THE HONG KONG POLYTECHNIC UNIVERSITY

HUNG HOM, KOWLOON, HONG KONG

THE DEPARTMETN OF COMPUTING

DESIGN OF HIGH PERFORMANCE MOBILE COMMUNICATION PROTOCOLS

By

Liang Zhang

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF PHILOSOPHY

December, 2004

Abstract

The notion of mobility takes various forms in networking and distributed systems. On one hand, the development of mobile and wireless communication networks allows users to change their access points of connecting to the network while maintaining their current sessions. On the other hand, mobile code and mobile agents provide high flexibility for the configuration of distrusted systems. In this thesis, a *Mobile Object* (MO) denotes a mobile entity which can be a mobile phone, a mobile node or a mobile agent.

In recent years, researchers have proposed a wide range of schemes for the location management and the message delivery in mobile communication. However, each scheme has it own assumptions, design goals and methodology, making it difficult to adapt to different application requirements. In this thesis, we aim at discovering new approaches in designing adaptive mobile communication protocols that are more reliable in a fault-prone environment, more asynchronous with less constrains on MO's mobility, more efficient in terms of communication costs and more scalable to future system expansion. We propose a novel *mailbox based scheme*, which associates each MO with a mailbox while allowing them to be decoupled when needed. Message passing between the sender and the receiver is now divided into two steps: 1) from the sender to the receiver's mailbox, and 2) from the mailbox to the receiver. During each migration, a MO can decide whether to bring its mailbox to the new location or to leave the mailbox at its current location. This flexible approach allows us to design a variety of protocols that can be made adaptive to specific application requirements by properly evaluating tradeoffs among various design considerations.

To explore the benefits of the mailbox based scheme, we apply it to two different contexts: mobile agent and Mobile IP. In the mobile agent context, we propose an *Adaptive and Reliable Protocol* (ARP), which is proved to possess many desirable

characteristics such as location transparency, asynchrony, scalability and adaptability. By properly determining the mailbox's migration pattern, the ARP can outperform both the *home-server based scheme* and the *distributed-registration based scheme*. To further enhance efficiency and reliability, we extend the ARP with a path pruning algorithm and a two-layer fault-tolerant architecture. In the path pruning algorithm, the location management cost is greatly reduced as the algorithm can effectively identify and remove the redundant hosts on the mailbox's migration path so that fewer messages are required for the location management. In the two-layer fault-tolerant architecture, the low layer abstracts away network failures and provides reliable point-to-point message passing between two *Mobile Agent Platforms* (MAP); the high layer overcomes message loss caused by host failures and accomplishes reliable end-to-end message delivery between two mobile agents.

In the Mobile IP context, by introducing the mailbox, we can achieve adaptive location management that enables dynamic tradeoff between the packet delivery cost and the location management cost so as to minimize the total communication cost. Since the mailbox is located somewhere in the network close to the receiver, the packet retransmission cost could also be reduced.

Certificate of Originality

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material which has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Liang Zhang (Name of Student)

Publications

1. Liang Zhang, Jiannong Cao and Sajal K. Das, "A Mailbox-based Scheme for Improving Mobile IP Performance", *The 2003 Workshop on Mobile and Wireless Networks (MWN 2003) in conjunction with The 23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, May 19-22, 2003, Providence, Rhode Island, USA.
2. Jiannong Cao, Liang Zhang, Xinyu Feng and Sajal K. Das, "Path Compression in Forwarding-based Reliable Mobile Agent Communications", *The 32nd International Conference on Parallel Processing (ICPP 2003)*, October 6-9, 2003, Kaohsiung, Taiwan.
3. Jiannong Cao, Liang Zhang, Xinyu Feng and Sajal K. Das, "Adaptive and Reliable Message Delivery for Mobile Objects", *The 2003 IEEE Global Communications Conference (GLOBECOM 2003)*, December 1-5, 2003, San Francisco, California, USA.
4. Jiannong Cao, Liang Zhang, Jin Yang and Sajal K. Das, "A Reliable Mobile Agent Communication Protocol", *The 24rd International Conference on Distributed Computing Systems (ICDCS 2004)*, March 23-26, 2004, Tokyo, Japan.
5. Jiannong Cao, Liang Zhang, Sajal K. Das and Henry Chan, "Design and Performance Evaluation of an Improved Mobile IP Protocol", *The 23rd IEEE Conference on Computer Communications (INFOCOM 2004)*, March 7-11, 2004, Hong Kong.
6. Jiannong Cao, Liang Zhang, Xinyu Feng and Sajal K. Das, "Path Pruning in Mailbox-based Mobile Agent Communications", *Journal of Information Science and Engineering – Special Issue on Mobile Computing, 20(3):405-424*, May 2004.

Acknowledgements

First of all, I would like to take this opportunity to express my sincere gratitude towards my supervisor, Prof. Jiannong Cao, for his invaluable support and guidance throughout the last two years. It was his constant encouragement and wonderful association that has enabled me to achieve the objectives of this work.

I would also like to thank Prof. Sajal K. Das at The University of Texas at Arlington, who has provided me remarkable insights into many aspects of my academic life.

I would like to thank Xingyu Feng for his previous work enlightening me on new ideas and improvements.

My parents and girl friend have been a constant source of love and affection. I am eternally grateful to them for always being with me whenever I need them.

Finally, I would like to express my special thanks to the following friends: Jingyang Zhou, Wei Xu, Xuhui Li, Jin Yang, Xianbing Wang and Yudong Sun.

Liang Zhang

April 18, 2004

Contents

List of Figures

List of Tables

Chapter 1. Introduction

Traditionally, computer communications require all the participants to be statically attached to the communication network. The physical location of each participant is fixed during the whole session of the communication or even during the participant's life-long period. Any attempt to relocate any participant on the fly will fail the communication. We call this type of communications *static communications*.

However, with the emergence of different portable and mobile devices ranging from laptop and palmtop computers to mobile phones, and the development of various wireless and mobile communication technologies including *Wireless LAN* (WLAN), Bluetooth, Mobile IP and cellular technology, users are able to change their access points of connecting to the network while keeping their current sessions [1]. At the same time, mobile code and mobile agents that can move among and execute on different network locations present a new paradigm of building distributed systems, which provides high flexibility for system configurations [2, 3]. Therefore, we have encountered a new type of communications – *mobile communications* where partial or all of the communication participants can move and their point-of-attachments to the network can change from time to time. The participant could be a mobile phone, a mobile node or a mobile agent, and in this thesis, we will uniformly denote it as a *Mobile Object* (MO).

In this chapter, we will first provide an overview of mobile communications, and then describe the requirements of designing a mobile communication protocol.

1.1 New Trend – Mobile Communication

As a MO can migrate from one network location to another while it is communicating with other MOs, a mobile communication protocol is necessary for any mobile

communication system to manage the location information of all the MOs within the system on one hand, and deliver messages to the proper destination on the other hand. More specifically, a mobile communication protocol should effectively support two operations: 1) *migrating* that facilitates the movement of a MO to a new location, and 2) *message-forwarding* that locates a specified MO and delivers messages to it. During the migrating operation, the new location of the MO may be updated to the network, while during the message-forwarding operation, the current location of the receiver MO is enquired from the network. The special network sites identified for storing and managing the location information of MOs are usually referred to as *location servers*.

Apart from the location management, the message delivery should also be given special attention. As a MO can be on the move when a message is delivered to it, it is possible that when the message arrives at its target location, the MO has already left for another location. One simply solution to this problem is to locate and forward the message again to the current location of the MO. However, the previous situation may happen again. This is also well-known as the *message chasing problem*. To deal with this problem, a certain degree of coordination between the message delivery and the MO's migration must be enforced into the mobile communication protocol.

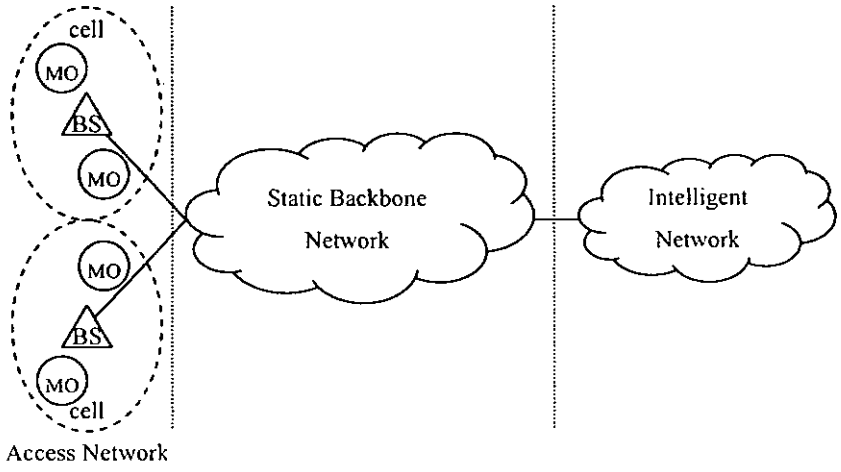


Figure 1. **The network infrastructure of a mobile communication system**

The network infrastructure of a mobile communication system adopted in this thesis is depicted in Figure 1, which consists of three network sections: *access network*, *static backbone network* and *intelligent network*. The access network is the interface between the MOs and the static backbone network. On one hand, it provides resources and working environment for the MOs. On the other hand, it assists the static backbone network to manage the MOs. For example, in a mobile telephone or Mobile IP system, the access network is the wireless network connecting mobile phones or mobile nodes through the air such as cellular network, WLAN and Bluetooth; in a mobile agent system, the access network is the *Mobile Agent Platform* (MAP) which provides the concepts and the mechanisms for mobility and communication on one hand, and security of the underlying system on the other hand [2, 3]. Usually, the access network is geographically made up of a group of *cells* to cover the service area of the mobile communication system. Each cell is supported by a *Base Station* (BS), which can be viewed as the point-of-attachment of a MO to the static backbone network and possesses all the necessary facilities to communicate and coordinate with the MO. A MO can communicate directly with a BS only if it is residing within the same cell as the BS. For example, in a mobile telephone system, we refer to the *Mobile Switching Centers* (MSC) as the BSs; in a Mobile IP system, the *mobility agents* (routers to handle the Mobile IP specific procedures [4]) could be viewed as the BSs; in a mobile agent system, both the BSs and their supporting cells are collocated at the MAPs. The intelligent network is the network to connect all the location servers within the system, which carries only the location management related traffic. It could be part of or a separate network connecting with the static backbone network. For example, in a mobile telephone system, the separate *Signaling System No. 7* (SS7) network is the most common candidate for the intelligent network; in a Mobile IP or mobile agent system, the intelligent network is built upon the static backbone network to connect all the mobility agents or the MAPs within the system.

With such an infrastructure, the location management involves the interactions between the MOs and the intelligent network, and the message delivery involves the

message transmission at the static backbone network. Of course, all these operations must also involve the assistance of the access network, even if the two communicating MOs are within each other's access range. However, this is not the sole infrastructure for mobile communication. For example, in *wireless ad hoc communication* [5], mobile nodes can communicate directly with each other without involving any static network infrastructure. In this thesis, all our discussions are based on the Figure 1's infrastructure. Apart from those described above, other example systems include a *Small Message Service* (SMS) or *Multimedia Message Service* (MMS) system where the BSs represent the SMS/MMS centers for mobile devices, and a *Wireless ATM* (WATM) system where the BSs are regarded as the zone managers for mobile terminals.

1.2 Requirements of Mobile Communication Protocols

The presence of mobility introduces new challenges and requirements in designing communication protocols. These are described as follows.

- **Location transparency.** Since a MO can move from one location to another, it is not reasonable, if not impossible, to require the MO to have apriori knowledge about its communication peers' locations before sending messages. Therefore, the first requirement of a practical mobile communication protocol is to allow MOs to communicate in a location transparent way, i.e., a MO can send messages to other MOs without knowing their current locations. It is the responsibility of the mobile communication protocol to keep track of the locations of all the MOs.

- **Reliability.** A desirable requirement for any communication protocol is reliability. Programming primitives that can guarantee reliable message delivery greatly simplify programming tasks and make the applications developed more robust. Traditionally in static communication, reliability is achieved by providing some

degree of fault tolerance to the underlying network. However, the fault tolerance is not sufficient to guarantee reliability in mobile communication since messages may face the chasing problem so that they cannot always catch up with the MO and will be dropped eventually. Therefore, the challenge to reliable mobile communication persists even under the assumption of a fault-free transmission mechanism. It is the presence of mobility, not the presence of network failures that undermines reliability. Coordination between the message delivery and the MO's migration is necessary to prevent the message chasing problem from taking place. In this thesis, reliability mainly refers to the following property: no matter how frequently the target MO migrates, messages can be delivered to it in a bounded number of hops.

- **Asynchrony.** It is desirable that a mobile communication protocol allows MOs to migrate freely to other locations whenever they want. This is because mobility is often considered as the greatest merit of mobile communication systems. However, due to the reliability constraint, sometimes a MO has to wait until the arrival of certain messages before it can initiate the migration. Communications between MOs should be as asynchronous as possible so that the MO's mobility is not over-constrained by frequent and tight coordination.

- **Efficiency.** The cost of a protocol is characterized by the number of messages sent, the message size and the distance traveled by the messages. An efficient protocol should attempt to minimize all these quantities. As mentioned before, a protocol should support two operations: migrating and message-forwarding. However, the objective of minimizing the cost of these two operations results in conflicting requirements [6]. To illustrate this tradeoff, let us consider two extreme strategies, namely the *full-information strategy*, in which every site in the network maintains the up-to-date information about the whereabouts of all the MOs, and the *no-information strategy*, which does not require any location update during each MO's migration. Clearly, the former strategy makes the

message-forwarding operation cheap, but the migrating operation very expensive because it is necessary to update the location information at every site during each movement. With the latter strategy, the migrating operation has a zero cost, but the message-forwarding operation imposes a very high overhead because it requires the searching over the entire network. In general, a protocol should perform well for some specific communication and migration pattern, achieving a balance of tradeoff between the migrating cost and the message-forwarding cost.

- **Scalability.** A successful mobile communication protocol may be required to leave rooms for the future expansion of the mobile communication system to support more MOs without severe performance degradation. As we know, a distributed approach of system design is more scalable than a centralized one, which is applied to mobile communication that the location information of MOs would better be stored in and processed by distributed location servers. Basically, there are two methods of distributing location information. One is *partition*, meaning that the entire location information of MOs within the system is divided into several portions with each portion maintained by a single location server, and the other is *replication*, meaning that the location information of a MO is copied into several location servers. The replication can also make the protocol more tolerant to location server failures because it introduces redundancy and leaves the protocol alternative options of resolving a MO's location if the primary location server fails. Both the partition and the replication methods may affect the protocol efficiency. On one hand, a good location information distribution strategy will bring down the cost of tracking a MO. On the other hand, more cost may be incurred in updating and maintaining certain degree of coherence among the replicated location information within the system.

- **Adaptability.** Although it would be ideal to address and satisfy all of the above requirements when designing a mobile communication protocol, in practice, enhancing one capability may be at the expense of another. For example, in some

applications, reliability is more important and therefore, asynchrony should be sacrificed to provide coordination between the message forwarding and the MO's migration; efficiency also exhibits a dilemma between the migration cost and the message-forwarding cost; in some applications, efficiency and scalability cannot be fulfilled at the same time and a tradeoff has to be made. Although protocols can be designed for specific applications to achieve optimal performance, it is desirable to have an adaptive protocol which is suitable for as many applications as possible.

These requirements can serve as a guideline for the design of mobile communication protocols, which will be presented in the following chapters.

1.3 Contributions of the Thesis

Although a wide range of schemes for the location management and the message delivery in mobile communication have been proposed in recent years, each scheme has it own assumptions, design goals and methodology, making it difficult to adapt to different application requirements defined above. In this thesis, we aim at developing approaches to designing adaptive mobile communication protocols. We adopt a novel *mailbox based scheme*, which associates each MO with a mailbox while allowing them to be decoupled. Message passing between the sender and the receiver is now divided into two steps: 1) from the sender to the receiver's mailbox, and 2) from the mailbox to the receiver. During each migration, a MO can decide whether to bring its mailbox together with it to the new location or simply leave the mailbox at the current location. This flexible approach allows us to design a variety of protocols that can be made adaptive to specific application requirements by properly evaluating tradeoffs among various design considerations.

To explore the benefits of the mailbox based scheme, we apply it to two different

contexts: mobile agent and Mobile IP. In the mobile agent context, we propose an *Adaptive and Reliable Protocol* (ARP), which is proved to possess many desirable characteristics such as location transparency, asynchrony, scalability and adaptability. By properly determining the mailbox's migration pattern, the protocol can outperform both the home-server based scheme (to be introduced in Section 2.1.1.1) and the distributed-registration based scheme (to be introduced in Section 4.1). To further enhance efficiency and reliability, we extend the protocol with a path pruning algorithm and a two-layer fault-tolerant architecture. In the path pruning algorithm, the location management cost is greatly reduced as the algorithm can effectively identify and remove those redundant hosts on the mailbox's migration path so that fewer messages are required during each location management. In the two-layer fault-tolerant architecture, the low layer abstracts away network failures and provides reliable point-to-point message passing between two MAPs; the high layer overcomes message loss due to host failures and accomplishes reliable end-to-end message delivery between two mobile agents. Finally, we consider how to dynamically determine whether to move a mailbox when its owner MO is going to migrate to a new location.

In the Mobile IP context, by introducing the mailbox, we can achieve adaptive location management that enables dynamic tradeoff between the packet delivery cost and the location management cost so as to minimize the total communication cost. Since the mailbox is located somewhere in the network close to the receiver, the packet retransmission cost could also be reduced.

1.4 Organization of the Thesis

The remaining part of this thesis is organized as follows. Chapter 2 presents a review of related work. Chapter 3 introduces our mailbox based scheme. Chapter 4 applies the mailbox based scheme to the mobile agent context and proposes the ARP, the path

pruning algorithm and the two-layer fault-tolerant architecture. Properties are proved and experiments are conducted to evaluate the performance of our protocol. Chapter 5 applies the mailbox based scheme to the Mobile IP context. A numerical model of our protocol is constructed, based on which we evaluate and compare the performance of our protocol with that of the traditional Mobile IP. The final chapter provides the concluding remarks and the directions of our future work.

Chapter 2. Related Work

In recent years, researchers have proposed a wide range of schemes in different contexts for the effective location management and message delivery in mobile communication. In the following subsections, we review these schemes according to the requirements defined in Chapter 1.

2.1 Location Management Schemes

Basically, a complete location management scheme needs to define both the location update strategy and the location enquiry strategy. The location update strategy defines how a MO updates its location information to appropriate location servers within the system, and the location enquiry strategy defines how a sender can resolve the receiver MO's location by enquiring suitable location servers plus some additional searching algorithms. The searching algorithms are used to handle those situations when enquiring the location servers does not produce the current location of the receiver. Each of the two strategies is associated with some cost and complexity. Although it is desirable to optimize both strategies, there actually exhibit a tradeoff that the more the location update proceeds, the less the location enquiry handles, and vice versa. Two extreme cases of the location management have been described in Chapter 1: full-information and no-information, where the full information case produces a large overhead in the location update to all the location servers, but the location enquiry reduces to enquire only the closest location server; on the other hand, the no-information case spends no cost on the location update, but requires extremely high cost on the location enquiry at all the network locations. Between these two extremes, various approaches that balance the cost of the location update against the cost of the location enquiry are possible, and we name these approaches *partial-information*. Next we provide a survey of several major schemes using the partial-information strategy, and categorize them according to the logical topology of

the location servers within the system.

2.1.1 Flat Schemes

In this category, there is no special organization of the location servers within the system. In other words, all the location servers are just situated in a single flat layer. With such an organization, the location update strategy and the location enquiry strategy are to determine, either statically or dynamically, a write set and a read set of location servers for each MO, respectively. The write set defines those location servers for a MO to update its location information, and the read set defines those location servers for a sender to enquire the receiver MO's location. The read set and the write set are also referred to as *quorums* [18]. Therefore, a feasible location management scheme in this category should guarantee that for every MO, its associated write set intersects with any other MO's read set so that it is always possible for a sender to enquire the receiver MO's location since the receiver has already update its location information to at least one location server queried by the sender. There are two major schemes: *home server* and *fair distribution*.

2.1.1.1 Home Server

In this scheme, only one location server, known as the home server, is statically associated with each MO. During the location update phase, the MO updates its location information to its home server; during the location enquiry phase, the sender enquires the receiver MO's location from the receiver's home server. The sender can resolve where the receiver's home server locates by performing a *naming lookup service* [7, 8] at the receiver's identity. Therefore, both the read set and the write set contain only one location server, i.e., the home server where they must intersect with each other.

There are many systems adopting this scheme because it is simple to implement. For example, in mobile telephone systems, the two prevailing existing standards, the *Interim Standard 41* (IS-41) used in North America and the *Global System for Mobile Communications* (GSM) used in Europe, both support the location management by a so-called *two-tier strategy* [9], which is actually a variant of the home-server based scheme. In the two-tier strategy, a home database called *Home Location Register* (HLR) is associated with each mobile user. It maintains the user's current location as part of the user's profile. The location update and the location enquiry coincide with those of the home-server based scheme. When a user x moves to a new zone, x's HLR is contacted and updated to record the new location. When another user y wants to communicate with x, x's HLR is identified and queried for x's current location. As an enhancement, *Visitor Location Registers* (VLR) are maintained at each zone to store copies of profiles of users who are not at their home location and currently residing inside that zone. When a call is placed from zone i to the user x, the VLR at zone i is queried first, and only if x is not found there, is x's HLR contacted. When x moves from zone i to zone j, in addition to updating x's HLR, the entry for x is deleted from the VLR at zone i, and a new entry for x is added to the VLR at zone j.

Mobile IP also makes use of the home-server based scheme. It associates each mobile node with a *home agent* in its home network. Once the mobile node leaves its home network and arrives at a foreign network, it will at once send back to its home agent a binding update message which contains its current IP address called the *care-of address*. The home agent records this care-of address and from now on, it will behave as if it was the mobile node in the home network. As usual, packets for the mobile node are sent to its home network where they are intercepted by the home agent. Later the home agent will tunnel these packets to the care-of address of the mobile node where they are received eventually. In this way, the senders are sheltered from the current location of the mobile node and will simply rely on the home agent for the packet delivery.

Besides, many mobile agent systems apply the home-server based scheme as their primary mechanism for the inter-agent communication. Since any mobile agent is associated with a dedicated place-of-born called the *agent home*, it is natural to let the agent home serve as the home server. Examples include Aglet [10], JATLite [11], Mogent [8], Mole [12] and Voyager [13].

Although the home-server based scheme is simple to implement, it has the following problems. As the home server acts as the sole location server for a group of serving MOs, it may suffer from the single point of failure problem. It can also be a performance bottleneck if the number of the serving MOs becomes large while migrations and communications are frequent. Caching based mechanisms [14], to some extent, can alleviate this problem as a sender can now keep records of the receiver's location so that it is not necessary to enquire the home server each time there is a message to send. If a cache miss occurs, the home server is then queried for the correct location. Although the caching based mechanisms can effectively reduce the number of location enquires, they cannot reduce the number of location updates because all the MOs have to report their updated locations whenever they move to a new site, even if the home server may be residing at a very far away place. The home-server based scheme also suffers from the well-known *triangle routing problem* [4] if the home server is a message relaying station (like the home agent in Mobile IP). In this case, messages have to always take a detour through the home server, which may be very far away, to reach the destination. Again, the caching based mechanisms can help. For example, in Mobile IP route optimization [15], when a sender wants to communicate with a mobile node, the first packet is sent to the receiver's home network. If the mobile node has already roamed outside its home network, an update message is sent back to the sender informing it of the current location of the mobile node. In this way, any subsequent packet can be sent directly to the mobile node without bypassing the home network. However, all these protocols cannot handle message loss caused by the MO's mobility.

2.1.1.2 Fair Distribution

In this scheme [16], the location management task is fairly distributed to all the location servers within the system. This is achieved in the following way. During the location update phase, a group of location servers is dynamically determined as the write set, which is based on both the MO's identity and its current location. This is because if only the MO's identity is used for determining the write set (just like the home-server based scheme), the scheme will fail to exploit the *locality* characteristic of mobile communication: a MO is more likely to communicate with its vicinity MOs. Hence it would better include some of the vicinity location servers in the write set. However, if only the current location is used for the determination, the scheme will lead to uneven distribution of responsibility. For example, when there is a high concentration of MOs in one network location, all these MOs will be mapped to the same location servers which will be overburdened. Similarly, during the location enquiry phase, a read set is dynamically determined also based on the receiver MO's identity but the sender's current location. To guarantee fairness, this scheme assigns multiple virtual identities to those hot MOs which are enquired more frequently than others. For each of such virtual identities, a write set will be generated. That means more location servers for the hot MOs are required to update in order to distribute the heavy workload of the location enquiry. Also the scheme employs the principle of open addressing hash function, specifically the double hashing [17] to determine the read set and the write set, which can make sure the results be uniformly spread over the entire location servers. Interesting readers may refer to [16] for details.

Although this scheme is free of the performance bottleneck and single point of failure problems, it significantly increases both the location update cost and the location enquiry cost. Also this scheme cannot guarantee that the derived write set of location servers will be close to those active senders. As a conclusion, this scheme enhances reliability and scalability at a cost of efficiency.

2.1.2 Hierarchical Schemes

In this category, the location servers within the system are organized in a hierarchical manner, i.e., some location servers may be logically situated at a higher layer to cover a bigger area but usually maintain coarser location information of MOs within its coverage, while some other location servers may be located at a lower layer with smaller area to take care and much more fine-grained location information to maintain. The bottom layer will maintain the precise location information of each supported MO. With such an organization, the location update strategy and the location enquiry strategy are, for each MO, to determine at each layer a write set and a read set of location servers, respectively. More specifically, during the location update phase, the MO will first determine till which layer its new location has to be updated. After that, it will derive and update the write set of location servers for each relevant layer. During the location enquiry phase, the sender will enquire the location of the receiver MO starting from the bottom layer. If the bottom layer does not find the receiver, the location enquiry will be propagated to the upper layers until an entry for the receiver is found. Of course, at each layer, a read set of location servers are determined and enquired. Therefore, a feasible location management scheme in this category should guarantee that for every pair of MOs, there must be a layer at which the write set of one MO intersects with the read set of the other MO. There are two major kinds of schemes: *tree hierarchy* and *non-tree hierarchy*.

2.1.2.1 Tree Hierarchy

In this scheme, a tree-like hierarchy of location servers forms a location directory similar to that of the *Domain Name Service* (DNS). Each region corresponds to a sub-tree in the directory. For each MO, there is a unique path of forwarding pointers traversing from the root to leaf of the tree that can resolve the actual location of the MO. This scheme and its variants have been widely used to keep track of mobile

phones [19, 20], mobile nodes [21, 22] and mobile agents [7, 13].

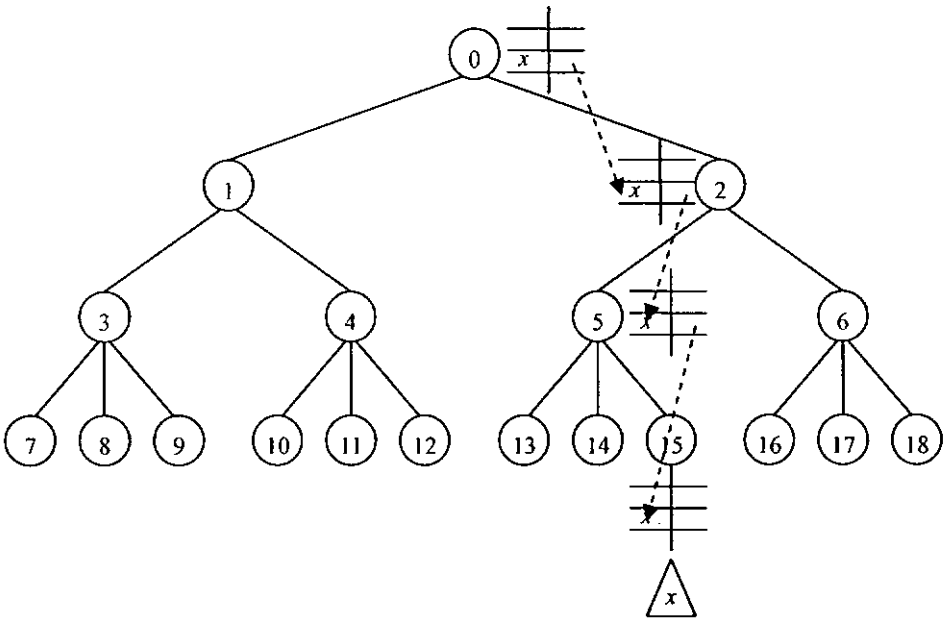


Figure 2. **The tree-hierarchy based scheme**

Figure 2 depicts an example of the location management in the tree-hierarchy based scheme. Suppose a MO x is currently residing under the location server 15. We can see from the figure a path of forwarding pointers starting from the root 0, through the intermediate location servers 2 and 5, and finally pointing to the location server 15. Now let us suppose x moves to the location server 17. Then all the following location servers have to be updated: the entry for x is deleted from the location servers 15 and 5, the entry for x at the location server 2 is updated, and the entry for x is added to the location servers 6 and 17. After that, a sender under the location server 13 wants to communicate with x. The following location servers have to be enquired to find x: the sender polls but finds no entry for x at the location servers 13 and 5, finds the first entry for x at the location server 2, and then follows the forwarding pointers downwards to the location server 17 where the precise location of x is found.

In this scheme, both the read set and the write set at each layer contain only one location server along the tree branch and, because of the tree structure, they are guaranteed to intersect with each other at certain layer. In the worst case, the

intersection meets at the root. The advantage of this scheme is obvious. First, there is no need to statically bind a MO with a home server. Second, it scales much better and costs less than other schemes because it exploits the locality characteristic. Actually, there are two types of locality: *location update locality* and *location enquiry locality*, which respectively mean that all the location updates and the location enquiries are handled in a local region. Both may greatly reduce the communication overhead and latency. However, the tree-like hierarchy is not always easy to construct, especially in the Internet environment. Even if the hierarchy is constructed, it will fail in adapting to new patterns of locality, in which situation both the location update and the location enquiry may have to proceed on a long path of location servers, even till the root of the tree. Therefore, the location servers at higher layers may need to handle a relative larger workload. Also this scheme has large storage demands.

2.1.2.2 Non-tree Hierarchy

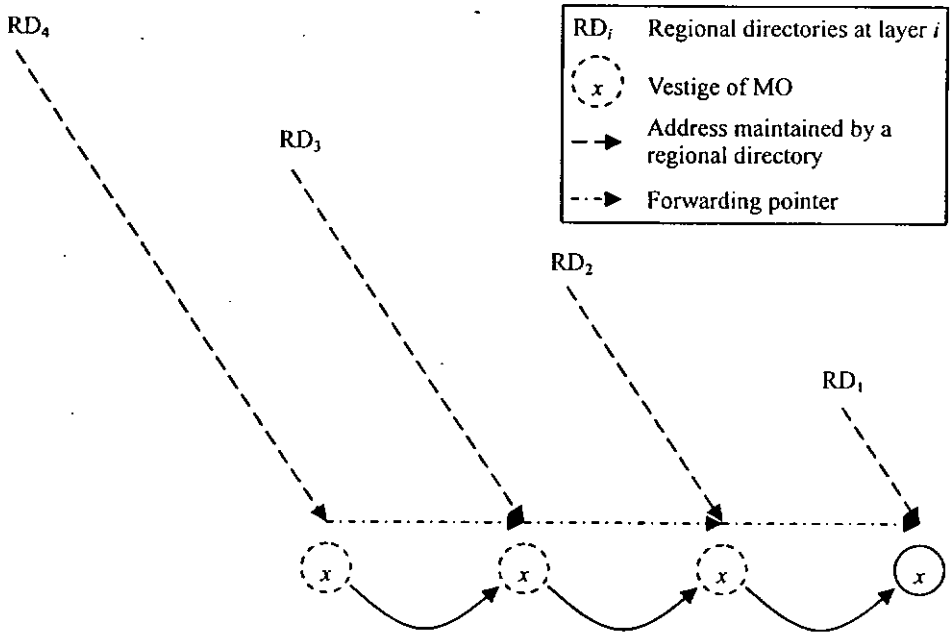


Figure 3. **The non-tree hierarchy scheme**

In this scheme [6], the location servers, specifically called *regional directories*, are organized in a non-tree hierarchy. In particular, a hierarchy of δ layers is built, where $\delta = \log d$ and d is the maximal distance between any two network locations. The

purpose of a regional directory at layer i is to enable a potential sender to find any receiver MO within the distance 2^i from it. Figure 3 illustrates the general structure of this scheme. Whenever a MO moves to a new location at a distance k away, only the log$k$ lowest layers of the hierarchy are updated to point directly to the new location. Entries at the higher layer regional directories will continue pointing to the old location with only a forwarding pointer left for the new one. When a sender wants to communicate with the receiver MO x at a distance k away, it will enquire the regional directories starting from the bottom layer till the layer log$k$ where x's (possibly) old location is found. Then it has to follow a chain of forwarding pointers before it can finally locate x. Interesting readers may refer to [6] for details.

In this scheme, more than one location servers may be included in the read set and the write set for each hierarchy layer, which to some extent deals with the single point of failure problem exhibited in the tree-hierarchy based scheme. However, it is much more complicated to implement and assumes a requirement not quite reasonable that MOs can retrieve the geographical location information from its working environment so as to calculate their traveling distance. Also the possible long chain of forwarding pointers required in this scheme will degrade the performance.

