SPREAD:
SECURE PROTOCOL FOR RELIABLE DATA DELIVERY

By

WENJING LOU

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2003

To
My husband, Jianrong Hong
My mother, Suwan Huang
My father, Guanmu Lou
and
My daughter, Joyce Xinyi Hong

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Yuguang Fang, for his invaluable advice, encouragement and motivation during the course of this work. This dissertation would not have been possible without his guidance and support. I also thank him for his philosophical advice on both my academic and non-academic life, which made me more mature, scholastically and personally.

I also thank Prof. Jose Fortes, Prof. Janise McNair, Prof. Shigang Chen, and Prof. Richard Newman, for serving in my committee and for their attention and advice. Thanks also go to Prof. Tan Wong and Prof. John Shea, for their many constructive suggestions and advice.

Many of my colleagues and friends have contributed to this dissertation. I would like to thank Wei Liu for the assistance with the simulation. I also thank Xiang Chen, Younggoo Kwon, and many other students in the Wireless Networks Laboratory (WINET) for the years of friendship and many helpful discussions.

Last but not least, I am indebted to my family. Every member of my family (my husband, my parents, and my young daughter) made countless sacrifices in one way or another to ensure my Ph.D dream coming true. This dissertation is dedicated to them. It is their love, belief, and constant support that made everything I have possible.

This material is based upon work supported by the Office of Naval Research Young Investigator Award under grant N000140210464, and the National Science Foundation Faculty Early Career Development Award under the contract ANIR0093241.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research or the National Science Foundation.

TABLE OF CONTENTS

page

LIST OF TABLES

LIST OF FIGURES

Figure page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

SPREAD:
SECURE PROTOCOL FOR RELIABLE DATA DELIVERY

By

Wenjing Lou

August 2003

Chair: Yuguang "Michael" Fang
Department: Electrical and Computer Engineering

Security is a critical issue in a mobile ad hoc network (MANET). The current research and development in this area is still in its infancy. In this dissertation, we proposed and investigated a novel scheme, Secure Protocol for REliable dAta Delivery (SPREAD), to improve network security by enhancing the data confidentiality service in a MANET. The proposed SPREAD scheme aims to provide further protection to keep secret messages from being compromised (or eavesdropped) when they are delivered across insecure networks. The basic idea is to transform a secret message into multiple shares by secret sharing schemes; and then deliver the shares via multiple paths to the destination, so that even if a small number of the nodes used to relay messages are compromised, the secret message as a whole is not compromised.

This dissertation focuses mainly on developing SPREAD in a MANET environment. We presented the overall system architecture and investigated three major design issues. We first described how to obtain message shares using the secret sharing

schemes. Then we studied the appropriate choice of secret sharing schemes and the optimal allocation of message shares onto each path so that the security can be maximized. Results show that SPREAD is more secure and also provides a certain degree of reliability because of the redundancy introduced without compromising the security. Third, we discussed the routing and multiple path routing techniques in a MANET; and developed a path set optimization algorithm to find the multiple paths with the desired properties (i.e., as many as possible independent paths and at the same time as secure as possible). Simulation results justify the feasibility of the SPREAD approach and show the effectiveness and other performance metrics.

The major difference between developing SPREAD in a MANET and in a wired network lies on the routing protocols. In the last part of this dissertation, we expand the SPREAD idea into wired networks (the Internet) by presenting a multipath routing protocol which is suitable to wired networks.

CHAPTER 1
INTRODUCTION

1.1 Motivation

A mobile ad hoc network (MANET) is a self-configurable, self-organizing, infrastructureless multi-hop mobile wireless network. By self-configurable and self-organizing, we mean that a MANET can be formed, merged together, or partitioned into separate networks on the fly, depending on networking needs; and few administrative actions need to be performed for network setup and maintenance. By infrastructureless, we mean that a MANET can be promptly deployed without relying on any existing infrastructure (such as base stations for wireless cellular networks). By multi-hop wireless, we mean that in a MANET, the routes between end users may consist of multiple wireless hops, as compared to the single wireless hop in a wireless local area network (WLAN) or a cellular network, where only the last hop (e.g., from the end user to the access point or the base station) is wireless; and all the links beyond that point remain wired. In addition, each node in an ad hoc network is capable of moving independently; thus the network topology can change continuously and dramatically. Each node also functions as a router that discovers and maintains routes to other nodes and forwards packets for other nodes. Rapidly deployable and self-organizing features make the ad hoc network very attractive in tactical and military applications, where fixed infrastructures are not available or reliable; and fast network establishment and self-reconfiguration are required. Primary applications of an ad hoc network include tactical communication in a battlefield, disaster rescue after an earthquake, and so on, where the

environment is hostile and the operation is security-sensitive. Recent availability of wireless communication devices that operate in the ISM (Industrial, Scientific, and Medical) band has extended the interest in ad-hoc networks to civilian life such as on-the-fly setup for conferencing and home-area wireless networks.

Although mobile ad hoc networks have attracted tremendous attention in the past few years, most research efforts focused on developing the network architecture itself -- particularly network routing protocol and medium access control (MAC) protocol design. Relatively little work addressed security considerations. Recent history with the Internet and cellular networks tells us that if a given network architecture is not designed with security consideration from the very start, the security vulnerabilities will be exploited by malicious users, and the network might be paralyzed by various types of attacks. Moreover, addressing security issues as an after thought can be very painful, expensive, and inefficient. Thus, incorporating security aspects into the currently formalized ad hoc networking architecture is of paramount importance.

Computer network and information security have been studied extensively in the wired Internet context in the past. A number of effective security services and mechanisms are already in place. However, due to the salient features (infrastructureless, wireless, mobile, self-organizing, etc.) of a MANET, security solutions that are valid in the Internet may not be fully applicable in a MANET environment. Compared with a fixed network or a wired network, the characteristics of an ad hoc network pose many new challenges in security. First of all, the wireless channels suffer from poor protection and are more susceptible to attacks such as passive eavesdropping, active signal interference, and jamming. Secondly, most ad hoc routing protocols are cooperative in

nature and rely on implicit trust relationships to route packets among participating nodes. This cooperative nature makes ad hoc protocols more vulnerable to data tampering, impersonation, and denial-of-service (DoS) types of attacks. Thirdly, the lack of a fixed infrastructure and a central concentration point makes many conventional security solutions difficult to apply. For example, it makes it difficult for an intrusion detection system to collect audit data, and also impedes the deployment of widespread asymmetric cryptography because of the lack of a PKI (Public Key Infrastructure), where a centralized certificate authority (CA) is needed. Fourthly, mobile devices tend to have low battery power, limited memory, slow processing capability, and finite radio transmission bandwidth, which limit the practical deployment of computationally intensive or more comprehensive security schemes in MANET environments. Finally, continuous and unpredictable ad hoc mobility clouds the distinction between normalcy and anomaly, which makes it difficult to detect malicious behaviors [1].

Because of these new challenges, many security solutions that have been effective in a wired network become inapplicable in a MANET. Much effort has been made to develop applicable security solutions dedicated to a MANET environment. Among them, key management, probably the most critical and fundamental security issue in a MANET, has attracted much attention [2-4]. A number of secure routing protocols have also been proposed to protect the correctness of different types of ad hoc routing protocols (both table-driven/on-demand; and distance vector/source routing) [5-8]. Issues addressed particularly for ad hoc networks include handling node misbehavior [9-11], intrusion detection [12], and so on [1] (see section 1.3.1).

The scheme suggested in this dissertation addresses an important issue in network security: data confidentiality service in a MANET. Data confidentiality is protecting transmitted data from passive attacks, such as eavesdropping. Sensitive information, such as tactical military information transmitted across a battlefield (an ad hoc network), requires confidentiality. Leakage of such information to enemies could cause devastating consequences. The wireless channel in a hostile environment is vulnerable particularly to the eavesdropping. Messages transmitted over the air can be eavesdropped from anywhere without having the physical access to the network components. Conventionally, confidentiality is achieved by cryptography. However, the limited resources, such as the limited battery power and processing capability, restrict the use of computationally intensive encryption schemes in a MANET. The computationally efficient encryption schemes sometimes are not secure enough. For example, the WEP (Wired Equivalent Privacy) protocol defined in IEEE 802.11 uses RC4 algorithm, which is a stream cipher and computationally efficient. However, it has been discovered that the cipher text can be decrypted through traffic analysis and dictionary-building attack. After analysis of about a day's worth of traffic, real-time automated decryption of all traffic is made possible [13]. A more severe problem in a MANET is that mobile nodes usually reside in an open and hostile environment. Nodes themselves might be compromised. For example, in the battlefield scenario, nodes might be captured. In this case, all the credentials stored in the nodes would be compromised, including the keys used to encrypt the message. Any encryption scheme, no matter how secure it is, would not help.

Based on these observations, we proposed a novel scheme, Secure Protocol for REliable dAta Delivery (SPREAD), to statistically enhance data confidentiality in a

MANET. The fundamental idea of SPREAD is shown in Figure 1-1. Assume that we have a secret message. If we send it through a single path, the enemy can compromise it by compromising any one of the nodes along the path. However, if we divide it into multiple pieces, and send the multiple pieces via multiple independent paths, then the enemy would have to compromise all of the pieces from all of the paths to compromise the message. Improved security can be achieved by this means.

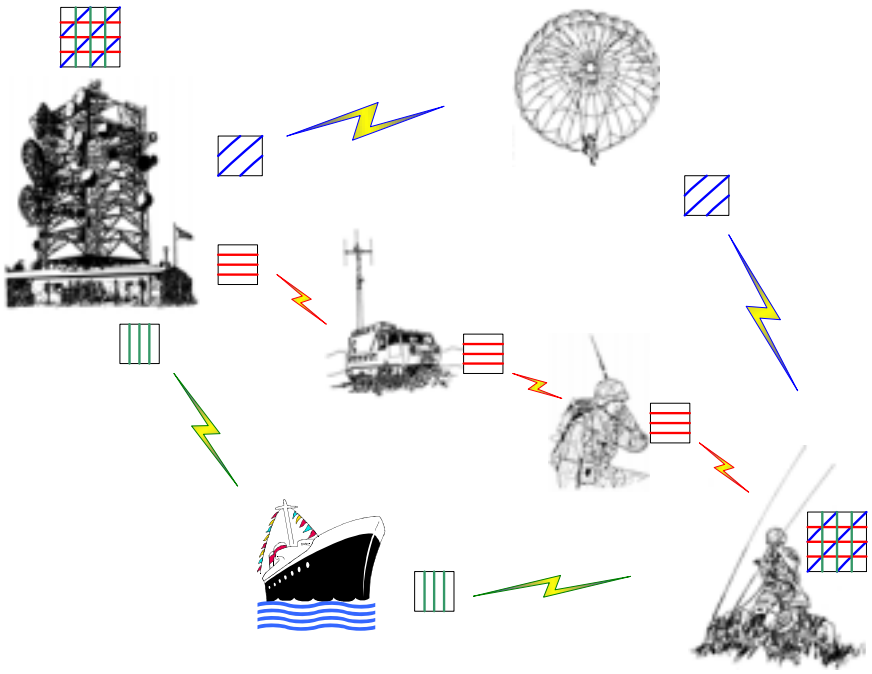


Figure 1-1. Fundamental idea behind SPREAD.

Here, to compromise the message, the enemy must accomplish at least two things. First, the enemy must physically intercept all pieces of the message. This can be done by either eavesdropping or compromising nodes. Either way, by spreading the message pieces over multiple paths, the enemy would have more difficulty collecting all of the

pieces. Secondly, we assume link encryption between neighboring nodes; each link has different keys. Key management is problematic in a MANET, however, the establishment of a shared session key between neighboring nodes is not that difficult [3]. So even if the enemy has collected all of the pieces, he/she must decrypt them. The decryption can be done either by compromising the nodes to get the keys; or by a brute-force type attack or traffic analysis. A brute-force type attack or traffic analysis requires the collection of a large amount of encrypted data by the same key. The more data, the better chance the decryption. Spreading the traffic onto multiple paths makes it harder for the enemy to decrypt the message. Improved security can be expected from both aspects.

The SPREAD idea is based on two principles, secret sharing and multipath routing. The secret message is divided into multiple shares (pieces) by secret sharing systems. A threshold secret sharing scheme allows us to divide a secret into multiple shares and requires the knowledge of a certain number (threshold) of shares to reconstruct the original secret. Any less than the threshold number of shares provide its holder no greater chance of recovering the secret than what an outsider who knows nothing at all about the secret sharing system. The secret sharing scheme is computationally efficient and unconditionally secure in the sense that the security it provides is independent of the computing time or power that an opponent may bring to subvert the system, or put in another way, even with infinite computing power, the cheater can do no better than guess [14].

Multipath routing, sometimes called traffic dispersion, has been one of the most important current directions in the area of routing. In a reasonably well-connected network, there would be several paths between any source-destination pair. While the

current routing is based on the single shortest path routing, multipath routing gives the source node a choice at any given time of multiple paths to a particular destination by taking advantage of the connectivity redundancy of the network. Multipath routing has been studied for various network control and management issues in various types of networks. For example, this approach has been applied to aggregate bandwidth and minimize delay, to support Quality of Service (QoS) routing, to smooth the burstiness of the traffic, to alleviate network congestion, and to improve the fault tolerance, etc. [15]. Multipath routing has been shown to be effective in coping with the frequent topological changes and improving resilience to node/link failure in a MANET [16-19]. In our SPREAD scheme, by using multipath routing to deliver the shares, the security obtained from the secret sharing system can be enhanced while the shares are transmitted across the network.

Basically there are three major design issues in the implementation of the SPREAD idea: how to divide the secret message into multiple shares using secret sharing principle, how many paths do we need and how to allocate and shares onto each paths, and how to find the desired multiple paths in a MANET. We will discuss these questions one by one in this dissertation.

1.2 Scope and Organization of the Dissertation

In this dissertation, we focus on how to exploit this SPREAD idea to develop a secure protocol to enhance the data confidentiality in mobile ad hoc networks. We address the improved security by dealing with the compromised nodes and eavesdropping problem. We evaluate the performance statistically for both individual attacks and colluded attacks. For the individual attack we assume that each adversary is working on his/her own. While for the colluded attack, we assume that multiple compromised nodes

are working together to recover the message. Once the total number of shares they compromise exceeds the threshold, the message is considered compromised. We only consider the security when messages are transmitted across the network, assuming that the source and destination are trusted. The protection of each individual node from being compromised is out of the range of this dissertation. We also assume that the adversaries, after compromising the nodes, will try to remain in the network by launching only passive attacks in order to acquire more secure information. If the compromised nodes launch active attacks, such as stopping forwarding packets for other nodes or altering the information when forwarding, some intrusion detection mechanism [12] or the misbehavior detection schemes such as a watchdog proposed in [9] can be used to identify the compromised node quickly so that it will be excluded from the network.

This dissertation is organized as follows.

Chapter 1 introduces the motivation and basic idea of the proposed SPREAD scheme. We also describe the scope and organization of this dissertation and summarize the related work in this chapter. The related work is organized into two categories. First we briefly review the wireless security research in mobile ad hoc networks. Then we review the related work that uses the similar multipath routing approaches as our SPREAD scheme, although for different objectives.

Chapter 2 describes the system architecture of the proposed SPERAD scheme and outline three major design issues in the implementation of SPREAD scheme: how to divide the secret message into multiple shares, how many paths do we need and how to allocate and shares onto each paths, and how to find the desired multiple paths in a MANET.

The first issue is addressed in more detail in Chapter 3. We gave a brief introduction to the threshold secret sharing system and describe how it is applied to the secret message to be protected in our SPREAD scheme.

Chapter 4 is dedicated to address the second issue, how to select the paths and how to choose an appropriate secret sharing algorithm and how to allocate the shares onto each selected paths such that the maximum security can be achieved. We also show that the share allocation scheme can be designed in such a way that it provides certain degree of reliability without sacrificing security.

We address the third question in Chapters 5 and 6 by discussing the routing and multipath routing techniques in a MANET and the path set optimization techniques for our SPREAD scheme. In Chapter 5, we study the performance of an ad hoc on-demand routing protocol, Dynamic Source Routing (DSR) [20], and present a link-cache route caching strategy based on DSR to improve the routing efficiency. The proposed route caching strategy can be incorporated to many other multipath routing protocols and it provides a partial view of the network topology. It allows the multipath optimization algorithm to be applied independent of the routing protocols used. Then in Chapter 6, we study a maximal node-disjoint path finding algorithm, which is able to find as many as possible paths and at the same time as secure as possible.

