYANARAYANAN, M., ET AL. Andrew: A distributed personal computing environment. *Communications of the ACM 29*, 3 (March 1986), 184–201.

28. MUMMERT, L., AND SATYANARAYANAN, M. Variable granularity cache coherence. *Open System Review 28*, 1 (January 1994), 55–60.

29. MYLES, A., AND SKELLERN, D. Comparing four IP based mobile host protocols. *Computer Networks and ISDN Systems 26* (1993), 349–355.

30. OPPEN, D., AND DALAL, Y. The Clearinghouse: A decentralized agent for locating named objects in a distributed environment. *ACM Transactions on Office Information Systems 1*, 3 (July 1983), 230–253.

31. OUSTERHOUT, J., CHERENSON, A., DOUGLIS, F., ET AL. The Sprite network operating system. *IEEE Computer* (February 1988), 23–36.

32. OUSTERHOUT, J., COSTA, H., HARRISON, D., ET AL. A trace-driven analysis of the Unix 4.2 BSD file system. Tech. Rep. UCB/CSD 85/230, UCB, 1985.

33. PATEL, B. V., BHATTACHARYA, P., REKHTER, Y., AND KRISHNA, A. An architecture and implementation toward multiprotocol mobility. *IEEE Personal Communications* 2, 3 (June 1995), 32–42.

34. PERKINS, C., MYLES, A., AND JOHNSON, D. IMHP: A mobile host protocol for the internet. *Computer Networks and ISDN System 27* (1994), 479–491.

35. PERKINS, C., MYLES, A., AND JOHNSON, D. B. The internet mobile host protocol (IMHP). In *Proceedings of INET'94/JENC5* (1994).

36. PERKINS, C., AND REKHTER, Y. Draft RFC, July 1992.

37. PERKINS, C., AND REKHTER, Y. Support for mobility with connectionless network layer protocols (transport layer transparency). Draft RFC, January 1993.

38. PERKINS, C. E. Ipv4 mobility support, rfc2002, October 1996.

39. PERLMAN, R. *Interconnections — Bridges and Routers*. Addison-Wesley, 1994.

40. PISCITELLO, D. M., AND CHAPIN, A. L. *Open Systems Networking*. Addison-Wesley, Reading, Massachusetts, 1993.

41. PITOURA, E., AND BHARGAVA, B. Dealing with mobility: Issues and research challenges. Tech. Rep. CSD-TR-93-070, Department of Computer Sciences, Purdue University, November 1993.

42. QU, X., MACLEOD, I., AND JIANG, H. A practical method to achieve portable communication in internet context. In *Proceedings of GlobeCom'95* (Singapore, November 1995), IEEE, pp. 1512–1516.

43. Requirements for internet hosts — communication layers, RFC 1122, October 1989.

44. Requirements for internet hosts — application and support RFC1123, October 1989.

45. RFC1518, an architecture for IP address allocation with CIDR, September 1993.

46. Classless Inter-Domain Routing (CIDR): an Address assignment and Aggregation Strategy, September 1993.

47. DHCP Options and BOOTP Vendor Extensions: RFC1533, October 1993.

48. Dynamic Host Configuration Protocol: RFC1541, October 1993.

49. RFC1631, the IP Network Address Translator (NAT), May 1994.

50. RFC1723, RIP version 2: Carrying additional information, November 1994.

51. RFC1771, a Border Gateway Protocol 4 (BGP-4), March 1995.

52. Domain Name System Security Extensions: RFC2065, January 1997.

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

54. Secure Domain Name System Dynamic Update: RFC2137, April 1997.

55. RFC2178, OSPF version 2, July 1997.

56. Internet protocol(IP), RFC 791, September 1981.

57. Internet Control Message Protocol, September 1981.

58. Transmission control protocol, RFC 793, September 1981.

59. Ethernet Address Resolution Protocol RFC1533, November 1982.

60. RFC950, internet standard subnetting procedure, August 1985.

61. Rumor project. `http://ficus-www.cs.ucla.edu/rumor/`.

62. SANDBERG, R., ET AL. Design and implementation of Sun Network File System. In *USENIX Summer Conference Proceedings* (June 1985), USENIX Association.

63. SATYANARAYANAN, M. Mobile information access. Tech. Rep. CMU-CS-96-107, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, 1996.

64. SATYANARAYANAN, M., KISTLER, J., P.KUMAR, ET AL. Coda: A highly available file system for a distributed workstation environment. *IEEE Transactions on Computer Systems 39*, 4 (April 1990), 447–459.

65. SATYANARAYANAN, M., KISTLER, J. J., ET AL. Experience with disconnected operation in a mobile computing environment. Tech. Rep. CMU-CS-93-168, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, June 1993.

66. SIMPSON, W. IPng Mobility Considerations, RFC 1688, August 1994.

67. Ip mobility support, version 2, November 1997.

68. STEVENS, W. R. *UNIX Network Programming.* Pretice Hall, 1990.

69. STEVENS, W. R. *TCP/IP Illustrated, Volume 1, The Protocols.* Addison-Wesley, 1994.

70. STEVENS, W. R. *TCP/IP Illustrated, Volume 3, TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols.* Addison-Wesley, 1996.

71. TAIT, C., AND DUCHAMP, D. An efficient variable-consistency replicated file service. In *Proceedings of File System Workshop* (Ann Arbor, MI, USA, May 1992), USENIX, pp. 111–126.

72. TANENBAUM, A. S. *Computer Networks,* second ed. Prentice-Hall, 1989.

73. TERAOKA, F., YOKOTE, Y., AND TOKORO, M. A network architecture providing host migration transparency. *ACM Computer Communications Review 21,* 1 (January 1991), 209–220.

74. WRIGHT, G. R., AND STEVENS, W. R. *TCP/IP Illustrated, volume 2, The Implementation.* Addison-Wesley, 1995.