2.2 Message Delivery Schemes

As mentioned in Chapter 1, even under the assumption of a fault-free transmission mechanism, message loss may still occur due to the MO's mobility. It is possible that when the message arrives at its target location (where the MO resided previously), the MO has already left for another location. Various approaches have been proposed to overcome this problem.

2.2.1 Partially Reliable Schemes

2.2.1.1 Forwarding Pointer

Before each migration, a MO will leave a forwarding pointer in its current network location pointing to the target location. In this way, when a message is sent to an obsolete address of the receiver MO (this address can be obtained by any of the location enquiry strategies introduced in Section 2.1), the message can be routed along the forwarding pointer. The forwarding-pointer based scheme is often used in combination with the caching mechanism. An example is the *Internet Mobile Host Protocol* (IMHP) [23].

Although messages can be routed along the forwarding pointer, there is not an upper bound on the number of hops a message will take before it can reach the receiver. If the receiver migrates frequently, the messages may keep chasing the receiver and will never be received. Therefore, the forwarding-pointer based scheme can only partially overcome the message loss caused by the MO mobility.

2.2.1.2 Resending

To implement reliable message delivery for MOs, resending-based TCP-like protocols [24, 25, 26] are proposed. If a message is missed because of the migration of the receiver MO, the sender can detect the message loss and will resend the message to the new address of the receiver. By making use of the sliding window algorithm, these protocols can not only overcome message loss caused by the network faults and the receiver's migration, but also guarantee the *First In First Out* (FIFO) order of the message delivery. However, as in the forwarding-pointer based scheme, when the receiver migrates frequently, there will be no upper bound on the number of message

forwardings. Therefore, it cannot satisfy the requirement of reliability mentioned in Chapter 1.

2.2.2 Fully Reliable Schemes

2.2.2.1 Broadcast

In this scheme, if the sender maintains an obsolete address of the receiver MO, the messages sent to that address will not be able to reach the destination and will be broadcasted to all the network locations within the system. For example, in Emerald [27] broadcast is used to find an object if a node specified by a forwarding pointer is unreachable or has stale data. According to Murphy [28], however, the simple broadcast cannot avoid the message loss caused by the MO mobility. This happens when the MO is traveling along a path in the reverse direction with respect to the propagation of the broadcast messages, or more generally, when the MO moves from the region ahead of the broadcast propagation to a region behind the propagation. In such a situation, the MO and the broadcast messages might cross in a link instead of a host, and the broadcast messages can never be received. To deal with this problem, Murphy proposed a snapshot based broadcast to guarantee reliable message delivery to highly mobile objects. The basic idea is to force the MO out of the links and into regions from which they cannot escape without receiving a copy of the messages.

2.2.2.2 Synchronization

From the perspective of concurrency control, the message loss or the chasing problem in the message delivery process is caused by the concurrent and asynchronous access to the location information of the target MO. The MO's migration and the message delivery processes can be regarded as two types of database operations. The migration of the MO changes its actual address, which can be regarded as a write operation of

the location information. The message delivery process needs the target MO's actual address, which is in fact a read operation of the location information. Strategies are proposed to coordinate and synchronize the MO's migration and the message delivery so that messages can reach the target within bounded number of hops.

One widely used synchronization strategy is implemented as follows. Before each migration, the MO will inform all the sites that might send messages to its current address and waits for acknowledgement (ACK) from each site. The ACK will usually contain the information about the number of messages that have been sent out to the MO from that site. The MO then has to wait for these messages due to arrive. On the other hand, after replying the ACK, the site will suspend the message forwarding and may start buffering messages. After the MO finishes its migration, it will register its new address with those sites so that they can restart the message delivery to the correct location. Variations of the strategy are proposed. For instance, if the FIFO message order is maintained in the underlying transport layer, the ACK does not need to contain the information about the number of messages sent and the MO can leave for the target location as soon as it has collected all the ACKs. In the Mogent system [8], a synchronized home-server based protocol is proposed to track mobile agents and can guarantee the reliable message delivery.

The synchronization based scheme can guarantee that messages are routed to its target MO within bounded number of hops. However, the MO has to wait for the arrival of all the ACKs and the on-the-fly messages. This constraint to the MO mobility could be very prohibitive.

Chapter 3. A Novel Mailbox Based Scheme

In Chapter 2, we have reviewed the major schemes used for the location management and the message delivery. However, all of these schemes have their own assumptions, design goals and methodology, making it difficult to adapt to different application requirements. Therefore, new approaches need to be discovered of designing adaptive mobile communication protocols. In this chapter, we describe such an approach: a novel mailbox based scheme.

3.1 The Scheme

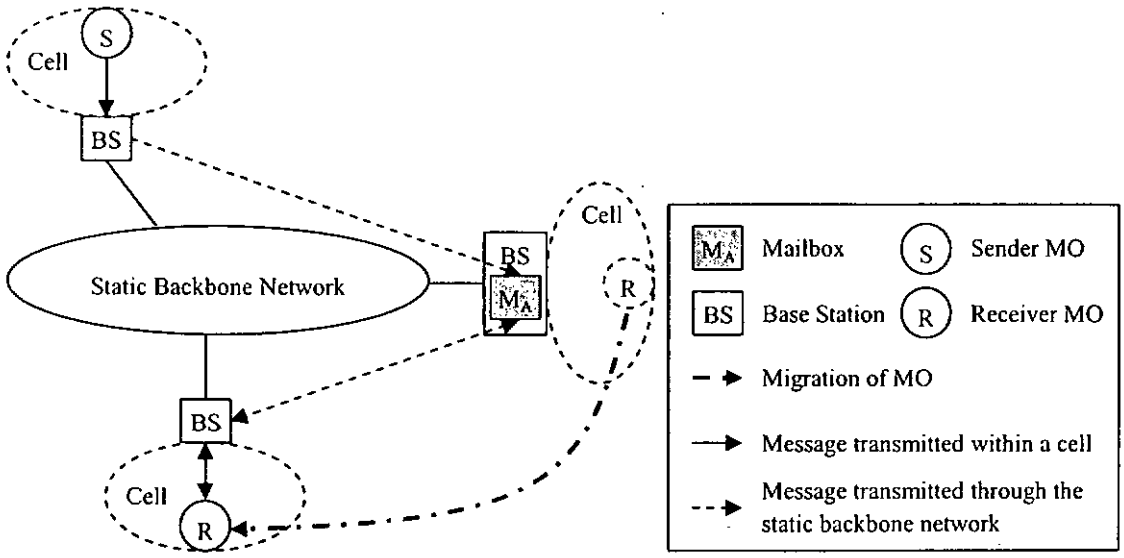


Figure 4. **The mailbox based scheme**

In our scheme, each MO is associated with a mailbox to buffer incoming messages for it, which is actually a data structure residing in a BS. As shown in Figure 4, the sender sends messages to the receiver's mailbox first, and later the receiver uses either a push or pull operation to obtain the messages from its mailbox. Although logically being part of the MO, the mailbox can be detached from its owner MO in the sense that the MO can leave its mailbox at one site while it is leaving for another site. During each migration, a choice can be made by the MO regarding whether to bring its mailbox

together with it to the new site or leave the mailbox at the current location. As a mailbox can move, it can also be viewed as a MO but without autonomy because its migration is controlled by its owner.

With the mailbox scheme, message passing between MOs consists of two parts: communication between the sender MO to the receiver MO's mailbox, and communication between the receiver MO and its mailbox. Existing mobile communication protocols can be applied to both parts. For the ease of exposition in the rest of the thesis, we make the following definitions.

Definition 1. The migration path of a MO A, denoted as $Path_a(A)$, is an ordered list of BSs $(h_{a0}, h_{a1}, \ldots, h_{an})$ whose supported cells have been visited by A in sequence. The set of BSs on the path is denoted as $S_a(A) = \{h_{ai} \mid h_{ai}$ is on $Path_a(A)$ where $1 \leq k \leq n\}$.

Definition 2. The migration path of the mailbox of a MO A, denoted as $Path_m(A)$, is an ordered list of BSs $(h_{m0}, h_{m1}, \ldots, h_{mn})$ that the mailbox has visited in sequence. The set of BSs on the path is denoted as $S_m(A) = \{h_{mi} \mid h_{mi}$ is on $Path_m(A)$ where $1 \leq k \leq n\}$. By definition, $S_m(A) \subseteq S_a(A)$ and $h_{m0} = h_{a0}$ since initially A and its mailbox are collocated within the *home cell*.

Definition 3. The function $f_A : S_a(A) \rightarrow S_m(A)$ maps from the location of a MO A to that of its mailbox so that for every $h_{ak} \in S_a(A)$, we have:

$$f_A(h_{ak}) = \begin{cases} h_{ak} & k = 0, \text{ or } k > 0 \text{ and A migrates with its mailbox} \\ f_A(h_{a(k-1)}) & k > 0 \text{ and A migrates without its mailbox} \end{cases}$$

3.2 A Three Dimensional Framework for Protocol Design

To meet the specific requirements of an application, the mailbox-based scheme offers choices in three aspects of the protocol design:

- **Mailbox Migration Frequency.** A MO might 1) always move alone, leaving its mailbox at the home cell with *No Migration* (NM); or 2) always migrate with its mailbox, resulting in a *Full Migration* (FM) pattern; or 3) determine dynamically upon each migration called *Jump Migration* (JM).
- **Mailbox-to-MO Message Delivery.** As mentioned before, messages destined to a MO are first sent to its mailbox and later received by the MO with either a push or pull operation. In the *Push* (PS) mode, the mailbox keeps the address of its owner and forwards every incoming message to it. In the *Pull* (PL) mode, the MO keeps the address of its mailbox and retrieves messages from it whenever needed.
- **Migration-Delivery Synchronization.** Users can determine whether they need reliable message delivery or not. If they need high reliability, message loss can be overcome by 1) *Synchronizing the Sender's message delivery with the Mailbox's migration* (SSM), or 2) *Synchronizing the Mailbox's message delivery with the Receiver's migration* (SMR), or 3) both, known as *Full Synchronization* (FS). The extreme case of *No Synchronization* is denoted as NS.

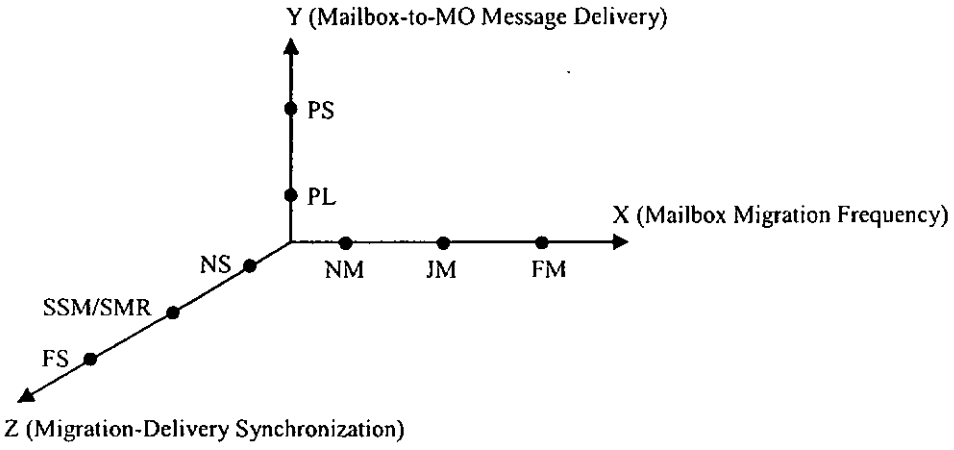


Figure 5. **The 3D design space**

We therefore have a three-dimensional model for protocol design space, as shown in Figure 5. Combining parameters from the three dimensions yields a taxonomy of mobile communication protocols. A string of the format XX-YY-ZZ expresses a protocol, in which XX represents the mailbox migration frequency (e.g., NM, JM or

FM), YY stands for the mailbox-to-MO message delivery (e.g., PL or PS), and ZZ symbolizes the migration-delivery synchronization (e.g., NS, SSM, SMR or FS). A protocol's overall configuration has a special value for each of the three parameters. The framework not only covers several well-known protocols but also allows for the design of new ones that can fulfill various application requirements. Interesting readers may refer to [29] for detailed discussions about the parameter combinations and the corresponding protocols.

3.3 Summary

In this chapter, we describe a three-dimensional framework for the design of adaptive mobile communication protocols. The framework uses a mailbox based scheme, which associates each MO with a mailbox while allowing the decoupling between them. Based on the framework, a taxonomy of mobile communication protocols can be developed, which makes it possible to describe or design a mobile communication protocol by simply selecting and combining parameters from all three dimensions. In the following two chapters, we will try to explore the benefits of the mailbox based scheme by respectively applying it to two different contexts: mobile agent and Mobile IP.

Chapter 4. A Mailbox Based Mobile Agent Communication Protocol

In this chapter, we are to illustrate how the mailbox based scheme can be applied to and benefit the design of mobile agent communication protocols. First, the background knowledge about mobile agent is introduced. The remaining part of this chapter focuses on ARP, an adaptive and reliable protocol derived from the mailbox based framework for message communication between mobile agents. We describe the ARP with proof of its properties and evaluation of its performance. A path pruning algorithm is proposed to improve the ARP with the reduced location management cost. The ARP's performance highly depends on the mailbox's migration policy and it uses a threshold to determine the mailbox's migration. Also the performance of the path pruning algorithm depends on the setting of certain parameters. We therefore propose an adaptive algorithm that can dynamically assign values to the threshold and some other parameters in order to achieve the optimal performance. Next, we propose a fault-tolerant architecture to enhance the resilience of the ARP. Lastly, an alternative way of choosing the optimal mailbox's migration policy is explored by making use of the dynamic programming [48].

4.1 Background

A mobile agent is essentially a program that is able to move autonomously around the network during its execution to finish the tasks assigned by its owner. Mobile agents have been recognized as a promising technology of developing applications in heterogeneous and distributed networking environments. Examples of applications include e-commerce, information retrieval, process coordination, mobile computing, personal assistance and network management. Specifically, mobile agents provide the following benefits of creating distributed systems [30].

- **Reduced network load and network latency.** Mobile agent allows a user to package a conversation and dispatch it to the destination where interactions can take place locally. In this way, remote interactions over the network decrease and thus the network load also decreases. By moving computation to data rather than data to computation, mobile agent can also reduce the flow of raw data in the network and therefore, overcome the network latency which is especially critical to real-time applications.

- **Protocol encapsulation.** Protocol enables the components of a distributed system to communicate and coordinate their activities. However, after a protocol evolves over a period of time, new features may be introduced into it. It is a cumbersome task to upgrade the protocol at all the locations in the distributed system. Mobile agent offers an ideal solution to this problem that it encapsulates the protocol into its code section. During the protocol upgrading, only the mobile agent is modified, cloned and dispatched to appropriate locations for execution.

- **Asynchronous and autonomous execution.** In traditional client-server systems, the client performing a task must always be available to receive and react to incoming messages. Therefore, a continuous connection between the client and the server is necessary. However, in mobile environment, wireless connection is often expensive and fragile, and the requirement of a continuous connection is probably neither economical nor practical. To solve this problem, we can embed the task in a mobile agent which is then dispatched into the network. After that, the mobile agent becomes independent of the process that created it, and can operate asynchronously and autonomously. The mobile device can disconnect the network as soon as the mobile agent has been dispatched, and later reconnect for the results produced by the mobile agent.

- **Dynamic adaptation.** Mobile agents can sense their execution environment (e.g., local computing environment, network bandwidth and local data information) and

react autonomously to changes. The ability of migration also enables mobile agents to dynamically distribute themselves within the network so as to maintain the optimal configuration of solving a particular problem.

- **Natural heterogeneity.** Network computing is fundamentally heterogeneous, often from both hardware and software perspectives. Since mobile agents are generally computer and transport-layer independent (dependent only on their execution environment), they provide a natural solution for seamless system integration.

- **Robustness and fault tolerance.** Mobile agent's ability of reacting dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems. For example, if a host is being shut down, all the mobile agents residing on that host are warned and dispatched to another host where they can resume their operations.

Many mobile agent systems have been developed. They differ mostly in application areas and thus have different emphases. For example, Telescript [31] is currently used in network management, active e-mail, e-commerce and business process management; Tacoma [32] is used in StormCast, a system for distributed weather simulation; *Mobile Service Agent* (MSA) [33] has been primarily used in "follow-me" computing, in which an application moves to the location of the user; Agent Tcl [34] has been widely used in information retrieval. Despite of all these differences, there is one fundamental feature, namely inter-agent communication that must be provided by any mobile agent system. This feature allows mobile agents to cooperate with each other by sharing and exchanging information, and collaboratively making decisions.

Most of the mobile agent systems use the home-server based scheme as their primary mechanism for the inter-agent communication. This is probably because for every mobile agent, there is an agent home associated with it which serves as an ideal

candidate for the home server. However, as mentioned in Chapter 2, the home-server based scheme cannot guarantee the reliable message delivery. The target mobile agent might be leaving for another host, while messages are being forwarded to it. To deal with this problem, the synchronization mechanism is necessary to coordinate the message forwarding and the agent's migration. Before each migration, the mobile agent sends a "DEREGISTER" message to its agent home and waits for a "REPLY" message. After the migration, the mobile agent registers its new address with its agent home by sending a "REGISTER" message. Messages destined to the mobile agent during its migration are blocked and buffered at the agent home, and forwarded to the new location after the migration.

Apart from the reliability problem, the triangle routing problem in this scheme increases the communication overhead, especially when there is a large number of mobile agents and frequent inter-agent communications. The agent home may become a performance bottleneck and a single point of failure. Therefore, a distributed scheme is needed to reduce the workload and the reliance on the agent home.

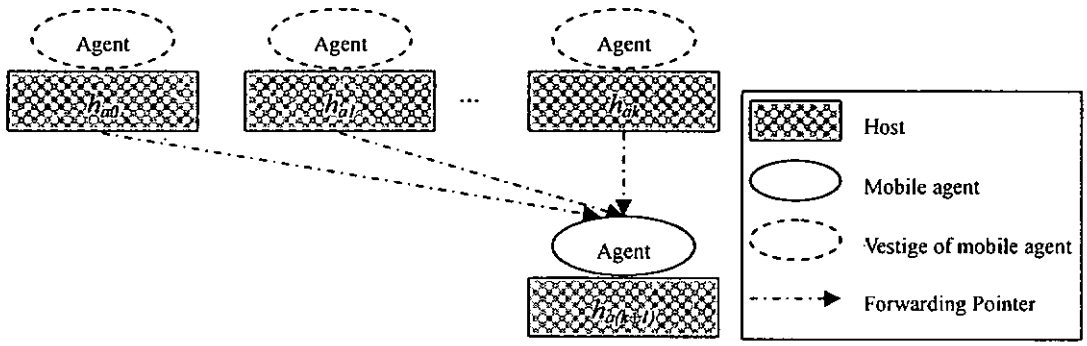


Figure 6. **The distributed-registration based scheme**

One way to distribute the workload on the agent home is to allow each host on the agent's migration path to maintain the most updated address of the mobile agent, which is called the *distributed-registration based scheme* [27, 35]. As shown in Figure 6, during each migration, the mobile agent synchronizes its location information with not only its agent home but also all the hosts on its migration path. The caching

mechanism is used here. When a sender wants to send a message to the agent A, it will first check whether or not A's address has already been cached locally. If so, it will send the message to the cached address, say h_{ak}; otherwise, the message will be sent to A's agent home, say h_{a0}. If A has moved elsewhere, h_{ak} or h_{a0} will forward the message to A's current address and notify the sender of this new address. Accordingly, the sender will update its cache.

Although the distributed scheme can effectively reduce the workload on the agent home, it incurs an unaffordable amount of location registration cost when the migration path of the mobile agent is long. Some techniques are therefore required to reduce the location registration cost while keeping the merits of the distributed scheme. In the next two sections, we will propose two such techniques: the basic mailbox based ARP and a path pruning algorithm as an improvement to the ARP.

4.2 ARP: An Adaptive and Reliable Protocol

One way to reduce the location registration cost is to apply our mailbox based scheme. In particular, we propose a new adaptive protocol called ARP, which represents the JM-PL-SSM combination of parameters. Since the mailbox will normally migrate at a relatively lower speed compared with its owner agent, both the location registration frequency (as the location registration now only occurs at each mailbox's migration instead of each agent's migration) and the number of messages required during each registration (since the mailbox's migration path is much shorter than the agent's migration path) can be reduced. Therefore, the total location registration cost will remarkably decrease.

4.2.1 The ARP

In the ARP, each host on $Path_m(A)$ maintains the current address of the mailbox M_A in

an *address table* which consists of five attributes: 1) the ID of M_A, 2) the current address of M_A, 3) a *valid* tag, 4) the number of messages, *mNum*, that have been forwarded to M_A, and 5) a message block queue for M_A. The *valid* tag is used to indicate whether the address of M_A is outdated or not. Later we will see that the *valid* tag switches to *false* only when M_A is under migration. The message block queue is used to temporarily buffer messages to M_A when it is moving. The protocol also defines operations for two processes, *Migrating* and *Message-forwarding*, which are presented in the next two subsections.

4.2.1.1 Migrating

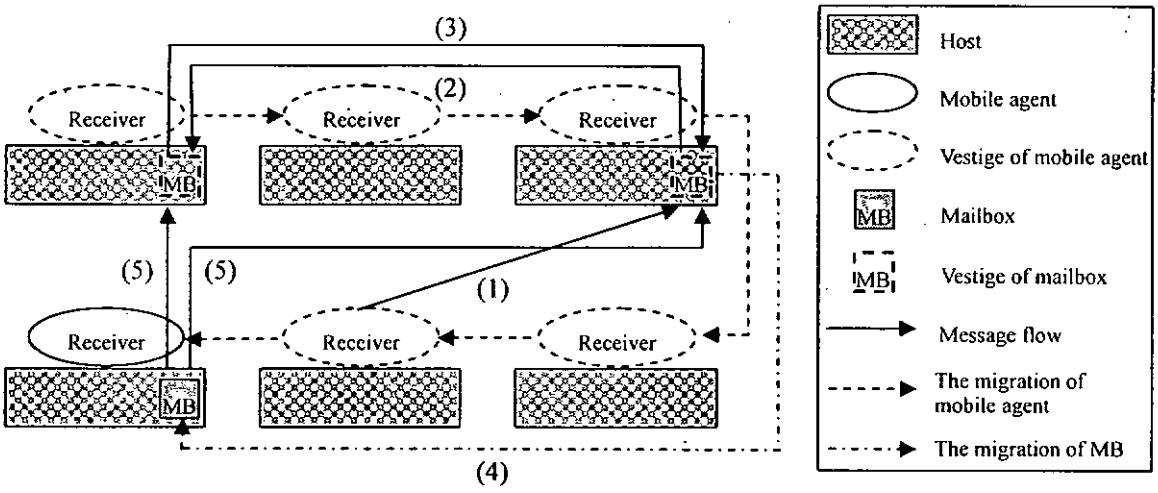


Figure 7. **The migrating process**

Figure 7 illustrates the mobile agent migrating process. Before moving to a new host h_{ak}, the agent A determines whether or not to take its mailbox M_A to the new location. If it decides to do so, it will send an "MVMB" message to M_A (step 1) informing it to migrate to h_{ak}. The "MVMB" message contains the address of h_{ak}. The pseudo code of this operation is shown in the function OnMigration_Agent() in Figure 8. In the pseudo code, the message sending function send() takes three parameters: the message type, the destination address and the content of the message.

```
OnMigration_Agent(){ // executed by the agent before moving
  if(f_A(h_ak) = h_ak)
    // the agent decides to move with its mailbox
    send("MVMB", f_A(h_a(k-1)), h_ak); // step 1 in Figure 7
  migrateTo(h_ak);
}
```

Figure 8. **OnMigration_Agent()**

As shown in Figure 9, on receiving the "MVMB" message, M_A will execute the function MVMBProcessing_MB(). It will send the "DEREGISTER" messages to all the hosts on $Path_m(A)$ (step 2) telling them to suspend the message forwarding and start to buffer incoming messages for it. It is not until M_A has collected all the "REPLY" messages (step 3) before it can prepare to move to the new location. Besides informing M_A about the reception of the "DEREGISTER" message, the "REPLY" message also carries information about the number of messages, *mNum*, that have been forwarded to M_A. In this way, M_A can tell whether or not there are still any data messages in transmission at the time it receives the "REPLY" message. If there do exist some, M_A has to also wait for these messages before it can finally perform the migration (step 4).

```
MVMBProcessing_MB(msg){ // executed by the mailbox before moving
  for(every host h_mj on Path_m(A))
    send("DEREGISTER", h_mj, null); // step 2 in Figure 7
  wait for all "REPLY" msgs and data msgs on the fly;
  // get the address of the target host: h_ak
  targetHost = msg.getContent();
  migrateTo(targetHost); // step 4 in Figure 7
}
```

Figure 9. **MVMBProcessing_MB()**

When M_A arrives at h_{ak}, it executes the function OnArrival_MB() as shown in Figure 10. It registers its new address by sending each host on $Path_m(A)$ a "REGISTER" message (step 5). This "REGISTER" message will restart the message forwarding.

```
OnArrival_MB(){ // executed by the mailbox after moving
  append h_ak to Path_m(A);
  for (every host h_mj on Path_m(A))
    send("REGISTER", h_mj, h_ak); // step 5 in Figure 7
}
```

Figure 10. **OnArrival_MB()**

The host is responsible for processing all the control messages. Its operation is shown in the function MessageProcessing_Host() in Figure 11.

```
MessageProcessing_Host(msg){ // executed by the host
  switch(msg.getKind()){
    case DEREGISTER:
      AgentID sender = msg.getSender();
      AddressEntry entry = addressTable.getAddr(sender);
      entry.VALID = false;
      // step 3 in Figure 7
      send("REPLY", msg.getSenderAddr(), entry.mNum);
    case REGISTER:
      AgentID sender = msg.getSender();
      AddressEntry entry = addressTable.getAddr(sender);
      if (entry == null){
        // "REGISTER" msg is from the local host, create a
        // new entry in the address table for the mailbox
        entry = new AddressEntry(sender);
        insert entry into the address table;
      }
      entry.address = msg.getContent();
      entry.VALID = true;
      while(there are messages in the block queue){
        Message blockedMsg = entry.blockQueue.getNextMsg();
        send("AGENTMSG", entry.address, blockedMsg);
        entry.mNum++;
        // increase the number of data msgs that
        // have been forwarded to the mailbox by 1
        send("UPDATE", blockedMsg.getSenderAddr(), entry.address);
        // update the address cached by the sender
      } // end of while
      entry.blockQueue.clear();
  } // end of switch
}
```

Figure 11. **MessageProcessing_Host()**

4.2.1.2 Message-forwarding

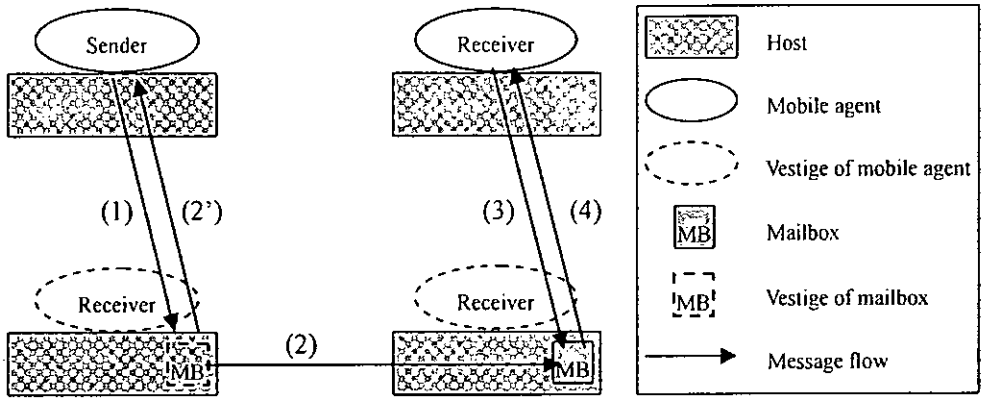


Figure 12. **The message-forwarding process**

Figure 12 illustrates the message-forwarding process. Suppose a mobile agent wants to send a message to the agent *A*. Referring to the function MessageSending_Sender() in Figure 13, it will first check *A*'s address from its cache. If it exists, the message will be sent to the cached address. Otherwise, it will be sent to *A*'s home.

```
MessageSending_Sender(msg){ // executed by the sender
  if (the receiver's address is in local cache)
    send("AGENTMSG", address in cache, msg);
  else
    send("AGENTMSG", the receiver's home address, msg);
}
```

Figure 13. **MessageSending_Sender()**

When a host receives a message destined to *A*, it will check from the address table whether M_A is residing locally or not. If so, it will directly insert the message into M_A. Otherwise, it will forward the message to M_A's current address (step 2), while at the same time, it will send back an "UPDATE" message to the sender (step 2a) to refresh its cache about M_A's current location. See the function MessageRouting_Host() in Figure 14 for details. Finally, the messages in M_A are received by the receiver (step 4) with periodic pull operations (step 3).

```
MessageRouting_Host(msg){ // executed by the host
  AgentID receiver = msg.getReceiver();
  if (the receiver's mailbox is local)
    insert msg into the mailbox;
  else{
    AddressEntry entry = addressTable.getAddress(receiver);
    if (entry.VALID){
      send("AGENTMSG", entry.address, msg); // step 2 in Figure 12
      mNum++;
      // increase the number of data msgs that
      // have been forwarded to the mailbox by 1
      send("UPDATE", msg.getSenderAddr(), entry.address);
      // step 2' in Figure 12
    }else{ // valid tag is false: mailbox is migrating
      entry.blockQueue.insert(msg);
      // insert the message into the block queue;
    }
  }
}
```

Figure 14. **MessageRouting_Host()**

4.2.2 Properties of the ARP

In this section, we will prove by argument the properties of the ARP, which illustrate that the ARP can fulfill the following requirements described in Chapter 1: location transparency, reliability, asynchrony and scalability. For the other two requirements, i.e., efficiency and adaptability, we will give detailed discussions in Section 4.2.3.

Theorem 1 (Location Transparency). With the ARP, a sender agent can send its messages without knowing where the target agent is currently residing.

Proof. According to the function MessageSending_Sender(), when the sender wants to send a message, it will first check whether or not the receiver's address has been cached locally. If there caches such address, it will send the message to this address directly without worrying about whether it is outdated or not. Otherwise, it will resolve the receiver's home address from the receiver's ID and send the message to it. In neither case, the sender has to specify the current location of the receiver whenever it wants to send a message. ■

Lemma 1. For all $h_{mi} \in Path_m(A)$, the *valid* tag of M_A in the address table is *true* only if the address reflects exactly the current location of M_A, i.e., the address kept in the address table is not outdated.

Proof. Suppose the address of M_A kept in h_{mi}'s address table is h_{mj}. If the *valid* tag is *true*, according to the function MessageProcessing_Host(), we can conclude that h_{mi} must have received M_A's "REGISTER" message from h_{mj}, while the "DEREGISTER" message has not yet arrived. So h_{mi} cannot send out the "REPLY" message to h_{mj}. According to the function MVMBProcessing_MB(), M_A cannot leave h_{mj} until it collects all the "REPLY" messages from the hosts on $Path_m(A)$ including h_{mi}. Therefore, the address h_{mj} kept in the address table of h_{mi} must reflect the current location of M_A. ■

Theorem 2 (Reliability). All the data messages can be delivered to the receivers' mailboxes by being forwarded at most once.

Proof. Suppose a sender S is sending a message m to a receiver A and A's mailbox M_A is locating at the host h_{mj}. According to the function MessageSending_Sender(), S will check get A's location, say h_{mi}, from its cache, and send the message m to it (if there is no record about A's address in the cache, m will be sent to A's home h_{m0}). When m arrives at h_{mi}, three cases could happen:

- $i = j$. The message m will be directly inserted into M_A without being forwarded.
- **$i < j$ and h_{mi} has not received M_A's "DEREGISTER" message from h_{mj}.** In this case, the *valid* tag for M_A is *true*. Therefore, m is immediately forwarded to h_{mj}. According to *Lemma 1*, M_A must now reside in h_{mj} and it cannot leave for $h_{m(j+1)}$ during the transmission of any data messages to it by referring to the function MVMBProcessing_MB(). So m must meet M_A at h_{mj}. We can see in this case, m is forwarded only once before reaching M_A.

- **$i < j$ and h_{mi} has received M_A's "DEREGISTER" message from h_{mj} while the "REGISTER" message from $h_{m(j+1)}$ has not yet arrived.** In this case, the *valid* tag for M_A is *false*. The message m will be put into the message block queue as shown in the function MessageRouting_Host(). It will not be forwarded until M_A reaches $h_{m(j+1)}$ and the "REGISTER" message arrives at h_{mi}. After that, m will be forwarded to $h_{m(j+1)}$ in the same way as discussed in the second case. Therefore, in this case, m is also forwarded only once.

From the above discussions of all the three cases, we can conclude that all the data messages can be delivered to the receivers' mailboxes by being forwarded no more than once. This property also guarantees the reliable message delivery: no matter how frequently the target agent migrates, messages can be routed to it in a bounded number of hops. ■

Theorem 3 (Asynchrony). With the ARP, constraints on agent's mobility are released since a mobile agent can migrate to a new place without waiting for the messages in transit.

Proof. Throughout the functions described in Section 4.2.1, we can see that a mobile agent does not need to perform any procedure before it migrates to a new location, i.e., the mobile agent requires no synchronization between its migration and the message forwarding. All the synchronization procedures are shielded by the mailbox, which definitely improves the mobility of the mobile agent. The only thing that the mobile agent needs to consider before the migration is when to move its mailbox. If it decides to do so, it will send a "MVMB" message to the mailbox and all the subsequent procedures of migration and synchronization will be accomplished by the mailbox. ■

There exists another advantage of relaying the synchronization duty to the mailbox. As we can see, it is not every mobile agent's migration that will trigger the mailbox's migration. Therefore, the migration frequency of the mailbox is much lower than that

of the mobile agent. The cost for conducting synchronization will dramatically drop comparing with the distributed-registration based scheme where synchronization is required each time the mobile agent migrates.