The first six chapters of this dissertation are dedicated to develop SPREAD scheme in a MANET environment. Security is an important issue in wired networks (Internet) too. In Chapter 7, we expand the SPREAD idea into wired networks scenario. The major difference between the deployment of SPREAD idea in wired networks and MANETs lies in the routing protocols. So in Chapter 7 we propose and study a distributed multipath

routing protocol which is suitable for stable wired networks. We show that, with comparable low complexity, the proposed protocol is able to find multiple node-disjoint paths for any source-destination pair. We also show the effectiveness of the SPREAD idea in wired networks. Conclusions are drawn in Chapter 8.

1.3 Related Work

1.3.1 Wireless Security in Mobile Ad Hoc Networks

Security is an important issue in a mobile ad hoc network. As we mentioned earlier, compared with a wired or infra-structured network, the characteristics of a MANET pose many new challenges in security that made many security solutions effective in a wired network not applicable in a MANET. Much work has been done to develop security solutions customized for a MANET. Two research focuses in the current literature are key management and secure routing protocols.

Key management is possibly the most critical and complex issue when talking about security in a mobile ad hoc network. The applicability of many other security services, such as confidentiality and authentication, relies on effective and efficient key management. For efficiency reasons, the parties involved in a secure communication usually need to share a common secret key. Public key cryptography has made key distribution easier among those parties in the wired Internet. For example, some public key cryptography based key exchange algorithms, such as the Diffie-Hellman key exchange algorithm [21], have been widely adopted to establish the session keys between parties without transmitting the keys themselves in the network. However, the management of the public keys usually involves a centralized trusted control point, called a *certificate authority (CA)*. Such centralized trust control contradicts the design goal of MANETs, where there is no infrastructure. Some research has been carried out to address

the public key management issue in MANETs. Basically, there are two major research directions along this line. One is to retain the *certificate authority (CA)* concept, but distribute its functionality into multiple servers (or trusted nodes) [2,3]. In this way, both the availability and the security of the CA can be improved. This approach is also based on the secret sharing and multipath routing principles. Each server holds one share of the system key, multiple servers collaborate through multipath routing to perform the functions of the CA. of Another approach is to discard the centralized CA, and instead, create a totally distributed and self-organized key management system [4], similar to the approach used in PGP, a widely used Email security system [22].

Secure routing protocols is another important issue in a mobile ad hoc network. Routing in ad hoc networks is more vulnerable than its counterpart in wired networks. Correct routing can be disrupted in many ways or be disabled by denial of service attacks. Much effort has been made to protect routing protocols. Vulnerabilities of ad hoc routing protocols have been identified [1] and generally classified into four categories: modification, fabrication, replay, and denial of service. Several secure routing protocols that aim to protect the correctness of routing protocols have been proposed. Traditional source authentication and message integrity measures have been adopted [23] to prevent modification of the routing information and limit the fabrication of the routing information. Particularly, the use of a one-way hash chain was proposed to protect the unauthorized modification of the mutable fields, such as the hop count, in a routing packet [5]. In response to the key management difficulty and the limited resource restrictions, authentication mechanisms without using public key cryptography are also proposed [5-8]. Different mechanisms have been proposed for different types of ad hoc

routing protocols (table-driven and on-demand, distance vector and source routing). Both preventive schemes and reactive schemes are developed. For example, in [7], a secure routing protocol (SRP) was proposed to provide end-to-end authentication for source routing based protocols. The scheme assumes a prior security association (SA) between source and destination while the existence of SAs with any of the intermediate nodes is not necessary. In [6], another secure routing protocol (Ariadne) was proposed based on Dynamic Source Routing (DSR). A per-hop hashing technique was proposed to verify that no node is missing from the node list in the route request message and all the nodes listed in the route request are legitimate nodes. The use of a broadcast authentication scheme, called TELSA [24], which requires loose time synchronization, was introduced.

Another interesting research area in wireless security is the handling of node misbehavior [9-11]. An interesting mechanism, called *watchdog*, was proposed to identify misbehaving nodes [9]. The watchdog's mechanism is based on the promiscuous mode of radio interface: the receiver of one node could listen to the transmission of any of its neighbors, regardless of the intended destination of that transmission. Thus, when a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet correctly by overhearing the next node's transmission. If the next node does not forward the packet(s) correctly, then it is misbehaving. This mechanism takes advantage of the broadcast wireless channel particularly.

Some other research works that have been done to address the security issue in ad hoc networks include authentication [25,26], intrusion detection [12], preventing traffic analysis [27,28], and so on. For a more detailed survey on wireless security in mobile ad hoc networks, readers are referred to [1].

The SPREAD scheme presented in this dissertation is a novel approach and addresses another important issue - data confidentiality while data are transmitted across the network.

1.3.2 Multipath Routing

Multipath routing, sometimes called traffic dispersion, has been one of the most important current directions in the area of routing [15]. In a reasonably well-connected network, there maybe exist several paths between a source-destination pair. While the current routing is based on the single shortest path routing, multipath routing gives the source node a choice at any given time of multiple paths to a particular destination by taking advantage of the connectivity redundancy of the network. The traffic may take one of the multiple paths at a time or it can flow through multiple paths simultaneously.

Multipath routing (or *dispersity routing* as called by the author) was first proposed by Maxemchuk to spread the traffic from a source in space rather than in time as a means for load balancing and fault handling in packet-switched networks [29-31]. The method was shown to equalize load and increase overall network utilization. With redundancy, it improved the delay and packet loss properties at the expense of sending more data through the network. After that, the multipath routing technique has been applied for many other objectives, to aggregate bandwidth and minimize delay, to support Quality of Service (QoS) routing, to smooth the burstiness of the traffic, to alleviate network congestion, and to improve the fault tolerance, etc [32-36]. Most of the work has been focused on evaluating gain by dispersion with various granularities. Many topics still remain to be investigated, such as the routing algorithms capable of finding right number of paths with desired properties, what the desired properties of the paths are, partially disjoint paths, synchronization problem, and so on.

Multipath routing has attracted extensive attention in mobile ad hoc networks recently. The network topology of a MANET can change frequently and dramatically. One reason is that nodes in a MANET are capable of moving collectively or randomly. When one node moves out of/in the transmission range of the other, the link between the two becomes down/up. Another reason that causes the topological changes is the unstable wireless links, which might become up and down due to the signal fading (obstacles between the two end nodes), signal interference, or the changing of transmission power level. Most of the mobile nodes are battery powered, when the nodes run out of the battery power, the node failure will also cause the topological changes. Much work has been done to examine the potential of the multipath routing technique in coping with this topological dynamics [16-18]. Several multipath routing protocols have been developed [19,37-39] for MANET environment and the simulation results proved that the multipath routing is effective in improving the robustness of data delivery, balancing the traffic load thus balancing the power consumption among nodes, reducing the end-to-end delay, and so on.

A few efforts have been made to improve the network security by using multipath routing. Yang et al. [40] proposed to improve the network security by traffic dispersion. They provided an analytical framework to study and evaluate the security performance provided by multipath traffic dispersion. However, their scheme did not integrate the secret sharing or any other source coding scheme and their system model is different from ours. Zhou and Hass [2] used replication and threshold cryptography and built a highly secure and highly available key management service to deal with the denial of service attacks in a MANET. Tsirigos and Haas [16,17] provided an analytical evaluation

of a framework for multipath routing in mobile ad hoc networks. Their scheme applied a diversity coding at the source and used multipath routing to route the packets to the destination. Papadimitratos, Hass et al. [41] continued the work by developing an algorithm to select the multiple edge-disjoint paths in mobile ad hoc networks. To the best of our knowledge, their works are the most relevant work to ours. However, the goal of their work is to cope with the frequent route failure and to improve the reliability. Although we apply similarly multipath routing approach, the detailed technologies used are different.

CHAPTER 2
SYSTEM MODEL

The fundamental idea of SPREAD comes from the following observation: a messenger who carries the full message from one place to another across hostile ground may reveal the message easier if he/she is captured, while the message will not be fully recovered if multiple messengers are deployed to carry only partial information and go through different routes across the hostile ground. Based on this observation, the proposed SPREAD scheme works as follows: if a source node wants to send a message to a destination node securely, the source can use a multipath routing algorithm to find multiple paths from the source to destination with certain properties (for example, node disjoint paths in certain sense), then, depending on the required message security level and the availability of the multiple paths, the source determines a scheme, say threshold secret sharing scheme, to divide the message into multiple pieces and routes them to the destination through the selected multiple paths. The destination, upon receiving a certain number of correct shares, recovers the original secure message.

The fundamental idea behind the SPREAD scheme is to enhance the security by distribute the secret in the network. It is intuitive that by spreading the information pieces over multiple paths, it would make it more difficult for an adversary to intercept and compromise the message. Several issues need to be addressed for SPREAD scheme in order to maximize the security. First, how do we divide the secret message into multiple pieces? Secondly, how many paths do we need and how the message pieces should be allocated onto each selected path? Thirdly, how do we discover multiple paths in a

MANET and how do we optimize the path set used to deliver the message pieces? We briefly discuss these issues in this chapter and elaborate each of them in the following chapters.

2.1 Threshold Secret Sharing System

The first issue is how to divide the message into multiple pieces? Simply chopping the message into multiple segments involves the least processing overhead. However, it does not provide extra security protection. Since each segment contains partial content of the message, the partial content might be used to infer the content of the whole message. It is also difficult to protect the integrity of the message. In our SPREAD scheme, we use the threshold secret sharing algorithm to divide the secret message into multiple pieces. Threshold secret sharing algorithms could divide a secret into N pieces, called *shares* or *shadows*. Each of N participants of the system holds one share of the secret respectively. Any less than T participants cannot learn anything about the system secret, while with an effective algorithm, any T out of N participants can reconstruct the system secret. This is called a (T,N) threshold secret sharing scheme [14,42,43]. Thus with a (T,N) secret sharing algorithm, the secret message can be divided into N message shares such that in order to compromise the message, the enemy has to compromise at least T shares. With less than the threshold, T, shares, the enemy could learn nothing about the message and he has no better chance to recover the secret than an outsider who knows nothing at all about the message. This gives us the desired security properties. Another reason that we use secret sharing is that the generation of the message shares and the reconstruction of the message are all linear operations over a finite field (Shamir's Lagrange interpolating polynomial scheme [42]). The computational overhead is trivial. In addition, the secret sharing scheme can be designed with cheating detection and cheater identification [44]. It

is possible that after compromising a node, the adversary tries to cheat our system by sending us the faked or altered message shares. By embedding the cheater detection and identification, we can deterministically detect cheating and identify the cheater, no matter how many cheating shares are involved in the secret reconstruction. This would be a very useful detective mechanism in an unreliable ad hoc network environment and it also helps to protect the integrity of the message transmitted.

2.2 Share Allocation

The second issue is how to select the paths, how to choose an appropriate value of (T,N), and how to allocate the shares onto each selected path such that the maximum security can be achieved? We consider the case that a message is compromised due to compromised nodes. We assume that if a node is compromised, all the credentials of that node are compromised. So the message shares traveling through that node are all compromised. Given the available independent paths and their corresponding security characteristics, the fundamental principle is that, to maximize the security, the shares should be allocated in such a way that the adversary has to compromise all the paths to compromise the message. The simplest and most intuitive share allocation scheme is to choose N as the number of available paths, apply (N,N) secret sharing, and allocate one share onto each path. This will achieve the desired maximum security with least processing cost. However, in an ad hoc network, wireless links are instable and the topology changes frequently. Sometimes packets might be dropped due to the bad wireless channel condition, the collision at MAC layer transmission, or stale routing information. In the case that packet loss does occur, this type of non-redundant share allocation will disable the reconstruction of the message at the intended destination. To deal with this problem, it is usually necessary to introduce some redundancy (e.g., $T<N$)

in the SPREAD scheme to improve the reliability (i.e., the destination would have better chance to receive enough shares for reconstructing the message). Generally speaking, the security and the reliability are two contradictive design goals - more redundancy implies better reliability but worse security. However, due to the salient feature of the threshold secret sharing, we develop the redundant SPREAD share allocation which could tolerate certain packet loss while at the same time maintain the maximum security (i.e., forcing the adversary to compromise all the paths to compromise the message). We formulate the share allocation into a constrained optimization problem, with the objective to minimize the message compromise probability. Our investigation to the optimal share allocation reveals that, by choosing an appropriate (*T,N*) value and allocating the shares onto each path carefully, we could improve the reliability by tolerating certain packet loss without sacrificing the security. The maximum redundancy we can add to the SPREAD scheme without sacrificing security is identified. The optimal share allocation is proposed. More details about share allocation are presented in Chapter 4.

2.3 Multipath Routing and Path Set Optimization

The third issue is the multipath routing in ad hoc networks – how to find/optimize the desired multiple paths in a mobile ad hoc network and how to deliver the shares to the destination using these paths? Routing in a mobile ad hoc network presents great challenge because the nodes are capable of moving and the network topology can change continuously, dramatically and unpredictably. A great effort has been made in designing ad hoc routing protocols in response to the frequent topological changes. Multipath routing technique is a promising choice since the use of multiple paths in a MANET could diminish the effect of unreliable wireless links and the frequent topological changes. Several multipath routing schemes have been proposed to improve the

reliability, fault-tolerance, end-to-end delay for bursty traffic, as well as to achieve load balancing etc. [16,17,37-39].

For our SPREAD scheme, we need independent paths, more specifically, node disjoint paths, because we are dealing with node compromising problem. Several multipath routing protocols have been proposed in MANET with the design goal to find node-disjoint paths, such as the split multipath routing [37], the diversity injection technique [38], and the on-demand multipath routing [39]. The dynamic source routing (DSR) protocol [20] itself is also capable of maintaining multiple paths from the source to a destination. Those proposed protocols are all on-demand, due to the network bandwidth limitation, and source routing type, as the source routing provides the source with the maximal capability of controlling the disjointness of the paths. Those on-demand protocols work by broadcasting the route inquiry messages throughout the network and then gathering the replies from the destination. Although those routing protocols are able to find multiple node-disjoint paths, the paths found directly by them might not be optimal for our SPREAD scheme as the path selection is usually based on the hop count or propagation delay, not necessary the security. For our SPREAD scheme, we take a similar on-demand and source routing type of approach. However, we make use of the "link cache" organization we proposed in [45] where each path returned to source is decomposed into individual links and represented in a unified graph data structure. Using such a link cache organization allows us to further optimize the path set used for SPREAD. Although we rely on an underlying routing protocol to provide us with a partial view of network topology, the optimization of the path set can be done independent of the routing protocols used, based on the discovered partial network

topology. We study the effects of the proposed “link cache” on the performance of ad hoc on-demand routing protocols in Chapter 5. Then in Chapter 6, we discuss the multipath routing and path set optimization. We propose a security related link cost function such that the path can be found according to their security level (i.e., the probability that the path might be compromised). Then we propose a maximal node disjoint paths finding algorithm to find as many as possible paths and at the same time as secure as possible.

CHAPTER 3
THRESHOLD SECRET SHARING SYSTEM

Since the secret sharing is the fundamental of our scheme, in this chapter, we give a brief introduction to the threshold secret sharing system, which is used to generate the shares from a message (messages). Details can be found in [14]. We also present how the secret sharing scheme is applied to the secret message in our SPREAD scheme.

Secret sharing principle has been well developed as a means for increasing the confidence in the proper functioning of the information based systems. Secret sharing scheme is popularly used in secret key management [14,46]. Suppose that we have a system secret K and we divide it into N pieces, S_1, S_2, ..., S_N, called *shares* or *shadows*. Each of N participants of the system, P_1, P_2, ..., P_N, hold one share of the secret respectively. Any less than T participants cannot learn anything about the system secret, while with an effective algorithm, any T out of N participants can reconstruct the system secret K. This is called a *(T,N)* threshold secret sharing scheme [14]. The simplest example of this principle is the well-known two-man control rule that the Unite States enforces for critical military actions. Each of the two men knows a private piece of information, only when combined with the piece known to the other man does it suffice to allow access to a weapons system. However, each piece of the information individually provides its holder no greater chance of access or ability to use the weapon than what an outsider who knows nothing at all about the secret controlling information would be.