Theorem 4 (Scalability). The ARP is more scalable than both the home-server based scheme and the distributed-registration based scheme.

Proof. As mentioned before, the agent home in the home-server based scheme may become the performance bottleneck, especially in a system with a large number of mobile agents and frequent inter-agent communications. This is because the agent home has to handle all the location registration and message delivery processes. However, in the ARP, the workload on the agent home is remarkably alleviated and properly distributed to all the hosts on the mailbox's migration path. First, since not every mobile agent's migration will trigger the mailbox's migration, the frequency of location registration to the agent home is reduced. Second, the caching mechanism used in the ARP allows the message delivery traffic to be distributed to multiple forwarding hosts along the mailbox's migration path. Therefore, the ARP is more scalable than the home-server based scheme.

In the distributed-registration based scheme, it will incur an unaffordable amount of location registration cost when the migration path of the mobile agent is long, which means the scheme can only suitable for applications with short agent life cycle and limited number of migrations. In the ARP, the location registration occurs at each mailbox's migration which may be much infrequent than the agent's migration. Therefore, the ARP may be more scalable than the distributed-registration based scheme (depending on the mailbox's migration frequency). ■

However, as the mailbox's migration path getting longer, the ARP may suffer from the performance bottleneck. We will introduce in Section 4.3 a path pruning algorithm to maintain the set of hosts required for the location registration within an affordable

size no matter how long the mailbox's migration path really is. In this way, our proposed protocol can become truly scalable.

4.2.3 Performance Evaluation

In this section, we evaluate the ARP's performance through simulation experiments. The performance metric used is the communication cost incurred by the migrating and message-forwarding procedures. The cost is characterized by the number of messages sent and the message size.

4.2.3.1 Simulation Model

(1,6)	(2,6)	(3,6)	(4,6)	(5,6)	(6,6)
(1,5)	(2,5)	(3,5)	(4,5)	(5,5)	(6,5)
(1,4)	(2,4)	(3,4)	(4,4)	(5,4)	(6,4)
(1,3)	(2,3)	(3,3)	(4,3)	(5,3)	(6,3)
(1,2)	(2,2)	(3,2)	(4,2)	(5,2)	(6,2)
(1;1)	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)

Figure 15.**Network topology setting**

Our simulation is based on the *Network Simulator 2* (NS2) [36] developed by the Lawrence Berkeley National Laboratory. We have incorporated the ARP into the simulator. The network configuration of the simulation is shown in Figure 15. There are altogether 36 hosts, each of which is associated with a coordinate (x, y). They are all interconnected with each other. As the two most important arguments required by NS2, the bandwidth and the propagation delay between any two hosts are configured as follows. Any two hosts in adjacent are considered as if they are within a local LAN environment which has a bandwidth of 10Mbps and a propagation delay of 2ms. The two hosts with the largest distance, i.e., the host (1, 1) and the host (6, 6) are treated as

if they reside on each half of the earth with a small bandwidth of 64Kbps and a large propagation delay of 100ms. To make it simple, we assume that both the bandwidth and the propagation delay between any two hosts are linearly distributed according to their distance. So we can derive the bandwidth and the propagation delay between host A and B as follows:

$$\frac{\text{Bandwidth}(A,B)-64\text{Kbps}}{\text{Dist}(A,B)-1}=\frac{10\text{Mbps}-\text{Bandwidth}(A,B)}{6\sqrt{2}-\text{Dist}(A,B)}\Rightarrow$$
$$\text{Bandwidth}(A,B)=(\text{Dist}(A,B)-1)\frac{10\text{Mbps}-64\text{Kbps}}{6\sqrt{2}-1}+64\text{Kbps} \quad (1)$$

$$\frac{\text{Delay}(A,B)-2\text{ms}}{\text{Dist}(A,B)-1}=\frac{100\text{ms}-\text{Delay}(A,B)}{6\sqrt{2}-\text{Dist}(A,B)}\Rightarrow$$
$$\text{Delay}(A,B)=(\text{Dist}(A,B)-1)\frac{100\text{ms}-2\text{ms}}{6\sqrt{2}-1}+2\text{ms} \quad (2)$$

$$\text{Dist}(A,B)=\sqrt{(A(x)-B(x))^2+(A(y)-B(y))^2} \quad (3)$$

Let us first define the parameters used in our simulation model as shown in Table 1. Here the receiver mobile agent and its mailbox are denoted as A and M_A, respectively.

Table 1. **Parameters for our simulation model**

Parameter	Description
S	the set of senders in the network that might send messages to A
t_{s_i}	the interval that the sender $s_i \in S$ sends two consecutive messages to A
$f_{s_i}(t)$	we assume a negative exponential probability distribution function of the message sending interval of s_i
$p_{s_i}(t,n)$	the probability distribution (Poisson) function of the number of messages sent from s_i during the time interval t
λ_{s_i}	the mean message sending rate of s_i, i.e., $f_{s_i}(t)=\lambda_{s_i}e^{-\lambda_{s_i}t}$, $p_{s_i}(t,n)=(\lambda_{s_i}t)^n e^{-\lambda_{s_i}t}/n!$
t_r	the residence time A spends in a host
$f_r(t)$	we also assume a negative exponential probability distribution function of an agent's residence time at a site
$1/\mu$	the mean residence time, i.e., $f_r(t)=\mu e^{-\mu t}$

In the simulation, A sequentially migrates from the host (1, 1) to the host (6, 6), while it actively communicates with 5 senders which are randomly distributed within the system. A periodically pulls messages from M_A. The message pulling rate is set to 1.5 times the mean message arrival rate ($5\lambda_{s_i}$) so that messages can be delivered in time. Since control messages such as "MVMB" and "REGISTER" may be much smaller in size than data messages, they should not be counted in the same way. We thus uniformly denote the cost of a control message as a normalized value 1 and the cost of a data message as a relative value 2.

Till now, we have not discussed the criteria that determine the migration of M_A. Actually, there are many such criteria that A might consider. For example, if A seldom receives messages from others at the new host h_{ai}, it does not need to take M_A with it. On the other hand, if A is expected to frequently receive messages from others while h_{ai} is far away from $f_A(h_{a(i-1)})$, leaving M_A unmoved will incur a lot of cost on the suboptimal route of message delivery (sender $\rightarrow M_A \rightarrow A$) and it would be more efficient for A to collocate with M_A. In our simulation study, we use a fixed threshold T on the number of messages to make the decision. Before the ith migration, A estimates the number of messages it will receive at h_{ai}, which is the multiplication of the mean message arrival rate ($5\lambda_{s_i}$) and the mean residence time ($1/\mu$). If the estimated number exceeds the threshold T, it will send the "MVMB" message to fetch the mailbox. Otherwise, it will migrate alone.

4.2.3.2 Simulation Results

Three parameters can affect the performance of the ARP: λ_{s_i}, μ and T. We pay extra attention to T since it can be adjusted dynamically according to the network environment. We design the following experiments to visualize its impact.

In Section 4.2.1, we have described that the ARP consists of two processes –

migrating and message-forwarding, and we know that these two processes cannot be optimized at the same time [6]. Therefore, we separately measure the cost of these processes. The total communication cost is the summation of the costs of these processes.

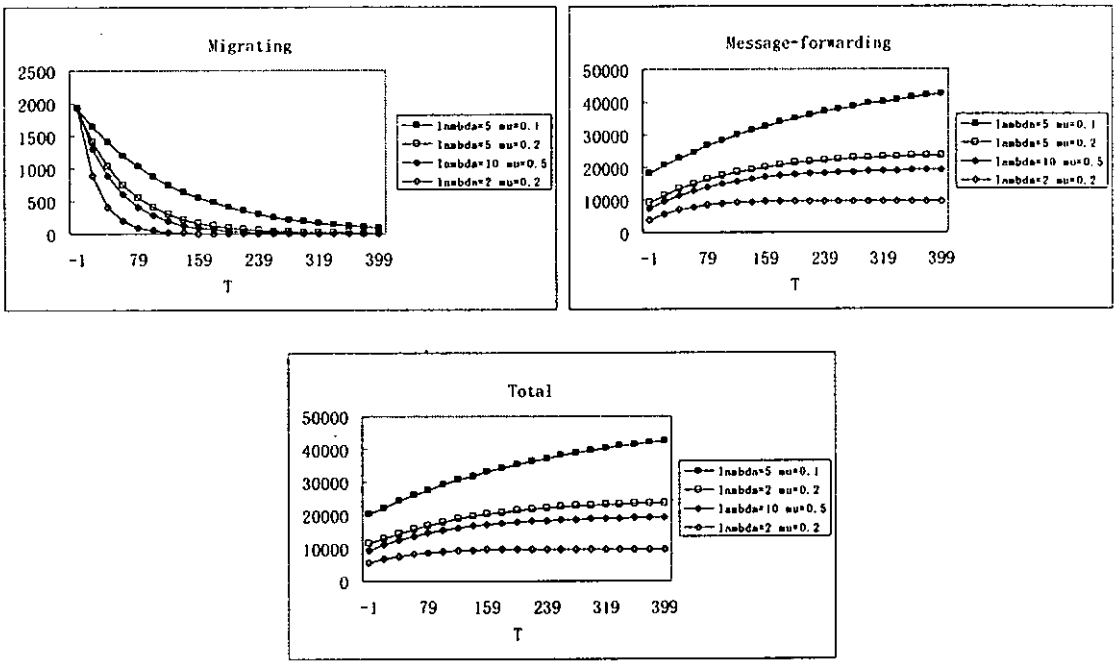


Figure 16. **The performance of the ARP – Scenario 1**

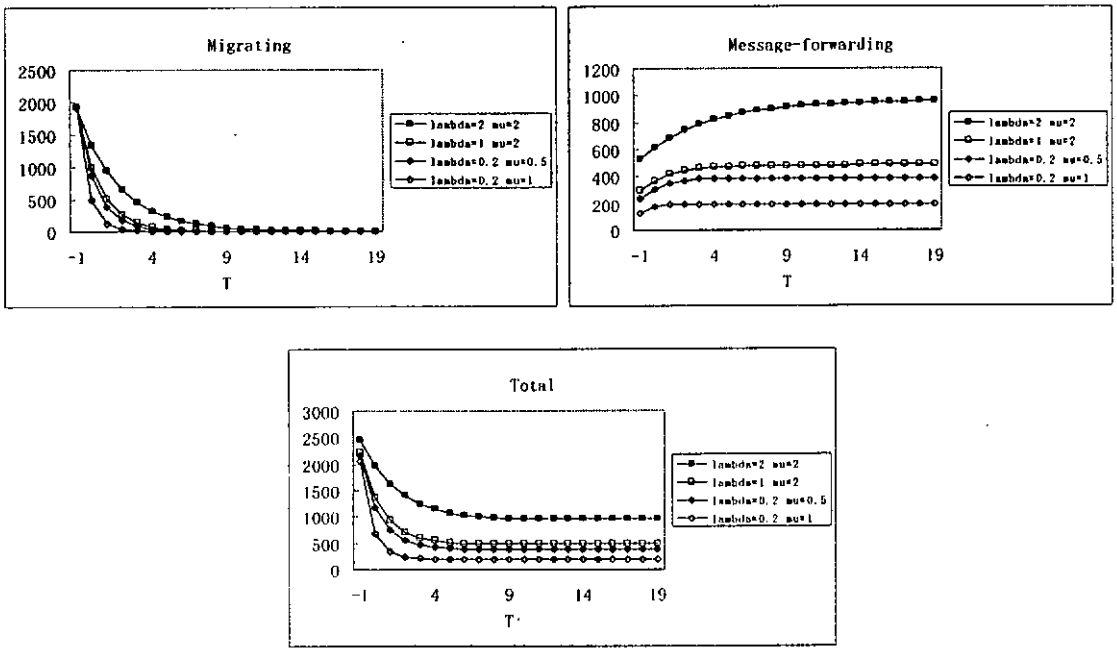


Figure 17. **The performance of the ARP – Scenario 2**

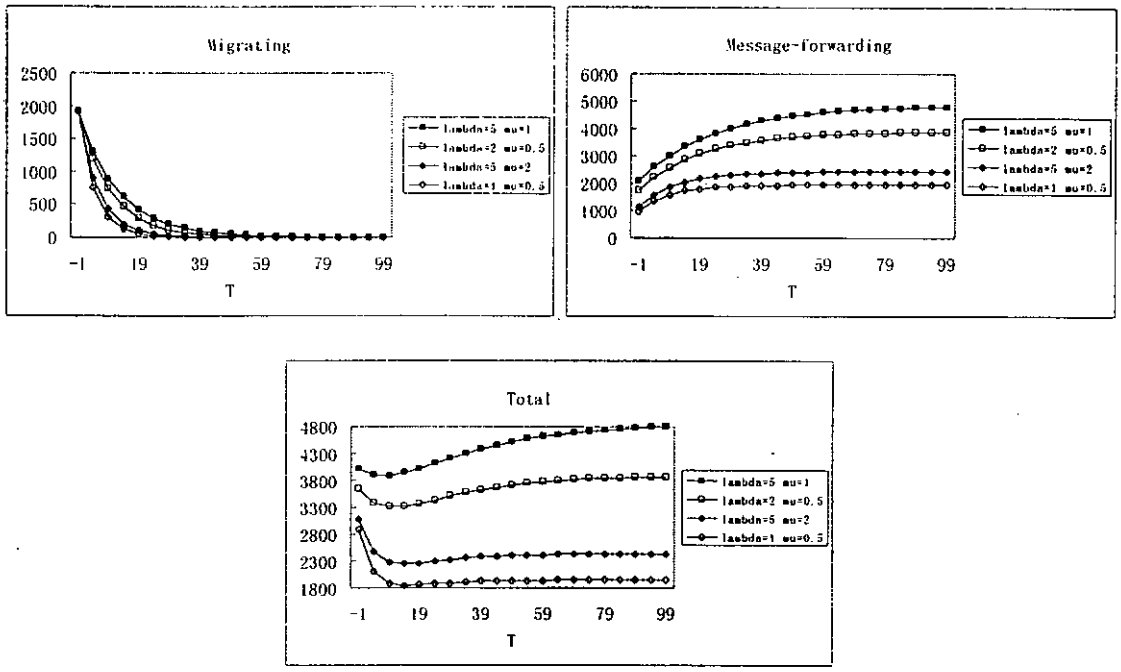


Figure 18. **The performance of the ARP – Scenario 3**

Figures 16 – 18 show three sets of experimental results obtained under three different scenarios with respect to different (λ_{s_i} , μ) pairs. The three scenarios exhibit the three possible locations of the optimal value of T and will be explained in detail soon. Note that both the home-server based scheme and the distributed-registration based scheme can actually be viewed as special cases of our ARP by assigning ∞ and -1 as the value of T, respectively. This is because when $T = \infty$, it is impossible to have the expected number of messages exceed the value of T, and the mailbox will have to stay at the agent home, which results in the home-server based scheme. However, when $T = -1$, the threshold condition is always met and the mailbox should always move with its owner agent, which is quite similar to the distributed-registration based scheme. Therefore, we can view the right-hand side of the curves in the above figures as the (approximate) performance of the home-server based scheme and the left-hand side as that of the distributed-registration based scheme.

We observe that in all settings, the cost of the migrating process decreases, while the cost of the message-forwarding process increases, as T increases. This is because

when T is small, it is more likely that the mailbox collocates with its owner, which results in an optimal route between the sender and the receiver. Therefore, the message-forwarding cost is low. However, the frequent migration of the mailbox consequentially increases the migrating cost. As T becomes larger, the migrating cost drops while the message-forwarding cost increases because of the triangle routing. It is reasonable to expect that there exists an optimal value of T where the total communication cost reaches its minimum. There are three possible locations of the optimal T, which are respectively exhibited in the above figures. It is possible that the message-forwarding cost dominates the total cost so that the decrement of the migrating cost cannot remedy the increment of the message-forwarding cost. This happens when there is dense message sending but sparse agent's migration. In this case, the best location of T is -1. It is also possible that the migrating cost takes a relatively higher weight than the message-forwarding cost. This happens when there is sparse message sending but dense agent's migration. In this case, the best location of T is ∞. The last case is when both the migrating cost and the message-forwarding cost share relatively the same importance. In this case, the total cost will drop first and rise later, and therefore, the best location of T is somewhere in the middle between -1 and ∞. Should an adaptive algorithm be figured out to calculate the optimal T, the ARP must perform no worse than both the home-server based scheme and the distributed-registration based scheme. We will discuss this adaptive algorithm in Section 4.4.

4.3 A Path Pruning Algorithm

Although the ARP has already significantly cut down the location registration cost, it may suffer from performance problems when the mailbox's migration path becomes longer. Therefore, the ARP is better for applications with short agent life cycle and limited number of migrations. For applications with frequent and long-term agent's migration, techniques must be figured out so as to maintain the set of hosts required

for registration within an affordable scale no matter how long the mailbox's migration path really is.

Fortunately, we find that since the cache maintained by the sender is updated whenever it sends a message to an outdated address of M_A, it is very likely that each sender will refer to the latest host on $Path_m(A)$ as the current address of M_A. The hosts on the front of $Path_m(A)$ may no longer be referred to by any sender after certain period. These hosts can be safely removed from $Path_m(A)$ to reduce the registration overhead.

Another important issue is the management of the cache. If an agent S sends messages to an agent A, the host in which S resides will cache the address of A's mailbox M_A. However, when S leaves the host, the address of M_A in the cache may no longer be used by any other agent. On the other hand, after the death of A, its mailbox's address cached by other hosts will also become useless. Some garbage collection mechanism is needed to remove these useless addresses in the cache to save space.

In this section, we propose an algorithm to prune the mailbox's migration path and also to remove useless addresses maintained in a cache.

4.3.1 The Algorithm

We first define the terminologies, which will be used in the following discussions.

Definition 4. Let H denote the set of all the hosts in the network and A denote a mobile agent. The function $R_A : H \rightarrow S_m(A)$, which defines the place where a host will forward a message to A, can be expressed as follows:

$$R_A(h_s) = \begin{cases} h_{mk} & h_{mk} \in S_m(A) \text{ and } h_{mk} \text{ is cached by } h_s \text{ as the current address of } M_A \\ h_{m0} & \text{otherwise} \end{cases}$$

Definition 5. The set $S_R(A, h_{mi}) = \{h_s \mid R_A(h_s) = h_{mi}\}$ contains all the hosts that refer to h_{mi} as the current address of M_A. In other words, if a host h_s sends a message to A through h_{mi}, it must belong to $S_R(A, h_{mi})$.

Definition 6. For each host $h_{mi} \in S_m(A)$ and $i > 0$, h_{mi} is called a *redundant host* in $S_m(A)$ if it will no longer receive any messages destined to A unless M_A revisits it.

By definition, we know that redundant hosts can be safely removed from $S_m(A)$ without affecting the reliability of message delivery. Therefore, the objective of our path pruning algorithm is to identify and remove those redundant hosts from $S_m(A)$. We first extend the data structure of the ARP as follows.

- Each cached address is associated with a timer which is initially set to 0 and starts as soon as the address is added to the cache. Each time the address is accessed (either updated or requested by the sender), the timer resets to 0. When the timer reaches TTL, the address is removed.

- Each host $h_{mi} \in S_m(A)$ maintains a reference table $T(A)$ for the agent A which consists of a set of addresses and a *closed* tag. Later in *Lemma 3*, we will show that $T(A)$ maintains the addresses of those hosts in the set $S_R(A, h_{mi})$. The table is created when M_A visits h_{mi} but $T(A)$ does not exist yet. On creation, the address set is empty, the *closed* tag is set to *false* and the timer is in an idle state. Besides, a timer is associated with $T(A)$ and is activated when the host h_{mi} receives the "REGISTER" message for the first time.

Besides, we extend the migrating process and the message-forwarding process of the ARP as follows.

- When $h_{mi} \in S_m(A)$ receives M_A's "REGISTER" message from h_{mj} $(i \neq j)$, it will

check the reference table *T(A)*. If *T(A).closed* is *true*, nothing will be done. Otherwise, h_{mi} sets *T(A).closed* to *true* and starts the timer associated with *T(A)*. Both *T(A)* and the record of M_A in the address table are removed from h_{mi} as soon as *T(A)* becomes empty or *T(A).timer* reaches TTL+MTL, where MTL denotes the maximum message transmission latency of the network.

- When a host receives a message destined to *A* from *S* residing at the host h_s, it will first check the reference table *T(A)*. If *T(A)* has already been removed from h_s due to either of the reasons described in the last paragraph, it will forward the message to *A*'s home which always maintains the current address of M_A. (The agent home should not perform the path pruning algorithm. Otherwise, it would be possible for the agent home not knowing the mailbox's location.) Otherwise, it will check *T(A).closed*. If *T(A).closed* is *false* and h_s is not in *T(A)*, h_s is added to *T(A)*. If *T(A).closed* is *true* and h_s is in *T(A)*, h_s is removed from *T(A)*.

- When $h_{mi} \in S_m(A)$ receives M_A's "DEREGISTER" message sent from h_{mj} $(i \neq j)$ and finds that the record of M_A has been removed from its address table, it will send back an "NAK" message, instead of the "REPLY" message, to h_{mj}. Upon receiving the "NAK" message, M_A removes h_{mi} from its migration path $Path_m(A)$.

The path pruning algorithm can be integrated into the original ARP as shown in the following pseudo code where the instructions in boldface represent the modifications.

```
OnArrival_MB(){ // executed by the mailbox after moving
  append h_ak to Path_m(A);
  // The creation of the referenced table at a new host
  RefTable t = new RefTable();
  for (every host h_mj on Path_m(A))
    send("REGISTER", h_mj, h_ak); // step 5 in Figure 7
}
```

Figure 19. **OnArrival_MB() extension**

```
MessageProcessing_Host(msg){ // executed by the host
  switch(msg.getKind()){
    case DEREGISTER:
      AgentID sender = msg.getSender();
      AddressEntry entry = addressTable.getAddr(sender);
      // check to remove the reference table
      RefTable t = getRefTable(sender);
      if (msg.getSenderAddr() != localHost){
        if ((t is empty) or (t.timer > TTL+MTL)){
          // send "NAK" instead of "REPLY"
          send("NAK", msg.getSenderAddr(), entry.mNum);
          addressTable.remove(sender);
          removeRefTable(t);
          return;
        }
      }
      entry.VALID = false;
      // step 3 in Figure 7
      send("REPLY", msg.getSenderAddr(), entry.mNum);
    case REGISTER:
      AgentID sender = msg.getSender();
      AddressEntry entry = addressTable.getAddr(sender);
      if (entry == null){
        // "REGISTER" msg is from the local host, create a
        // new entry in the address table for the mailbox
        entry = new AddressEntry(sender);
        insert entry into the address table;
      }
      entry.address = msg.getContent();
      entry.VALID = true;
      // the handling of "REGISTER" message
      RefTable t = getRefTable(sender);
      if ((msg.getSenderAddr() != localHost) and
          (t.closed == false)){
        t.closed = true;
        t.timer.start();
      }
      while(there are messages in the block queue){
        Message blockedMsg = entry.blockQueue.getNextMsg();
        send("AGENTMSG", entry.address, blockedMsg);
        entry.mNum++;
        // increase the number of data msgs that
        // have been forwarded to the mailbox by 1
        send("UPDATE", blockedMsg.getSenderAddr(), entry.address);
        // update the address cached by the sender
      } // end of while
      entry.blockQueue.clear();
    // the handling of "NAK" message
    case NAK:
      delete msg.getSenderAddr() from Path_m(A);
  } // end of switch
}
```

Figure 20. **MessageProcessing_Host() extension**

```
MessageRouting_Host(msg){ // executed by the host
  AgentID receiver = msg.getReceiver();
  RefTable t = getRefTable(receiver);
  if (t != null){
    if (the receiver's mailbox is local)
      insert msg to the mailbox;
    else{
      AddressEntry entry =
          addressTable.getAddress(receiver);
      if (entry.VALID){
        send("AGENTMSG", entry.address, msg);
        // step 2 in Figure 12
        mNum++;
        // increase the number of data msgs that
        // have been forwarded to the mailbox by 1
        send("UPDATE", msg.getSenderAddr(), entry.address);
        // step 2' in Figure 12
      }else{ // valid tag is false: mailbox is migrating
        entry.blockQueue.insert(msg);
        // insert the message to the block queue;
      }
    }
    if ((t.closed == false) and (msg.getSenderAdd() ∉ t))
      t.add(msg.getSenderAddr());
    if ((t.closed == true) and (msg.getSenderAdd() ∈ t))
      t.remove(msg.getSenderAddr());
  }else
    send("AGENTMSG", the receiver's home address, msg);
}
```

Figure 21. **MessageRouting_Host() extension**

4.3.2 Correctness of the Algorithm

Here we present an informal proof of the correctness of the algorithm. First, we present two basic assumptions. Later we will show in the simulation that with a minor revision, the algorithm can work even without these assumptions, although the performance is slightly degraded.

Assumption 1. The message transmission latency of the network is no larger than MTL.

Assumption 2. The interval that a sender sends two consecutive messages destined to

the same receiver agent is no less than 2MTL.

Lemma 2. For any host $h_{mi} \in S_m(A)$ and $i > 0$, after a host h_s is removed from the table $T(A)$, no message to the agent A from h_s will be sent to h_{mi}.

Proof. According to the path pruning algorithm, h_s is removed from $T(A)$ only when h_{mi} receives a message destined to the agent A from h_s but finds out that the *closed* tag of $T(A)$ is *true*. h_{mi} will send an "UPDATE" message to h_s informing it of the new address of M_A. By *Assumptions 2*, h_s must have received the "UPDATE" message from h_{mi} and updated its cache before sending the next message. In other words, $R_A(h_s)$ must have changed to the new address of M_A when h_s sends the next message to A. ■

Lemma 3. For all the hosts $h_s \in S_R(A, h_{mi})$, either h_s is already in $T(A)$, or it will be added to $T(A)$ within MTL.

Proof. Suppose currently M_A is residing at h_{mi}, the sender agent is residing at h_s, and $R_A(h_s) = h_{mj}$ $(i \neq j)$. h_s will send the message destined to the agent A to h_{mj}. According to our algorithm, h_{mj} will forward the message to h_{mi} and send an "UPDATE" message to h_s. Upon receiving the "UPDATE" message, h_s will update its cache and let $R_A(h_s) = h_{mi}$ (therefore, we have $h_s \in S_R(A, h_{mi})$). After h_{mi} receives the data message from h_{mj}, it will add h_s to the reference table $T(A)$. If the arrival of the data message at h_{mi} is earlier than the arrival of the "UPDATE" message at h_s, h_s would have been added to $T(A)$ before it updates its cache. Otherwise, since the "UPDATE" message and the data message are sent out by h_{mj} at almost the same time and the transmission time of each message is no larger than MTL, we can easily reach the conclusion that h_s will be added to $T(A)$ within MTL after $R_A(h_s)$ turns to h_{mi}, i.e., $h_s \in S_R(A, h_{mi})$. ■

Lemma 4. No new hosts will join the set $S_R(A, h_{mi})$ after the *closed* tag of $T(A)$ turns to *true*.

Proof. Turning the *closed* tag of $T(A)$ to *true* implies that M_A has left h_{mi} for the host $h_{m(i+1)}$ and h_{mi} has received the "REGISTER" message from M_A at $h_{m(i+1)}$. Therefore, all the hosts $h_{mj} \in S_m(A)$ must have received the "DEREGISTER" message from M_A at h_{mi}. Either M_A's address maintained in the address table of h_{mj} has been updated (h_{mj} has also received the "REGISTER" message from M_A at $h_{m(i+1)}$), or the *valid* tag of M_A's address in the address table of h_{mj} is *false*. In neither case will h_{mj} return to any message sender an "UPDATE" message containing h_{mi} as the current address of M_A. Therefore, no new hosts will join $S_R(A, h_{mi})$. ■

Theorem 5. h_{mi} is a redundant host in $S_m(A)$ if the *closed* tag of $T(A)$ is *true* and the address set in $T(A)$ is empty.

Proof. By *Lemma 3* and *Lemma 4*, if the *closed* tag of $T(A)$ is *true* and the address sent in $T(A)$ is empty, we have $S_R(A, h_{mi}) = \varnothing$. From *Lemma 2*, we know that h_{mi} will no longer receive any message destined to the agent A unless M_A revisits it. Therefore, h_{mi} is a redundant host in $S_m(A)$. ■

Theorem 6. h_{mi} is a redundant host in $S_m(A)$ if the timer of $T(A)$ reaches TTL+MTL.

Proof. If the timer of $T(A)$ has reached TTL+MTL, we know the *closed* tag of $T(A)$ is *true*. This is because the *T(A).timer* is started only after the *closed* tag of $T(A)$ is set *true*. If $T(A)$ is empty, we have proved that h_{mi} is a redundant host (*Theorem 5*). Otherwise, suppose h_s is in $T(A)$. According to our algorithm, we know that h_{mi} has not received any messages to the agent A from h_s since the *closed* tag of $T(A)$ has turned *true*. This implies that:

- The address of M_A cached by h_s, if not being cleared, is h_{mi} (i.e., $R_A(h_s) = h_{mi}$) and it has not been updated since the *closed* tag of $T(A)$ turns *true*.
- The address of M_A cached by h_s has not been requested for a period of TTL since the *closed* tag of $T(A)$ turns *true*. This is because the address of M_A cached by h_s is requested only if there are messages sent from h_s to the agent A. Since the

period of TTL+MTL has passed and the transmission time of a message is less than MTL, we know that there have been no messages sent from h_s to the agent A for, at least, a period of TTL.

From the above two points, we know the physical address of M_A in the cache of h_s has been neither updated nor requested for at least the period of TTL. Therefore, h_s must have removed the address of M_A from its cache. By definition, we have $h_s \notin S_R(A, h_{mi})$ and h_s can be safely removed from $T(A)$. In this way, we can safely empty $T(A)$. By *Theorem 5*, we know that h_{mi} is a redundant host. ■

Proofs of *Theorem 5* and *Theorem 6* depend on both *Assumption 1* and *Assumption 2*. Without these assumptions, messages to the agent A may arrive at h_{mi} even after h_{mi} has been removed from $S_m(A)$. In this case, we let h_{mi} forward the message to h_{m0}, i.e., the home of the agent A. Since h_{m0} should always know the physical address of M_A, it can finally forward the message to M_A. Although messages may be forwarded once more to reach the target agent and the workload on A's home increases, the reliability of message delivery can be maintained. Since the address in a cache is removed if it is not accessed within TTL, the algorithm also provides a way to clear useless addresses maintained in the cache.

The value of TTL affects the performance of the path pruning algorithm greatly. If TTL is very large, the probability that a sender cannot find a receiver's address in the cache is small and there is small increment of the workload on the receiver's home. However, the redundant hosts on the migration path of the receiver's mailbox may not be removed in time and we cannot achieve much reduction in the location registration cost. On the other hand, with a small TTL, the location registration cost can be greatly reduced by path pruning, but more messages must be forwarded by the receiver's home. Therefore, how to select the best value of TTL remains a problem to be discussed in Section 4.4.

4.3.3 Performance Evaluation

In this section, we evaluate the performance of the path pruning algorithm. We adopt the same simulation model as that used in the performance evaluation of the ARP except for the new parameter MTL being set to 1sec.

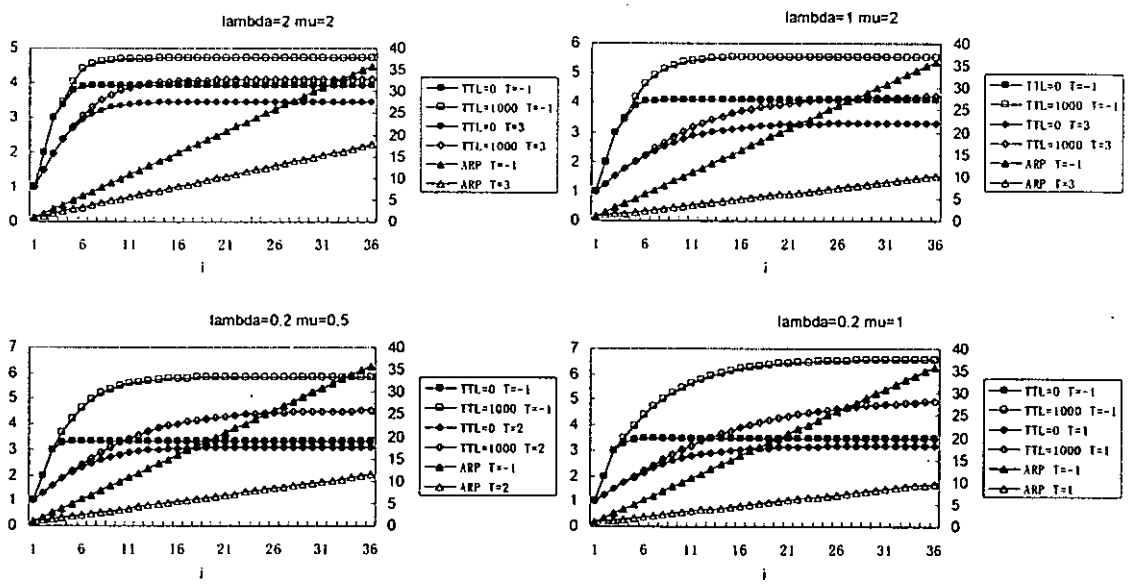


Figure 22. **The length of the mailbox's migration path**

First of all, we try to look at how effectively the algorithm works, as the goal of the path pruning algorithm is to remove the redundant hosts along the mailbox's migration path. Therefore, we conduct experiments to produce Figure 22, which shows the amount of path reduction the algorithm makes. The x-axis is the number of migrations and the twin y-axis is the length of the mailbox's migration path maintained by the mailbox. The left-hand side y-axis is for the path pruning algorithm while the right-hand side y-axis is for the ARP. We have intentionally chosen the same set of (λ_{s_i} , μ) pairs as in Figure 17 where the cost of the migrating process dominates the total cost, i.e., the path length is long. The first observation is that the algorithm greatly reduces the path length and makes it maintain at a small scale, no matter how many migrations the mobile agent has made. Even when TTL is very large, the algorithm can effectively identify and remove redundant hosts. This is not like what

we originally expect, that is, the performance of the algorithm will degrade as TTL increases until no pruning is performed when TTL approaches ∞. The reason for this phenomenon is due to the second condition that removes a host out of the mailbox's migration path: *T(A)* becomes empty. Even when TTL is ∞, *T(A)* will become empty as long as every sender maintained in *T(A)* sends a message to the agent after the mailbox migrates to a new host (as referred to the process extension for details).

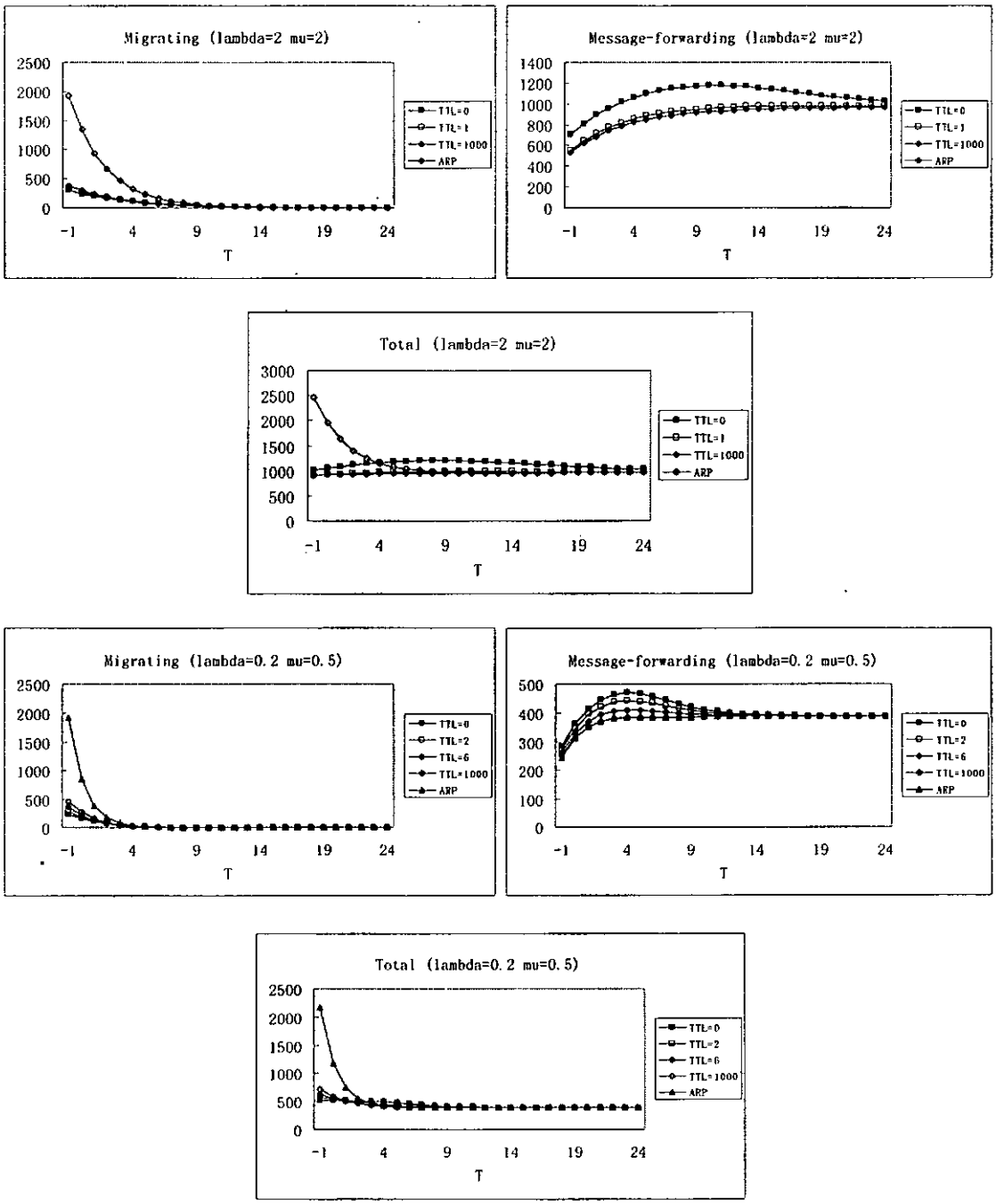


Figure 23. **The effect of *T***

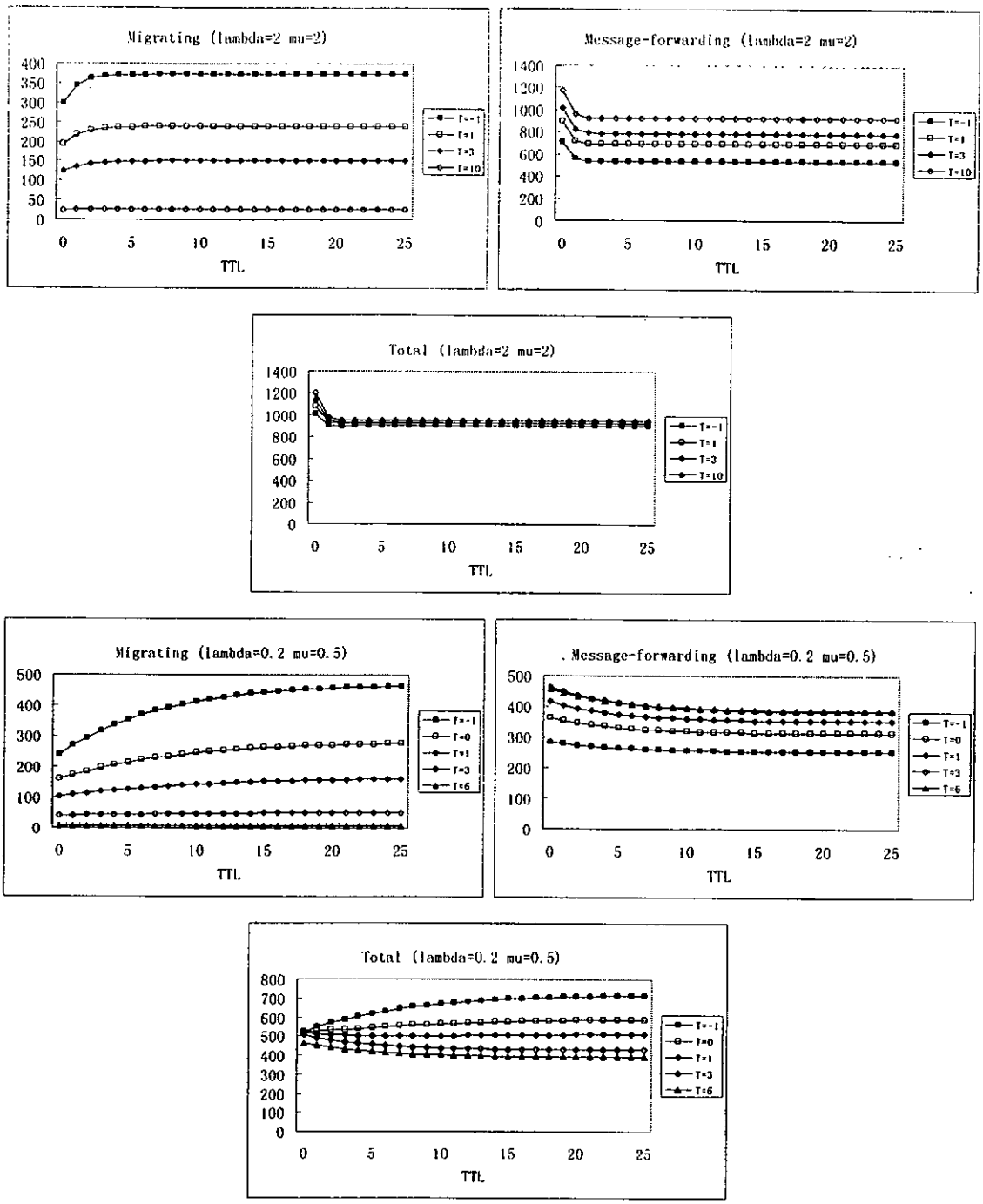


Figure 24. **The effect of TTL**

Finally, we plot Figure 23 and Figure 24 show the difference between the path pruning algorithm and the ARP. In Figure 23, we use T as the x-axis and discover that the path pruning algorithm dramatically reduces the migrating cost, especially when T is small. For the message-forwarding cost, it is interesting to note that when TTL is small, the curve increases first and then drops. The reason is as follows. Let us assume

TTL is 0, i.e., the sender always refers to the agent home for the message forwarding. When T is small, the mailbox collocates with its owner and requires two times of message forwarding (sender → agent home → (mailbox) receiver) for sending a message. As T increases, the mobile agent starts to migrate alone and therefore needs three times of message forwarding (sender → agent home → mailbox → receiver). When T continues to increase, the mailbox does not migrate any more and just stays at home. In this case, two times of message forwarding (sender → agent home (mailbox) → receiver) are necessary. However, when TTL is large, the sender can make use of its cache to record the current location of the mailbox so that it is more likely that the sender could send the message directly to the mailbox instead of going through the agent home. When TTL is approaching ∞, the message-forwarding cost coincides with that of the ARP. Finally, should T and TTL be properly chosen, the total cost with the path pruning algorithm would outperform the ARP. Figure 24 shows how TTL affects the cost. In this figure, TTL is used as the x-axis. For the same T, the migrating cost increases as TTL increases, while at the same time, the message-forwarding cost drops. The reason has been explained before.

4.4 An Adaptability Algorithm

In the previous sections, we have shown that by properly choosing the values of T and TTL, the ARP with path pruning could perform optimally in terms of the total cost. In this section, we develop an adaptive algorithm to find out the optimal values of both T and TTL.

4.4.1 The Algorithm

Basically, the adaptive algorithm consists of two parts. First, we develop an analytical model for the ARP with path pruning with T and TTL being parameters. Second, based on the analytical model, we try to find a method to derive the optimal values of

T and TTL so that the analytical model can produce the best result.

4.4.1.1 Analytical Model

First of all, let us define some new parameters used in the analytical model:

- p_i: the probability that A will not take M_A to the new location during its ith migration, i.e., $p_i = \text{Prob}(f_A(h_{ai}) = f_A(h_{a(i-1)}))$.
- p_{hit}: the probability that the mailbox's location information cached by the sender is correct.
- p_a: the probability that the forwarding host has not been prematurely removed from the migration path of M_A.

By removing the redundant hosts, the migration path maintained by M_A is reduced. To differentiate from the actual M_A's migration path $S_m(A)$, we create another symbol $S_p(i)$ to denote the set of hosts maintained by M_A after the ith migration of A. Therefore, in the ARP, we have:

$$S_p(i) = \begin{cases} S_p(i-1) & i > 0 \text{ and } f_A(h_{ai}) = f(h_{a(i-1)}) \\ S_p(i-1) \cup \{h_{ai}\} & i > 0 \text{ and } f_A(h_{ai}) = h_{ai} \end{cases}$$

and in the ARP with path pruning, we have:

$$S_p(i) = \begin{cases} S_p(i-1) & i > 0 \text{ and } f_A(h_{ai}) = f(h_{a(i-1)}) \\ S_p(i-1) \cup \{h_{ai}\} - R(i) & i > 0 \text{ and } f_A(h_{ai}) = h_{ai} \end{cases}$$

where $R(i)$ denotes the set of redundant hosts identified during the ith migration of A.

If A takes M_A to its target host in the ith migration, the migration overhead involves the communication cost of the following messages: "MVMB", "DEREGISTER", "REPLY" or "NAK", and "REGISTER". Otherwise, the migration overhead is zero. Therefore, the overhead of A's ith migration can be denoted as:

$$\begin{aligned} C_{mig}(i) = \; & (1-p_i)(p_{i-1}C_{mvmb} + (|S_p(i-1)|-1)C_{deregister} + \\ & (|S_p(i-1)|-1)C_{reply/nak} + (|S_p(i)|-1)C_{register}) \end{aligned} \quad (4)$$

where $|S_p(i)|$ means the number of hosts existing in the set $S_p(i)$. The total migration

overhead during the life cycle of A is then given by:

$$C_{mig} = \sum_{i=1}^{N} C_{mig}(i) \tag{5}$$

where N is the total number of migrations performed by A.

While A is residing at the host h_{ai}, the cost of message delivery from the sender s_j involves the cost of message passing from s_j to M_A and then from M_A to the receiver:

$$C_{del}(i, j) = n_{i,j}(C_{s_j \to M_A}(i) + C_{M_A \to receiver}(i)) \tag{6}$$

where n_{ij} denotes the number of messages sent from s_j to A when it is residing at h_{ai}. The average value of n_{ij} is λ_{s_j}/μ. For the cost of message passing from s_j to M_A, it might only require one message if a cache hit occurs, or it might require two messages if a cache miss occurs but still the cached host is included in $S_p(i)$, or it might require three messages if the cached host has been prematurely removed as a redundant host from $S_p(i)$:

$$\begin{aligned} C_{s_j \to M_A}(i) = \; & p_{hit} C_{s_j \to M_A} + (1 - p_{hit} - p_a)(C_{s_j \to h_{mk}} + C_{h_{mk} \to M_A}) + \\ & p_a (C_{s_j \to h_{mk}} + C_{h_{mk} \to h_{m0}} + C_{h_{m0} \to M_A}) \end{aligned} \tag{7}$$

For the cost from M_A to the receiver, we have:

$$C_{M_A \to receiver}(i) = p_{i-1}(C_{M_A \to receiver} + \alpha C_{query}) \tag{8}$$

where α is the ratio of the number of query messages to the number of messages obtained from M_A. The total message delivery cost is then given by:

$$C_{del} = \sum_{i=1}^{N} \sum_{j=1}^{M} C_{del}(i, j) \tag{9}$$

where M is the number of mobile agents that might send messages to A..

The total communication cost is therefore given by:

$$C_{total} = C_{mig} + C_{del} \tag{10}$$

As mentioned in Section 4.2.3.1, we use a fixed threshold T number of messages to determine the migration of the mailbox. Therefore, we have:

$$p_i = \text{Prob}(f_A(h_{ai}) = f_A(h_{a(i-1)})) = \text{Prob}(\sum_{j=0}^{M} n_{i,j} \le T) \tag{11}$$

We know that the message generation of each sender follows a Poisson distribution. To simplify the analysis, we assume that the combination of the message generation of all the senders also follow a Poisson distribution, i.e., the incoming messages seen by the mailbox follow a Poisson distribution with the mean message incoming rate $M\lambda_{s_j}$. Therefore, (11) can be rewritten as:

$$\begin{aligned} p_i &= \int_{t_r=0}^{\infty} \left(\sum_{n=0}^{T} \frac{(M\lambda_{s_j} t_r)^n}{n!} e^{-M\lambda_{s_j} t_r}\right) f_r(t_r) dt_r \\ &= \int_{t_r=0}^{\infty} \left(\sum_{n=0}^{T} \frac{(M\lambda_{s_j} t_r)^n}{n!} e^{-M\lambda_{s_j} t_r}\right) \mu e^{-\mu t_r} dt_r \\ &= 1 - \left(\frac{M\lambda_{s_j}}{M\lambda_{s_j} + \mu}\right)^{T+1} \end{aligned} \tag{12}$$

Since p_i has the same value for a fixed threshold T, for all i, we simply use p to denote the probability, i.e., for all i, we have $p_i = p$.

Now let t_h denote the residence time of the mailbox seen by an incoming message. Since the mobile agent's migration is assumed to be a Poisson process and the probability that the mobile agent will move its mailbox is $1 - p$, the migration of the mailbox is a stream split from the agent's migration with the probability $1 - p$. So the migration of the mailbox is also a Poisson process and the residence time is negative-exponentially distributed with the mean value $1/(1 - p)\mu$. From the residual lifetime property [37], the probability distribution function of the residence time t_h is represented as $f_h(t_h) = (1-p)\mu e^{-(1-p)\mu t_h}$. So we can represent p_{hit} as:

$$\begin{aligned} p_{hit} &= \text{Prob}(t_{s_j} < t_h)\text{Prob}(t_{s_j} < \text{TTL}) \\ &= \int_{s_j=0}^{\infty} f_{s_j}(t_{s_j}) \int_{h=t_{s_j}}^{\infty} f_h(t_h) dt_h dt_{s_j} \times \int_{s_j=0}^{TTL} f_{s_j}(t_{s_j}) dt_{s_j} \\ &= \int_{s_j=0}^{\infty} \lambda_{s_j} e^{-\lambda_{s_j} t_{s_j}} \int_{h=t_{s_j}}^{\infty} (1-p)\mu e^{-(1-p)\mu t_h} dt_h dt_{s_j} \times \int_{s_j=0}^{TTL} \lambda_{s_j} e^{-\lambda_{s_j} t_{s_j}} dt_{s_j} \\ &= \frac{\lambda_{s_j}}{\lambda_{s_j} + (1-p)\mu} \times (1 - e^{-\lambda_{s_j} TTL}) \end{aligned} \tag{13}$$

Suppose before the ith migration of A, M_A has made g number of migrations, i.e., the actual migration path of M_A contains $g + 1$ hosts (h_{m0}, h_{m1}, ..., h_{mg}) where h_{m0} is the home of A. Some of these $g + 1$ hosts have been removed as redundant hosts while others remain on the path $S_p(i)$ maintained by M_A. According to the path pruning algorithm, we know that for any host except for h_{m0} and h_{mg}, there are two reasons for it being removed from $S_p(i)$: either the timer of its reference table reaches MTL + TTL, or the addresses maintained in the reference table is empty. Therefore, the probability that h_{mk} ($0 < k < g$) has been removed from the actual migration path of M_A after the ith migration of A can be represented as follows:

$$p_{remove}(k) = 1 - (1 - p_{timerexpire}(k))(1 - p_{emptyaddr}(k)), \quad (14)$$

where $p_{timerexpire}(k)$ denotes the probability due to the timer expiration and $p_{emptyaddr}(k)$ denotes the probability due to the empty address list.

According to the path pruning algorithm, the timer for the reference table in h_{mk} is started when it first receives a "REGISTER" message from the other host. In other words, the timer is started after M_A moves from h_{mk} to $h_{m(k+1)}$. Therefore, we can express $p_{timerexpire}(k)$ as:

$$p_{timerexpire}(k) = Prob(\sum_{x=k+1}^{g} t_h(x) > MTL + TTL), \quad (15)$$

where $t_h(x)$ means the residence time of M_A at h_{mx}. We know that $t_h(x)$ follows a negative exponential distribution. According to [37], the random variable $t_{sum} = \sum_{x=k+1}^{g} t_h(x)$ has a k-Erlang distribution with the probability density function:

$$F_h(t_{sum}) = \frac{(1-p)\mu}{(g-k-1)!}((1-p)\mu t_{sum})^{g-k-1} e^{-(1-p)\mu t_{sum}} = \frac{((1-p)\mu)^{g-k}}{(g-k-1)!} t_{sum}^{g-k-1} e^{-(1-p)\mu t_{sum}} \quad (16)$$

Similarly, $t'_{sum} = \sum_{x=k}^{g} t_h(x)$ has the probability density function:

$$F'_h(t'_{sum}) = \frac{(1-p)\mu}{(g-k)!}((1-p)\mu t'_{sum})^{g-k} e^{-(1-p)\mu t'_{sum}} = \frac{((1-p)\mu)^{g-k+1}}{(g-k)!} t'^{g-k}_{sum} e^{-(1-p)\mu t'_{sum}} \quad (17)$$

Now we can derive:

$$
\begin{aligned}
p_{\text{timerexpire}}(k) &= 1 - \text{Prob}\Big(\sum_{x=k+1}^{g} t_h(x) < \text{MTL} + \text{TTL}\Big) \\
&= 1 - \text{Prob}(t_{sum} < \text{MTL} + \text{TTL}) \\
&= 1 - \frac{((1-p)\mu)^{g-k}}{(g-k-1)!} \int_{sum=0}^{MTL+TTL} t_{sum}^{\,g-k-1} e^{-(1-p)\mu t_{sum}} dt_{sum}
\end{aligned} \tag{18}
$$

Suppose the address of the sender s_a is in the reference table of h_{mk}. This implies that s_a sends at least one message to A during the period when M_A resides in h_{mk} and after M_A leaves for other hosts, no messages are sent from s_a. Therefore, the probability that s_a exists in the reference table can be expressed as:

$$
\begin{aligned}
p_{exist}(s_a) &= (1 - p_{s_a}(t_h(k),0)) p_{s_a}(t_{sum},0) \\
&= \frac{\lambda_{s_a}}{\lambda_{s_a} + (1-p)\mu} \int_{sum=0}^{\infty} p_{s_a}(t_{sum},0) F_h(t_{sum}) dt_{sum} \\
&= \frac{\lambda_{s_a}}{\lambda_{s_a} + (1-p)\mu} \int_{sum=0}^{\infty} e^{-\lambda_{s_a} t_{sum}} \frac{((1-p)\mu)^{g-k}}{(g-k-1)!} t_{sum}^{\,g-k-1} e^{-(1-p)\mu t_{sum}} dt_{sum} \\
&= \frac{\lambda_{s_a}}{\lambda_{s_a} + (1-p)\mu} \frac{((1-p)\mu)^{g-k}}{(g-k-1)!} \int_{sum=0}^{\infty} t_{sum}^{\,g-k-1} e^{-(\lambda_{s_a} + (1-p)\mu) t_{sum}} dt_{sum}
\end{aligned} \tag{19}
$$

Thus, we have:

$$
p_{\text{emptyaddr}}(k) = \prod_{x=1}^{M} (1 - p_{exist}(s_x)) \tag{20}
$$

After resolving $p_{remove}(k)$, we can now express $|S_p(i)|$ as:

$$
|S_p(i)| = \begin{cases}
1 & i = 0 \\
|S_p(i-1)| & i > 0, \text{ and } M_A \text{ does not migrate during A's ith migration} \\
3 + \sum_{k=1}^{g-1} (1 - p_{remove}(k)) & i > 0, \text{ and } M_A \text{ migrates during A's ith migration}
\end{cases} \tag{21}
$$

where the term 3 in the last case means the three hosts h_{m0}, h_{mg} and $h_{m(g+1)}$ that must exist in $S_p(i)$.

During the period when A resides at the host h_{ai}, the sender s_j might send messages to A. However, the forwarding host cached by s_j might have already been removed from $S_p(i)$, which will cause once more message forwarding to the home of A. To derive the

probability of such premature removal of the forwarding host, we use the following equation:

$$1-p_a=\sum_{k=1}^{g-1}\text{Prob}(s_j \text{ caches } h_{mk} \text{ as the forwarding host})p_{remove}(k) \tag{22}$$

In order that s_j caches h_{mk} as the forwarding host, the last message s_j sent to A is during the period when M_A resides in h_{mk} and the time interval must be within TTL. Otherwise, the cached address is expired and s_j will refer to h_{m0} as the forwarding host. So we can further express (22) as:

$$p_a=\sum_{k=1}^{g-1}\text{Prob}(\sum_{x=k+1}^{g}t_h(x)<t_{s_j}<\sum_{x=k}^{g}t_h(x))\text{Prob}(t_{s_j}<\text{TTL})p_{remove}(k), \tag{23}$$

where we apply the same technique as in (15) to solve the above equation:

$$\begin{aligned}\text{Prob}(\sum_{x=k+1}^{g}t_h(x)<t_{s_j}<\sum_{x=k}^{g}t_h(x)) &= \text{Prob}(t_{s_j}>\sum_{x=k+1}^{g}t_h(x))-\text{Prob}(t_{s_j}>\sum_{x=k}^{g}t_h(x))\\ &= \text{Prob}(t_{s_j}>t_{sum})-\text{Prob}(t_{s_j}>t'_{sum})\end{aligned} \tag{24}$$

$$\begin{aligned}\text{Prob}(t_{s_j}>t_{sum}) &= \int_{t_{sum}=0}^{\infty}F_h(t_{sum})\int_{t_{s_j}=t_{sum}}^{\infty}f_{s_j}(t_{s_j})dt_{s_j}dt_{sum}\\ &= \frac{((1-p)\mu)^{g-k}}{(g-k-1)!}\int_{t_{sum}=0}^{\infty}t_{sum}^{\,g-k-1}e^{-(\lambda_{s_j}+(1-p)\mu)t_{sum}}dt_{sum}\end{aligned} \tag{25}$$

$$\begin{aligned}\text{Prob}(t_{s_j}>t'_{sum}) &= \int_{t'_{sum}=0}^{\infty}F'_h(t'_{sum})\int_{t_{s_j}=t'_{sum}}^{\infty}f_{s_j}(t_{s_j})dt_{s_j}dt'_{sum}\\ &= \frac{((1-p)\mu)^{g-k+1}}{(g-k)!}\int_{t'_{sum}=0}^{\infty}t'^{\,g-k}_{sum}e^{-(\lambda_{s_j}+(1-p)\mu)t'_{sum}}dt'_{sum}\end{aligned} \tag{26}$$

$$\text{Prob}(t_{s_j}<\text{TTL})=1-e^{-\lambda_j\text{TTL}} \tag{27}$$

Finally, because $f(h_{a(i-1)}) = h_{mg}$, we have g as the number of migrations the mailbox has performed before the agent's ith migration. Since the probability of the mailbox's migration is $1-p$, we conclude that g is binomially distributed with the function:

$$p(i-1,g)=C_{i-1}^{g}(1-p)^g p^{i-1-g}, 1<g<i-1, \tag{28}$$

where the expectation of g is $(1-p)(i-1)$.

Till now, we have completely constructed the analytical model of the ARP with path pruning. Based on the model, we now try to derive the optimal values of T and TTL

so as to minimize the total communication cost. Before each migration, the mobile agent decides whether or not it should move with its mailbox by estimating the number of messages it will receive at the new host and comparing it with T. If the number is smaller than T, it will do nothing and simply migrate to the new place. Otherwise, it will calculate the new optimal values of T and TTL according to the current network and communication circumstances. It will also embed the new value of TTL into the "MVMB" message. The mailbox learns the new value of TTL and will propagate this knowledge to all the hosts on $S_p(i)$ by including it in the "REGISTER" messages. Finally, all the senders will learn this new TTL after receiving the "UPDATE" message from its corresponding forwarding host. In this way, all the relevant entities learn the most updated values of T and TTL and will work in an optimal manner.

4.4.1.2 Threshold Optimization

There are several alternatives to derive the optimal T and TTL. First, we can pre-construct a lookup table stored in every host within the system. Once a new pair of values is required, the mobile agent can simply lookup that table. This method has low computational complexity and thus causes minimal delay for a mobile agent's migration. The downside of this method is that it requires extra space for storing the lookup table in each host.

We can also dynamically calculate these two values as follows. Since T and TTL can only be discrete values, the total communication cost is not a continuous function of these parameters. Thus it is not appropriate to take derivatives with respect to T and TTL. We can use an iterative algorithm similar to the one proposed in [38] although it may result in a local minimum. The algorithm starts with $T = -1$ and TTL = 0. We iteratively increase T by d_t until the cost difference between the systems with the current and previous (T, TTL) pairs starts to be positive, i.e., T has reached its optimal value. We then apply the same technique again to find out TTL's optimal value by

each time increasing TTL by d_{ttl}. The value of d_t and d_{ttl} could be adjusted according to the accuracy required since smaller d_t and d_{ttl} imply greater accuracy but require more iterations. Here we empirically choose both d_t and d_{ttl} as the number of iterations performed. The mathematical presentation of the algorithm is as follows:

- Define the cost difference function:

  $$\Delta(T, TTL, T', TTL') = C_{total}(T, TTL) - C_{total}(T', TTL'),$$

  where (T,TTL) and (T',TTL') represents the current and previous values of T and TTL, respectively.

- Initialization: $T = -1$; TTL = 0;

- Step 1:

  while $\Delta(T + d_t, TTL, T, TTL) < 0$

  $T = T + d_t$;

  $T_{optimal} = T$;

- Step 2:

  while $\Delta(T_{optimal}, TTL + d_{ttl}, T_{optimal}, TTL) < 0$

  $TTL = TTL + d_{ttl}$;

  $TTL_{optimal} = TTL$;

4.4.2 Performance Evaluation

In this section, we evaluate the effectiveness of the adaptive algorithm under different communication and migration patterns by comparing the optimal total communication cost derived from this algorithm with that obtained from the experiments. From Table 2, we can conclude that the developed adaptive algorithm is sufficiently good to be implemented into the operating system of the network host for the dynamic calculation of the optimal (T, TTL) pair.

Table 2. **Comparison of the analytical value with the simulation value**

(λ, μ)	$(T_{optimal}, TTL_{optimal})$		Total Communication Cost		
	Simulation	Analytical	Simulation	Analytical	Difference
5,2	(-1,1)	(-1,1)	1363.76	1409.81	2.57%
5,1	(-1,3)	(-1,3)	2448.65	2362.82	-0.04%
5,0.5	(0,3)	(-1,3)	4139.10	4212.57	3.99%
5,0.2	(0,3)	(-1,3)	9202.43	9659.69	6.13%
5,0.1	(0,4)	(-1,3)	18176.00	18679.45	0.35%
2,2	(-1,4)	(-1,3)	908.26	898.42	6.80%
2,1	(0,3)	(-1,3)	1339.29	1295.26	5.88%
2,0.5	(0,8)	(-1,6)	2212.70	2071.55	2.89%
2,0.2	(0,12)	(-1,15)	4581.41	4318.81	-1.21%
2,0.1	(0,22)	(-1,28)	8116.89	7971.66	-1.53%
1,2	(27,1)	(27,0)	476.23	486.58	4.83%
1,1	(13,1)	(-1,3)	880.07	912.04	6.74%
1,0.5	(12,4)	(-1,6)	1361.91	1309.18	6.14%
1,0.2	(5,11)	(-1,15)	2561.28	2472.04	5.30%
1,0.1	(-1,30)	(-1,36)	4374.20	4341.56	4.35%
0.2,2	(6,0)	(9,0)	99.12	97.28	-4.10%
0.2,1	(12,1)	(14,0)	197.30	194.57	2.73%
0.2,0.5	(15,2)	(20,0)	389.55	389.69	-3.97%
0.2,0.2	(10,4)	(-1,10)	926.71	901.42	-3.09%
0.2,0.1	(3,18)	(-1,21)	1326.82	1312.35	1.00%
0.1,2	(4,0)	(5,0)	51.17	48.73	-3.49%
0.1,1	(7,1)	(9,0)	97.16	97.28	-4.39%
0.1,0.5	(10,1)	(14,0)	207.93	194.57	-2.08%
0.1,0.2	(6,6)	(-1,0)	612.00	577.92	-4.05%
0.1,0.1	(3,10)	(-1,15)	901.56	899.21	1.46%

4.5 A Fault-tolerant Architecture

Throughout the previous discussions, in order to concentrate on the mobility problems, we have ignored the failures of links and hosts. However, these failures do exist in the real world and the separation of the mailbox from its owner agent increases the ARP's vulnerability to failures. In this section, we propose a fault-tolerant architecture for the ARP, which can keep the ARP working in the presence of failures.

4.5.1 The Architecture

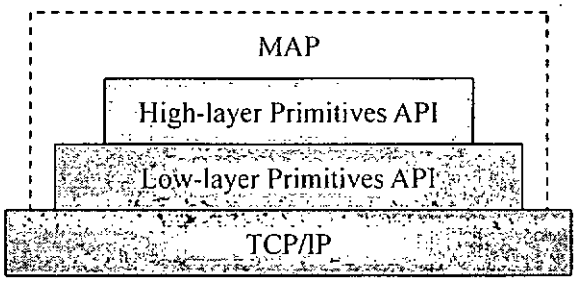


Figure 25. **The two-layer architecture for fault tolerance**

In general, a fault tolerant protocol should have three crucial capabilities: detecting failures, bypassing failures and recovering from failures. To equip the ARP with these capabilities, we propose a two-layer architecture to support fault tolerance as shown in Figure 25. Failures can be detected by the low-layer primitives that are responsible for the reliable point-to-point message passing between MAPs, and bypassed and recovered by the high-layer primitives that implement the reliable end-to-end message delivery between mobile agents. Mobile agents call the high-layer primitives to pass messages between each other. The high-layer primitives determine the next host to which the messages are to be forwarded and call the low-layer primitives to carry out the message delivery to that host. In the next two subsections, we will present the implementation of these two layers.

4.5.1.1 Implementation of the Low-layer Primitives

To increase fault tolerance, all messages in our system must be placed into stable storage before they are considered to be safely received. Although protocols like TCP can guarantee the arrival of a message at a target host, the host may crash before the message is delivered to the MAP where it is saved into the disk. Therefore, instead of using TCP directly, we have to implement a separate layer. The implementation of the low-layer primitives is quite similar to a simplified TCP by making use of the sliding window algorithm. Therefore, the low-layer primitives can be used for detecting host

failures. If the sender does not receive acknowledgement from the receiver after several retransmissions, the sender will treat the receiver unreachable and throw exceptions to be handled by the high-layer primitives.