The first secret sharing scheme (called threshold scheme) was invented independently by Shamir [42] and Blakley [43] for the same application: robust key management for cryptosystems. Shamir's construction is algebraic in nature - based on interpolation of a polynomial defined over a finite field, *GF(q),* while Blakley's construction is geometric. Although their approaches to solving the problem were quite different, the essential notion is the same in both cases. The secret sharing scheme in SPREAD approach is derived from Shamir's LaGrange interpolating polynomial scheme. So we describe Shamir's approach in more detail here. Other secret sharing schemes can be easily incorporated into SPREAD.

3.1 Shamir's Secret Sharing Scheme

A secret sharing scheme consists of two algorithms. The first is called the *dealer*, which generates and distributes shares among the participants. The second is called the *combiner*, which collects shares from the participants and recomputes the secret. It produces the secret *K* from any *T* correct shares. The combiner fails to recompute the secret if the number of the correct shares is less than *T*.

Shamir's construction for (*T,N*) secret sharing scheme is algebraic and is based on the polynomial interpolation. Assume K is the secret to be shared among N participants, P_1, P_2, ..., P_N. The dealer obtains the *i*th participant P_i's share S_i by evaluating a polynomial of degree (*T*-1)

$$f(x) = (K + a_1 x + ... + a_{T-1} x^{T-1}) \bmod p$$

at x=i (i=1,2,…,N):

$$P_i \rightarrow S_i = f(i)$$

Here $a_1, a_2, \ldots, a_{T-1}$ are all randomly chosen coefficients, p is a randomly chosen prime number which is greater than any of the coefficients and must be made available to both dealer and combiner [46].

At the combiner, with the knowledge of a minimum number of T shares, $f(i_1)$, $f(i_2), \ldots, f(i_T)$, the original polynomial $f(x)$ can be recovered by Lagrange interpolation.

$$f(x) = \sum_{j=1}^{T} S_{i_j} \cdot l_{i_j}(x) \bmod p \qquad \text{where} \qquad l_{i_j}(x) = \prod_{k=1, k\neq j}^{T} \frac{x - i_k}{i_j - i_k}$$

Particually, the original secret K can be recovered by calculating $f(0)$.

Efficient ($O(n \log^2 n)$) algorithms for polynomial evaluation and interpolation have been discussed [47]. Even the straightforward quadratic algorithms are fast enough for practical key management systems.

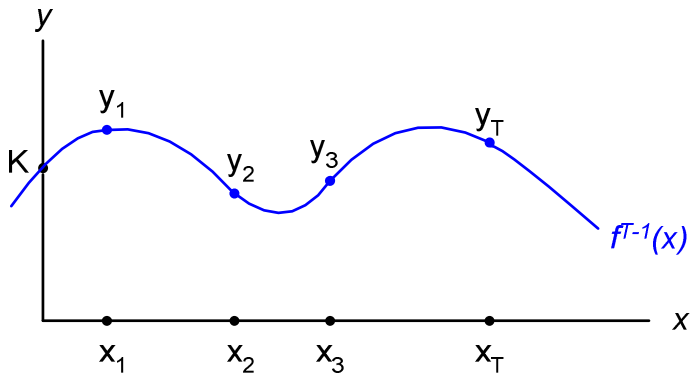


Figure 3-1. Fundamental idea of Shamir's secret sharing scheme.

Figure 3-1 illustrates the basic idea of Shamir's secret sharing scheme. Given T points in the two-dimensional plane, (x_i, y_i) $i=1,2,\ldots,T$, there is a unique $(T-1)$st degree

polynomial $f(x)$, for which $f(x_i)=y_i$ for all i. Given any subset of T points on $f(x)$, it is computationally easy using Lagrange interpolation to determine the polynomial and to solve for K, that is, to recover the secret. However, given only $T' \leq T-1$ points on $f(x)$, K could equally likely be any element in $GF(p)$ since for each choice of a point y' on the y-axis, there will be equally likely many $(T-1)$st degree polynomials through the subset of T' points and y'. Consequently, any collusion of fewer than T of the shares will have no better chance of determining the secret than an outsider who has no information at all. In this sense, Shamir's polynomial interpolation secret sharing scheme is *perfect*.

3.2 Secret Sharing in SPREAD

Consider a source node intends to send data to a destination node securely over the distributed insecure networks. In this application, the dealer process is implemented at the source node while the combining is done at the destination node.

To save network bandwidth, we make an improvement on Shamir's scheme by assigning secrets to all coefficients instead of one coefficient commonly used in key management. So the coefficients $a_0(K)$, $a_1, a_2, \ldots, a_{T-1}$ are all secrets here as shown in Figure 3-2. If a message is too large, it can be chopped up as we normally do in the transport layer. During this process, some scrambling may be helpful. Limited by the size of the chosen prime number p, the secret sharing is applied on a block by block basis, which is similar to any block cipher used to encrypting a large message. A group of T blocks, which correspond to the T secrets $a_0, a_1, a_2, \ldots, a_{T-1}$, are sent to the T inputs of the dealer process at one time. The output of the dealer process is a group of N blocks, where the ith output corresponds to the value of $f(i)$. Each block can form a single IP packet or a number of blocks can be concatenated to form a longer IP packet. Identifying indices need to be added to the packets so that they can be identified at the receiver end.

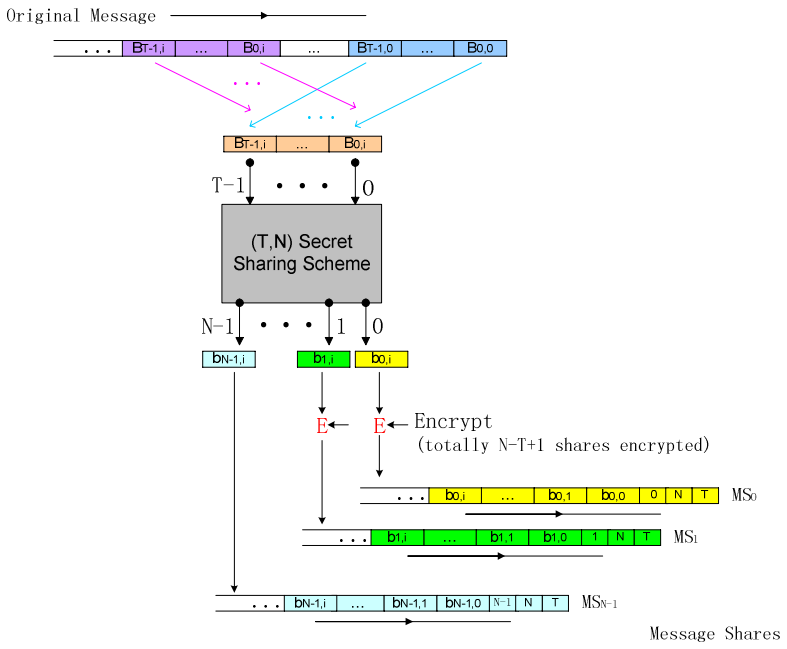


Figure 3-2. (*T*,*N*) secret sharing system in SPREAD.

At the network layer, the multipath routing protocol will allocate the *N* message shares (packets) in one group onto different paths to the destination according to the share allocation scheme. To maximize the security, the path assignment could be scrambled. Source routing can be used to route the packets via the specified paths across the network.

A buffer at the destination node is needed to temporally store the received packets. Here we assume the capacity of the buffer is not an issue so that re-sequencing of the packets could be done (Further implementation issues such as synchronization will be investigated in the future). For each group of blocks, as long as *T* blocks of them have been received, the combiner can reconstruct the original blocks by Lagrange interpolation as we described before. It can be also illustrated as solving a set of linear equations over a finite field. For example, let the received *T* blocks be $f(i_1)$, $f(i_2)\ldots, f(i_T)$, let

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{T-1} \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ i_1 & i_2 & \cdots & i_T \\ \vdots & \vdots & & \vdots \\ i_1^{T-1} & i_2^{T-1} & \cdots & i_T^{T-1} \end{bmatrix}$$

$$F = \begin{bmatrix} f(i_1) & f(i_2) & \cdots & f(i_T) \end{bmatrix}$$

then, the original blocks, $a_0, a_1, a_2, \ldots, a_{T-1}$, can be determined by solving a set of linear equations in matrix form

$$AB = F$$

This equation has a unique solution over the finite field *GF(p)*. Therefore, the reconstruction always succeeds if the combiner has at least *T* different shares, but fails if the number of shares is less than *T* [46]. The reconstructed blocks will be concatenated and passed to the higher layer.

3.3 Cheating Detection and Cheater Identification

Another way to improve the security is through the careful design of the secret sharing algorithm. It is possible for an attacker, after compromising an intermediate router, to further cheat our system by sending the faked message share. Secret sharing algorithm can be designed so that if some of the shares are lost or stolen – invalidated, the remaining shares can be modified in a way that the invalid shares cannot be used. Secret sharing schemes that tolerate the loss of a specified number of shares are said to have disenrollment capability [48]. Cheating can be prevented with modifications to such scheme. Another variation of secret sharing scheme is that, by embedding the cheater detection and identification mechanism, SPREAD scheme is also helpful in identify

malicious or compromised nodes. For example, one cheater detection and cheater identification scheme works as follows [44]:

At the dealer (source):

Choose a one-way function h() and a prime number q such that h() <q; Computer

$$T = \sum_{i=1}^{N} h(s_i) p^{2(i-1)} + \sum_{i=1}^{N-1} cp^{2i-1}$$

where c is a positive constant randomly chosen over GF(p); Publish T and p.

Then at the combiner (destination):

Upon receiving all the shares S_1^*, S_2^*, ..., S_N^*, compute

$$T^* = \sum_{i=1}^{N} h(s_i^*) p^{2(i-1)}$$

For each S_i^*, check

$$\left\lfloor \frac{T - T^*}{p^{2(i-1)}} \right\rfloor (\mathrm{mod}\, p) == 0$$

If the equation holds, S_i^* is correct; otherwise, it is a cheating share.

This scheme can deterministically detect cheating and identify the cheater, no matter how many cheating shares are involved in the secret reconstruction. It can be inferred that one or more nodes along the path where the cheating share coming from are compromised.

CHAPTER 4
SHARE ALLOCATION SCHEMES

How to choose the appropriate values of (*T,N*) and how to allocate the *N* shares onto each selected paths is another important issue in SPREAD design. Provided the available paths and their corresponding security characteristics, the first objective is to maximize the message security. In this section, we discuss the share allocation schemes which exploit the redundant secret sharing scheme (where $T<N$) to achieve both data confidentially and reliability.

4.1 Problem Formulation and Notations

Assume that (*T,N*) secret sharing algorithm is applied to the message to be protected at source node. In the network layer, we assume that there are totally *M* node disjoint paths, *path 1, path 2, …, path M*, available from the source to the destination. We use vector $\underline{p} = [p_1, p_2, \cdots, p_M]$ to denote the security characteristics of the paths, where p_i $(i = 1,2,\cdots,M)$ is the probability that path *i* might be compromised. Without loss of generality, we further assume $p_1 \le p_2 \le \cdots \le p_M$, which means that the paths are ordered from more secure one to less secure one. Note that the path security information, $\underline{p}$, is available at source from the multipath routing protocols we will discuss in the following two chapters. We assume that if one node were compromised, all the shares traveling through that node would be compromised. Therefore, we define that a path is compromised as when any one or more of the nodes along the path is compromised. For each path, we consider that if it were compromised, all the shares allocated to it would be

compromised. Otherwise, if the path were not compromised, all shares on that path would be safe. As those paths are node disjoint, we further assume that the probability that one path is compromised is independent of others. We should point out that the SPREAD scheme only enhances the data confidentiality statistically when the data are transmitted across the network. Thus the probability p_i does not include the probability that the source or the destination node is compromised, e.g we assume source and destination are trustworthy.

A share allocation scheme is used to allocate the *N* shares onto the *M* available paths. Denote the share allocation as $\underline{n} = [n_1, n_2, \cdots, n_M]$, where n_i is the number of shares allocated to path *i*, n_i is an integer, $n_i \geq 0$, $\sum_{i=1}^{M} n_i = N$. According to the secret sharing algorithm, the probability that the message is compromised equals to the probability that *T* or more shares are compromised. We denote the probability that the message is compromised in terms of the share allocation $\underline{n}$ as $P_{msg}(\underline{n})$. Then the share allocation can be formulated to a constrained optimization problem

minimize $P_{msg}(\underline{n})$

subject to $\sum_{i=1}^{M} n_i = N$, n_i is an integer, $n_i \geq 0$

4.2 Maximum Security without Redundancy

Let define $r = 1 - T/N$ as the redundancy factor of the (*T,N*) secret sharing scheme. A non-redundant SPREAD scheme is one where *r*=0 (e.g., *N*=*T*). It is easy to derive that given the number of available paths, *M*, and the corresponding path security characteristics $\underline{p} = [p_1, p_2, \cdots, p_M]$, the non-redundant (*N,N*) (*N*≥*M*) secret sharing

scheme would give the maximum security (e.g., minimum message compromise probability), when at least one share and at most *T-1* shares are allocated to each of the available paths, i.e.

$$\begin{cases} 1 \le n_i \le T-1, & \text{i} = 1,...,\text{m} \\ \sum_{i=1}^{m} n_i = N \end{cases}$$

This share allocation forces the adversary to compromise all the paths to compromise the message. This probability equals to the probability that all the paths are compromised.

$$P_{msg}(\underline{n}) = \prod_{i=1}^{M} p_i$$

It is noted that the maximum security provided only depends on the paths chosen. As p_i is a probability satisfying $0 \le p_i \le 1$, the more paths we use to distribute the shares, the less the probability is, and the more secure the message delivered. Thus, if given required security level (in terms of message interception ratio) γ_{P_n}, the SPREAD scheme should chose the first *m* paths, path *1*, path *2*, …, path *m*, which satisfying $P_{msg}(\underline{n}) = \prod_{i=1}^{m} p_i \le \gamma_{P_n}$, to deliver the message.

4.3 Maximum Security with Redundancy

It is intuitive that non-redundant secret sharing scheme provides the maximum security to the message. However, it requires the successful reception of all the shares in order to reconstruct the original message. In an ad hoc network, wireless links are not stable. Packets might be dropped due to broken links, heavy MAC layer collision, or wireless signal fading, etc. With our SPREAD scheme, as the shares are spread onto

multiple paths, long paths might be used. The reliability of the message, in terms of transmission error, packet lost ratio, etc., may be further degraded. It is usually necessary to add some redundancy for reliability.

Redundancy is a common way to improve the reliability. It is based on the idea of sending more information than minimum requirement, so that the original message can be reconstructed in the event of loss in the network. It may be used independently of the multipath routing, by adding Forward Error Correction (FEC) code to each individual share. We denote this type of redundancy as serial redundancy. Serial redundancy is good for correcting noise-like random errors introduced to the bit stream, while helpless for the persistent errors or link failure. With a (*T,N*) secret sharing scheme, when $T<N$, we actually introduce redundancy from another dimension, the parallel redundancy. When this type of redundancy is used in combination with multipath routing, the system becomes more error tolerant, because a certain number (*N-T*) of message shares can be corrupted or lost without affecting the reconstruction of the original message. The system also becomes more fault-tolerant because a certain fraction of the paths can be affected by failure without interrupting the flow of the information.