4.5.1.2 Implementation of the High-layer Primitives

The high-layer primitives can prevent message loss due to host failures by providing failure bypassing and failure recovery capabilities.

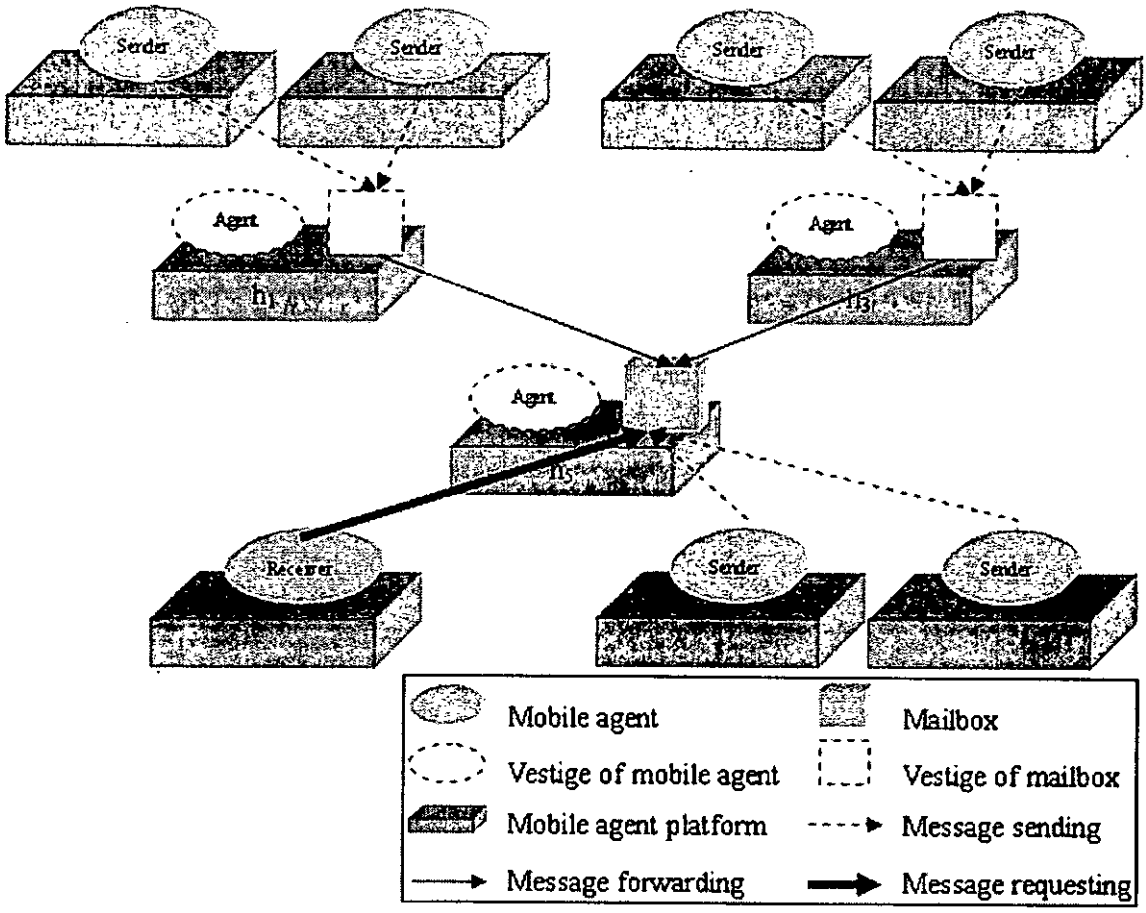


Figure 26. **The visualization of entity relationships**

Let us first talk about the failure bypassing capability. After the low-layer primitives report a host failure, steps must be taken to bypass the failed host in order to avoid any interruption to communications. One problem is that the failed host may still hold some mailboxes. The ARP will then not be able to keep working because 1) the sender can no longer send messages if its cached address were the failed host, 2) the hosts on

the migration path of a mailbox trapped in the failed host can no longer forward messages to the mailbox, and 3) the receiver can no longer pull messages from the mailbox. These three problems, illustrated in Figure 26, can be addressed as follows.

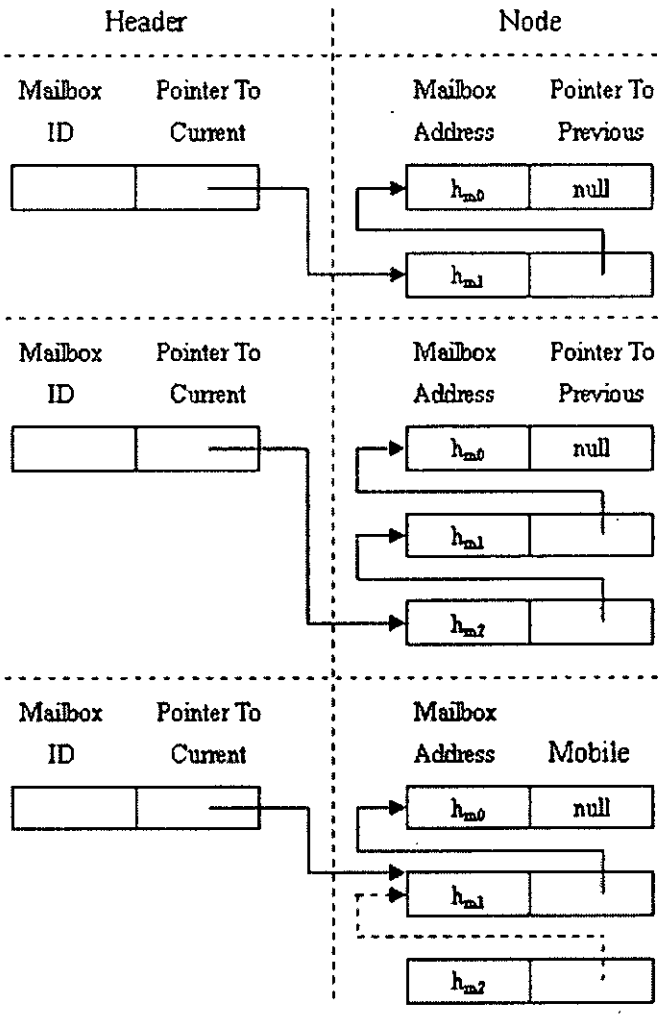


Figure 27.**The linked list cache**

- **Failure bypassing by a sender.** In the ARP, a sender caches the mailbox address of the receiver and sends messages directly to it. If the cached address is a failed host, however, there will be no way for the sender to continue the communication with the receiver. We deal with this problem by requiring the sender to maintain in its cache not only the mailbox's latest address but also its superseded addresses. For instance, suppose initially the sender caches the address h_{m1}. Later it receives an "UPDATE" message from h_{m2}. Immediately, it adds h_{m2} to the cache as the current address of the mailbox. However, h_{m1} is not removed from the cache. As a

result, the sender will maintain a linked list of addresses associating with the receiver's mailbox. The last address in the linked list should be h_{m0}, i.e., the home of the receiver. As shown in Figure 27, when the sender finds that h_{m2} has become unreachable, it removes h_{m2} from the linked list and starts to send messages to h_{m1}. If all the hosts in the linked list become unreachable, it will suspend the message forwarding to that receiver.

- **Failure bypassing by the hosts on the mailbox's migration path.** If a host on the migration path of a mailbox wants to forward messages to the mailbox and finds it unreachable, it will block the message forwarding until either the mailbox becomes reachable again or a new mailbox has been constructed. The messages are inserted into the block queue and the *valid* tag is set to *false*.

- **Failure bypassing by the receiver.** If the receiver detects that its mailbox is unreachable, it will at once create a new mailbox locally and send "REGISTER" messages to all the hosts on the migration path of its old mailbox. This path also includes the failed host. The processing of the "REGISTER" message is the same as that of the ARP.

Now let us talk about the failure recovery capability. A failure could be permanent or transient. If the host failure is permanent, messages buffered in the host are lost forever. We deal with this problem by assigning a sequence number to each message. The sender sends a message and buffers it in an outgoing window until the arrival of an acknowledgement from the receiver. If the acknowledgment does not arrive within the sender's expected round trip time, the message is retransmitted. In this way, we can guarantee that the message will eventually be delivered to the receiver.

If the host failure is transient, we can make use the following techniques to not only speed up the process of recovering lost messages but also to reduce the retransmission costs. These techniques might be especially useful when the host failure time is so

short that the sender does not notice it and thus does not retransmit the lost messages. Suppose after some period of time, the host recovers. Since all data is in stable storage, the entire data set can be restored, although some of the data such as the address table may be out-of-date. To reconfigure the recovered host according to the current system state, we suggest the following two reconfiguration schemes.

- **Passive reconfiguration scheme.** A new attribute called FAILED is introduced into the address table. This *failed* tag suggests whether or not the mailbox has experienced a host failure. Therefore, the initial value of the *failed* tag is *false*. On recovery from a host failure, the host sets *true* to all the *failed* tags of the mailboxes carried by it before the failure. The *failed* tag of a mailbox will not be set back to *false* (i.e., the host will not resume the normal mailbox operation) until the host has received enough messages necessary for it to determine the current state of the mailbox. The procedures are as follows:
  - ✧ If the host receives a data message for the mailbox, it simply inserts the message into the mailbox and continues to wait for other messages.
  - ✧ If the host receives a "PULL" or "MVMB" message from the owner of the mailbox, it at once realizes that there was no new mailbox created during the period of the host failure, and it can resume the normal mailbox operation.
  - ✧ If the host receives a "REGISTER" or "DEREGISTER" message from another mailbox with the same owner, it knows a new mailbox has been created. It will start to forward the messages buffered in the old mailbox to the new one. After that, it will deconstruct the old mailbox and resume the normal operations.

- **Active reconfiguration scheme.** In the passive reconfiguration scheme, a host has to wait for the "REGISTER" or "DEREGISTER" messages from the newly created mailboxes before it can update its state. Also the scheme assumes that the host's failure time is long enough for the receiver to detect the failure and create a new mailbox. Therefore, the passive reconfiguration scheme requires a relatively

long time for recovery. In the active reconfiguration scheme, after the host h_m recovers, it will actively check whether or not a new mailbox has been created. The checking algorithm is simple:

- ✧ h_m checks the address table in the agent home of the receiver for the current address of the mailbox.
- ✧ If the address of the mailbox is h_m, it means that there was no new mailbox been created. So h_m can resume the operations for the mailbox as if there is no host failure.
- ✧ If the address is different from h_m, it indicates that a new mailbox has been created. So h_m should perform the following actions: 1) forwarding all the messages stored in the old mailbox to the new one, 2) deconstructing the old mailbox, and 3) updating the address table accordingly.

There are a few more remarks about the fault-tolerant architecture. First, since we use the retransmission mechanism to deal with the permanent host failure, duplicate messages may exist in our system. To remove these duplicate messages, the receiver can make use of the sequence number assigned to each message. Second, the property that a message will be forwarded at most once before it arrives at the target mailbox can no longer be satisfied. After a host recovers from a failure, it may still receive data messages. However, the current address of the receiver's mailbox maintained by the host may be out-of-date because it may have missed several "REGISTER" and "DEREGISTER" messages. As a result, it will forward the messages to the outdated address and cause once more message forwarding.

4.5.2 Performance Evaluation

In this section, we evaluate the performance of the fault tolerant architecture. Similarly, we adopt the same simulation model as that used in the performance evaluation of the ARP plus the following failure model.

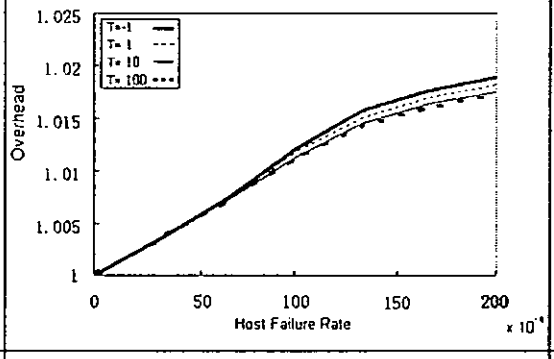
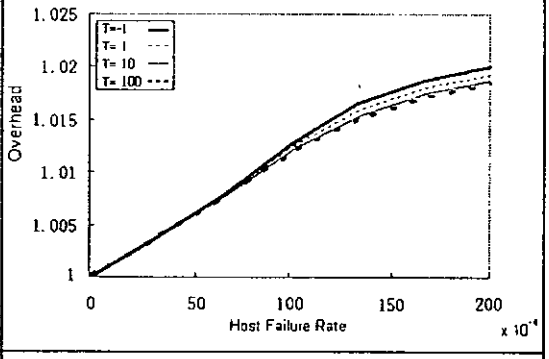
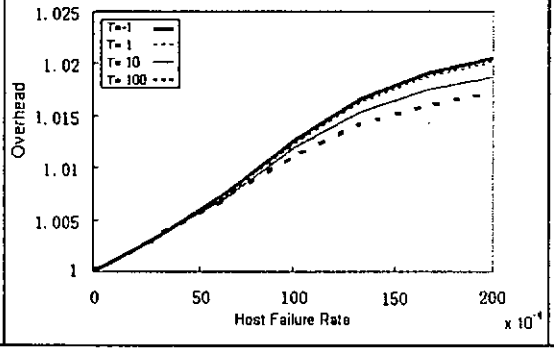
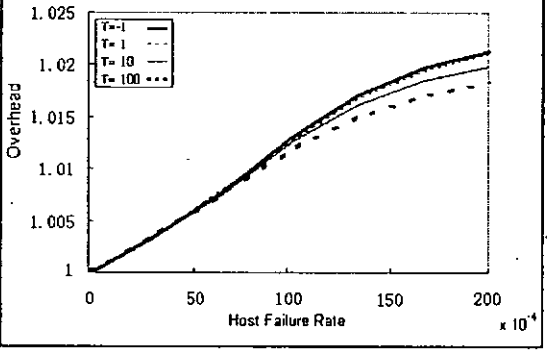
A host continues functioning for a period of time of $1/\alpha$ in average until a host failure occurs. The average failure recovery time is $1/\beta$. Therefore, the probability of host failure p_f can be represented as

$$p_f = \frac{1/\beta}{1/\alpha + 1/\beta} = \frac{\alpha}{\alpha + \beta} \tag{29}$$

We assume the host failure and the failure recovery follow the negative exponential distribution with α and β being their average values, respectively. We also define another very important variable called message-to-migration ratio $\eta = \lambda/\mu$, which is equal to the expected number of messages to be received during the residence time at a host. Two other parameters used in our simulation are the retransmission timeout which is set to 20sec, and the maximum number of retransmissions before a host failure is detected which is set to 3.

Table 3 lists the expected cost required for both the active and passive reconfiguration schemes under various failure probabilities. The purpose is to see the comparison between the communication cost with host failures and that without host failures. We find that the overhead approximately scales to the host failure rate p_f when p_f is small. This is because in this condition our failure handling algorithms generate a more or less constant traffic overhead each time a failure is detected. So the overhead scales to the total number of failures detected and thus scales to the host failure rate. As p_f increases, we can see that the slope of the curves first increases and then decreases. The reason might be as follows. The increment of p_f makes it more likely that there are more than one host failures at a time and the sender has to perform the failure bypassing twice or even more times. Thus the slope of the curves increases. However, this increment cannot last long since the overhead is bounded by the detection of either network partition or receiver failure, i.e., as several times of end-to-end retransmission fails, an exception is reported to the sender and the sender will suspend the message sending. So the slope drops eventually.

Table 3. **The overhead of the communication cost with host failure**

η	Passive Reconfirmation Scheme	Active Reconfiguration Scheme
1		
10		

Although the costs for both reconfiguration schemes increase as the host failure rate increases, this increment remains very small. The active reconfiguration scheme costs a little bit more than the passive reconfiguration scheme. This is because in the active reconfiguration scheme, the failed host after recovery has to send additional request to the agent home to reconfigure the current state, while in the passive reconfiguration scheme, no reconfiguration message is sent out and the failed host only needs to listen to the existing messages for it to determine the current state. On the other hand, the active reconfiguration scheme reduces the average message delivery time as shown in Table 4. Since we can observe similar curves to those in Table 3 with approximately the same reasons, we skip the discussions of Table 4.

Table 4. **The overhead of the message delivery latency with host failure**

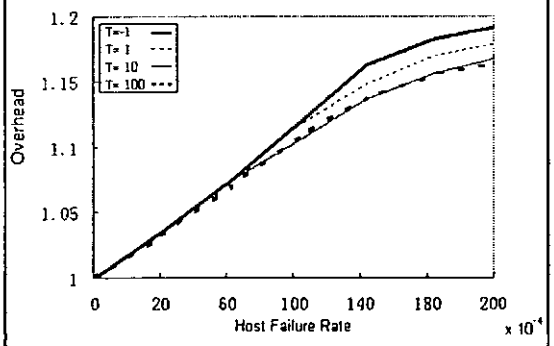
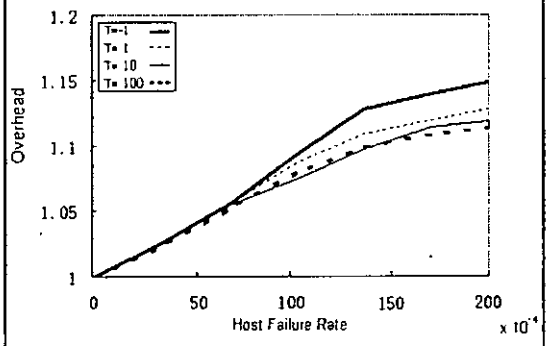
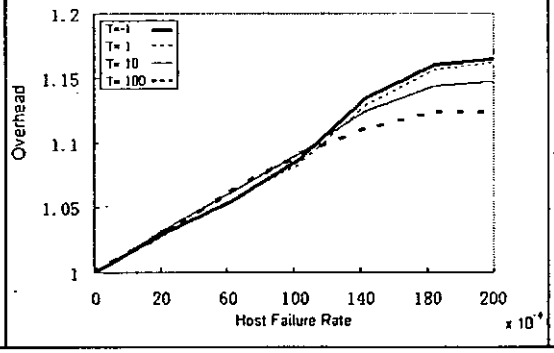
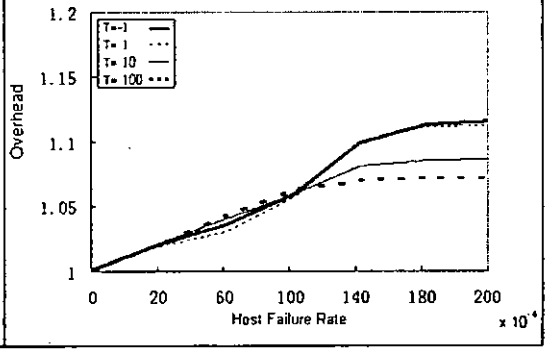
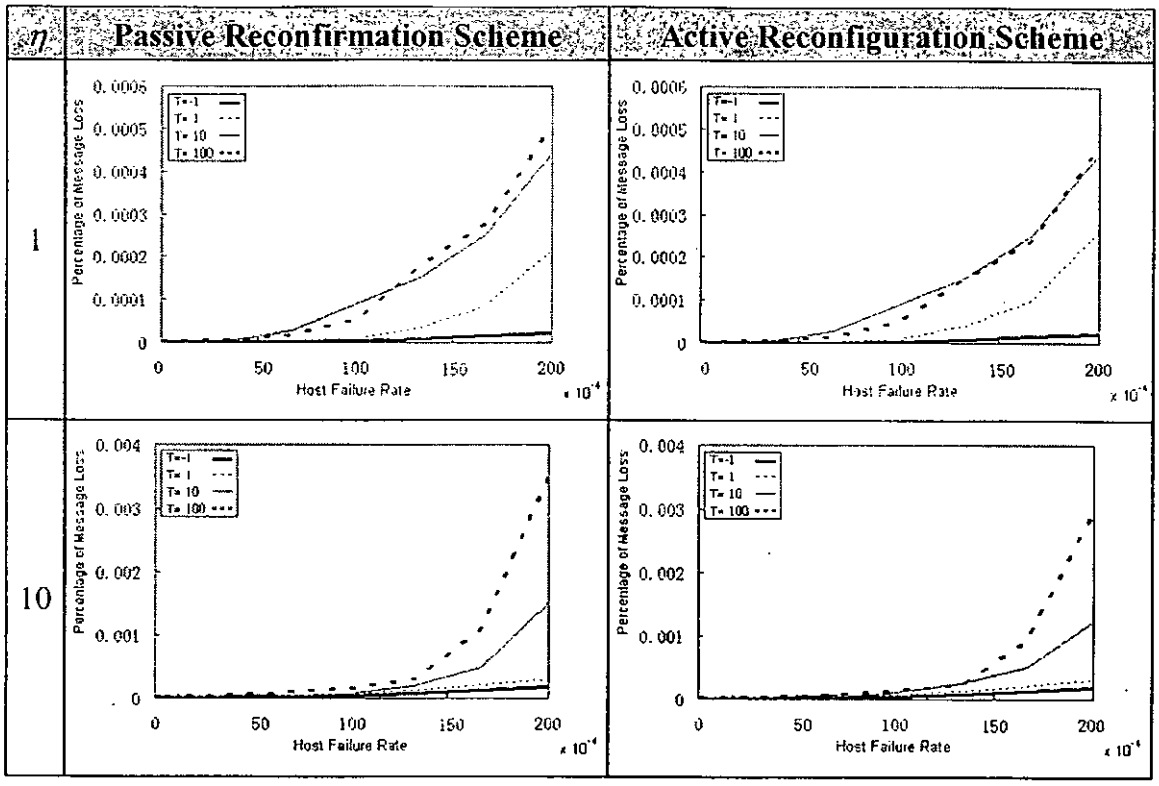
n	Passive Reconfirmation Scheme	Active Reconfiguration Scheme
1		
10		

Table 5 shows the percentage of messages that are not deliverable due to the sender's detection of either network partition or receiver failure. As mentioned before, the home-server based scheme and the distributed-registration based scheme are two special case of our proposed protocol with $T = \infty$ and $T = -1$, respectively. In the home-server based scheme, the mailbox always stays at the agent home. Therefore, it suffers from a single point of failure, i.e., when the agent home fails, senders will find no way else to forward messages because the agent home is the last address in the cached linked list of addresses associating with the receiver's mailbox. However, in the distributed-registration based scheme, the length of the linked list is normally quite long, i.e., there are many backup hosts that can forward messages to the receiver's mailbox. Therefore, we can observe a much higher percentage of messages not deliverable when T is small than that when T is big. But in all cases, this percentage remains below 0.5%.

Table 5. **The percentage of messages not deliverable**



4.6 An Dynamic Programming Approach of Optimization

As discussed before, we use a fixed threshold T number of messages to make the decision about the mailbox's migration in our ARP, and then we develop an analytical model and an optimization algorithm to derive the best value of T. To make it adaptive, we require new T to be calculated each time the old T has been exceeded. However, this is not a truly adaptive approach since the value of T is updated only at some selective agent's migration. A truly adaptive approach should be able to adjust the optimal values of relevant parameters of a protocol during each agent's migration. In this section, we are to propose one such approach to our ARP by making use of the dynamic programming technique [48]. We will first briefly introduce how the dynamic programming works. Readers may refer to [48] for more information about the dynamic programming. Then we will mathematically formulate our performance optimization problem as a dynamic programming problem, based on which the optimal mailbox's migration policy can be established. Lastly, we will conduct

experiments to evaluate the performance of our ARP under this new approach.

4.6.1 The Dynamic Programming

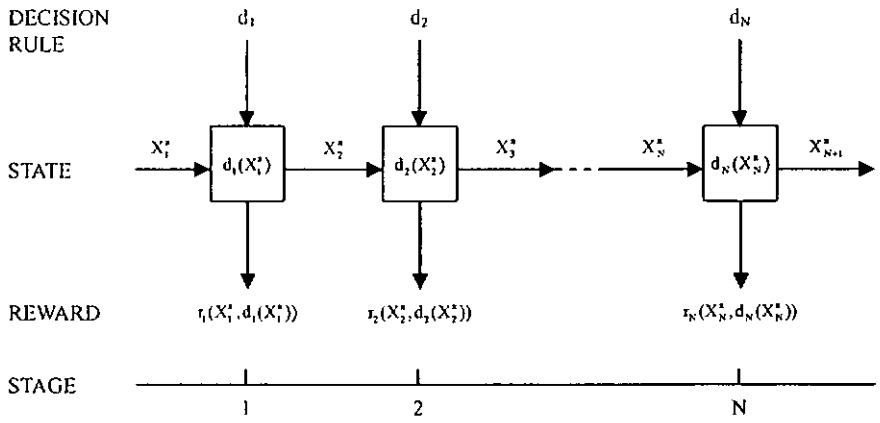


Figure 28. **The model of a typical dynamic programming problem**

In general, the dynamic programming is a mathematical tool for making a sequence of interrelated choices with the goal of optimizing the system performance over the entire decision making horizon. Figure 28 depicts the model of a typical dynamic programming problem. First, the dynamic programming defines a set of time points at which decisions can be made. These time points are also referred to as stages and are denoted by $TP = \{1, 2, \ldots, N\}$. TP could be either finite or infinite; it could also be either a discrete set or a continuum. Then for each stage, a set of allowable states S_{tp} are identified. In Figure 28, we use X_{tp}^{π} to denote the state the system occupies at the time point $tp \in TP$. The meaning of the superscript π will be explained later.

Now suppose a system is in the state $s \in S_{tp}$ when a decision has to be made. In the dynamic programming, we use $A_{s,tp}$ to represent the set of allowable decisions, or say actions, at the time point tp with the current system state s. Among all these candidate actions $A_{s,tp}$, only one action a is selected and performed. Usually, the selection of the action a only depends on the current system state s. If this is true, a decision rule d_{tp} can then be defined upon each decision point tp. In other words, $d_{tp}(s)$ specifies the

action a. The sequence of decision rules $\{d_1, d_2, \ldots, d_N\}$ over the entire decision making horizon is called a policy, which is conventionally denoted as π.

After performing the action a, an immediate reward $r_{tp}(s, a)$ is received and the system moves to the next stage. This reward could be anything that you want your system to gain and it could be either positive or negative. For a negative reward, it can be thought of as a cost. Furthermore, the choice of action could affect the system evolution either deterministically or probabilistically. If it is a deterministic case, the choice of action would determine the state of the system at the next stage with certainty. Therefore, a transfer function $w_{tp}(s, a)$ can be defined as the mapping from the current state s at the time point tp to the next state at the time point $tp+1$ if the action a is performed, and we now have $w_{tp}(X_{tp}^{\pi}, d_{tp}(X_{tp}^{\pi})) = X_{tp+1}^{\pi}$.

By specifying a policy at the start of a deterministic system, the decision maker completely determines the future evolution of the system: the sequence of states the system will occupy is known with certainty, and hence the sequence of rewards the decision maker will receive is also known. With this property, the decision maker can now evaluate different policies by comparing their total rewards over the decision making horizon. The following equation expresses the total rewards.

$$v^{\pi}(X_1^{\pi}) = \sum_{tp=1}^{N} r_{tp}(X_{tp}^{\pi}, d_{tp}(X_{tp}^{\pi})) \tag{30}$$

The decision maker's problem is now reduced to choose at time point 1 a policy π to make (30) as large as possible and to find the maximal reward as expressed in the following equation.

$$v^{*}(X_1^{*}) = \max_{\pi \in \Pi} v^{\pi}(X_1^{\pi}), \tag{31}$$

where Π denotes the set of all the possible policies.

To solve the above equation, the backward induction algorithm [48] is usually used. The backward induction algorithm is a procedure that uses the functional equation in

an iterative fashion to find the optimal policy for a finite-horizon sequential decision problem so as to obtain the optimal total value. The procedures of this algorithm are described as follows:

a. Set $tp = N+1$ and $v_{tp} = 0$.

b. Substitute $tp - 1$ for tp and compute $v_{tp}(s)$ for each $s \in S_{tp}$ by using the following equation.

$$v_{tp}(s) = \max_{a \in A_{s,tp}} \{r(s,a) + v_{tp+1}(w_{tp}(s,a))\} \quad (32)$$

Put the action a^* that maximizes (32) into the optimal action set $A^*_{s,tp}$.

c. If $tp = 1$, stop; otherwise, return to step b.

After performing the backward induction algorithm, both the optimal total reward $r_1(s)$ and the optimal action set $A^*_{s,tp}$ are derived. The optimal policy can now be easily obtained by always selecting an action in the set $A^*_{s,tp}$. The formal proof of the correctness of the above procedures is given in the Principle of Optimality [48].

4.6.2 Modeling of Our Optimization Problem

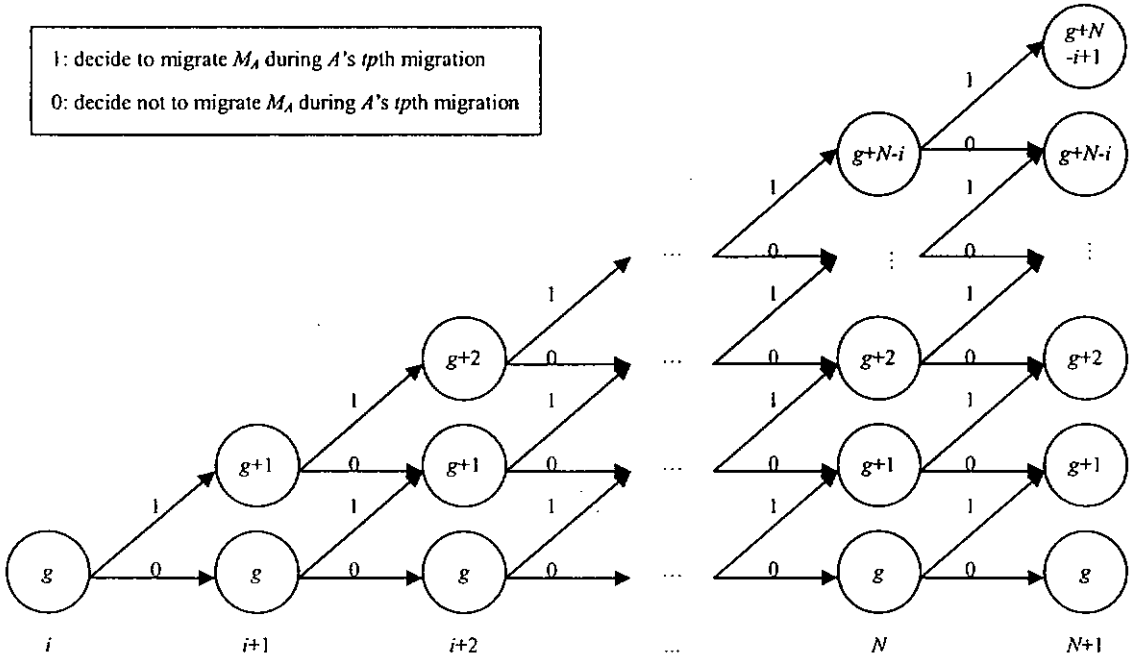


Figure 29. **The model of our optimization problem**

Now we are to model our optimization problem as a dynamic programming problem. To minimize the total communication cost over the life cycle of an agent A, the best we can do is to perform the dynamic programming each time before A wants to migrate. In this way, we can always make sure that the current decision will bring the remaining communication cost in optimum, assuming that the communication and migration pattern of A will never change for the remaining life time. Also performing the algorithm during each agent's migration guarantees that the derived decision policy can always adapt to the changing environment.

Figure 29 graphically depicts the model of our cost optimization problem before the agent's ith migration. The circle represents the state, which records the number of migrations the mailbox M_A has performed, and the arrow represents the action, which could be either moving with M_A or moving alone. Next we will formally define the items that are required by the dynamic programming.

Decision Points:

$TP = \{i, i+1, i+2, \ldots, N\}$

where N represents the total number of migrations during the life cycle of the agent A.

States:

$S_i = \{g\}$; $S_{i+1} = \{g, g+1\}$; $S_{i+2} = \{g, g+1, g+2\}$; ...; $S_N = \{g, g+1, g+2, \ldots, g+N-i\}$

where we assume that the mailbox M_A has performed g migrations before the agent's ith migration.

Actions:

$A_{s,tp} = \{1, 0\}, s \in S_{tp}$

where an action 1 stands for A migrating with M_A and an action 0 stands for A migrating alone.

Rewards:

The reward in our problem is defined as the communication cost between the current and the next agent's migration. This cost includes both the location registration cost should M_A moves and all the data messages' forwarding cost. Since a cost is normally treated as a negative reward, we therefore append a negative sign before each cost function as shown below.

$$r_{tp}(s,a) = \begin{cases} -Cost_{move}(s,\eta) & s \in S_{tp} \text{ and } a = 1 \\ -Cost_{unmove}(s,\eta) & s \in S_{tp} \text{ and } a = 0 \end{cases}$$

where $Cost_{move}(s, \eta)$ and $Cost_{unmove}(s, \eta)$ are defined as (33) and (34), respectively.

$$\begin{aligned} Cost_{move}(s,\eta) = \; & Cost_{mvmb} + s \times Cost_{deregister} + s \times Cost_{reply} + (s+1)Cost_{register} + \\ & M \times (Cost_{sender \to h_{ms}} + Cost_{h_{ms} \to M_A} + Cost_{update}) + \\ & (\eta - M) \times Cost_{sender \to M_A} \end{aligned} \quad (33)$$

$$Cost_{unmove}(s,\eta) = \eta \times (Cost_{sender \to M_A} + Cost_{M_A \to A} + \alpha \times Cost_{query}) \quad (34)$$

where η is the expected number of messages to be received during the residence time at a host, which is also known as the message-to-mobility ratio; M is the number of senders in the system that might send messages to A; and α is the ratio of the number of query messages to the number of messages obtained from M_A.

The meaning of these two equations is straightforward. In (33), A takes its mailbox during its tpth migration. It will first send an "MVMB" message ($Cost_{mvmb}$) to M_A. M_A, on receiving "MVMB", will send the "DEREGISTER" messages ($Cost_{deregister}$) to all the hosts but itself on $Path_m(A)$ ($Path_m(A)$ contains s+1 hosts after M_A has performed s migrations, and therefore, s "DEREGISTER"s are sent out.) and wait for "REPLY"s ($Cost_{reply}$) from them. After M_A performs the migration, the "REGISTER" messages ($Cost_{register}$) are sent out to make the registration. Since M_A has moved to its owner, the location information cached by the senders must be outdated. For the first data message sent out by each of the M senders, it is first routed to the old place of M_A ($Cost_{sender \to h_{ms}}$) and then forwarded to the current location of M_A ($Cost_{h_{ms} \to M_A}$), while at the same time, an "UPDATE" message ($Cost_{update}$) is sent back to the sender. After

that, the data messages can be sent directly to M_A ($\text{Cost}_{\text{sender}\rightarrow M_A}$) since all the senders have updated their caches. Now that M_A and A are collocated with each other, no extra step is required to retrieve the data messages from M_A. In (34), since M_A does not move, no synchronization is required and the only cost is to send the data messages. To receive the data messages in time, the receiver periodically pulls its mailbox ($\text{Cost}_{\text{query}}$) at a rate of α times the mean message arrival rate. Therefore, the mean message retrieval cost for a data message from M_A is $\text{Cost}_{M_A \rightarrow A} + \alpha \times \text{Cost}_{\text{query}}$.

The parameter η needs a bit more explanation. By definition, it means the expected number of messages to be received during the residence time at a host. As the formulation of our problem assumes that η will never change for the remaining life time of A, the proper choice of the value of η is very important to the accuracy of the derived decision policy. The more the real situation follows this assumption, the more likely the derived decision policy coincides with the universally optimal decision policy, and the closer the total communication cost approaches the universal minimum. Since η is affected by both the message arrival rate λ and the agent's migration rate μ, it is also called message-to-mobility ratio. Let us denote λ_j and μ_j as the message arrival rate and the agent's migration rate when A resides at the host h_{aj}, respectively. Suppose A is now residing at h_{ai}. Then we use the following equation to represent the expected number of messages to be received at each host afterwards, where W is the weighting factor.

$$\eta_i = \begin{cases} W \times \lambda_i/\mu_i + (1-W) \times \eta_{i-1} & i \neq 0 \\ \lambda_0/\mu_0 & i = 0 \end{cases} \tag{35}$$

Transfer function:

$$w_{tp}(s,a) = \begin{cases} s+1 & s \in S_{tp} \text{ and } a = 1 \\ s & s \in S_{tp} \text{ and } a = 0 \end{cases}$$

Now that we have defined all the necessary items for the dynamic programming, we

can apply the backward induction algorithm to derive the optimal decision policy.

4.6.3 Performance of the ARP with the Dynamic Programming

In this section, we will experimentally evaluate the efficiency of our ARP by making use of the dynamic programming for the policy selection. Similarly, we adopt the same simulation model as that used in the performance evaluation of the ARP except for the weighting factor W being set to 0.5.

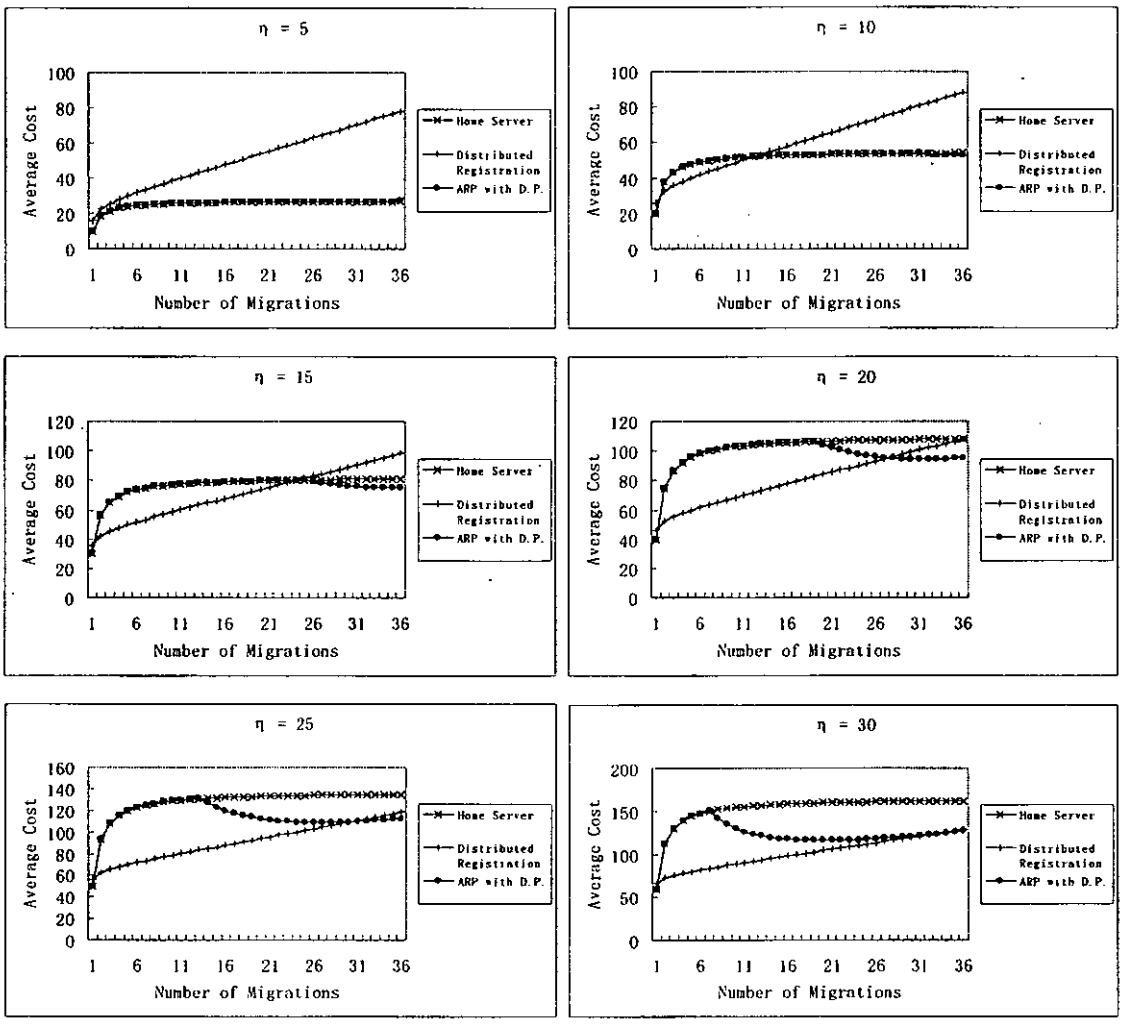


Figure 30. **The ARP with the dynamic programming for constant η**

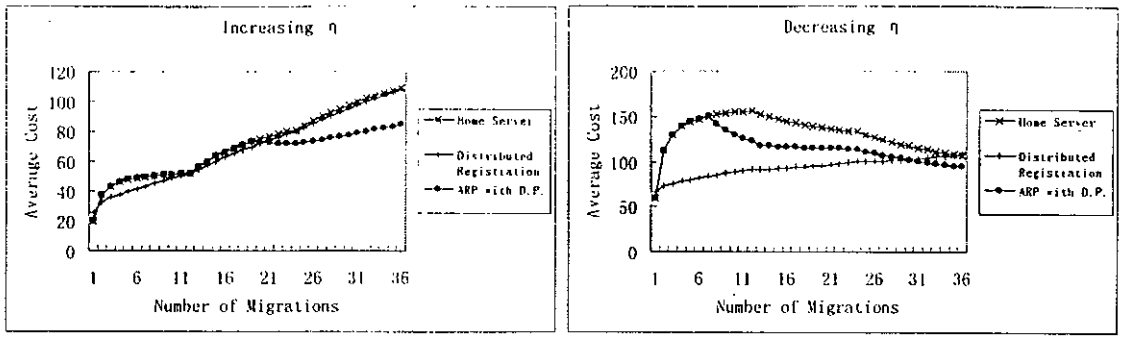


Figure 31. **The ARP with the dynamic programming for variable η**

Figure 30 and Figure 31 show the experimental results that compare the performance of our ARP with that of the home-server based scheme and the distributed-registration based scheme. The x-axis is the number of migrations and the y-axis is the average cost per-migration which is the total communication cost divided by the number of migrations performed. The reason behind using the number of migrations as the x-axis is that we would like to visualize the trend of the curves. As we know that the home-server based scheme and the distributed-registration scheme are two extreme cases of our ARP, if the curve of the ARP follows that of the home-server based scheme, we can judge that no M_A migration is performed; otherwise, we may estimate some M_A migrations are carried out. Figure 30 is generated under the circumstance of the constant message-to-mobility ratio, while Figure 31 is produced with the variable message-to-mobility ratio.

In Figure 30, we can discover that when η is small, the home-server based scheme performs much better than the distributed-registration based scheme, which means it is preferred that M_A dose not make any movement if A seldom receives messages. Otherwise, the cost of M_A migration will overwhelm the saving of the optimal route achieved by moving the mailbox. We observe that the ARP intelligently discovers this rule and adapts to the home-server based scheme. However, when η increases, the advantage of the distributed-registration scheme becomes evident. By moving the mailbox, a larger cost will be saved since a great number of messages will benefit from the optimal route between the sender and the receiver. Despite of the mailbox's

migration cost, it is still economical to move the mailbox. But this does not mean the mailbox should always move when η is large. As the mailbox's migration path becomes longer, the migration cost also grows. Eventually, it will exceed the cost saving and no further M_A migration is desirable. The larger η is, the longer the distributed-registration scheme outperforms the home-server based scheme.

Now look at our ARP. When η is large, our ARP chooses not to migrate the mailbox first by adapting to the home-server based scheme and then starts M_A migration as we can see the departure of our ARP from the home-server based scheme. The larger η is, the earlier this departure starts, which means a larger number of M_A migrations will be performed. This exactly coincides with what we have discussed before. Finally, our ARP always performs no worse than both the home-server based scheme and the distributed-registration based scheme at the end of the agent's lift cycle, and for some specific η, our ARP can perform much better than the other two schemes. Figure 31 shows two scenarios of the changing η. In the first scenario, η increases from 10 to 20 to 30 with each η lasting 12 agent's migrations. The second scenario is just the reverse situation of the first one. Again our ARP can outperform both the home-server based scheme and the distributed-registration based scheme.

Although we observe considerable performance improvement of our ARP over other schemes, there exists one critical problem that must be addressed. The problem is the huge computational complexity, one of the common problems in any practical dynamic programming application. This is because of the iterative algorithm used in the computation. The system has to actually traverse all the possible branches of the network (such as the one in Figure 29) to compute the final result. Therefore, when the number of stages, states and actions are large, the computational complexity will become unaffordable. By referring to (32), we can numerically predict the number of required summations as $\sum_{tp \in TP} \sum_{s \in S_{tp}} | A_{s,tp} |$ and the number of required max operations as $\sum_{tp \in TP} | S_{tp} |$, where $|A_{s,tp}|$ and $|S_{tp}|$ stand for the number of actions belonging to $A_{s,tp}$ and

the number of states belonging to S_{tp}, respectively. Specifically, to compute an optimal decision before the ith migration of A, the algorithm has to perform $(N-i+2)\times(N-i+1)$ summations and $(N-i+2)\times(N-i+1)/2$ max operations. The total number of summations and max operations during the life cycle of A are given by:

$$\mathrm{Num}_{sum} = \sum_{i=1}^{N}(N-i+2)\times(N-i+1) = \frac{N\times(N+1)\times(N+2)}{3} = O(N^3) \tag{36}$$

$$\mathrm{Num}_{max} = \sum_{i=1}^{N}(N-i+2)\times(N-i+1)/2 = \frac{N\times(N+1)\times(N+2)}{6} = O(N^3) \tag{37}$$

Therefore, the overhead may be large for applications with long-term agent migration (large N) and dense agent distribution. Considering the case that multiple agents perform migrations at about the same time in a host, a substantial computational delay will be imposed to each agent's migration, which not only decreases the application performance in terms of the response time but also, to some extent, restricts the agent's asynchronous mobility. In the next subsection, we will try to find a solution to alleviate this problem.

4.6.4 An Improved Dynamic Programming Approach

As inspired by (36) and (37), if we can reduce the number of stages in a dynamic programming problem, the computational complexity will be dramatically reduced. Also it is reasonable to assume that the message-to-mobility ratio η does not fluctuate too frequently and therefore, the dynamic programming is not necessary to get applied for each agent's migration. In the following improved approach, we take advantages of both the above techniques to reduce the computational complexity.

Instead of always optimizing the remaining life cycle of A, we now utilize the dynamic programming to just optimize a portion of A's life cycle: the period of A's next K migrations. For example, before the first migration, A applies the dynamic

programming to derive the decision policy that only runs until its fourth migration (K = 3) so that within this period of time, the communication cost is minimal. No dynamic programming is conducted for its second and third migrations. When the fourth migration starts, another round of the dynamic programming is conducted to minimize the communication cost until the seventh migration. Obviously, a significant reduction of the computational complexity can be achieved by using this approach. Theoretically, the number of summations and max operations required for each round of the dynamic programming are $K\times(K+1)$ and $K\times(K+1)/2$, respectively. The total number of summations and max operations required during the entire life cycle of A are given by:

$$\mathrm{Num}_{sum} = \mathrm{K}\times(\mathrm{K}+1)\times\lfloor \mathrm{N/K}\rfloor + (\mathrm{N}-\mathrm{K}\times\lfloor \mathrm{N/K}\rfloor)\times(\mathrm{N}-\mathrm{K}\times\lfloor \mathrm{N/K}\rfloor+1) = \mathrm{O}(\mathrm{K}\times\mathrm{N}) \quad (38)$$

$$\mathrm{Num}_{max} = \frac{\mathrm{K}\times(\mathrm{K}+1)\times\lfloor \mathrm{N/K}\rfloor + (\mathrm{N}-\mathrm{K}\times\lfloor \mathrm{N/K}\rfloor)\times(\mathrm{N}-\mathrm{K}\times\lfloor \mathrm{N/K}\rfloor+1)}{2} = \mathrm{O}(\mathrm{K}\times\mathrm{N}) \quad (39)$$

where $\lfloor N/K \rfloor$ means the max integer that does not exceed N/K.

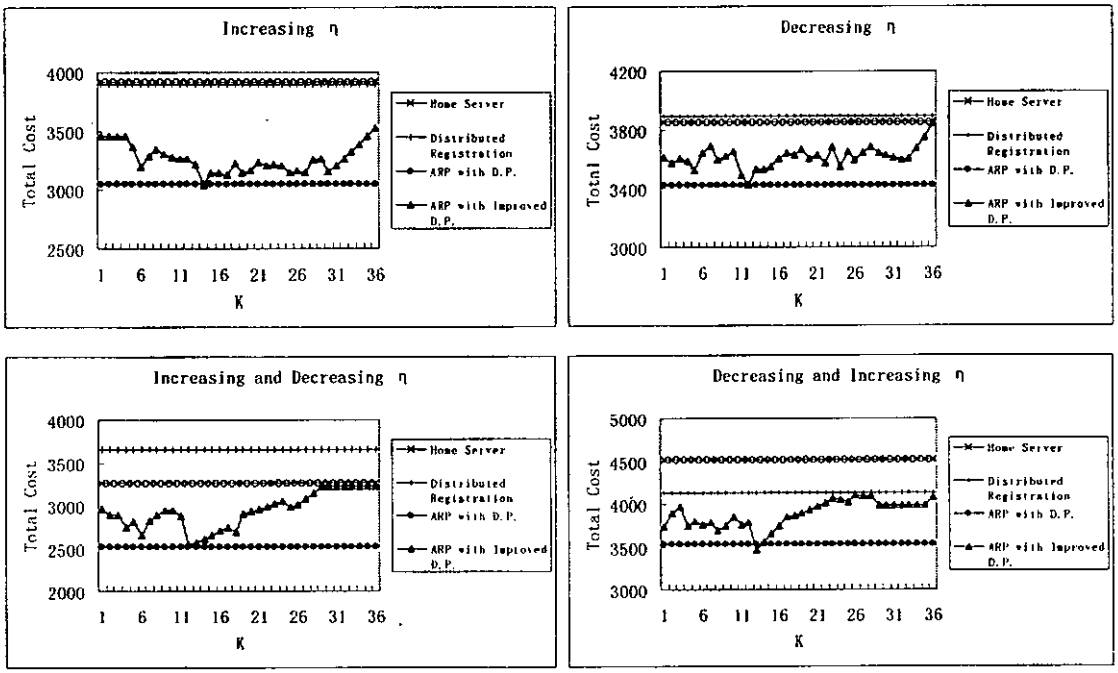


Figure 32. **An improved approach of using the dynamic programming**

Figure 32 shows the performance results with the x-axis now being the variable K and the y-axis being the total communication cost. All four plots demonstrate the scenario

of the changing η. We do not show the plots under the circumstance of the constant η because there are no differences between the original and the improved dynamic programming approaches. The first two scenarios are the same as those in Figure 31. The third one is about η rising first and then dropping (10→30→10) and the last one is about η dropping first and then rising (30→10→30).

All four scenarios conclude that the improved approach by introducing the variable K to the dynamic programming normally performs worse than the original approach, but still better than the home-server based scheme and the distributed-registration based scheme. This is probably because the improved approach restricts the optimization problem within just a small region K. In other words, the improved approach is myopic and hence considers only the short-term benefit. By increasing K, this *myopic problem* can be alleviated to some extent, but another problem comes out. As we assume that η does not fluctuate too much within the period of K, this assumption may not be satisfied if K is large and therefore, the derived policy cannot be accurate. We call it the *adaptability problem*. So a proper K, neither big nor small, can produce the best performance. This conclusion can also be examined from the figure, where the proper K in our example is around 12, which is exact the duration of each step of η. The reason could probably be that 12 is the largest K (to alleviate the myopic problem) that can catch up with the pace of the changing η (to deal with the adaptability problem). The only question remains now is how to select the value of K properly. In the next section, we will talk about this issue.

Table 6 concludes the pros and cons of selecting among different Ks. For a small K, it can reduce the computational complexity and can be more adaptive to the changing environment. The downside is that a small K may be more likely to have the myopic problem. A large K exhibits high computational complexity, low performance and low adaptability. Only a proper K is preferred since it achieves the best performance with acceptable computational complexity and adaptability.

Table 6. **The pros and cons about the size of K**

Characteristic	Small K	Proper K	Large K
Complexity	Low	Middle	High
Performance	Low	High	Low
Adaptability	High	Middle	Low

4.6.5 An Adaptive Dynamic Programming Approach

In the last section, we propose to use a value K to limit the scope of the dynamic programming. As concluded in Table 6, we need to find a proper K that is neither big nor small. In this section, we propose a simple algorithm to dynamically search for such a K. The algorithm is described as follows:

a. Set $K = 1$ and record the current η.

b. Apply the dynamic programming over the range of K and conduct the migrations based on the policy derived from the dynamic programming.

c. After the last migration defined by the policy is performed, check whether the life cycle of the mobile agent has reached. If it has reached, stop; otherwise, continue.

d. Record the current η and compare it with the last recorded η. If the difference between the two η exceeds a threshold, reduce K by half; otherwise, increase K by 1. Jump to step b.

We starts with a small $K = 1$ for one way search. Then we either increase K by 1 if there is no evident change about the message-to-mobility ratio, or decrease K by half if the message-to-mobility ratio fluctuates beyond a predefined threshold. The reason behind using the "smooth increase and sharp decrease" strategy is also because of the preference about K. We increase K very carefully but immediately decrease K if the message-to-mobility ratio changes.

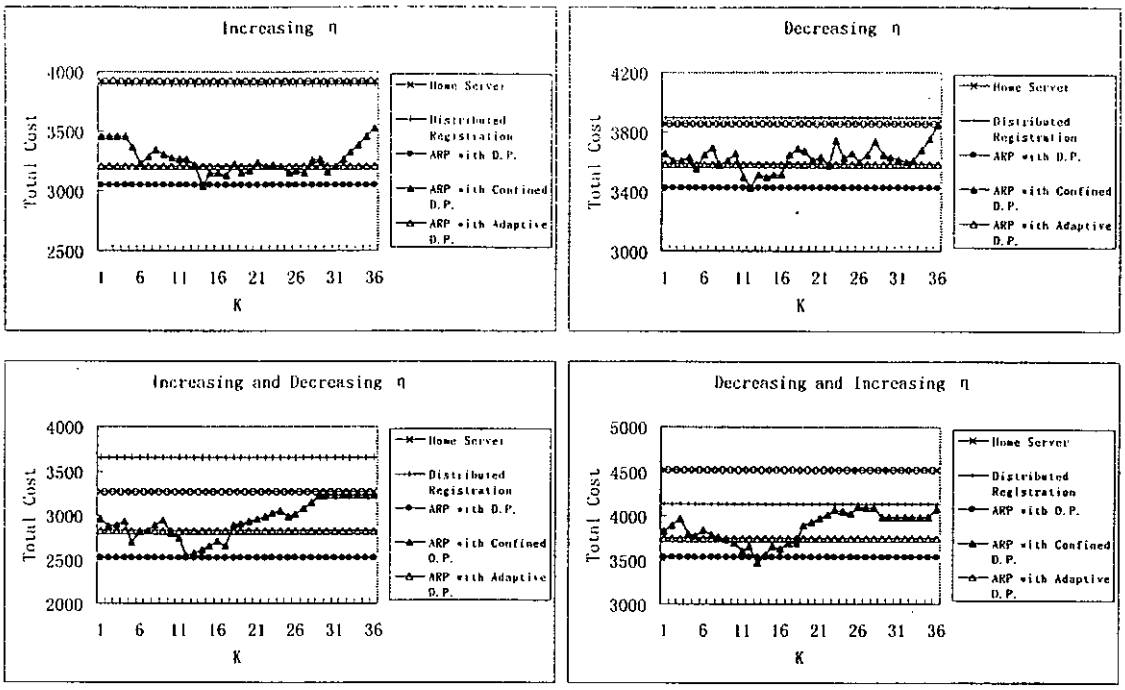


Figure 33. **An adaptive approach of using the dynamic programming**

Figure 33 shows the result of our third approach of using the dynamic programming. The threshold we have chosen in the experiment is 5, i.e., the expected number of messages η between the recent two records cannot exceed 5. It is pleased to discover that this approach can outperform the second approach for the majority of K. We conclude that our third approach does successful find and function around the optimal K, and produce a very sound performance.

4.7 Summary

In this chapter, a particular protocol called ARP is derived from the mailbox based framework for the mobile agent context. It represents the JM-PL-SSM combination of parameters and satisfies all the requirements defined in Chapter 1. Experiments are conducted and the results show that our ARP can always outperform both the home-server based scheme and the distributed-registration based scheme should the threshold T be properly settled.

To further enhance efficiency the ARP, a path pruning algorithm is proposed. With

this algorithm, all the redundant hosts along the mailbox's migration path can be effectively removed so that fewer messages are required during each location registration period. The byproduct of the algorithm is the garbage collection capability of removing the useless addresses maintained in a cache. The experimental results show that there exists an optimal value of TTL where the performance of the algorithm is maximal.

To enable the optimized performance, we build an analytical model for both the ARP and the path pruning algorithm, based on which the optimal (T, TTL) pair can be derived by using a simple iterative algorithm. Experiments are conducted to compare the derived optimal performance with the experimental one, and the results suggest that this analytical model is good enough to be implemented into the operating systems of network hosts for the dynamic calculation of the optimal (T, TTL) pair.

To further enhance reliability, a two-layer fault-tolerant architecture is proposed to handle both network failures and host failures. Specifically, the low layer abstracts away network failures and provides reliable point-to-point message passing between two MAPs; and the high layer overcomes message loss due to host failures and accomplishes reliable end-to-end message delivery between two mobile agents. The results show that this fault-tolerant architecture can effectively reduce the number of message loss while not introducing heavy overhead to both the communication cost and the message delivery latency.

Lastly, an alterative approach of optimizing the ARP's performance is proposed by applying the dynamic programming. In this approach, the optimization procedures are conducted during each agent's migration so that it can always adapt to the changes of the network environment and the communication pattern.

Chapter 5. A Mailbox Based Mobile IP

In this chapter, we are to illustrate how the mailbox based scheme can be applied to improve the performance of Mobile IP. Before that, let us first review some background knowledge.

5.1 Background

Mobile IP [4] is a recently proposed standard for the IP mobility support in the Internet by the *Internet Engineering Task Force* (IETF). Its goal is to provide seamless network access to a mobile node regardless of its current location. Without modifying the mobile node's long-term IP address during each migration, Mobile IP has the ability to keep track of the node and deliver packets to it. Specifically, Mobile IP has the following features.

- **No spatial and temporal limitations.** A user can take his/her laptop or palmtop computer to any place at any time without losing the connection to the home network.

- **Automatic configuration.** Mobile IP can automatically discover local mobility agents and connect mobile nodes to them.

- **No need to modify other nodes and routers.** Apart from mobile nodes and mobility agents, all the other nodes and routers remain unchanged, and keep using the current IP. Also Mobile IP leaves the transport and the higher-layer protocols unaffected.

- **No need to modify the current IP address and address format.** The current IP address and address format remain unchanged.

- **Security support.** Authentication is performed to ensure security protection.

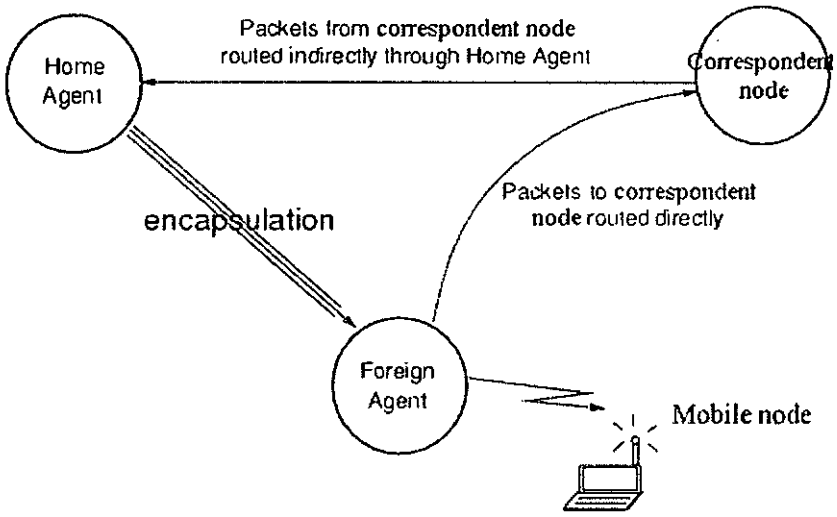


Figure 34.**Mobile IP**

In Mobile IP, each mobile node is assigned a long-term IP address called *home address* in its home network. While being away from the home network, the location of the mobile node is captured by a care-of address provided by a *foreign agent* in the foreign network. A *home agent* in the home network maintains a *mobility binding* between the home address and the care-of address in a *binding cache*. All packets destined to the mobile node are first routed with regular IP routing to its home network where they are captured by the home agent. The home agent then tunnels these packets to the foreign agent which, in turn, forwards them to the final destination.

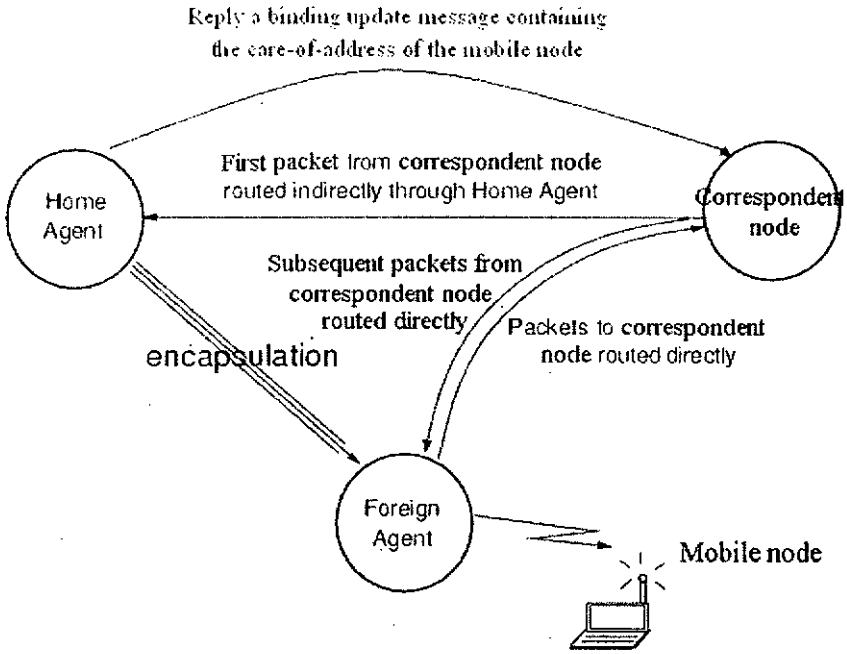


Figure 35. **Route optimization**

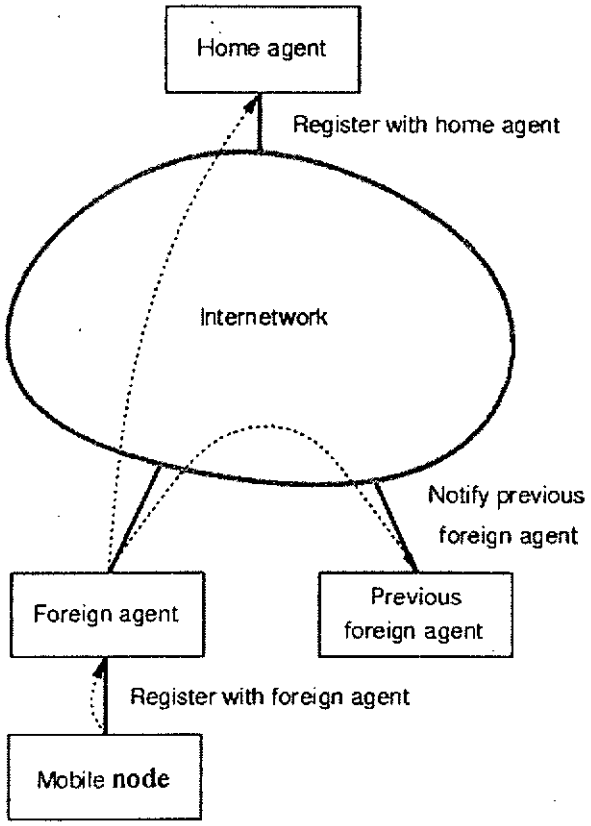


Figure 36. **Smooth handoff**

However, Mobile IP suffers from the triangle routing problem. Therefore, Mobile IP route optimization [15] as shown in Figure 35 has been proposed to alleviate this

problem. Any node that is willing to communicate with a mobile node maintains a binding cache. When the home agent intercepts a packet for the mobile node outside the home network, it may send a *binding update* message to the sender, informing it of the mobile node's current care-of address. The sender then updates its binding cache and tunnels any ensuing packet for the mobile node directly to its care-of address. An extension to the registration process, called *smooth handoff* [15], enables foreign agents to also make use of binding updates to reduce packet loss during a handoff. The mobile node may ask the new foreign agent to send a *Previous Foreign Agent Notification* message, which includes a binding update, to the previous foreign agent. The previous foreign agent then updates its binding cache and re-tunnels any packets for the mobile node to its new care-of address..

Although Mobile IP and the route optimization provide general mechanisms for mobility support in the Internet, there are still several performance problems that need to be addressed.

First, both Mobile IP and the route optimization require that a mobile node's home agent be notified of every location change. The route optimization further requires that every new location be registered with nodes that are actively communicating with the mobile node. Should the smooth handoff extension be implemented, an extra update message would be sent to the previous foreign agent. All these signaling messages for location management waste a lot of bandwidth.

Second, Mobile IP suffers from slow handoffs since the home agent has to handle all the handoffs, even though it may be far away from the current location of the mobile node. The network delay adds to the slow handoffs. All these will cause more packet loss since these packets are routed based on the outdated location information, which is especially harmful to real-time applications such as *Voice over IP* (VoIP) [44] and video streaming. Although the smooth handoff can, to some extent, reduce packet loss, tunneled packets that arrive at the previous foreign agent before the *Previous Foreign*

Agent Notification message are still lost since the previous foreign agent has not yet been aware of the migration of the mobile node. It will then rely on the upper-layer protocols such as TCP to retransmit these lost packets from the possibly distant source. This will incur the latency of the communication. TCP-based connections will also suffer from throughput reduction since the lost packets may be mistakenly treated as congestion and will result in TCP's slow start mechanism.

Third, since all handoffs are handled by the home agent, they cause lots of signaling traffic between the mobile node and the home agent. In high speed LANs, this is not an issue, but when low speed WANs are involved and a lot of mobile nodes are performing simultaneous handoffs, network congestion may occur.

In the next section, we introduce the mailbox concept into Mobile IP to resolve the above performance problems. However, we find that some parameters in the mailbox based framework are not applicable to the Mobile IP context because of the following reasons. In Mobile IP, mobile node can always migrate freely, making it impossible to guarantee the synchronization between the mailbox's packet delivery and the mobile node's migration. Sometimes, it is possible that the mobile node moves so fast that there is not enough time for the synchronization process to complete before the mobile node connects to a new foreign network. Also it is (almost) required that packets in Mobile IP are delivered to the destination as fast as possible, which means that the push mode of message delivery should be used. Therefore, in the Mobile IP context, only the following parameters in the mailbox based framework are meaningful: 1) NM, JM and FM in the mailbox migration frequency dimension, 2) PS in the mailbox-to-MO message delivery dimension, and 3) NS and SSM in the migration-delivery synchronization dimension.