Using the same path chosen criteria (i.e., choosing the first *m* most secure paths which satisfy the required security level), it is intuitive to show that, in order to achieve the maximum security, the total number of shares allocated to any *m-1* or less paths should be less than *T.* Again, this share allocation forces the adversary to compromise all the *m* paths to compromise the message. This is also a necessary and sufficient condition to achieve the maximum security. This condition can be simplified as

$$\begin{cases} N - n_i < T, & \forall i \in (1,2,\cdots,m) \\ n_1 + n_2 + \cdots + n_m = N \end{cases}$$

Recall $r = 1 - T/N$ is the redundancy factor of the secret sharing scheme. Then we could derive a necessary condition for achieving the maximum security, i.e.

$$r < 1/m \quad (m \geq 2)$$

This is an important condition as it defines the maximum redundancy we can add to the SPREAD scheme without sacrificing the security. It indicates that to maintain the maximum security achievable from the chosen path set, the maximum redundancy we can add to the secret sharing algorithm is bounded by $r < 1/m$, where m is the number of chosen paths $(m \geq 2)$. In other words, we could claim that for a r-redundancy SPREAD scheme, the maximum security can be achieved only if the redundancy factor r satisfies $r < 1/m \quad (m \geq 2)$. Then by choosing an appropriate (T,N) value which satisfies

$$T \geq N\frac{m-1}{m} + 1 \qquad (m \geq 2)$$

an optimal share allocation can be designed such that the maximum security can be achieved while at the same time certain (r) redundancy can be provided. Any allocation that conforms to the constraints

$$\begin{cases} N - T + 1 \leq n_i \leq T - 1, & \text{i} = 1,...,\text{m} \\ \sum_{i=1}^{M} n_i = N \end{cases}$$

is an optimal share allocation in terms of security. An optimal share allocation will force the adversary to compromise all the paths to compromise the message, while at the same time, it can tolerate a certain number (N-T) of share lost during the transmission. The optimal share allocation is not unique. Other optimization objectives, such as the

minimal delivery cost, balanced bandwidth usage, or maximum reliability, might be set to further optimize the share allocation for other purposes.

4.4 General Case : Calculation of $P_{msg}(\underline{n})$

In this subsection we discuss the optimal share allocation for arbitrary (*T,N*) values and derive the generalized calculation of $P_{msg}(\underline{n})$. We consider the possible network security states in terms of each path state. Let $\underline{s} = [s_1, s_2, \cdots, s_M]$ denote a network security state, where s_i is the state of path *i*. $s_i = 0$ indicates that path *i* is not compromised while $s_i = 1$ indicates that path *i* is compromised. We assume that s_i follows a Bernoulli distribution: s_i=0 with probability $(1 - p_i)$, and s_i=1 with probability p_i. The probability that the network is in a particular state $\underline{s}$ can be calculated as

$$P_{state}(\underline{s}) = \prod_{i=1}^{M} p_i^{\,s_i} (1 - p_i)^{1 - s_i} \tag{4.1}$$

Since we assume that if a path *i* is compromised, all the n_i shares allocated to path *i* would be compromised. Otherwise, if path *i* is not compromised, no share is lost. Thus the total number of shares compromised when the network is in state $\underline{s}$ can be calculated as

$$N_{lost}(\underline{s}, \underline{n}) = \underline{n} \bullet \underline{s} = \sum_{i=1}^{M} n_i s_i$$

We create a set S

$$S = \{\underline{s} : N_{lost}(\underline{s}, \underline{n}) \geq T\}$$

which includes all the possible network security states $\underline{s}$ that satisfy the constraint $N_{lost}(\underline{s},\underline{n}) \geq T$. Then the probability of the message to be compromised can be calculated as

$$P_{msg}(\underline{n}) = \sum_{\underline{s} \in S} P_{state}(\underline{s})$$

It is impossible to derive a closed form solution of the above formula, mainly because the selection of $\underline{s}$'s is dependent on $\underline{n}$. However, in reality, the number of available paths *M* would not be large. The values of *N* and *T* are also not necessary to be large. Given $\underline{p}$, for a fixed arbitrary value *N* and *T*, the search of optimal share allocation (or sub-optimal share allocation which aims to minimize $P_{msg}(\underline{n})$ when optimal share allocation does not exist) based on $P_{msg}(\underline{n})$ can be carried out by exhausted search. In the following subsection, we describe a straightforward algorithm to find the optimal share allocation.

4.5 Share Allocation Algorithm

Given secret sharing parameters *T* and *N*, the available number of paths *M*, and their corresponding security characteristics $\underline{p} = [p_1, p_2, \cdots, p_M]$, the following algorithm is designed to find the optimal or sub-optimal share allocations efficiently. The algorithm can find all the optimal share allocations if they exist. If the optimal share allocation does not exist, it will attempt to find near optimal share allocations with low probability $P_{msg}(\underline{n})$.

- ***Step 1:*** Create set S, which includes all the possible network security state vectors $\underline{s} = [s_1, s_2, \cdots, s_M]$ except [0,0,…,0] and [1,1,…,1]. There should be totally $2^M - 2$ elements in set S. Calculate $P_{state}(\underline{s})$ for each element $\underline{s}$ according to Equation 4.1. Create set S', which includes [1,1,…,1] only initially.

- ***Step 2:*** Create set A, which include all the possible share allocation vectors $\underline{n} = [n_1, n_2, \cdots, n_M]$. The following constraints are applied to reduce the size of A.

$$\begin{cases} N \geq n_1 \geq n_2 \geq \cdots \geq n_M \geq 0 \\ \sum_{i=1}^{M} n_i = N \end{cases}$$

- ***Step 3:*** while ($S \neq \phi$), do the following
  - choose $\underline{s}$, $\underline{s} \in S$ and $P_{state}(\underline{s})$ is the maximum
  - for each $\underline{n} \in A$, mark $\underline{n}$ if $\underline{n} \bullet \underline{s} \geq T$
  - if (all the elements in A have been marked)
    {unmark all the elements in A;
    add $\underline{s}$ to S' ; delete $\underline{s}$ in S ; }

  else { delete all marked elements in A ;
  delete $\underline{s}$ in S ; }
- ***Step 4:*** All the remaining elements in A are optimal share allocations if [1,1,…,1] is the only element in set S'; or they are sub-optimal share allocations if more elements present in set S'; the probability $P_{msg}(\underline{n})$ for all the remaining $\underline{n}$'s is $P_{msg}(\underline{n}) = \sum_{\underline{s} \in S'} P_{state}(\underline{s})$.

An example of how the algorithm works is illustrated in figure 4-1. It is noted that when (*T,N*) is (9,10) and (8,10), *r*=0.9 and 0.8. The necessary and sufficient conditions are met so the optimal share allocations exist where maximum security can be achieved. For (7,10) SPREAD, although the necessary condition satisfies, the sufficient condition does not. The optimal share allocation to achieve the highest security does not exist. For (6,10) and (5,10) SPREAD, *r*<2/3, the optimal share allocation does not exist. Our algorithm finds a set of sub-optimal share allocations which maximize the achievable security.

4.6 Summary

In this chapter, we provide an analytical framework to study the SPREAD performance. The effect of share allocations on the security and reliability is discussed and optimal share allocation scheme is proposed. In addition, we pointed out that the

redundant SPREAD scheme is not only secure but also highly error-tolerant and fault-tolerant because of the parallel redundancy introduced. The redundant SPREAD could maintain the maximum security while at the same time provide a certain degree of redundancy for reliability purpose. The upper bound of the redundancy that could be added without sacrificing the security is identified. The necessary and sufficient condition for the optimal share allocation existing is identified. A share allocation algorithm is proposed to find the optimal share allocation when it exists or sub-optimal share allocation when the optimal one does not exist.

Initial set A

10 0 0	6 4 0
9 1 0	6 3 1
8 2 0	6 2 2
8 1 1	5 5 0
7 3 0	5 4 1
7 2 1	5 3 2
	4 4 2
	4 3 3

Initial set S

$\underline{s}$	$P_{state}(\underline{s})$ ($^0/_{00}$)
001	224
010	144
011	96
100	84
101	56
110	36

Final Results

(T,N)	*(9,10)*	*(8,10)*	*(7,10)*	*(6,10)*	*(5,10)*
set A	[6 2 2] [5 3 2] [4 4 2] [4 3 3]	[4,3,3]	[6 4 0] [5 5 0] [5 4 1] [4 4 2]	[5 5 0]	[10 0 0] [9 1 0] [8 2 0] [8 1 1] [7 3 0] [7 2 1] [6 4 0] [6 3 1] [6 2 2]
Set S'	111	111	111, 110	111, 110	111, 110, 101, 100
$P_{msg}(\underline{n})$	$p_1 \cdot p_2 \cdot p_3$ =0.024	$p_1 \cdot p_2 \cdot p_3$ =0.024	$p_1 \cdot p_2$ = 0.06	$p_1 \cdot p_2$ = 0.06	p_1=0.20

Figure 4-1. Share allocation scheme (assume M=3, $\underline{p}$=[p_1,p_2,p_3]=0.2, 0.3, 0.4).

Another way to improve the security is through the careful design of the secret sharing algorithm. It is possible for an attacker, after compromising an intermediate router, to further cheat our system by sending the faked message share. Secret sharing algorithm can be designed so that if some of the shares are lost or stolen – invalidated, the remaining shares can be modified in a way that the invalid shares cannot be used. Secret sharing schemes that tolerate the loss of a specified number of shares are said to have disenrollment capability. Cheating can be prevented with modifications to such scheme [44,48].

The above analysis is based on the idea to distribute the multiple shares across multiple paths one at a time, which is essentially a distribution of the secret in space domain. We could further distribute the secret in time domain by sending out the shares over a certain period of time and dynamically changing the paths during the transmission of different shares. Another improvement is to combine the SPREAD with the conventional data encryption. According to the features of the secret sharing algorithm, only partial of the message shares (N-T+1) need to be encrypted. Significant computation can be saved.

CHAPTER 5
ADAPTIVE ROUTE CACHING STRATEGY

The third design issue for the SPREAD is the multipath routing in ad hoc networks – how to find/optimize the desired multiple paths in a mobile ad hoc network and how to deliver the shares to the destination using these paths? Routing in a mobile ad hoc network presents great challenge because the nodes are capable of moving and the network topology can change continuously, dramatically and unpredictably. A great effort has been made in designing ad hoc routing protocols in response to the frequent topological changes. In this chapter, we discuss the routing in a MANET, particularly we study the performance of an popular ad hoc on-demand routing protocol, Dynamical Source Routing (DSR) [20]. We then propose an adaptive route caching strategy based on DSR to improve the routing efficiency in a MANET. The proposed link cache scheme is the basis of our multipath routing strategy for the SPREAD: it provides us a partial view of network topology so that the optimization of the path set can be done independent of the routing protocols. We will present the optimization of the path set, the maximal path finding algorithm, in the next chapter. These two chapters, combined together, address the third design issue of SPREAD, the multipath routing.

5.1 Routing in a Mobile Ad Hoc Network

An ad hoc network is an infrastructureless multi-hop mobile wireless network. In an ad hoc network, there is no fixed infrastructure such as base stations that function as routers in a wireless cellular network. Each node in an ad hoc network is capable of moving independently and functioning as a router that discovers and maintains routes and

forwards packets to other nodes. Since each node in an ad hoc network is capable of moving collectively or independently of other nodes, the network topology can be changed dramatically. Traditional Internet routing protocols are no longer effective in ad hoc network. It presents a great challenge for a routing protocol to keep up with the frequent and unpredictable topology changes.

Much work has been done since the mobile ad hoc networking (MANET) working group was formed within the Internet Engineering Task Force (IETF) to develop a routing framework for IP-based protocols in ad hoc networks. Existing ad hoc routing protocols can be generally categorized into two classes: table-driven (or proactive, such as DSDV[49] and WRP[50]) and demand-driven (or reactive, such as DSR[20] and AODV[51]) [52]. Similar to the routing protocols used in the wired network, table-driven routing protocols attempt to maintain consistent, up-to-date route information from each node to every other node, regardless of the need of such routes. They respond to topology changes by propagating updates throughout the network. An on-demand routing protocol differs in that it attempts to discover a route to a destination only when it is presented a packet for forwarding to that destination. Discovered routes are maintained by a route maintenance procedure until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. Most of the performance studies indicate that on-demand routing protocols perform better than table-driven routing protocols [53-55]. The major advantage of the on-demand routing comes from the reduction of the routing overhead, as high routing overhead usually has a significant performance impact in low bandwidth wireless networks.

However, on-demand routing also has its disadvantages. First of all, since a route is discovered on a need basis, the packet cannot be sent before such a route has been found. Since the source node has no idea about where the destination is, the protocol has to search the entire network to find the destination. This is a very costly operation. It also adds the latency to the packet delivered. Secondly, in order to avoid the need to rediscover each routing decision for each individual packet, any on-demand routing protocol must utilize some type of routing cache to store the routes previously discovered. However, due to the frequent topology changes and lacking of timely update mechanisms on a regular basis, the cache itself may contain out-of-date information, implying that links in the cache are actually no longer valid. This stale data represents a liability that may degrade performance rather than improve it [56]. There is a tradeoff in using the cached information: valid data may reduce route discovery cost which will contribute to the improved performance, while stale data will cause more errors, longer delay and higher packet loss.

In this chapter, we study the effects of different caching strategies on the performance of on-demand routing protocol in an ad hoc network. Two types of cache schemes, a *path cache* and a *link cache*, are investigated. A path cache is one in which each cache entry is a node list representing an entire path leading to a certain destination, while a link cache has a conventional graph data structure in which each individual link is referred to as a cached data unit. A link cache has the potential to utilize the obtained route information more efficiently. However, without a good link update mechanism, stale links may cause more route errors and consequently severe performance degradation. Marina and Das [57] studied a few techniques to improve cache correctness

in DSR. Their study was based on a path cache structure. Hu and Johnson [58] conducted a comprehensive study on the caching strategies in on-demand routing protocols for wireless ad hoc networks. However, their study on the timeout mechanisms is limited to a fixed level of node mobility. A static "optimal" lifetime was assigned to each node with this mobility level. Their adaptive timeout mechanism was also limited to fine tune-up of the lifetime within the same level of node mobility and does not adapt to various node mobility levels. In this chapter, we investigate two types of link update mechanisms, a static timeout mechanism and an adaptive timeout mechanism. The static timeout mechanism expires a link using a statically assigned lifetime, while in the adaptive timeout mechanism, we propose to adapt the link timeout interval to various node mobility levels based on the estimation from a moving average of real link lifetime statistics. The performance results of different cache schemes obtained from simulations will be presented.

Our simulation is based on the Dynamic Source Routing (DSR) protocol, since it is a well performed and entirely on-demand routing protocol. Previous studies have shown that a path cache DSR protocol has good performance in less "stressful" situations (i.e., smaller number of nodes and lower load and/or mobility) [53,59]. In this chapter, we will show that the performance of DSR under heavy traffic load situation could be much improved when incorporating a link cache scheme together with an adaptive timeout mechanism.

Although our simulations are based on DSR, the proposed adaptive link cache strategy does not tie to DSR protocol. Due to the on-demand nature, the on-demand routing protocols do not exchange routing information periodically. Even though some

protocols could detect the local link broken promptly with the use of *hello* messages, the bad news (e.g., broken link) propagates to other nodes very slowly. Timer-based stale route removal strategies are also used in other routing protocols, such as AODV, in an attempt to maintain the consistence of the route cache without consuming extra network bandwidth. We believe that our results could be generalized to other protocols that use similar caching schemes.

5.2 Route Caching Strategies

5.2.1 Overview of the DSR Protocol

DSR is an on-demand routing protocol that is based on the concept of source routing. The protocol is composed of two major mechanisms, *Route Discovery* and *Route Maintenance*, and three types of route control messages, *Route Request*, *Route Reply*, and *Route Error*. When a source node in the ad hoc network attempts to send a packet to a destination but it does not already have a route to that destination in its route cache, it initiates a route discovery process by broadcasting a route request packet. This route request packet contains the source node address, the destination node address, a unique sequence number, and an empty route record. Each intermediate node, upon receiving a route request for the first time, will check in its own route cache. If it has no route to the destination, the intermediate node will add its own address to the route record and rebroadcast the route request. If it has a route to the destination in its route cache, the intermediate node will append the cached route to the route record and initiates a route reply back to the source node. The route reply contains the complete route record from the source to the destination. The intermediate node ignores the latecomers of the same route request by examining the sequence number. If the node receiving the route request is the destination node, it will copy the route record contained in the route request and

send a route reply back to the source. In most simulation implementations, the destination node will reply to all the route requests received as DSR is capable of caching multiple paths to a certain destination and the replies from the destination most accurately reflect the up-to-date network topology. Figure 5-1 illustrates the operation of DSR protocol.