5.2 The Protocol

Derived from the revised mailbox based framework, we propose a new protocol with the combination JM-PS-SSM. In this protocol, all the modifications are done at mobility agents without any changes to the existing operating system of mobile nodes. Every mobile node is associated with a mailbox, which is a data structure residing at a mobility agent. If a sender wants to send a packet to a mobile node, it simply sends the packet to the receiver's mailbox. Later the receiver receives the packet from its mailbox. (There is only one exception that a packet will not be sent to the mailbox but delivered directly to the receiver. This happens when the mobile node resides in its home network and the sender does not maintain the mobility binding for the receiver. Unless otherwise stated, the following discussions will preclude this condition.)

Initially, the mailbox is residing on the same network as its owner. The mobile node realizes that it has entered a new foreign network when it receives an *Agent Advertisement* [39] message from a new foreign agent. It then sends registration message to the old foreign agent where its mailbox resides. The old foreign agent then decides whether to move the mailbox to the new foreign agent with the consideration of two factors: the distance to the new foreign agent and the communication traffic of the mobile node. Here the communication traffic is defined as the number of packets expected to receive during the residence time at the new foreign agent. If the mobile node is expected to receive many packets while the distance is long, it will be costly to forward all these packets to the new address and better to move the mailbox closer to the mobile node so as to achieve a more optimal route. On the other hand, if the mobile node seldom receives packets or the distance is quite short, it is economical to leave the mailbox at where it was in order to reduce the registration overhead. The distance may be obtained from the routing table of the old foreign agent if a link state routing protocol such as OSPF [40] is used. The communication traffic is easy to obtain because the mailbox acts as a relay and buffer station of the mobile node.

In this work, we use a pair of thresholds (d, n) to determine the mailbox's migration. If either the distance exceeds d or the communication traffic exceeds n, the mailbox will migrate to the new foreign agent. Otherwise, the mailbox stays at the old foreign agent. We therefore differentiate two types of handoff, i.e., handoff without mailbox and handoff with mailbox, and we refer them as *local handoff* and *home handoff*, respectively. Upon each home handoff, a new threshold pair is calculated based on the current circumstances, which will be discussed later.

Besides mailbox, another new data structure called *address table* is defined in each mobility agent. Each entry in the address table has six attributes: 1) the home address of the mobile node, 2) the mailbox's address, 3) a *valid* tag, 4) the threshold pair, 5) a pointer to the mailbox, and 6) the care-of address of the mobile node. The *valid* tag is used to indicate whether the mailbox's address is outdated or not. Table 7 gives an example of address table where the first entry shows a remote mailbox in another mobility agent and the second entry shows a local mailbox.

Table 7. **An example of address table**

Home Addr	Mailbox's Addr	Valid Tag	Threshold Pair	Pointer to Mailbox	Care-of Addr
Addr A	Remote Address	*true*	Null	Null	Null
Addr B	Local Address	*true*	(d, n)	A Pointer	Addr C

The scheme also defines two processes, *Migrating* and *Packet-Forwarding*, which are presented in the following two subsections.

5.2.1 Migrating

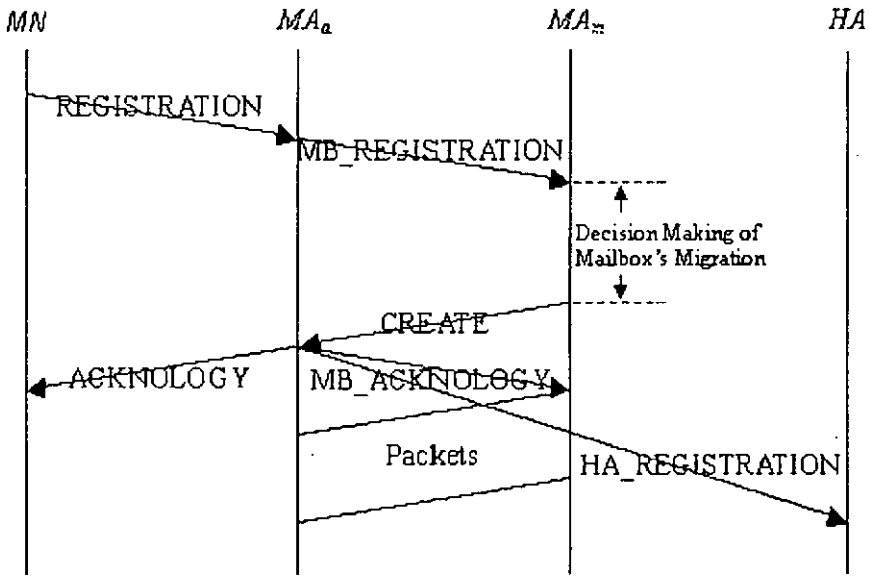


Figure 37.**Message exchange in registration**

Upon receiving the advertisement from a new mobility agent MA_a, the mobile node *MN* determines that it has crossed the boundary of the old network and roamed to a new one. It then initiates the registration process. It first uses gratuitous ARP[1] [41] to update the ARP caches of the nodes in the foreign network so that they associate *MN*'s link layer address with its home address. Then it sends a "REGISTRATION" message to MA_a. The key information contained in the "REGISTRATION" message is the address of the mobility agent MA_m where the mailbox *MB* currently resides. Figure 37 illustrates the message exchange in the registration process.

Upon receiving the "REGISTRATION" message, MA_a extracts the address of MA_m from the message, and sends a "MB_REGISTRATION" message to MA_m.

Upon receiving the "MB_REGISTRATION" message, MA_m decides whether to move *MB* to MA_a according to the threshold pair in the address table. In case when *MB* does

[1] Here the ARP refers to the Address Resolution Protocol instead of the Adaptive and Reliable Protocol proposed in Section 4.2.

not migrate, MA_m simply updates the care-of address of *MN* to MA_n. Otherwise, it will act as follows:

- setting the *valid* tag of the corresponding entry to *false* meaning that this mailbox will no longer be used; and
- sending a "CREATE" message to MA_n to request it to create a new mailbox *MB'* for *MN*.

Upon receiving the "CREATE" message, MA_n creates *MB'* and adds an entry to its address table recording this newly created *MB'*. Based on the current circumstances, it calculates a new threshold pair for the mobile node. It also sends three messages:

- an "ACKNOLOGY" message to *MN* informing it of the new address of its mailbox *MB'*,
- an "MB_ACKNOLOGY" message to MA_m telling it the creation of *MB'*, and
- an "HA_REGISTRATION" message to the home agent *HA* registering the new address of the mailbox.

After receiving the "MB_ACKNOLOGY" message, MA_m perform the following actions:

- updating the address of the mailbox in the address table to that of *MB'*;
- setting the *valid* tag to *true*;
- streaming every packet buffered in *MB* to *MB'*;
- informing the new address of the mailbox to the senders of the buffered packets in *MB* by sending "UPDATE" messages; and
- after all the packets have been streamed out, deconstructing the mailbox and setting null to the last three attributes in the address table, i.e., the threshold pair, the pointer to the mailbox and the care-of address.

After receiving the "HA_REGISTRATION" message, *HA* updates the address of the mailbox in its address table accordingly.

5.2.2 Packet-forwarding

If a correspondent node *CN* wants to send a packet to a *MN*, it first checks its binding cache to see whether or not the address of *MN* has been cached locally. If yes, it tunnels the packet to the cached address. Otherwise, it sends the packet with regular IP routing to the *MN*'s home address. Once the packet arrives at the home network, *HA* will intercept the packet since it acts as a proxy ARP server [41] for *MN*.

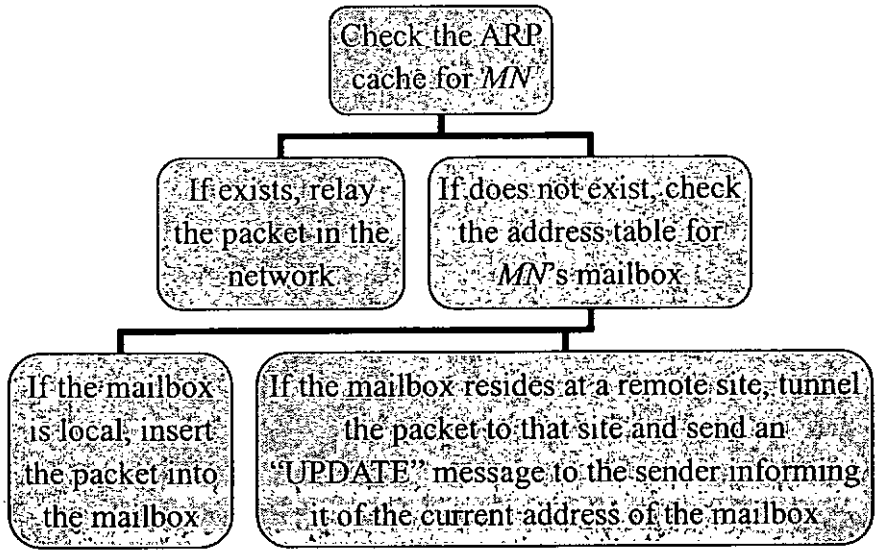


Figure 38. **Decision tree when receiving a packet**

When a mobility agent receives a packet destined to *MN*, it will perform actions according to the decision tree in Figure 38.

For a packet in *MB* to be forwarded to *MN*, the mobility agent MA_m first checks the *valid* tag. If it is *false*, i.e., the mailbox is migrating to the new foreign agent *FA*, the packet forwarding will be suspended and will rely on the migrating process to stream the packet to the new mailbox. If the *valid* tag is *true*, it tunnels the packet to the care-of address of *MN*.

5.3 Performance Modeling

In this section, we develop a performance model to be used for evaluating the proposed scheme. We first present a system model for a mobile network and two walk models for a mobile node, which are adopted in many existing studies such as [42] and [43]. Based on these models, we develop the cost functions for our scheme and the benchmark, which is the Mobile IP route optimization with the smooth handoff extension. The reason why we choose this as the benchmark is its relative high performance among the existing schemes. The cost function includes both the signaling transmission cost and the packet delivery cost.

5.3.1 System Model

The model assumes that the coverage area of the mobile network is partitioned into *cell*s. A cell is defined as the coverage area of a mobility agent that has the capability of exchanging packets with mobile nodes directly through the air. A mobility agent serves only one cell and cells do not overlap with each other.

A movement occurs when a mobile node moves from the residing cell to one of its adjacent cells. The distance between any two cells in the network is measured by the minimum number of cell boundary crossings required for a mobile node to travel from one cell to another. If we assume that a mobility agent is a router in a cell that can communicate directly with other mobility agents in the adjacent cells through wired link, the distance between two mobility agents can also be defined as the distance between their cells.

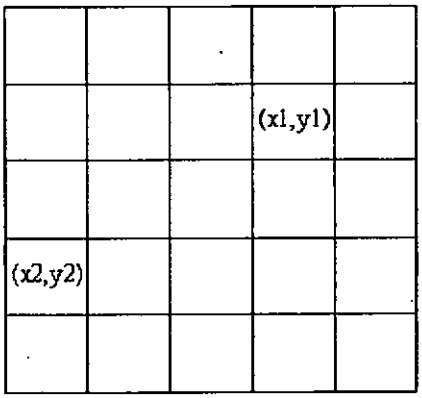


Figure 39. **Grid configuration of the mobile network**

We consider a grid configuration for the mobile network composed of equal-sized non-overlapping rectangular cells. In this grid configuration as shown in Figure 39, each cell has four neighbors. The distance between two cells with coordinates (x_1, y_1) and (x_2, y_2) is $|x_2-x_1|+|y_2-y_1|$.

5.3.2 Walk Models

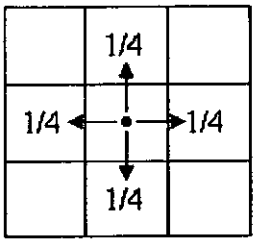


Figure 40. **Random walk model**

The cost function of our scheme depends on the walk model of a mobile node. Two walk models are considered here, namely *random walk model* and *directional walk model*. In the random walk model, a mobile node moves to one of its four neighbors with equal probability of 1/4 as shown in Figure 40.

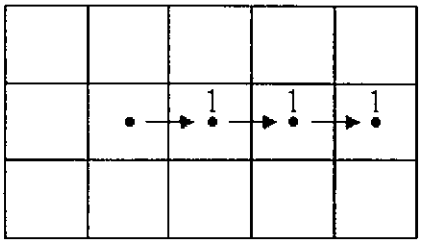


Figure 41. **Directional walk model**

The random walk model may be appropriate for pedestrian mobile users. For mobile vehicle users, a directional walk model is more appropriate. We assume that a mobile node moves towards only one direction in our analysis period. It moves to the next cell on the moving direction with a probability of 1 as shown in Figure 41.

In general, the cost function can be expressed as follows irrespective of what scheme is used.

$$Cost_{total} = Cost_{signaling} / ExpNum + Cost_{packet} \quad (40)$$

where ExpNum denotes the expected number of packets to be received within the residence time at a cell. We divide $Cost_{signaling}$ by ExpNum because we want the signaling cost of a handoff to be undertaken by all the packets delivered within the residence time after the handoff. Table 8 lists all the parameters for the performance model.

Table 8. **Parameters for our performance model**

Parameter	Explanation
$f_m(t)$	negative exponential probability distribution function of the packet's inter-arrival time
λ	mean packet arrival rate, i.e., $f_m(t) = \lambda e^{-\lambda t}$
$f_r(t)$	negative exponential probability distribution function of the mobile node's residence time at a cell
$1/\mu$	mean residence time at a cell, i.e., $f_r(t) = \mu e^{-\mu t}$
$Cost_{homehandoff}(X)$	signaling cost of the home handoff, given that it occurs at the mobile node's Xth migration
$Cost_{localhandoff}(i, X)$	signaling cost of the ith local handoff, given that the home handoff occurs at the mobile node's Xth migration
$Cost_{update}$	signaling cost of binding updates to the correspondent nodes

$Cost_{normal}(i, X)$	normal packet delivery cost during the period of the mobile node's $(i-1)$th and ith migration, given that the home handoff occurs at the mobile node's Xth migration
$Cost_{retransmission}(i, X)$	abnormal packet delivery cost due to packet loss and retransmission during the period of the mobile node's $(i-1)$th and ith migration, given that the home handoff occurs at the mobile node's Xth migration
$T_{normal}(i, X)$	time period that the mailbox has the correct care-of address of the mobile node during the period of the mobile node's $(i-1)$th and ith migration, given that the home handoff occurs at the mobile node's Xth migration
$T_{retransmission}(i, X)$	time period that the mailbox has the incorrect care-of address about the mobile node during the period of the mobile node's $(i-1)$th and ith migration, given that the home handoff occurs at the mobile node's Xth migration
$T(i, X)$	sum of $T_{normal}(i, X)$ and $T_{retransmission}(i, X)$, which is equal to $1/\mu$ on average
ExpMig	expected number of migrations that might trigger a home handoff
N	number of correspondent nodes
m	proportionality constant between the transmission cost (the transmission time) of the wireless link and that of the wired link
δ_{C_s}	proportionality constant between the signaling transmission cost and the transmission distance
δ_{C_p}	proportionality constant between the packet delivery cost and the transmission distance
δ_{T_s}	proportionality constant between the signaling transmission time and the transmission distance
$p(X)$	probability that the home handoff occurs at the mobile node's Xth migration
$p_d(X)$	probability that the home handoff occurs at the mobile node's Xth migration due to the threshold d
$p_n(X)$	the probability that the home handoff occurs at the mobile node's Xth migration due to the threshold n
$p_d^t(x)$	probability that the distance to the mailbox is t after the mobile node's xth migration, given that the distance threshold is d; as a special case, $p_d^d(x)$ means the probability that the distance to the mailbox after the mobile node's xth migration meets the threshold d which will trigger the home handoff
$f_X(x)$, $0 \le x \le X$	average distance to the mailbox after the mobile node's xth migration, given that the home handoff occurs at the mobile node's Xth migration

5.3.3 Cost Function of Our Protocol under the Random Walk Model

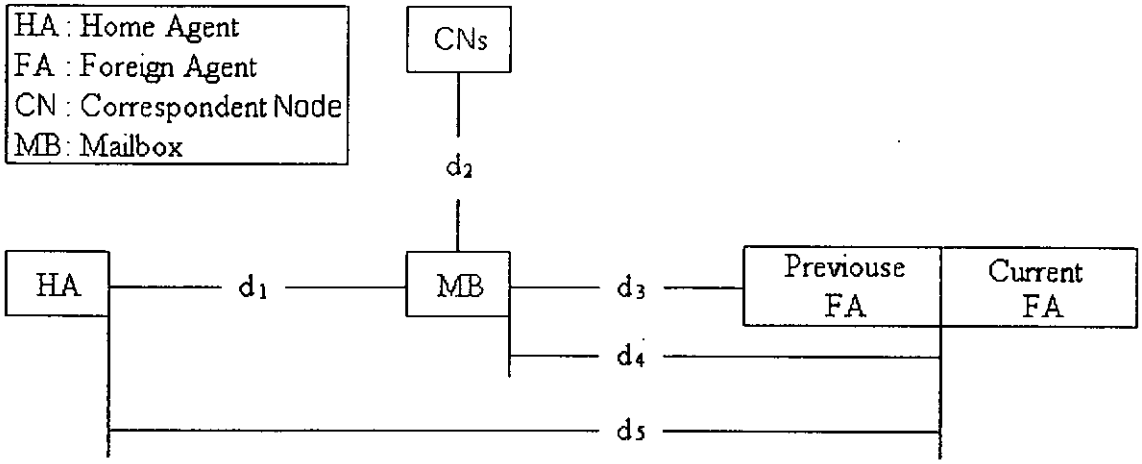


Figure 42. **Network scenario of our scheme**

Figure 42 depicts the network scenario about mobile node's migration in our scheme. The Current FA is the foreign agent after the mobile node's migration from the Previous FA. So the two foreign agents must be adjacent to each other, i.e., $d_4 = d_3 \pm 1$ where the ± sign depends on whether the mobile node migrates close to or far away from the mailbox. In case of the home handoff, the Current FA will become the new residing place for the mailbox. Here d_2 is the average distance to all the correspondent nodes.

Suppose the mailbox migrates at the mobile node's Xth migration, it is for sure that there are $X-1$ local handoffs and one home handoff. Thus $\text{Cost}_{\text{signaling}}$ is expressed as

$$\text{Cost}_{\text{signaling}} = \sum_{X=1}^{\infty} p(X) \frac{\text{Cost}_{\text{homehandoff}}(X) + \sum_{i=1}^{X-1} \text{Cost}_{\text{localhandoff}}(i, X) + \text{Cost}_{\text{update}}}{X} \tag{41}$$

where we can express $\text{Cost}_{\text{homehandoff}}(X)$, $\text{Cost}_{\text{localhandoff}}(i, X)$ and $\text{Cost}_{\text{update}}$ as follows (refer to Figure 37 for details).

$$\text{Cost}_{\text{homehandoff}}(X) = (2m + 3d_4 + d_5)\delta_{C_s} = (2m + 3f_X(X) + d_5)\delta_{C_s} \tag{42}$$

$$\text{Cost}_{\text{localhandoff}}(i, X) = (m + d_4)\delta_{C_s} = (m + f_X(i))\delta_{C_s} \tag{43}$$

$$Cost_{update} = N \times d_2 \times \delta_{C_s} \tag{44}$$

Since we assume a random packet incoming rate λ and a random migration rate μ, we have

$$ExpNum = \int_{t_r=0}^{\infty} \lambda t_r \times \mu e^{-\mu t_r} dt_r = \lambda/\mu \tag{45}$$

Similar to $Cost_{signaling}$, $Cost_{packet}$ is expressed as follows.

$$Cost_{packet} = \sum_{X=1}^{\infty} p(X) \frac{\sum_{i=1}^{X} \left(Cost_{normal}(i,X) \frac{T_{normal}(i,X)}{T(i,X)} + Cost_{retransmission}(i,X) \frac{T_{retransmission}(i,X)}{T(i,X)} \right)}{X} \tag{46}$$

where $Cost_{normal}(i, X)$ and $Cost_{retransmission}(i, X)$ are

$$Cost_{normal}(i,X) = (d_2 + d_3 + m)\delta_{C_p} = (d_2 + f_X(i-1) + m)\delta_{C_p} \tag{47}$$

$$\begin{aligned} Cost_{retransmission}(i,X) &= (d_2 + d_3 + m + d_4 + m)\delta_{C_p} \\ &= (d_2 + f_X(i-1) + m + f_X(i) + m)\delta_{C_p} \end{aligned} \tag{48}$$

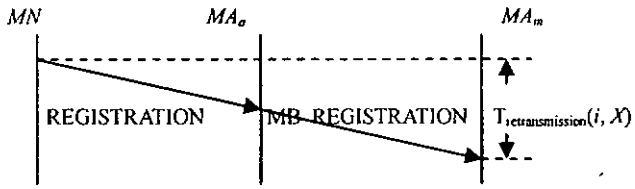


Figure 43. **Retransmission time**

According to the Figure 43, $T_{retransmission}(i, X)$ is the time period after the ith migration of the mobile node and before the "MB_REGISTRATION" message arrives at MA_m. So we have

$$T_{retransmission}(i,X) = (m + d_4)\delta_{T_s} = (m + f_X(i))\delta_{T_s} \tag{49}$$

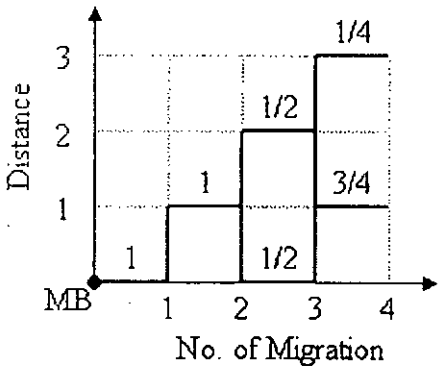


Figure 44. **Model of migration pattern**

Let us now try to derive *p(X)*. It is a little bit difficult and tricky to derive *p(X)*. We use Figure 44 to model the migration pattern of a mobile node.

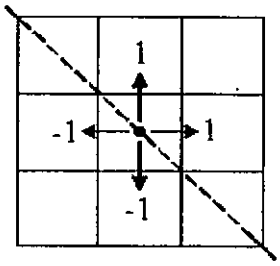


Figure 45. **Extending the y-axis to negative**

Initially, the mailbox and the mobile node are co-located. Therefore, when the x-axis is 0, the y-axis is also 0. After the first migration, the distance to the mailbox becomes 1 with a 100% probability. Thus we can see a value of 1 above the red stripe between 1 and 2 migrations. Next after the second migration, the distance could be either 2 or 0, each with a probability of 1/2. This probability distribution can be derived by using the following trick of extending the y-axis to negative. We define 1 to represent a movement to either up or right adjacent cell and −1 to represent a movement to either down or left cell as shown in Figure 45.

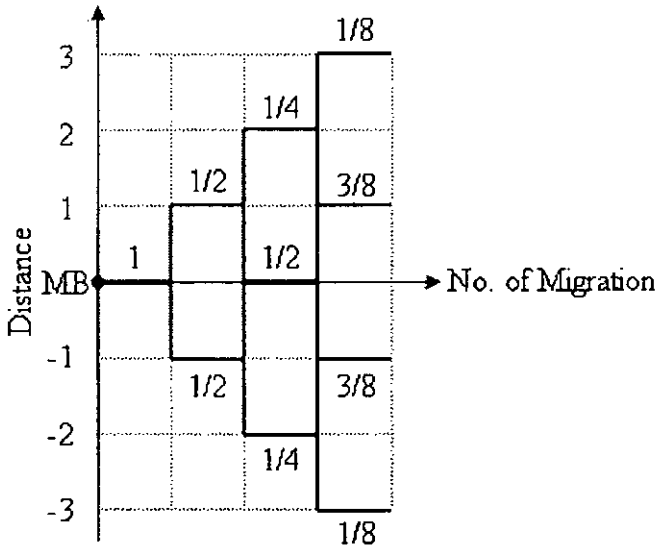


Figure 46. **Model of migration pattern with the new configuration**

With this new configuration, we can redraw Figure 44 as Figure 46 where Figure 44 can be viewed as the absolute of Figure 46.

So a mobile node's migration can be represented as a string of 1 and −1. To achieve a distance of t after the xth migration, we must have $(x+t)/2$ times of 1 and $(x-t)/2$ times of −1. This is a simple binomial distribution problem and we can express this probability distribution as the following equation.

$$\text{Prob}^{t}(x) = C_{x}^{\frac{x+t}{2}} / 2^{x}, \qquad t \in R \tag{50}$$

Now we can easily represent the probability distribution as shown in Figure 44 as follows.

$$p^{t}(x) = \begin{cases} \text{Prob}^{t}(x) + \text{Prob}^{-t}(x) = 2C_{x}^{\frac{x+t}{2}} / 2^{x}, & t > 0 \\ \text{Prob}^{0}(x) = C_{x}^{\frac{x}{2}} / 2^{x}, & t = 0 \end{cases} \tag{51}$$

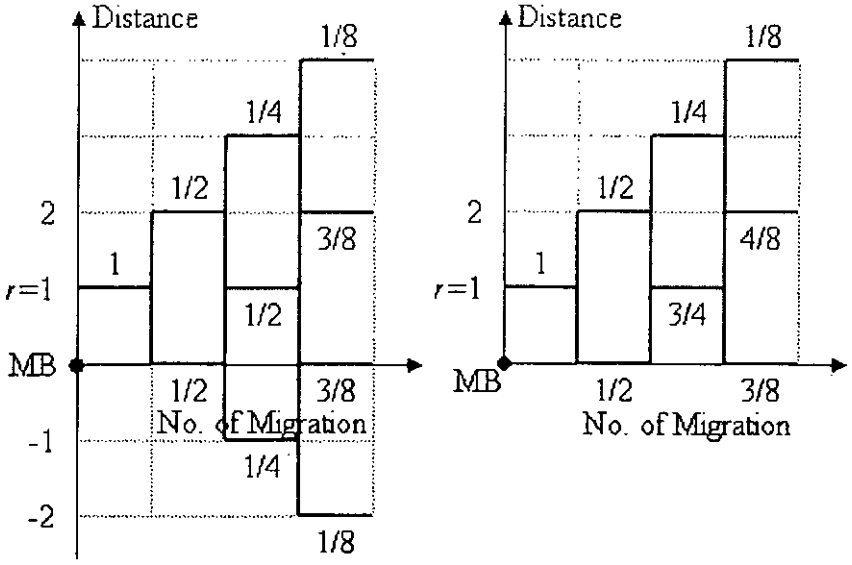


Figure 47. **Graphical representation of (52) and (53)**

The following two probability distribution functions are also useful in the derivation of *p(X)*. They are same as expressions (50) and (51) expect for the initial y-axis starting at *r*. Figure 47 graphically shows the two distributions with $r = 1$.

$$\mathrm{Prob}^{t,r}(x) = \mathrm{Prob}^{t-r}(x), \qquad t, r \in R \tag{52}$$

$$p^{t,r}(x) = \begin{cases} \mathrm{Prob}^{t,r}(x) + \mathrm{Prob}^{-t,r}(x) = \mathrm{Prob}^{t-r}(x) + \mathrm{Prob}^{-t-r}(x), & t > 0 \\ \mathrm{Prob}^{0,r}(x) = \mathrm{Prob}^{0-r}(x), & t = 0 \end{cases} \tag{53}$$

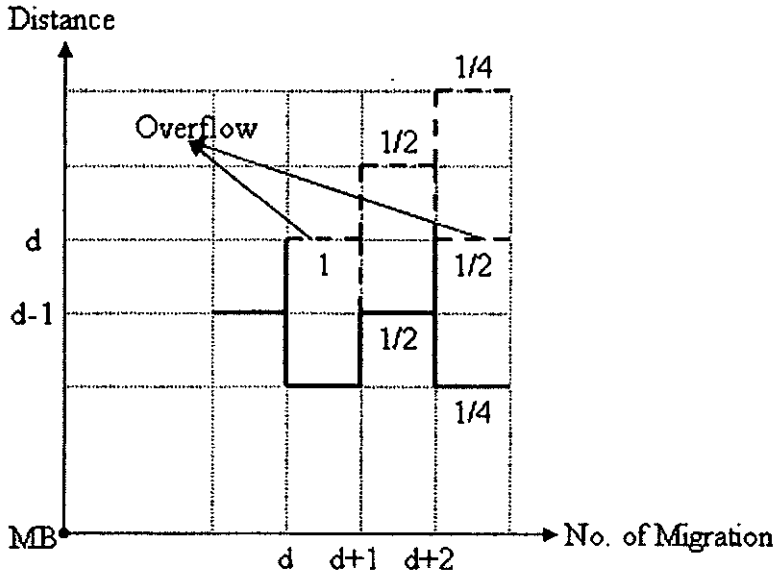


Figure 48. **The probability distribution of** $p_d^d(x)$

Next let us find out $p_d^d(x)$. Obviously, when $x < d$, we have $p_d^d(x) = 0$ and when $x = d$, we obtain $p_d^d(x) = p^d(X) = 1/2^{d-1}$. When $x = d+2k+1$ (for $k = 0, 1, \ldots$), $p_d^d(x)$ is also equal to 0 as shown in Figure 48. However, when $x = d+2k$ (where $k = 0, 1, \ldots$), $p_d^d(x)$ will not be easy to derive since once the threshold d is met, the probability overflow will take place.

If there is no overflow at the dth migration, we will have $p_d^d(d+2) = p^d(d+2)$. But now there is overflow, therefore, $p_d^d(d+2)$ is equal to $p^d(d+2)$ minus the part that the overflow $p_d^d(d)$ should contribute as shown in the red dash line in Figure 48, i.e., $p_d^d(d+2) = p^d(d+2) - p_d^d(d)/2$. In general, we can express $p_d^d(x)$ as

$$p_d^d(x) = \begin{cases} 0, & x < d \text{ or } x = d+2k+1 \\ p^d(x), & x = d \\ p^d(x) - \sum_{i=0}^{\frac{x-d}{2}-1} p_d^d(d+2i)p^{d,d}(x-d-2i), & x = d+2k \end{cases} \quad (54)$$

Similarly, $p_d^t(x)$ can be derived as $p^t(x)$ minus the part contributed by the overflow and we have

$$p_d^t(x) = \begin{cases} p^t(x), & x < 2d-t \\ 0, & x = 2d-t+2k+1 \\ p^t(x) - \sum_{i=0}^{\frac{x-2d+t}{2}} p_d^d(d+2i)p^{t,d}(x-d-2i), & x = 2d-t+2k \end{cases} \quad (55)$$

where $0 \le t \le d$.

Now we can finally derive $p(X)$. If the threshold $d = 1$, it is for sure that the home handoff takes place each time the mobile node migrates. So we have

$$p(X) = \begin{cases} 1, & X = 1 \\ 0, & X > 1 \end{cases} \tag{56}$$

When $d > 1$, the home handoff may not take place at each mobile node's handoff. Suppose the home handoff occurs at the mobile node's Xth handoff, there will be no home handoff at any of the previous $X-1$ handoffs. So

$$p(X) = (1 - \sum_{i=1}^{X-1} p(i))(1 - (1 - p_d(X))(1 - p_n(X))) \tag{57}$$

where $p_d(X)$ and $p_n(X)$ are expressed as follows.

$$p_d(X) = \frac{p_d^d(X)}{\sum_{t=0}^{d} p_d^t(X)} \tag{58}$$

$$\begin{aligned} p_n(X) &= \text{Prob(expected no. of packets} \geq n) \\ &= 1 - \text{Prob(expected no. of packets} < n) \\ &= 1 - \sum_{i=0}^{n-1} \text{Prob(expected no. of packets} = i) \\ &= 1 - \sum_{i=0}^{n-1} \left[\int_{t_r=0}^{\infty} \left(\frac{(\lambda t_r)^i}{i!} e^{-\lambda t_r} \right) \mu e^{-\mu t_r} dt_r \right] \\ &= \left(\frac{\lambda}{\lambda + \mu} \right)^n \end{aligned} \tag{59}$$

With the help of $p(X)$, we obtain the expected number of migrations that might trigger a mailbox's migration.

$$\text{ExpMig} = \sum_{X=1}^{\infty} X \times p(X) \tag{60}$$

Finally, let us derive $f_X(x)$. Suppose the mailbox will migrate at the mobile node's Xth migration. So when $x < X$, $p_d^d(x)$ must be 0. So

$$f_X(x) = \sum_{i=0}^{d-1} \frac{p_d^i(x)}{\sum_{t=0}^{d-1} p_d^t(x)} \times i, \qquad x < X \tag{61}$$

When $x = X$, a home handoff takes place. As mentioned earlier, this happens either because of d or n. If it is because of d, it is for sure that $f_X(x) = d$. If it is because of n, $f_X(x)$ is the same as when $x < X$. So

$$f_X(x) = \frac{p_d(x)}{1-(1-p_d(x))(1-p_n(x))} \times d + \frac{p_n(x)-p_d(x)p_n(x)}{1-(1-p_d(x))(1-p_n(x))} \times \sum_{i=0}^{d-1} \left(\frac{p_d^i(x)}{\sum_{t=0}^{d-1} p_d^t(x)} \times i \right), \quad x = X \tag{62}$$

5.3.4 Cost Function of Our Protocol under the Directional Walk Model

In the directional walk model, since the mobile node handoffs to its next cell along its moving direction, the number of migrations before a home handoff takes place is bounded by the distance threshold d. Therefore, we should rewrite (41), (46) and (60) as follows.

$$Cost_{signaling} = \sum_{X=1}^{d} p(X) \frac{Cost_{homehandoff}(X) + \sum_{i=1}^{X-1} Cost_{localhandoff}(i,X) + Cost_{update}}{X} \tag{63}$$

$$Cost_{packet} = \sum_{X=1}^{d} p(X) \frac{\sum_{i=1}^{X} \left(Cost_{normal}(i,X) \frac{T_{normal}(i,X)}{T(i,X)} + Cost_{retransmission}(i,X) \frac{T_{retransmission}(i,X)}{T(i,X)} \right)}{X} \tag{64}$$

$$ExpMig = \sum_{X=1}^{d} X \times p(X) \tag{65}$$

where all the terms but $p(X)$ and $f_X(x)$ remain unchanged.

For $p(X)$, only $p_d(X)$ requires modification as

$$p_d(X) = \begin{cases} 0, & X < d \\ 1, & X = d \end{cases} \tag{66}$$

For $f_X(x)$, since the distance to the mailbox increases by 1 each time a migration takes place, we thus have

$$f_X(x) = x \tag{67}$$

5.3.5 Threshold Optimization

The cost function of our scheme highly depends on the chosen threshold pair (d, n). Therefore, it is important to have a method to derive the optimal threshold pair to minimize the cost function. We apply the same iterative algorithm as that used in Section 4.4.1.2. We starting the algorithm with d = 1 and n = 1. We iteratively increase d by 1 until the cost difference between the systems with the current and previous threshold pairs starts to be positive, i.e., d has reached its optimal value. After that, we apply the same technique to find out the optimal value of n. The only difference is that we use k×ExpNum as the increment of n since n could be very large and depends on ExpNum. A scale k is used to control the accuracy of n. A smaller value of k implies greater accuracy but requires more iteration. In our experiment, we choose k = 0.2. The algorithm is shown below:

- Define the cost difference function:

  $$\Delta(d, n, d', n') = \text{Cost}_{\text{total}}(d, n) - \text{Cost}_{\text{total}}(d', n')$$

  where (d, n) and (d', n') represents the current and previous threshold pairs, respectively.
- Initialization: d = 1; n = 1;

- Step 1:

$$\text{while } \Delta(d+1, n, d, n) < 0$$
$$d = d + 1;$$
$$d_{optimal} = d;$$

- Step 2:

$$\text{while } \Delta(d_{optimal}, n + k \times ExpNum, d_{optimal}, n) < 0$$
$$n = n + k \times ExpNum;$$
$$n_{optimal} = n;$$

5.3.6 Cost Function of the Benchmark

Let us first define some new parameters:

- $Cost_{HARegistration}$: the signaling cost of registering the home agent;
- $Cost_{retransmission}$: the abnormal packet delivery cost due to packet loss and retransmission and it is important to notice that the packet is retransmitted from the sender instead of the mailbox as in our scheme.

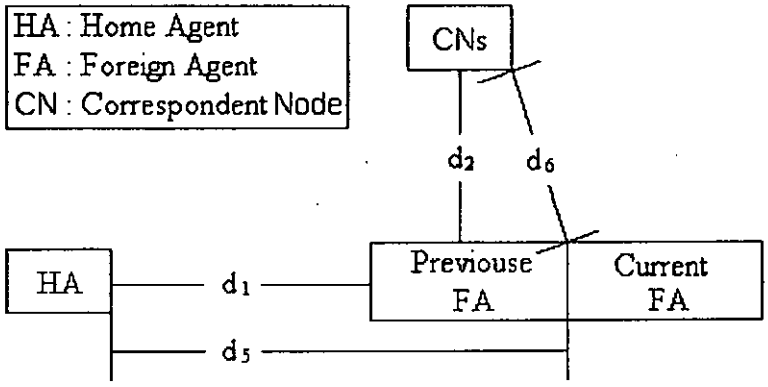


Figure 49. **Network scenario of the smooth handoff**

The above diagram depicts the network scenario about mobile node's migration in the smooth handoff. During each handoff, the Current FA will send registration messages to both the home agent and the Previous FA. It will also update each correspondent node's binding cache. So

$$Cost_{signaling} = Cost_{HARegistration} + Cost_{update} \quad (68)$$

$$Cost_{HARegistration} = (m + 1 + d_5)\delta_{C_s} \quad (69)$$

The packet forwarding cost is somewhat similar to that in our scheme when the home handoff occurs at its first handoff. The only difference is the retransmission cost.

$$Cost_{packet} = Cost_{normal}(1,1) \times \frac{T_{normal}(1,1)}{T(1,1)} + Cost_{retransmission} \times \frac{T_{retransmission}(1,1)}{T(1,1)} \quad (70)$$

$$Cost_{retransmission} = (d_2 + m + d_6 + m)\delta_{C_s} \quad (71)$$

5.4 Performance Evaluation

In this section, we compare the performance of our scheme under the two walk models with that of the benchmark. Notice that the values of expressions (42) and (69) vary as d_5 varies. Also (71) changes as d_6 changes its value. To be more convincing, we compare our scheme under the worst case with the smooth handoff under the best case. Therefore, we take the upper bound value of (42) when d_5 is in its maximum $d_1 + f_X(X)$, and the lower bound value of (69) and (71) when d_5 and d_6 are in their minimum $d_1 - 1$ and $d_2 - 1$, respectively. Table 9 lists some of the parameters used in our performance evaluation. We set δ_{C_s}, the signaling transmission cost per hop, as a normalized value 1. The packet delivery cost per hop δ_{C_p} is twice as high as the signaling cost. δ_{T_s} can be viewed as the signaling processing time on a router and we set it 0.05sec. We also assume the transmission cost (the transmission delay) in the wireless environment is twice as high (long) as that in the wired environment. Finally, there are five active correspondent nodes.

Table 9. **Parameters for performance evaluation**

δ_{C_s}	δ_{C_p}	δ_{T_s}	m	N
1	2	0.05sec	2	5

Other parameters that might affect the cost function are ExpNum, d_1 and d_2. We choose ExpNum as the x-axis and select two (d_1, d_2) pairs: (100, 0) and (0, 100). The first pair visualizes the scenario when the mobile node is far away from the home agent while close to its correspondent nodes and the second pair shows the scenario in a reverse condition.

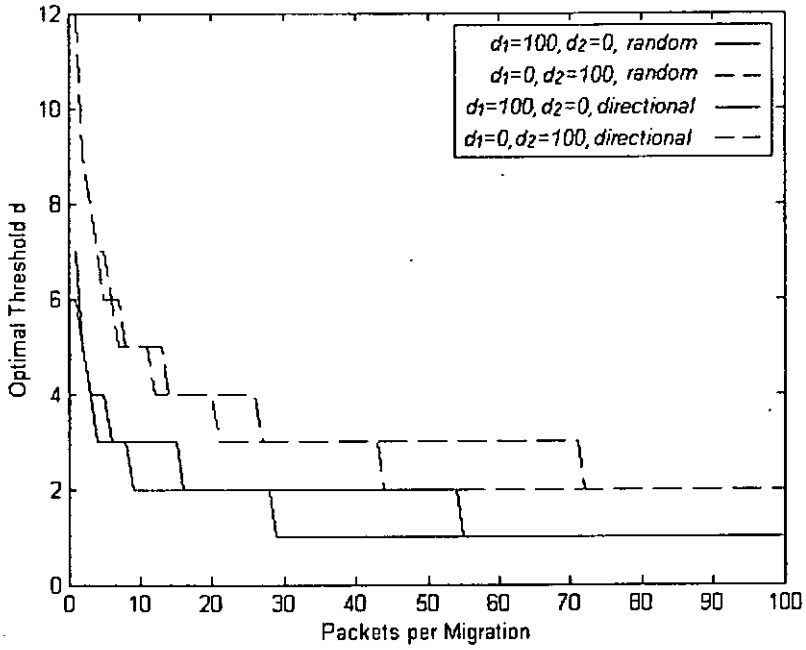


Figure 50. **Optimal threshold *d***

Figure 50 shows the changes of the optimal threshold d with different ExpNum. In general, d decreases as ExpNum increases no matter which distance pair and walk model are used. This is because when ExpNum gets bigger, that is to say, the mobile node is expected to receive more packets during its residence time in the new foreign network, it is thus not economical to leave the mailbox at its old place which will result in suboptimal route. By decreasing the value of d, it is more likely that this threshold is met and the home handoff occurs at an early time before the suboptimal route becomes severe. Under the same (d_1, d_2) pair, the directional walk model has a little bit higher value of d. This is because, should the same d be used, the random

walk model usually takes more local handoffs before a home handoff takes place than the directional walk model. Thus the directional walk model must not be in its optimum since the home handoff occurs so early that the cost saving due to the local handoffs cannot make up the cost wasting due to the home handoff. With a larger d, the directional walk model can achieve a balance between cost saving and wasting so as to achieve the optimum.

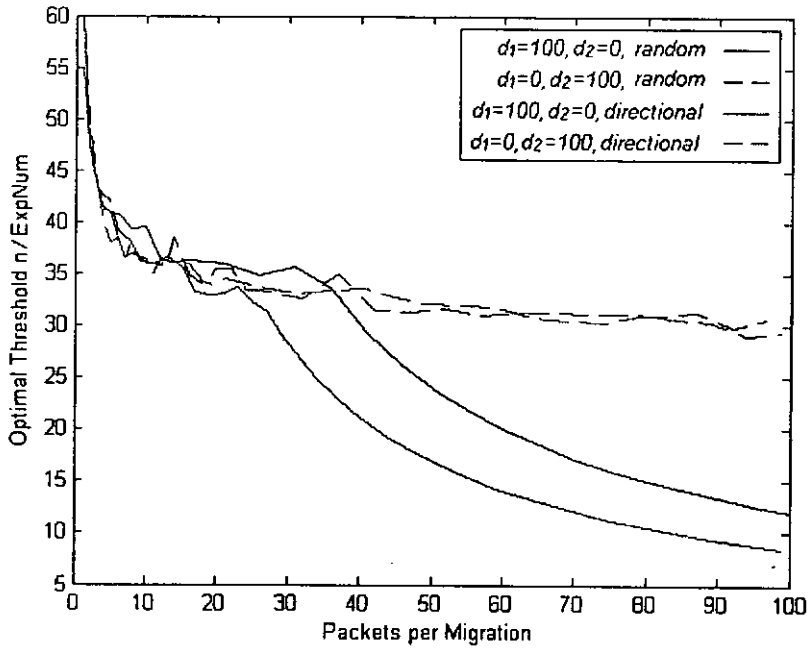


Figure 51. **Optimal threshold n/ExpNum**

Figure 51 demonstrates the changes in the optimal threshold n with different ExpNum and we can make similar observations as in Figure 50. However, we use n/ExpNum as y-axis instead of just n because as ExpNum increases, n should also increase. But the increasing rate of n is not as fast as that of ExpNum, which makes the home handoff take place more likely with larger ExpNum.

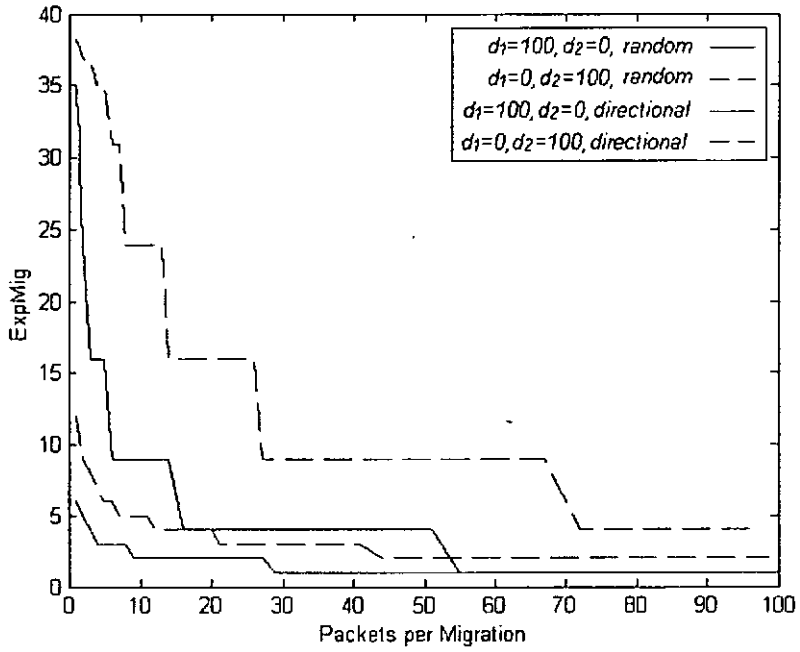


Figure 52.**ExpMig**

In Figure 52, we can observe that as the increment of ExpNum, the expected number of local handoffs before a home handoff takes place drops due to the decrease of both *d* and *n*.

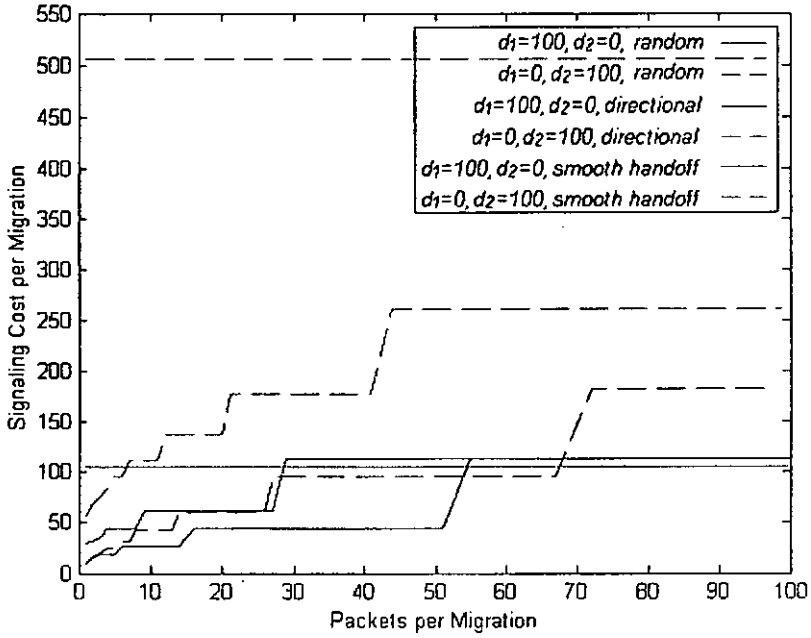


Figure 53.**Signaling cost per migration**

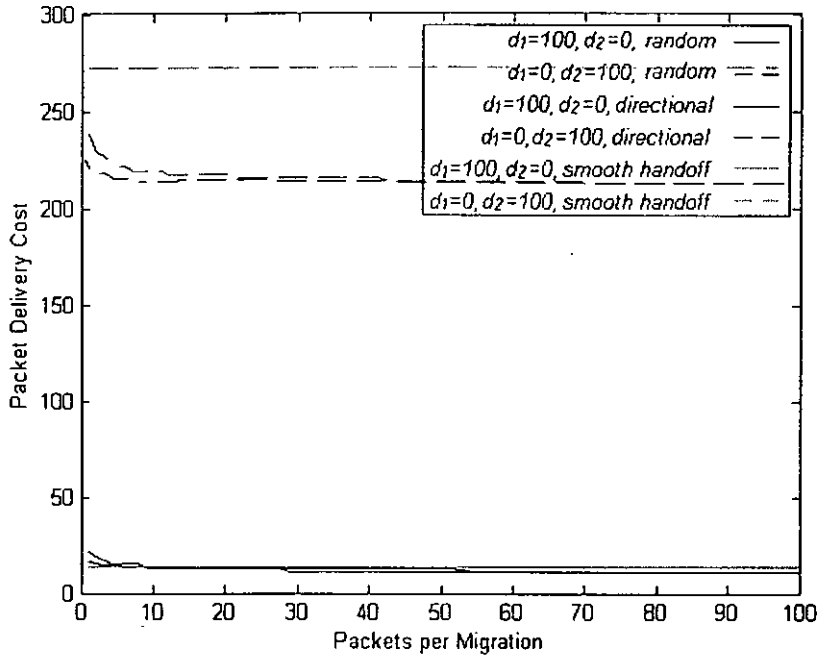


Figure 54. **Packet forwarding cost**

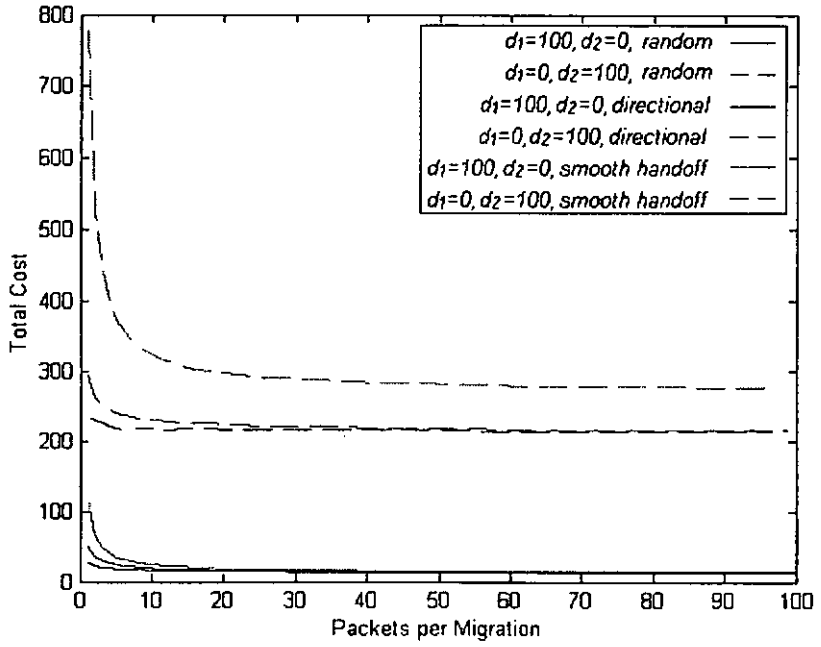


Figure 55. **Total cost**

The last three figures show the signaling transmission cost, the packet forwarding cost and the total cost, respectively. We observe that our scheme outperforms the smooth

handoff in all the three costs, especially when ExpNum is small. This is because with small ExpNum, our scheme takes more advantages of the local handoffs and thus saves more cost. As ExpNum increases, the signaling cost of our scheme increases to approach that of the smooth handoff, which means that it is getting more and more likely that the home handoff occurs after the first handoff of the mobile node just like the smooth handoff. The packet delivery cost is basically a decreasing curve in our scheme because of the more optimal route achieved as ExpNum becomes bigger. However, due to the large difference in the retransmission cost, the packet delivery cost of our scheme will outperform that of the smooth handoff provided that the correspondent nodes are not in the same network as the mobile node. The same reason applies to the total cost.

5.5 Summary

In this chapter, we introduce the mailbox concept to Mobile IP and propose a new protocol from the mailbox based framework, which represents the JM-PS-SSM combination of parameters. With this protocol, we can achieve adaptive location management that enables dynamic tradeoff between the packet delivery cost and the location management cost so as to minimize the total communication cost. The packet retransmission cost could also be reduced since normally the mailbox is located somewhere in the network close to the receiver. Like Chapter 4, we also build an analytical model for the protocol to derive the optimal threshold pair (d, n) where the performance is maximized.

Chapter 6. Conclusion and Future Work

6.1 Conclusion

As the emergence of mobile communication paradigm, many new challenges and requirements are introduced into the design of communication protocols for MOs, which includes location transparency, reliability, asynchrony, efficiency, scalability and adaptability. To optimize all these requirements often results in dilemma situation. More often than not tradeoffs enhance one capability at the expense of another. Most of the existing mobile communication schemes, however, do not handle the tradeoffs properly, making it difficult to adapt to different application requirements. Therefore, new approaches are required to design adaptive mobile communication protocols that can dynamically tradeoff among different application requirements so as to achieve the best performance. The novel mailbox based scheme proposed in this thesis is one of such desirable approaches.

In the mailbox based scheme, each MO is associated with a dedicated mailbox to buffer incoming messages for it. With such a configuration, message passing between the sender and the receiver requires two steps: 1) from the sender to the receiver's mailbox, and 2) from the mailbox to the receiver. Although logically part of the MO, the mailbox can be detached from its owner in the sense that the MO can leave its mailbox at one site while it is leaving for another site. During each MO's migration, a choice can be made whether to bring its mailbox together with it to the new site or simply leave along. It is just because of this flexibility about the mailbox's location that enables the protocols designed adaptive to specific application requirements by simply adjusting parameter values of the following design considerations: the mailbox migration frequency, the mailbox-to-MO message delivery and the migration-delivery synchronization.

To explore the benefits of the mailbox based scheme, we would like to apply the scheme to two specific contexts: mobile agent and Mobile IP. In the mobile agent context, we propose a new protocol called ARP which has been proved in the thesis to possess many desirable features. With the ARP, senders do not need to know the receiver's current location. Messages are first sent to the cached address or the receiver's home address, and later forwarded to the receiver's mailbox, where they are finally received by the receiver through a pull operation. Synchronization between the message delivery and the mailbox's migration is used to avoid message loss. The protocol guarantees that messages are forwarded at most once before they reach the mailbox of the receiver. Asynchrony is improved because constraints on agent's mobility are released as synchronization only involves mailbox, and mobile agent can migrate to new locations whenever they want without waiting for the messages in transit. Scalability is also improved since the workload on the agent home about the location management and the message delivery is now distributed to all the hosts along the mailbox's migration path. Most importantly, the protocol can also work in an adaptive way since it performs in a JM mode and the criteria that determine whether or not to move the mailbox remain unspecified. The designer can base on the particular requirements of an application to design a set of adaptive policies that govern the migration pattern of the mailbox to maximize the fulfillment of the requirements according to the real-time network environment.

To further enhance efficiency and reliability, we extend the ARP with a path pruning algorithm and a two-layer fault-tolerant architecture. In the path pruning algorithm, the location management cost is greatly reduced as the algorithm can effectively identify and remove those redundant hosts on the mailbox's migration path so that fewer messages are required during each location management. The byproduct of this algorithm is the garbage collection capability of removing the useless addresses maintained in a cache. In the two-layer fault-tolerant architecture, the low layer abstracts away network failures and provides reliable point-to-point message passing between two MAPs; the high layer overcomes message loss due to host failures and

accomplishes reliable end-to-end message delivery between two mobile agents.

In the Mobile IP context, we propose a new protocol which can achieve adaptive location management that enables dynamic tradeoff between the packet delivery cost and the location management cost so as to minimize the total communication cost. Since the mailbox is located somewhere in the network close to the receiver, the packet retransmission cost could also be reduced. To optimize the performance, we build an analytical model for the protocol, which can be used to derive the best threshold condition.

6.2 Future Work

6.2.1 Application to Next Generation Network

In this thesis, we do not explore the application of the mailbox based scheme to the voice transmission in mobile telephone systems because of the following two reasons. First, the voice communication is a real-time application which requires the instant transmission of voice data. In the current systems, once a connection is established, a dedicated direct link between the caller and the callee is reserved, which derives that should the mailbox based scheme be applied, the mailbox would always have to move and collocate with its owner mobile user, making it look dummy to insert a mailbox between the two communicating users. Second, most of the existing mobile telephone systems use *circuit switching* which is not feasible, if not impossible, to use a mailbox to buffer and forward voice data.

However, starting from *General Packet Radio Service* (GPRS), *packet switching* is getting more and more involved in mobile telephone systems. Currently, the *Third Generation* (3G) mobile telephone systems have already deployed packet switching in the backbone network. Technologies such as VoIP and SS7 over IP [45] make it

possible to allow both voice data and signaling data transmitted over a packet switching network. On the other hand, powerful mobile phones that can process more complicated applications and media types are continuously pushed into the market. It has been expected that in the near future, we will be able to experience all kinds of convenient mobile services through whatever access network available by simply operating at one single handheld mobile phone.

All the above new technologies, applications and devices make it both feasible and meaningful to exercise our mailbox based scheme in the next generation network. The preliminary idea is that the mailbox, acting as a virtual secretary, handles all types of messages for its owner. Different messages are categorized according to their relative importance and emergency, and the mailbox can intelligently choose the best way and the best time to deliver them. For example, for those emergent or real-time messages, the mailbox may at once push these messages to its owner or issue a SMS message to alert its owner to pick up them; for those un-emergent or non-real-time messages, they may be delivered to the owner at either non-peak or cheap-price hours; for those junk messages like advertisement, the mailbox can simply drop them. However, how to best configure the behavior of the mailbox remains a challenging task to get future researched.

6.2.2 Mailbox Based Multicast

Multicast is a group communication mechanism to send messages to a group of receivers called *members*. It is now getting more and more popular and applied to many mobile communication systems. For example, in a mobile telephone or Mobile IP system, multicast is often used in applications such as video conferencing, massive SMS/MMS delivery, advertisement and online gaming. In a mobile agent system, mobile agents are often grouped to perform certain operations cooperatively and will use multicast as an essential communication mechanism.

- In [46], the authors classified multicast requirements of different applications along four dimensions with each dimension describing a distinct feature:
  - **Delivery atomicity** refers to the feature that either all the group members receive a message safely or none at all.
  - **Delivery ordering** specifies the order in which multicast messages should be delivered to the group members. There are several degrees of message ordering: *arbitrary ordering*, *FIFO ordering*, *causal ordering* and *total ordering*, where the arbitrary ordering possesses the weakest degree and the total ordering possesses the strongest degree. A weaker degree of ordering is a subset of a stronger degree.
  - **Real-time delivery** specifies the time constraints within which a multicast session should complete.
  - **Fault tolerance** refers to the capability that a multicast protocol can continue functioning correctly in the presence of failures.

There has been a wide range of multicast mechanisms developed for different applications in the last decade. However, the presence of mobility makes it more difficult to design multicast protocols that can satisfy the above requirements. For example, it would be more difficult to ensure message delivery ordering and real-time delivery. One direction of our future research is to expand our mailbox based scheme for the multicast capability. The preliminary idea is that a group of MOs share a common mailbox dedicated for multicast. However, special algorithms should be figured out to overcome difficulties caused by the MO's mobility and to satisfy the requirements defined above.

References

[1] A. Samjani, "Mobile Internet Protocol", *IEEE Potentials*, 20(1):16-18, February/March 2001.

[2] M. Strasser, J. Baumann and F. Hohl, "Mole – A Java Based Mobile Agent System", *Proceedings of the 10th European Conference on Object Oriented Programming*, July 1996.

[3] A. Pham and A. Karmouch, "Mobile Software Agents: An Overview", *IEEE Communications Magazine*, 36(7):26-37, July 1998.

[4] C. Perkins, "IP Mobility Support", RFC 2002, October 1996.

[5] M. S. Corson, J. P. Macker and G. Cirincione, "Internet-Based Mobile Ad Hoc Networking", *IEEE Internet Computing*, 3(4):63-70, July/August 1999.

[6] B. Awerbuch and D. Peleg, "Online Tracking of Mobile Users", *Journal of the ACM*, 42(5):1021-1058, September 1995.

[7] W. V. Belle, K. Verelst and T. D'Hondt, "Location Transparent Routing in Mobile Agent Systems – Merging Name Lookups with Routing", *Proceedings of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems*, December 1999.

[8] X. Tao and X. Feng, "Communication Mechanism in Mogent System", *Journal of Software*, 11(8):1060-1065, August 2000.

[9] S. Mohan and R. Jain, "Two User Location Strategies for Personal Communication Services", *IEEE Personal Communications*, 1(1):42-50, First Quarter 1994.

[10] D. Lange and M. Oshima, "Programming and Deploying Mobile Agents with Aglets", Addison-Wesley, 1998.

[11] H. Jeon, C. Petrie and M. R. Cutkosky, "JATLite: A Java Agent Infrastructure with Message Routing", *IEEE Internet Computing*, 4(2):87-96, March/April 2000.

[12] J. Baumann, "Communication Concepts for Mobile Agent Systems", *Proceedings*

of the 1st International Workshop on Mobile Agents, April 1997.

[13]S. Lazar, I. Weerakoon and D. Sidhu, "A Scalable Location Tracking and Message Delivery Scheme for Mobile Agents", *Proceedings of the 7th IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, June 1998.

[14]J. Rain, Y. Lin, C. Lo and S. Mohan, "A Caching Strategy to Reduce Network Impacts of PCS", *IEEE Journal on Selected Areas in Communications*, 12(8):1434-1444, October 1994.

[15]C. Perkins and D. Johnson, "Route Optimization in Mobile IP", IETF Draft, November 1997.

[16]R. Prakash and M. Singhal, "A Dynamic Approach to Location Management in Mobile Computing Systems", *Proceedings of the 8th International Conference on Software Engineering and Knowledge Engineering*, June 1996.

[17]T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms", MIT Press and McGraw-Hill, 1990.

[18]W. K. Ng and C. V. Ravishankar, "Coterie Templates: A New Quorum Construction Method", *Proceedings of the 15th International Conference on Distributed Computing Systems*, May 1995.

[19]K. Ratnam, I. Matta and S. Rangarajan, "A Fully Distributed Location Management Scheme for Large PCS Networks", *Journal of Interconnection Networks*, 2(1):85-102, March 2001.

[20]P. Krishna, N. H. Vaidya and D. K. Pradhan, "Location Management in Distributed Mobile Environments", *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems*, September1994.

[21]M. Steen, F. Hauck, P. Homburg and A. Tanenbaum, "Locating Objects in Wide-Area Systems", *IEEE Communications Magazine*, 36(1):104-109, January 1998.

[22]C. Perkins, "Mobile IP: Design Principles and Practices", Addison Wesley, 1998.

[23]C. Perkins, A. Myles and D. B. Johnson, "IMHP: A Mobile Host Protocol for the Internet", *Computer Networks and ISDN Systems*, 27(3):479-491, December

1994.

[24]M. Ranganathan, M. Bednarek and D. Montgomery, "A Reliable Message Delivery Protocol for Mobile Agents", *Proceedings of the 2nd International Symposium on Agent Systems and Architectures and the 4th International Symposium on Mobile Agents*, September 2000.

[25]T. Okoshi, "MobileSocket: Session Layer Continuous Operation Support for Java Applications", *Transactions of Information Processing Society of Japan*, 41(2):222-234, February 2000.

[26]A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", Technical Report DCS-TR-314, Rutgers University, 1994.

[27]E. Jul, H. Levy, N. Hutchinson and A. Black, "Fine-grained Mobility in the Emerald System", *ACM transaction on Computer Systems*, 6(1):109-133, February 1988.

[28]A. Murphy and G. Picco, "Reliable Communication for Highly Mobile Agents", *Proceedings of the 1st International Symposium on Agent Systems and Architectures and the 3rd International Symposium on Mobile Agents*, October 1999.

[29]J. Cao, X. Feng, J. Lu and S. K. Das, "Mailbox-Based Scheme for Mobile Agent Communications", *IEEE Computer*, 35(9):54-60, September 2002.

[30]D. B. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents", *Communication of the ACM*, 42(3):88-89, March 1999.

[31]J. E. White, "Mobile Agents Makes a Network an Open Platform for Third-party Developers", *IEEE Computer*, 27(11):89-90, November 1994.

[32]D. Johansen, R. Renesse and F. B. Schneider, "Operating System Support for Mobile Agents", *Proceedings of the 5th Workshop Hot Topics in Operating Systems*, May 1995.

[33]B. Thomsen, L. Leth, F. Knabe and P. Y. Chevalier, "Mobile Agents", ECRC External Report, European Computer-Industry Research Center, 1995.

[34]E. Pitoura and B. Bhargava, "A Framework for Providing Consistent and Recoverable Agent-based Access to Heterogeneous Mobile Databases", *ACM*

SIGMOD Record, 24(3):44-49, September 1995.

[35]J. Desbiens, M. Lavoie and F. Renaud, "Communication and Tracking Infrastructure of a Mobile Agent System", *Proceedings of the 31st Annual Hawaii International Conference on System Sciences*, January 1998.

[36]Network Research Group, Lawrence Berkeley National Laboratory, ns-LBNL Network Simulator, URL: http://www-nrg.ee.lbl.gov/ns/.

[37]L. Kleinrock, "Queuing Systems - Volume I and II", John Wiley and Sons, New York, 1976.

[38]H. Xie, S. Tabbane and D. J. Goodman, "Dynamic Location Area Management Overhead and Performance Analysis", In Proceedings of the 43rd IEEE Vehicular Technology Conference, 1993.

[39]C. Perkins, "IP Mobility Support for IPv4", RFC 3220, January 2002.

[40]J. Moy, "OSPF Version 2", RFC 1274,.October 1996

[41]J. Postel, "Multi-LAN Address Resolution", RFC 925, October 1984.

[42]Y. Wang, W. Chen and J. Ho, "Performance Analysis of Mobile IP Extended with Routing Agents", *Proceedings of the 2nd European IASTED International Conference on Parallel and Distributed Systems*, July 1998.

[43]I. F. Akyildiz, J. S. M. Ho and Y. B. Lin, "Movement-Based Location Update and Selective Paging for PCS Networks", *IEEE/ACM Transactions on Networking*, 4(4):629-638, August 1996.

[44]C. Ibe, "Converged Network Architectures: Delivering Voice and Data Over IP, ATM, and Frame Relay", Wiley, November 2001.

[45]R. Stewart, Q. Xie and K. Morneault, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[46]X. Jia, J. Cao and W. Jia, "A Classification of Multicast Mechanisms: Implementations and Applications", *Journal of Systems and Software*, 45(2):99-112, March 1999.

[47]A. Tanenbaum, "Computer Networks", Prentice Hall, 2003.

[48]R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, N.J., 1957.