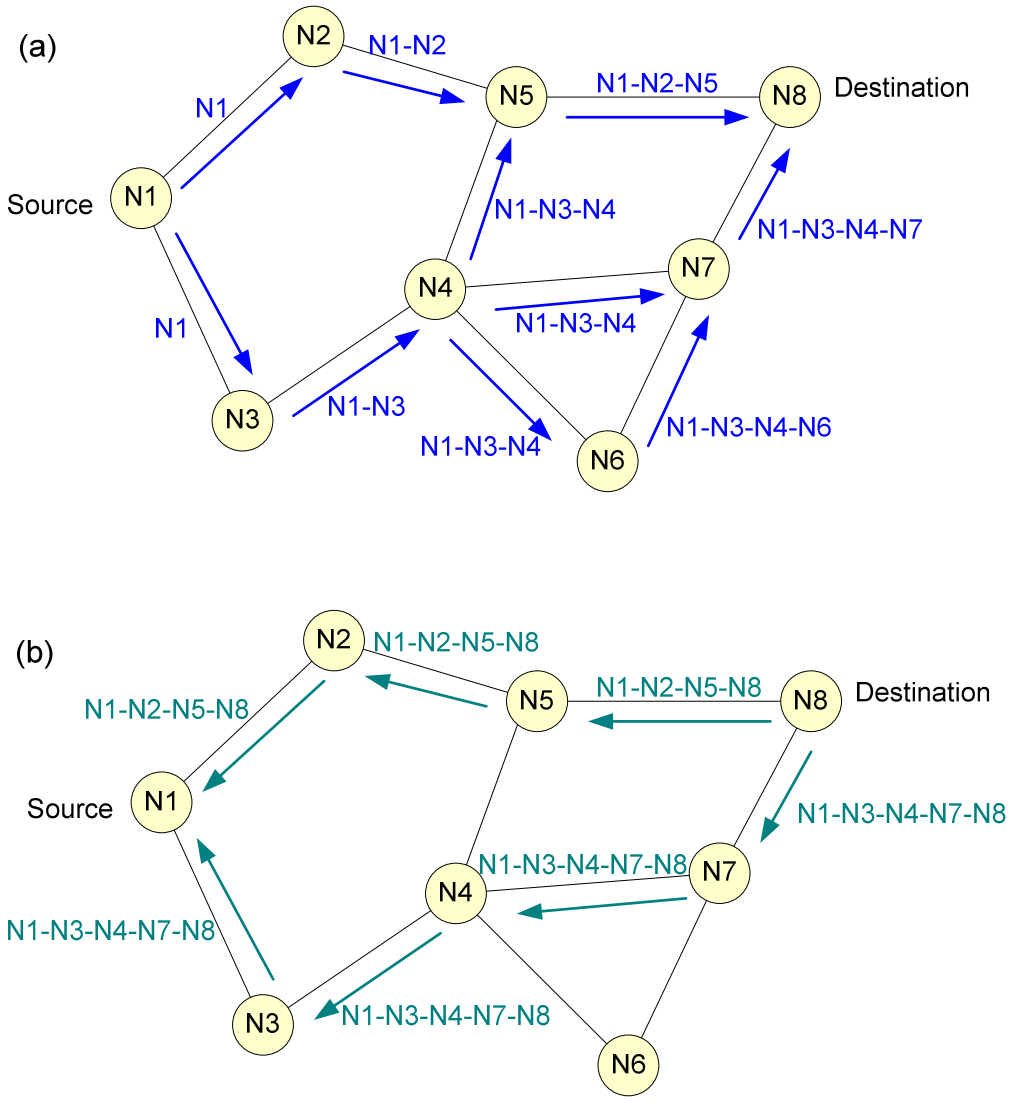


Figure 5-1. Operation of DSR protocol. (a) Route request. (b) Route reply. (c) Route error

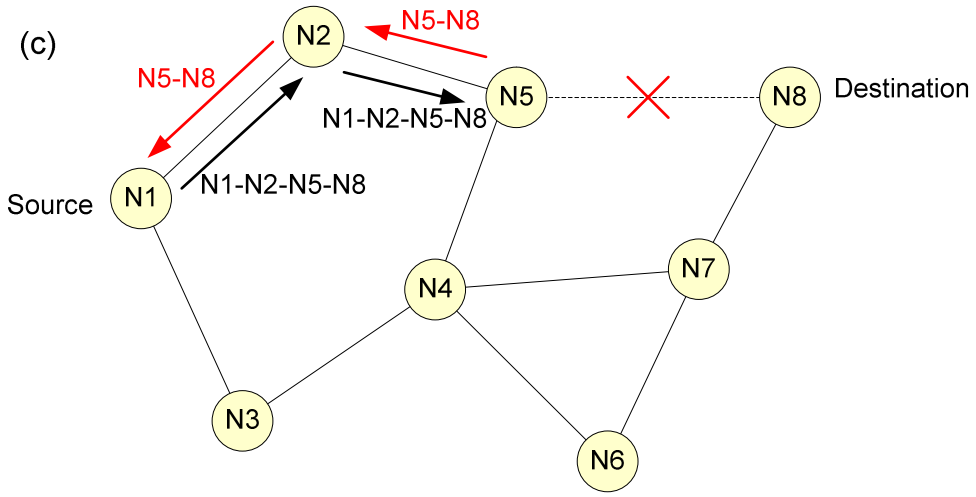


Figure 5-1. Continued

Due to the node movement, the routes discovered may no longer be valid over time. The route maintenance mechanism is accomplished by sending route error packets. When a link is found broken, a route error packet is sent back from the node that detects the link failure back to the source node. Each node, upon receiving the route error message, removes from its cache all the routes that contain the broken link [20]. Some ad hoc routing protocols, such as WRP [50] and AODV [51], use the periodic local broadcasts of the *hello* messages to ensure the local connectivity. Nodes learn of the existence of a neighbor from receiving or overhearing the packets transmitted by that node. When a node is not sending anything, it must send a *hello* message within a specified time period. Otherwise, failure to hear from a neighbor indicates a broken link between the two. DSR does not incorporate such local connectivity maintenance mechanism. In DSR, each node transmitting the packet is responsible for confirming that the packet has been received by the next hop along the source route. This can be done by either a link layer acknowledgement (as in IEEE 802.11), or a “passive

acknowledgement" (in which the first transmitting node confirms the receipt at the second node by overhearing the second node transmitting the packet to the third node), or a DSR-specific software acknowledgement returned by the next hop. Thus, once a route enters the cache, the failure of the route can only be detected when it is actually used to transmit a packet but fails to confirm the receipt by the next hop.

Besides the aforementioned basic functions, more optimization mechanisms are proposed and added to DSR protocol. These optimizations include gratuitous route replies, salvaging, gratuitous route errors, snooping, tapping, etc. [56]. Most of the optimizations are included in our simulation implementation.

5.2.2 Cache Organizations

In DSR, the route returned to the source is a complete path leading to the destination. By caching each of these paths separately, a "path cache" organization can be formed. A path cache is very simple to implement. When a route is needed, the path cache data structure can be efficiently searched for any path leading to that destination. This type of cache organization has been extensively studied through simulations [53,57,59]. In this paper, we consider an alternative type of organization, a "link cache", in which each node keeps a graph data structure representing this node's current view of the network topology, the route returned to this node is decomposed into individual links and represented in the graph data structure [58]. When a route is needed, a graph search algorithm, such as the Dijkstra shortest path algorithm or the breath-first-search (BFS) shortest path algorithm, is executed to find a path to the destination.

Compared with a path cache scheme, a link cache has the potential to utilize the route information more efficiently. Given the same amount of route reply information, the routes existing in a path cache can always be found in a link cache, while by

connecting individual links differently the link cache may have new or better routes that a path cache does not include. In a path cache, a complete route will be removed (or is truncated before the broken link, depending on the path representation in the cache) when one of its links breaks. While in a link cache, only the broken link is removed. The rest of the links on that route are still available to form new routes. Consider the situation illustrated in Figure 5-2, the link cache will produce a new and better path to E as A->C->E, which does not exist in the path cache. When link A->C is broken, there will be no route to destination D and F exist in a path cache. A route discovery process has to be initiated if a route to destination D or F is desired. While by removing only link A->C, a link cache still has route A->B->C->D or A->B->C->E->F to the destination D and F. A route discovery can be avoided. Therefore, with a link cache, potential reduction in the costly route discovery operation can be expected.

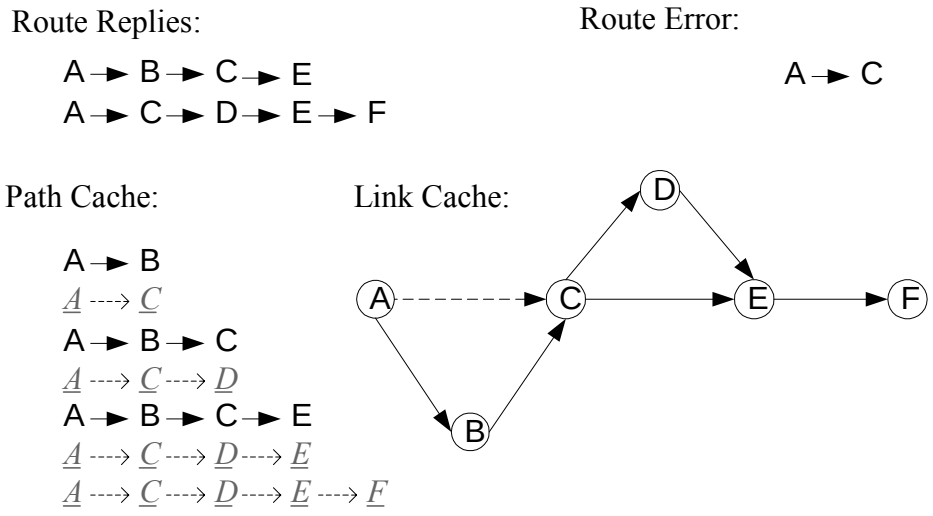


Figure 5-2. Path cache and link cache.

5.2.3 Link Timeout Mechanisms

As nodes in an ad hoc network are capable of moving independently, a link has a limited lifetime. Current existing link is no longer valid when the two end nodes moving out of the transmission range of each other. Due to the on-demand manner of the routing protocol, the link status will not be updated until they are used. However, using an actually broken route will cause a number of route errors to be generated and potential packet loss. So taking advantage of using active links individually has to be combined with a mechanism that removes stale links timely to avoid route errors. A natural choice is to combine a timeout policy to the link cache such that each link is given an appropriate lifetime when it enters the link cache, and is removed when its lifetime is running out. The lifetime estimation becomes a critical issue for such a link cache scheme. The lifetime assigned to the link should properly reflect the expected value of its real lifetime. If this value is assigned too small, links expire too soon before they really break, more costly route discoveries would have to be performed. On the other hand, if the value is too large, links break early before the timers expire, more route errors will be caused, which would also degrade the overall performance of the protocol.

In this paper, we first study the static lifetime assignment, in which when a link enters the link cache, it is assigned a predefined static lifetime *T1* seconds. During its lifetime, if the link is used to send packets, the lifetime of this link will be adjusted so that it won't expire in the future *T2* seconds. Then, based on the observations from the static lifetime experiments, we propose and study an adaptive lifetime estimation scheme that adaptively adjusts the link lifetime based on the moving average of the pervious collected lifetime statistics.

In our adaptive link lifetime scheme, each link (*i,j*) in the link cache is associated with three clock type attributes, *born*, *lastUsed*, and *liveTo*. Attribute born indicates the time when a new link enters the link cache. It is updated when a route reply is received and a new link is found in the route. Attribute *lastUsed* is the time stamp when the link is last used to forward a packet. Attribute *liveTo* is the predicted time at or after which the link expires. The statistical lifetime data are collected whenever a link is removed from the link cache. There are two situations that will cause a link to be removed from the link cache – a route error is received indicating that link is broken, or the *liveTo* attribute associated with that link expires. If a link is removed because of the reception of a route error, the lifetime *l* is calculated as

$$l = CurrentTime() - link[i, j].born$$

or if it is removed because of timeout, lifetime *l* is calculated only if this link has ever been used during its lifetime,

$$l = link[i, j].lastUsed - link[i, j].born$$

LIFETIME is the variable indicating the estimation of the link lifetime. It is initially assigned a static value and is adjusted dynamically using a moving average method whenever a lifetime datum *l* is collected,

$$LIFETIME = (1 - \alpha) * LIFETIME + \alpha * l$$

where α is the 1st order moving average parameter, which indicates the weight of the new datum in the total average. In our simulation, we set the initial *LIFETIME* as 50 seconds and α as 0.01.

The effects of different static lifetime values on the performance of routing protocol are investigated by simulations, which are reported in the next section.

5.3 Simulation and Results

5.3.1 Simulation Framework

The simulation of our link cache DSR protocol is implemented within the GloMoSim library [60,61]. The GloMoSim library is a scalable simulation environment for wireless network systems using the parallel discrete-event simulation language called PARSEC. The link layer model is the Distributed Coordination Function (DCF) of the IEEE 802.11 wireless LAN standard. The radio model uses the channel characteristics similar to Lucent's WaveLAN product. Radio propagation range for each node is 250 meters and channel capacity is 2 Mbits/sec. The network simulated consists of 50 nodes. The simulation area is 700×700 square meters. Each simulation is executed for 15 simulated minutes.

The random waypoint mobility model is used in the simulation [53]. In this model, a node selects a destination randomly within the simulated territory, moves to that destination at a speed uniformly distributed in [vmin, vmax] m/sec, stops there for a predefined pause time and then repeats this behavior for the duration of the simulation. In our simulations, [vmin, vmax] is fixed to [0,20]. The different node mobility levels are achieved by changing the values of pause time.

The traffic used in this simulation is CBR UDP sessions. The source-destination pairs are chosen randomly among the 50 nodes. The number of communication sessions is varied to change the offered load in the network. All the data packets are 64 bytes and are sent at a speed of 4 packets/sec. The reason that we choose 64 bytes small packet size is because our focus is on routing protocol's capability of tracking the topology change. Small packet size will factor out the effects of other reasons such as the network congestion [53].

We evaluate the performance of the link cache DSR protocol with static lifetime assignments and the proposed adaptive lifetime scheme. With static lifetime assignments, we execute multiple simulations for different (T1,T2) values as (3,1), (6,1), (12,2), (25,3), (50,5), (100,10), (900,-). For each (T1,T2) value, we execute multiple simulation runs with various traffic load conditions (10,20,30,40,50 sources) and various node mobility levels (pause time = 0s, 30s, 60s, 120s, 240s, 480s, 900s). For comparison purpose, we also evaluate the conventional path cache DSR protocol. The traffic load and the node mobility scenarios are identical across different variations of DSR protocol.

5.3.2 Effects on Routing Overhead

We first examine the effects of different cache schemes on the routing overhead generated. The routing overhead is calculated as the number of control packets transmitted by the protocol. It is counted as per hop basis. There are three types of routing packets, route requests, route replies, and route errors. Figure 5-3 plots the amount of each type of control packets generated when using different static lifetime values. The results confirm our expectation that small values of lifetime cause increased number of route requests but decreased number of route errors, while large values of lifetime cause decreased number of route requests but increased number of route errors. When the lifetime is within an appropriate range (in our experiments, 6<=T1<=50), the overall control messages are quite balanced. However, when the lifetime value becomes very large (in our experiments, T1>=50), the dramatically increased route errors would overwhelm the slightly decreased route requests. The total number of control messages increases significantly. Figure 5-4 gives the comparison of the overall routing overhead generated by each variation of the protocol. We observe that, in general, with an appropriate static link lifetime, the number of control packets used by the link cache

scheme is less than that used by the path cache scheme. With the increased network traffic load, the reduction becomes more obvious. However, without an appropriate timeout mechanism, the routing overheads are quite high. Among them, the majority is route error messages caused by the use of stale routes. We also observe that the proposed adaptive cache scheme tracks the "optimal" link lifetime quite well. Although it is not always optimal, it keeps the routing overhead "sub-optimally" low under various mobility level conditions.

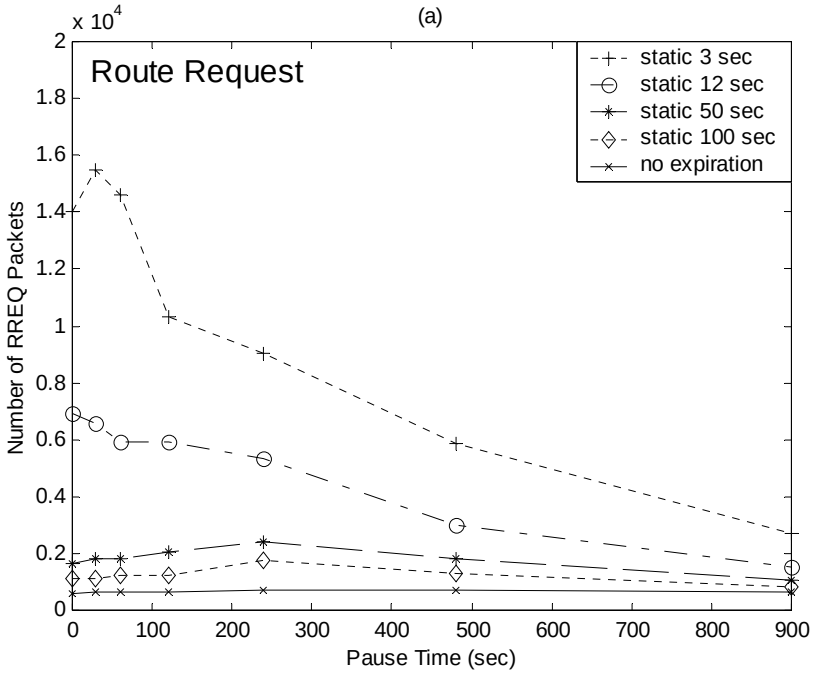


Figure 5-3. Separated routing overhead (50 sources). (a) Route request. (b) Route reply. (c) Route error.

The above observations indicate that a link cache scheme, particularly the link cache scheme with adaptive lifetime estimation, could make use of the available route information more efficiently. When the network traffic load is high, more route replies and more promiscuously overheard route information can be collected. The link cache can maintain more accurate and more up-to-date information of which the link cache

scheme can take advantage to reduce the route discovery processes. The reduced routing overhead, especially in heavy traffic situation, contributes to the improved performance, as discussed in the following subsections.

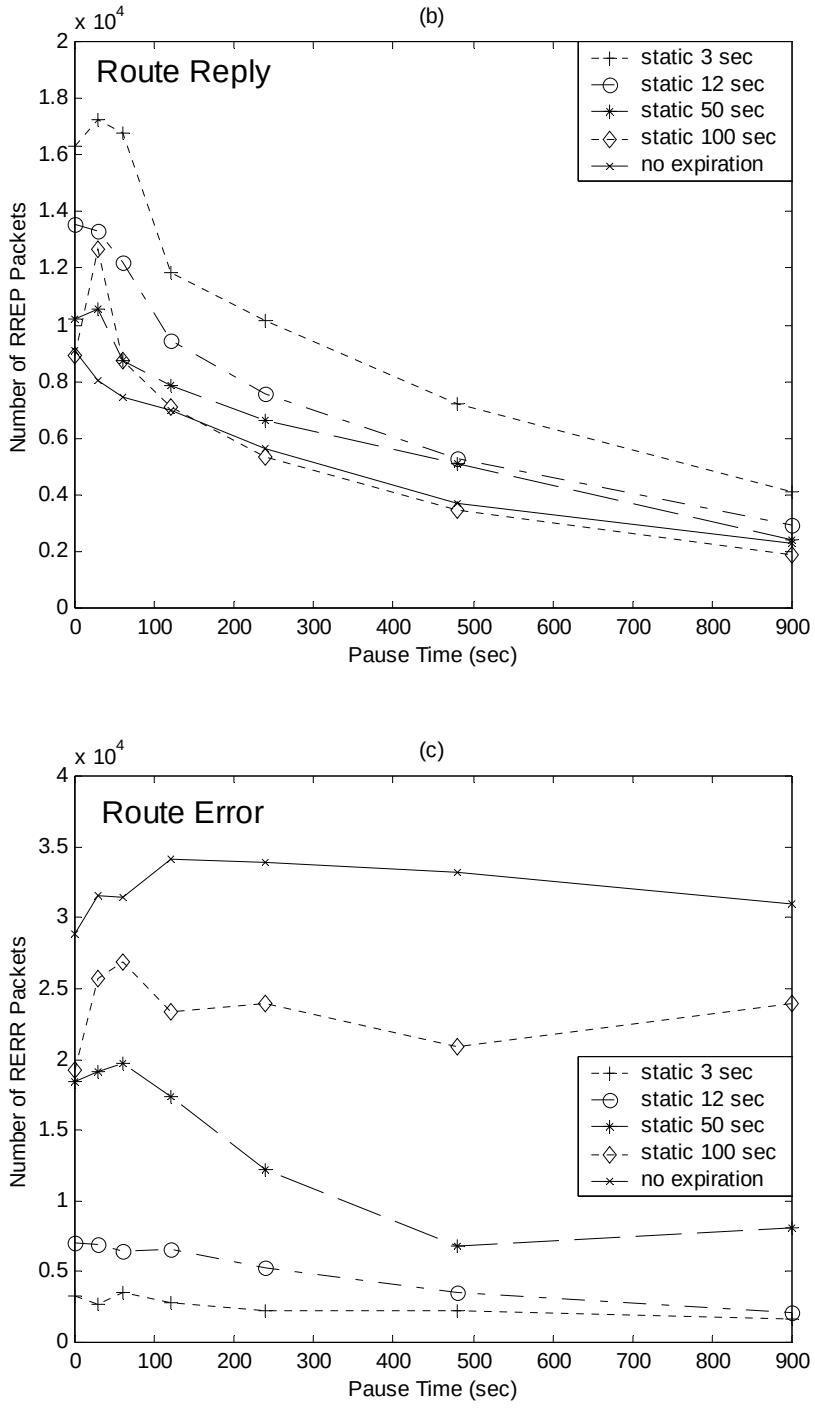


Figure 5-3. Continued

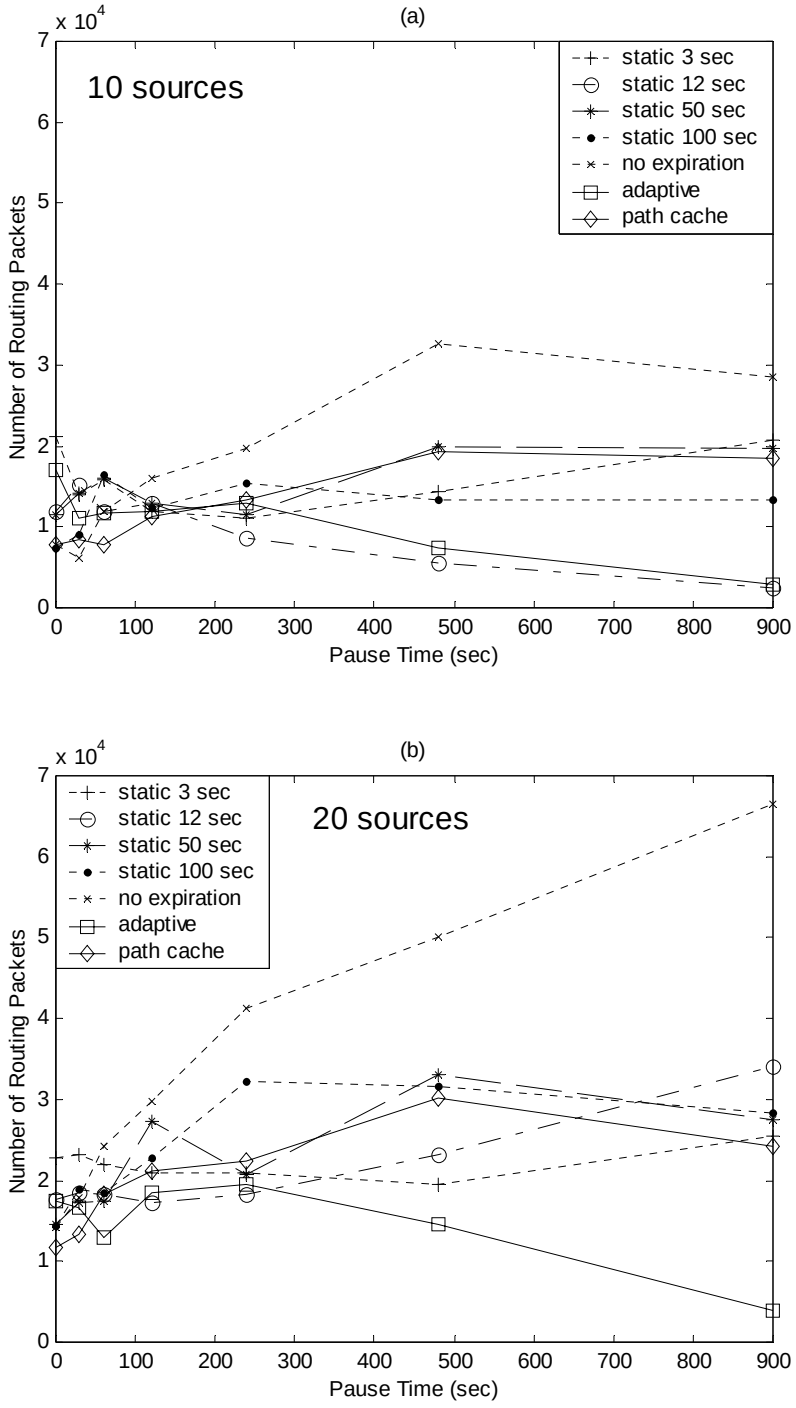


Figure 5-4. Comparison of total routing overhead. (a) 10 sources. (b) 20 sources. (c) 40 sources. (d) 50 sources

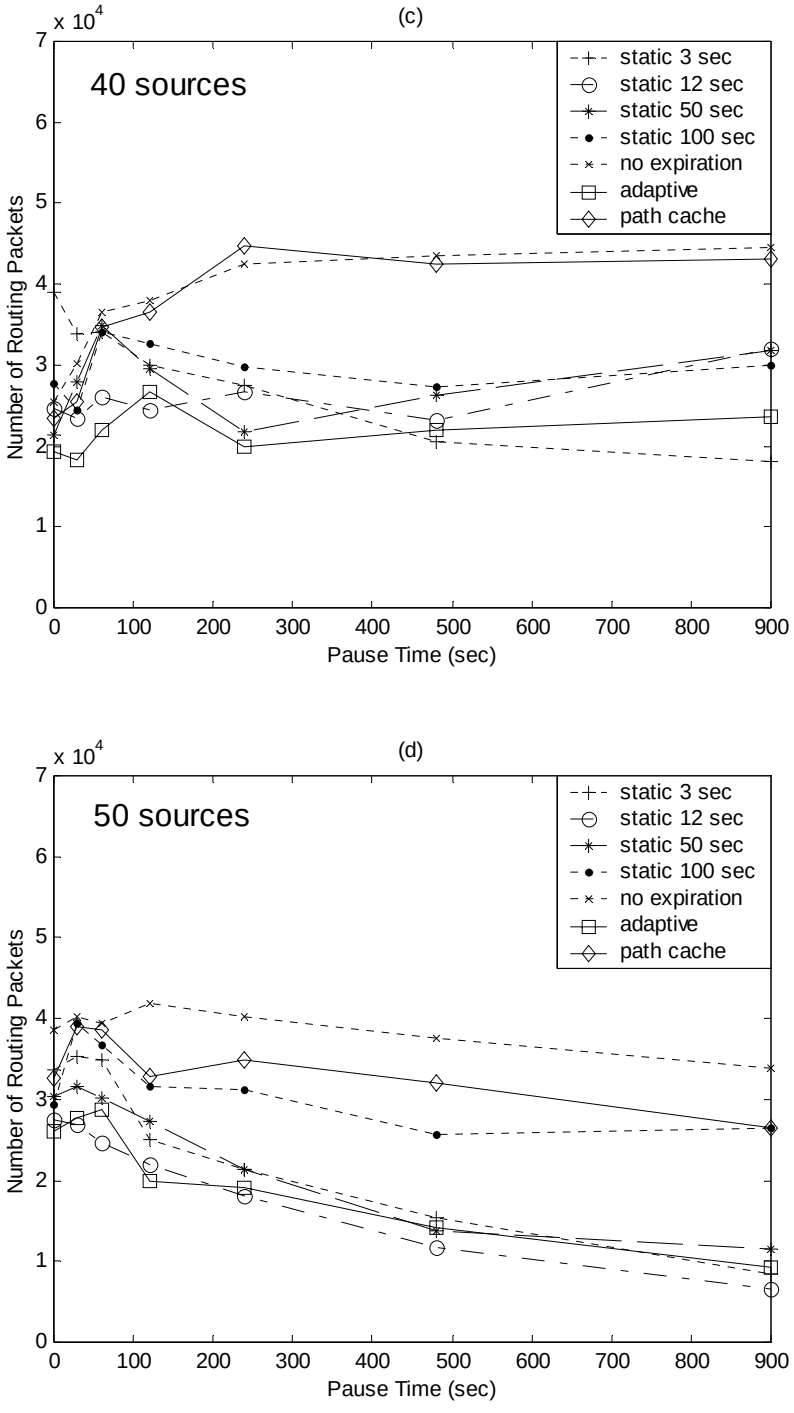


Figure 5-4. Continued

5.3.3 Effects on Packet Delivery Ratio

In this subsection, we evaluate the application level performance metric – packet delivery ratio (the end-to-end throughput). The packet delivery ratio is the fraction of packets that are received at corresponding destination over those sent at the source. It is the most important metric to evaluate the performance of an ad hoc routing protocol. There are two major situations that a packet may drop. One is due to the stale routes. A stale route in the cache may direct a packet to an actually broken link. When a node fails to forward a packet over a broken link, it will try to savage the packet by looking up in its own route cache for an alternative route to the packet's destination. If the alternative path does not exist or the packet has already been salvaged before, the packet is dropped. To reduce such packet drops, small lifetime value is preferred because small lifetime value could expire stale routes and reduce the chances that a stale route is used. The other type of packet loss is due to the heavy collisions in the MAC layer that cause a packet drop after failing a certain number of attempts to transmit the packet. A route discovery process can cause a large number of route requests and route replies generated within a short time, thus cause increased interference to data traffic at MAC layer. When the traffic load is high, the packet loss caused by collisions becomes more severe. To reduce the packet loss due to this reason, a large lifetime value is preferred because a large lifetime value could minimize the number of route discovery performed. Figure 5-5 summarizes the performance of packet delivery ratio under various network traffic loads and node mobility levels. We observe that, in general, the link cache scheme outperforms the path cache scheme when the network load is high. The heavier the network load, the clearer the trend, and the wider the performance gap. However, when the traffic load is

low, the performance comparison between the two types is not certain. A link cache does not always show advantage over the path cache, as shown in 10 sources case.

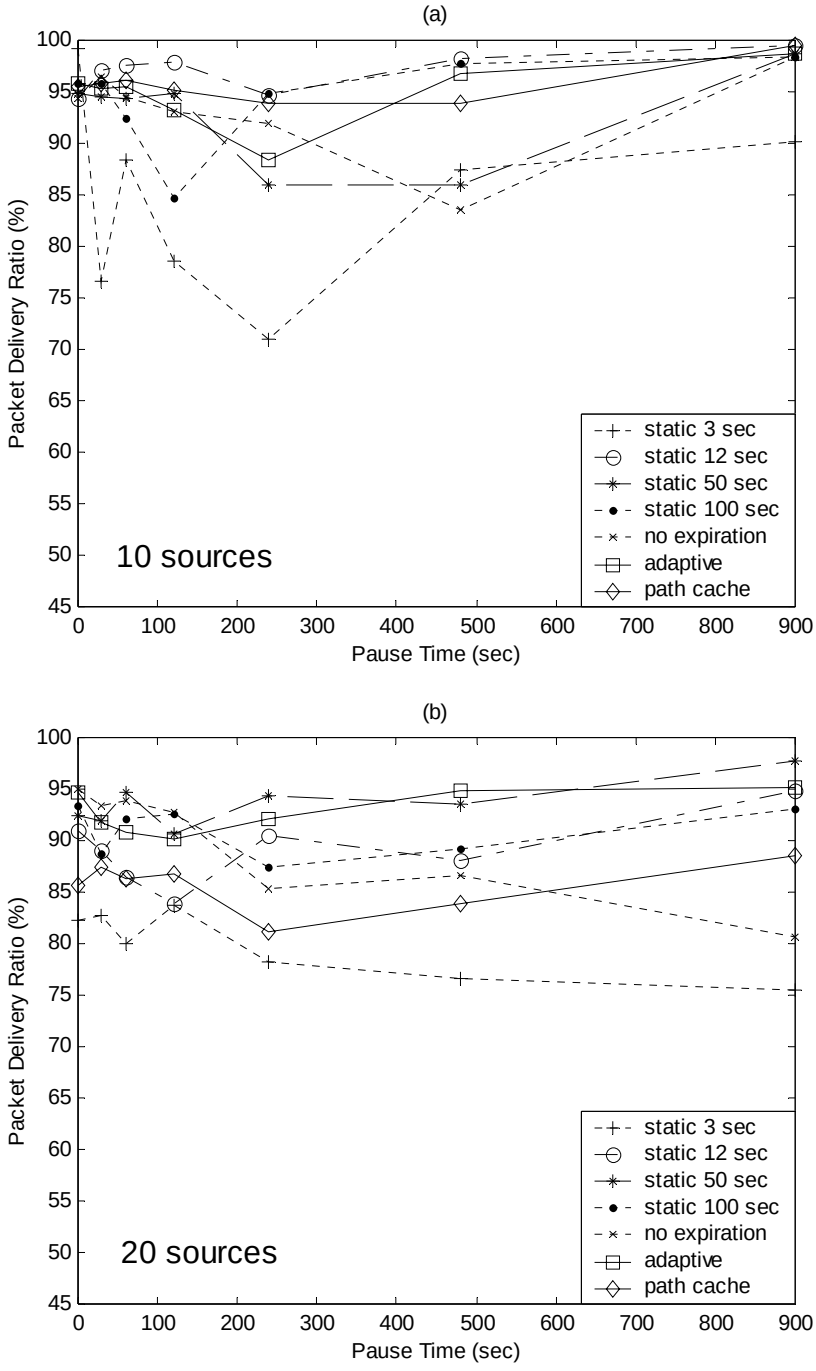


Figure 5-5. Comparison of packet delivery ratio. (a) 10 sources. (b) 20 sources. (c) 40 sources. (d) 50 sources.

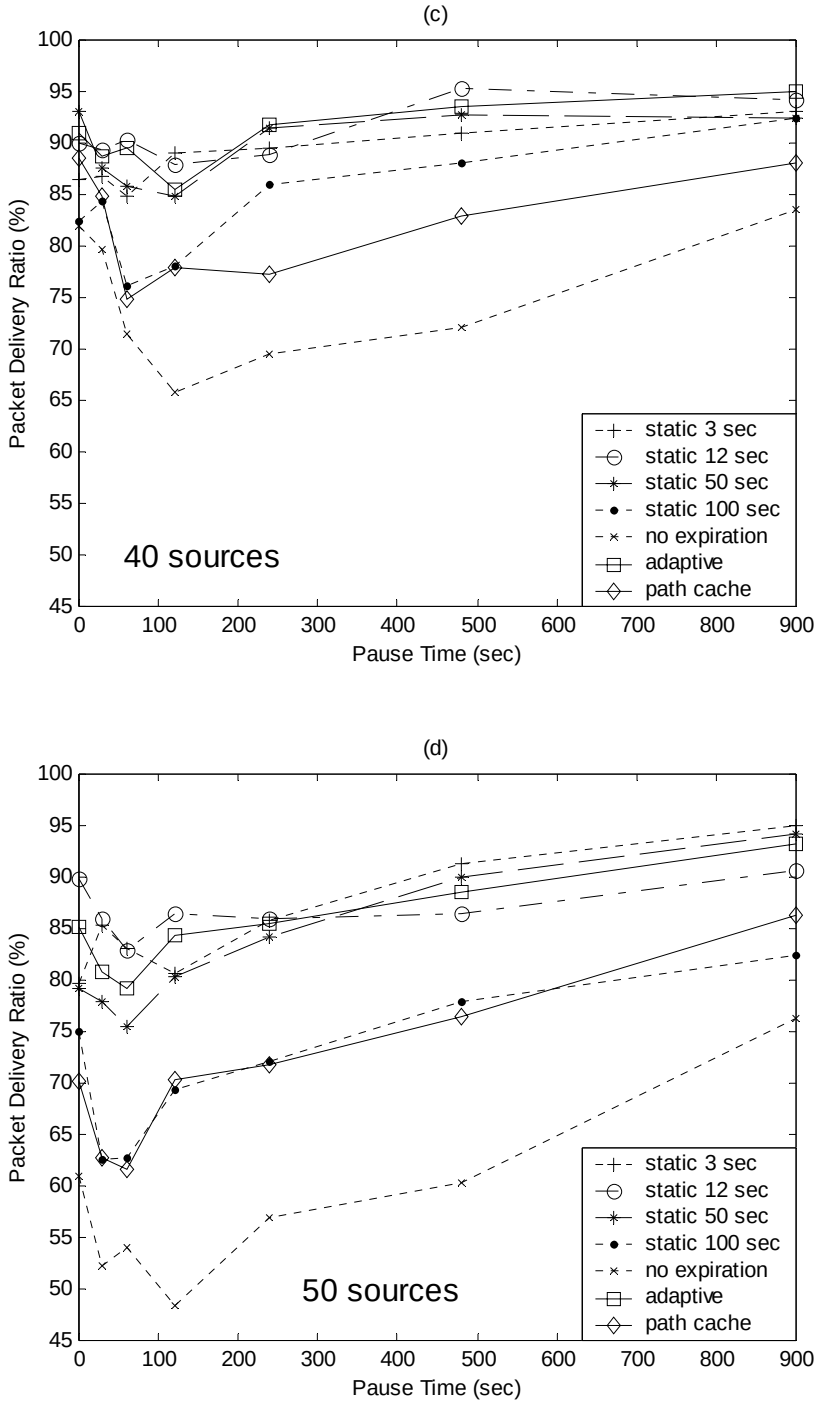


Figure 5-5. Continued

If we use the packet delivery ratio as the performance index, we also observe that the "optimal" link lifetime values vary at different node mobility levels. Nodes with high

mobility tend to perform better with shorter lifetime while nodes with low mobility tend to perform better with a little bit longer lifetime. No single static lifetime can provide optimal performance under all circumstances. Again we observe that the adaptive cache scheme could keep near optimal performance under various mobility scenarios. By studying the performance of packet delivery ratio and the routing overhead together, we find that the performance of packet delivery ratio in high traffic load scenarios is significantly affected by the routing overhead in ad hoc networks.

5.3.4 Effects on Packet Latency

Packet latency is another important performance metric. The packet latency is also called end-to-end delay, which is the latency between a packet sent at the source and received at the destination. The packet latency is only calculated for packets that are successfully delivered. Besides the ordinary transmission delay, propagation delay, and queuing delay, which widely exist in all IP networks, there are two types of latency caused particularly by ad hoc on-demand routing protocols. One is the latency the protocol takes to discover a route to a destination when there is no known route to that destination. This type of latency is due to the on-demand behavior of the routing protocol and exists in all such protocols. The other one is the latency for a sender to "recover" when a route used breaks. The latency resulting from broken routes could be very large because the amount of latency is the addition of the following three parts, the time for a packet travel along the route to the node immediately before the broken link, the time for that node to detect the broken link, and the time for a route error message to travel from that node back to the source node. Among them, the time to detect a broken link could be very large because the failure of the node can only be determined after having made a certain number of attempts to transmit the packet over the broken link but failed to

receive a passive or explicit acknowledgement of success. Figure 5-6 shows the performance comparison of packet latency across different variations of DSR protocol. We observe that, with an appropriate link lifetime, the packet latency incurred by the link cache scheme is kept relatively low compared to the path cache scheme. Especially when the network traffic load grows high, the advantage of the link cache becomes clearer. Since the small link lifetime value tends to expire links earlier than they actually break, the possibility of using broken links are reduced, thus the latency caused by broken routes is minimized. Therefore, we observe that small lifetime values result in low packet latency. Correspondingly, without an appropriate timeout mechanism (e.g. the link lifetime assigned is too large), packets suffer abnormally large latency because too many broken routes are used. Again we observe that the proposed adaptive cache scheme maintains comparably low latency under various mobility level conditions.

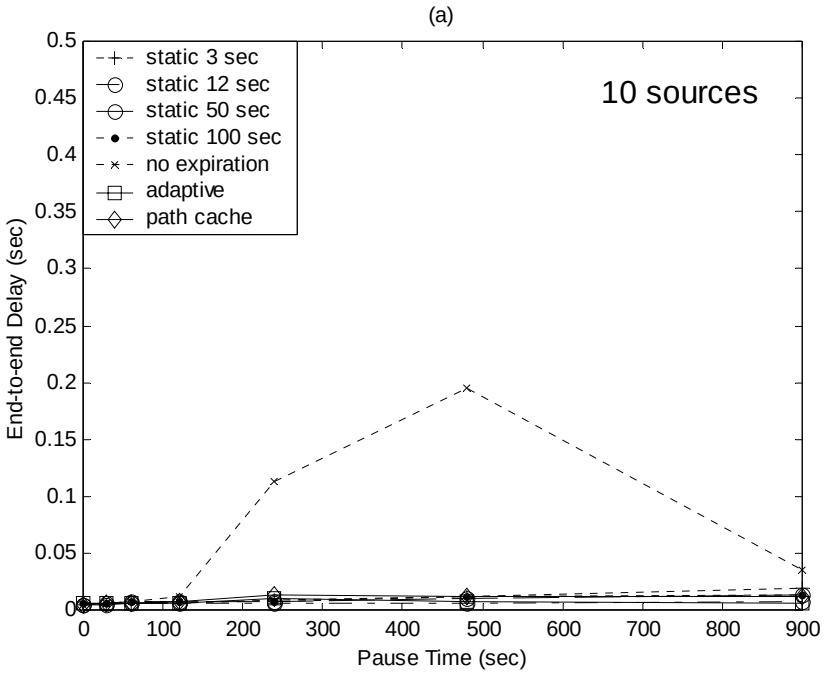
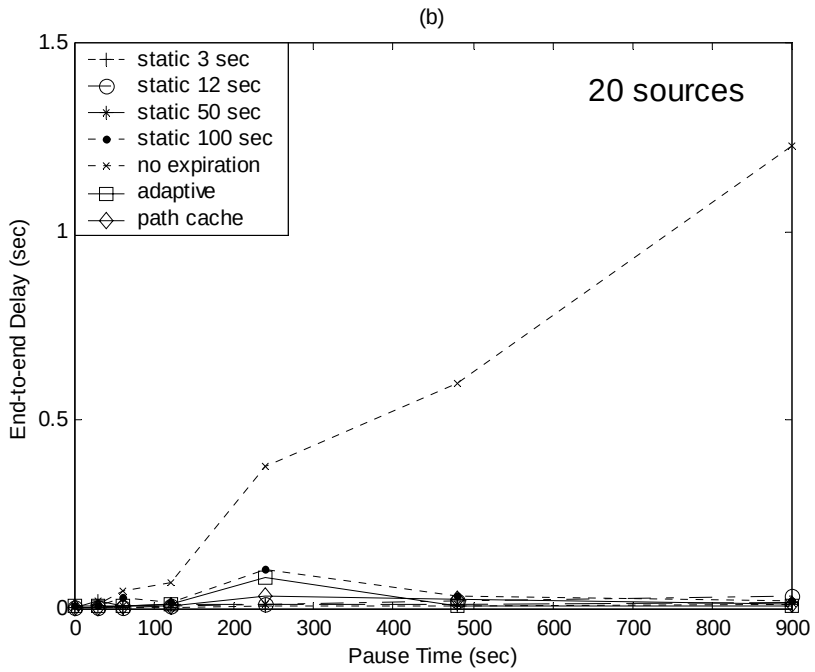


Figure 5-6. Comparison of end-to-end delay. (a) 10 sources. (b) 20 sources. (c) 40 sources. (d) 50 sources



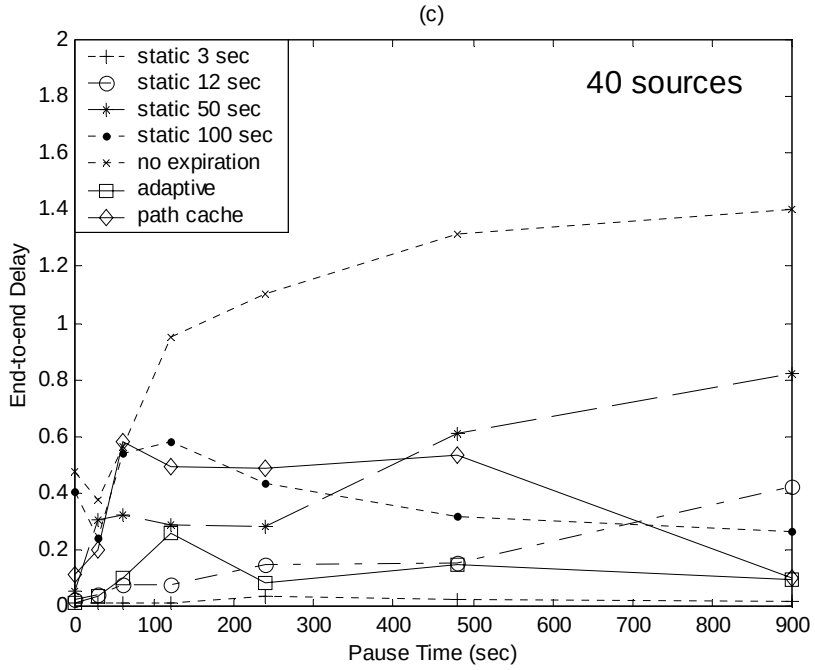


Figure 5-6. Continued

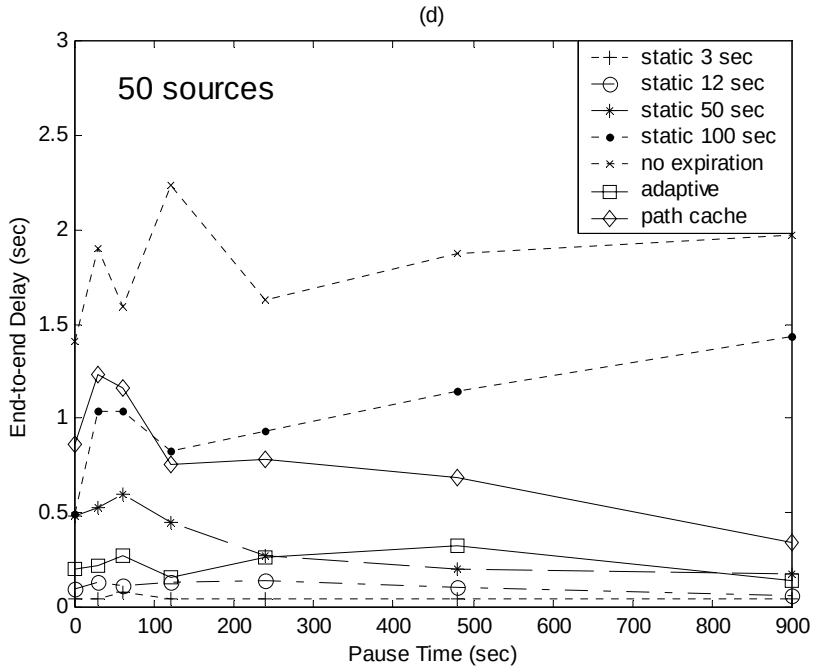


Figure 5-6. Continued

5.4 Summary

In this chapter, we study the effects of the route cache schemes on the performance of on-demand routing protocols in ad hoc networks. We base our simulation on DSR, the well-evaluated on-demand routing protocol in ad hoc networks. We first study the "link cache" performance with static lifetime assignments. The results indicate that an appropriate timeout mechanism is critical on the performance of such a link cache scheme. A link cache with an appropriate timeout mechanism could make use of the available route information more efficiently, thus improve the protocol performance. However, without an appropriate timeout mechanism, a link cache may cause dramatically increased route errors and consequently severe performance degradation. We also find that in general the link cache scheme performs better when network traffic load is high. Based on these observations, we propose an adaptive link timeout

mechanism in combination with a “link cache” organization (the adaptive link cache scheme) for DSR. The adaptive link lifetime estimation scheme aims at tracking the “optimal” link lifetime under various node mobility conditions. The performance of the proposed cache strategy is compared with the conventional “path cache” DSR. The results show that when the number of traffic sources increases, the proposed “link cache” DSR outperforms the “path cache” DSR, with wider performance gap with increasing load.

In this chapter, we investigate the link cache strategy. By using it in the traditional way (single shortest path routing), it shows that the performance of DSR under more stressful situation can be much improved. In addition, since a link cache is similar to a link state type of protocol which maintains the whole network topology, the link cache strategy makes it easier to find multiple node-disjoint paths for a single source and destination pair. The study on the link cache is the foundation of the multipath routing technique we discuss in Chapter 6.

CHAPTER 6
MULTIPATH ROUTING AND PATH SET OPTIMIZATION

As we discussed in the previous chapter, routing in ad hoc networks presents great challenge. The challenge comes mainly from two aspects: constant node mobility causes frequent topological changes while limited network bandwidth restricts the timely topological updates at each router. On-demand routing has been widely developed in mobile ad hoc networks in response to the bandwidth constraints because of its effectiveness and efficiency. The multipath routing technique is another promising technique to combat the frequent topological change and link instability problem in a MANET environment since the use of multiple paths could diminish the effect of possible link failures. Several multipath routing schemes have been proposed to improve the reliability, fault-tolerance, end-to-end delay for bursty traffic, as well as to achieve load balancing, etc. [16,17,37-39,62]. In this chapter, we review the multipath routing techniques in the current literature in a MANET environment and present the techniques we used to find and optimize the multiple paths desired in our SPREAD scheme to improve the network security.

6.1 Multipath Routing in Ad Hoc Networks

Multipath routing, sometimes called traffic dispersion, has been one of the most important current directions in the area of routing. In a reasonably well-connected network, there would be several paths between any source-destination pair. While most of the current routing protocols are based on the single shortest path routing, multipath routing gives the source node a choice at any given time of multiple paths to a particular

destination by taking advantage of the connectivity redundancy of the network. Multipath routing has been studied for various network control and management issues in various types of networks. For example, this approach has been applied to aggregate bandwidth and minimize delay, to support Quality of Service (QoS) routing, to smooth the burstiness of the traffic, to alleviate network congestion, and to improve the fault tolerance, etc. [15,33,35,36].

Multipath routing in a MANET has been widely developed as a means to provide route failure protection. For example, the Dynamic Source Routing (DSR) protocol [20] is capable of caching multiple routes to a certain destination. When the primary path fails, an alternate one will be used to forward the packet. The Temporally Ordered Routing Algorithm (TORA) [63] also provides multiple paths by maintaining a destination-oriented directed acyclic graph (DAG) from the source node. The performance studies of some on-demand multipath extensions for DSR [18] showed that providing a few alternative paths could keep an end-to-end connection for a longer time than using a single path. In other word, the routing overhead, particularly the costly route discoveries, can be reduced significantly. In these cases, the multiple paths are not used simultaneously. The traffic takes one of the multiple paths at a time. Other paths are kept as backup in case the used one is broken. When all known paths are broken, a new multipath discovery procedure is initiated.

Another way to use the multiple paths is that the traffic flows through multiple paths simultaneously. Simultaneous multipath routing in a MANET has been developed to improve the throughput, reliability and achieve load balancing. Some type of source coding scheme is usually incorporated, particularly for reliability. A certain amount of

redundancy is coded into the data traffic such that the successfully decoding could tolerate a certain amount of the lost data or even tolerate the failure of a path (paths). For example, Tsirigos and Haas [16,17] proposed a multipath routing scheme based on diversity coding [64] in which data load is split into multiple pieces (with redundancy) and distributed over multiple paths in order to minimize the packet drop rate, achieve load balancing, and improve end-to-end delay. In [65], Gogate et al. proposed an interesting application which combined the multiple path transport (MPT) and multiple description coding (MDC) in order to send video and image information in a MANET.

For our SPREAD scheme, we need independent paths, more specifically, node disjoint paths, because we are dealing with node compromising problem. Several multipath routing protocols have been proposed in the literature that are able to found node disjoint paths in an ad hoc network. In [38], Pearlman et al. investigated the Alternate Path Routing (APR) for load balancing. A set of candidate alternate routes is found using the "diversity injection" technique [66]. Their investigation of APR in the MANET environment revealed that APR provide effective load balancing, improved capacity, and improvements in end-to-end delay, particularly for bursty traffic. The impact of the "route coupling" problem was also discussed in the paper. In [37], Lee and Gerla proposed the "Split Multipath Routing" technique to find multiple routes of maximally disjoint paths. Two approaches of route Maintenance were also introduced. The performance study on the SMR showed the same effects: providing robustness to mobility, distributing the load to the network hosts, and reducing the end-to-end delay. In [39], Wu and Harms proposed another "on demand multipath routing" technique which is able to find multiple less correlated node-disjoint paths in a shared-channel MANET. The

objectives of the multipath routing were also to improve throughput, reduce routing overhead, achieve load balancing, and reduce end-to-end delay.

Most of the proposed protocols are on-demand, due to the network bandwidth limitation, and use source routing technique to control the disjointness of the paths at source node. For an on-demand routing protocol, whenever it needs a path to a certain destination but it does not know one, it starts a route discovery process by broadcasting the route inquiry messages throughout the network, the destination (or intermediate nodes that have a valid route to the destination) will reply by sending back the route. Some type of cache is necessary to store the routes previously found so that the node does not have to perform the costly route discovery for each individual packet. In DSR and the multipath extension of DSR, the route replies back to the source contain the complete node list from the source to the destination. By caching each of these paths separately, a "path cache" organization can be formed. This type of cache organization has been widely used in the proposed protocols. However, the paths found by this means might not serve our purpose best. They are not necessary the most secure paths. The link cache we designed in the previous chapter is an alternative cache organization that can be easily incorporated into other DSR like on-demand multipath routing protocols. By decomposing each candidate route into individual links and represented in a unified graph data structure, a link cache organization provides the source node a partial view of the network topology, similar to a link-state type of routing protocol. Thus, by using such a link cache, we could separate the routing and the path set optimization. Although we rely on an underlying routing protocol to provide us with a partial view of network topology,

the finding of the optimal paths can be done independent of the routing protocols used, but based on the discovered partial network topology.

In the rest of the chapter, we discuss how to choose the optimal multiple paths, given the (partial) view of the network topology. We first discuss how to find the secure path, and then we discuss how to find the multiple paths with the desired property: as many as possible and at the same time as secure as possible.

6.2 Security Related Link Cost Function

We introduce the security as a dimension of routing metric here and discuss how to select a path based on the node security property. Assume in the mobile ad hoc network, each node n_i is associated with a security related parameter q_i. For simplicity and consistence, we define this security related parameter q_i as the probability that node n_i might be compromised. In reality, the value of q_i indicates the security level of node n_i and it could be estimated from the feedback of some security monitoring software and/or hardware such as firewalls and intrusion detection devices. It could also be assigned manually by administrators based on the level of physical protection of nodes, the positions of nodes, or the rankings of nodes, and so on. For example, the headquarters in the rear has the highest security level (lowest q_i value) while the individual scout penetrating into enemy ground would have lower security level (higher q_i value). Or the officers with higher rank may have higher security level while the soldiers with lower rank may have lower security level. It is important that these security levels are immutable, e.g. nodes should not be able to change their security level arbitrarily in an unauthorized way. To ensure this, some form of authentication or temper resistant device is needed. We leave this to the related security researchers. Here we simply assume such mechanism is already in place.

Then the probability that a (*s,t*) path consisting of node *s, n_1, n_2, ..., n_l, t* might be compromised equals to

$$p = 1 - (1 - q_1)(1 - q_2)\cdots(1 - q_l)$$

So the objective of the most secure path routing algorithm is to find from source *s* to destination *t* the path that minimizes *p*, the probability that the path might be compromised. Notice that here we assume the source and destination are trusted.

The conventional shortest path algorithms cannot be applied to this problem directly because the cost function of the path is not additive. We apply a simple transform to the above equation and it becomes

$$-\log(1-p) = -\log(1-q_1) - \log(1-q_2) - \cdots - \log(1-q_l)$$

We can define the cost function of link between node n_i and n_j as

$$c_{ij} = -\log(1-q_j)$$

This will convert the link cost to additive function. However, this definition will cause asymmetric link costs, which is not desirable in many routing or path-finding algorithms.

We define the following symmetric link cost function to convert the security characteristics into an additive link cost function so that the shortest path algorithm is readily used as most secure path finding algorithm.

The cost of link between node n_i and n_j is defined as

$$c_{ij} = -\log\sqrt{(1-q_i)(1-q_j)}$$

Then the cost of the (*s,t*) path consisting of node *s, n_1, n_2, ..., n_l, t* is

$$\begin{aligned}
&cost(s,t) = c_{s1} + c_{12} + \cdots + c_{l-1,l} + c_{lt} \\
&= -\frac{1}{2}\log(1-q_s) - \log(1-q_1) - \log(1-q_2) - \cdots - \log(1-q_l) - \frac{1}{2}\log(1-q_t) \\
&= -\log\{(1-q_1)(1-q_2)\cdots(1-q_l)\} - \frac{1}{2}\log\{(1-q_s)(1-q_t)\}
\end{aligned}$$

With the shortest path algorithm, *s,t* are fixed,

$cost(s,t)$ is minimized, thus
$-\log\{(1-q_1)(1-q_2)\cdots(1-q_l)\}$ is minimized,
$(1-q_1)(1-q_2)\cdots(1-q_l)$ is maximized, then
$p = 1-(1-q_1)(1-q_2)\cdots(1-q_l)$ is minimized.

Thus with the defined security related cost function, the non-additive security metric is transformed into an additive one. The most secure path problem is transformed into a simple shortest path problem. Conventional shortest path algorithms can be applied directly.

If we treat the security as a dimension of quality of service (QoS) routing [67,68], many other QoS routing protocols [69] which are developed for other additive performance metrics, such as end-to-end delay, can also be applied to this new metric.

6.3 Maximal Paths Finding Algorithm

In this section, we present a maximal node-disjoint paths finding algorithm to optimize the path set for SPREAD based on the partial view of the network topology, which is discovered by any underlying routing protocol. The links are symmetric and the costs are defined using the security related cost function we described in the previous section.

Ideally, given a network, we wish to find an optimal path set, such that the probability *Pmsg* is minimized. Recall that $P_{msg}(\underline{n}) = \prod_{i=1}^{M} p_i$, intuitively, since p_i is a

probability that is always less than 1. The more items of p_i, the less the probability, the better the security. So the general goal of our path finding algorithm is to find as many as possible paths while at the same time as secure as possible.

The simplest way to find multiple node-disjoint paths is to find the first path, remove all intermediate nodes in the selected path, then find the next one from the remaining network, so on and so forth. However, this simple paths finding algorithm cannot find the maximal number of paths. For example, as shown in Figure 6-1, this simple path finding algorithm will fail to find a second path for figure 6-1(a) and fail to find a third path for figure 6-1(b).

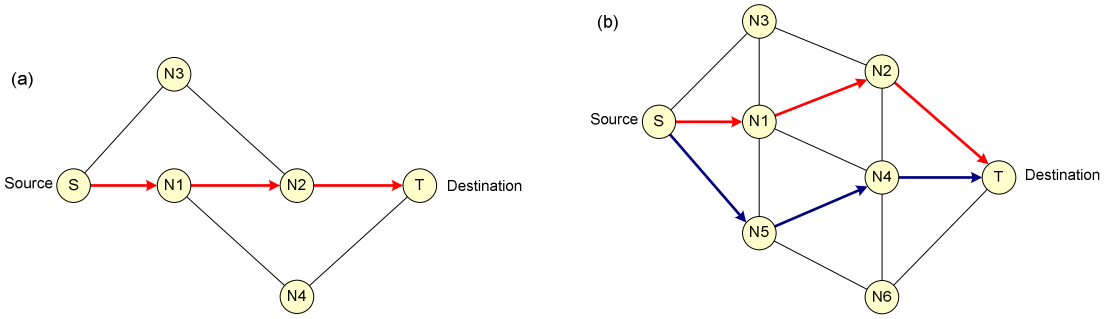


Figure 6-1. A simple node-disjoint paths finding algorithm

The maximal paths finding algorithm proposed for our SPREAD scheme is modified from the node disjoint shortest pair algorithm [70]. The basic idea of the algorithm is not simply removing the nodes on the selected paths; instead, a graph transform is applied so that the selected nodes and links can be temporarily reused. Then after the interlacing removal, when the graph is transformed back to the original format, the maximal number of paths can be found by regrouping the selected links. In [41], Papadimitratos et al. proposed a similar disjoint pathset selection protocol (DPSP) which

aims to select multiple disjoint paths in a mobile ad hoc network. The objective of their multipath routing is to combat the path failure due to topological changes. The disjoint paths they found are edge-disjoint while in our case we find node-disjoint paths.

Since the graph transformation will cause negative link costs, a modified Dijkstra algorithm is used so that the shortest path can be found correctly even when the negative links present (but no negative loop) in the graph [71]. For the non-negative graph, the algorithm reduces to the standard Dijkstra algorithm. Figure 6-2 summarizes the modified Dijkstra algorithm.

Let *c(i,j)* denote the cost between node *i* and *j*. If there is no link between node *i* and *j*, *c(i,j)=INFINITY*.
Let *cost(i)* denote the cost from source *s* to node *i.*
Let *P(i)* denote node *i*'s predecessor.
Let *L(i)* denote the label of node *i*, which is either "permanent" or "tentative".

Then the steps of the modified Dijkstra algorithm are:

Step1: Initialization

```
For each node i, do {cost(i)=c(s,i); P(i)=NONE; L(i)= tentative}
Start with n=s; cost(n)=0; L(n)= permanent;
```

Step 2:

```
For each node i, if (c(n,i)<INFINITY && cost(n)+c(n,i)<cost(i)) do
    { cost(i)=cost(n)+c(n,i); P(i)=n; L(i)=tentative}
```

Step 3:

```
Find node n, L(n)==tentative and cost(n)=min{cost(i)} for all i,  L(i)==tentative
L(n)=permanent;
If (n==t) return; else goto step 2;
```

Figure 6-2. Modified Dijkstra algorithm.

The maximal node-disjoint paths algorithm is then an iterative procedure. The most secure path is found first and added to the path set. Then in each iteration, the number of paths in the set will be augmented by one. Figure 6-3 summarizes the steps taken to find the maximal number of paths. Each time a new path is added to the selected path set, a

graph transformation is performed, which involves a vertex splitting of the nodes on the selected paths (except the source and destination node). Then the modified Dijsktra algorithm is executed to find the most secure path in the transformed graph. Then the split nodes are transformed back to the original one, any interlacing edges are erased, the remaining edges are grouped to form the new path set. For each iteration, the number of paths will be augmented by one.

Step 1. Find the first most secure path by modified Dijkstra algorithm, select the path

Step 2. Perform a graph transformation as follows

For each selected path:

a. Replace the links used in the path with directed arcs – for the arc that is directed towards the source, make its cost the negative of the original link cost; make the cost of the arc directed towards the destination infinite (i.e., remove it)

b. Split each node on the selected paths (except the source and destination) into two collocated subnodes; Connect the two subnodes by an arc of cost 0 and directed towards the source node.

c. Replace each external link that is connected to a node in the selected paths by its two component arcs of cost equal to the link cost – let one arc terminate on one subnode and the other one emanate from the other subnode such that along with the zero-cost arc, a cycle does not result.

Step 3. Run the modified Dijkstra algorithm, find the most secure path in the transformed graph

Step 4. Transform back to the original graph; erase any interlacing edges; group the remaining edges to form the new path set.

Step 5. Go to step 2, until no more path can be found or the new path set does not offer better security.

Figure 6-3. Maximal node disjoint path finding algorithm.

Figure 6-4 shows an example of the path finding algorithm. After finding the first two node-disjoint paths, the third one temporarily makes use of the selected nodes and links but reusing the selected links in the reverse direction. After the interlacing removal and regrouping, a path set consisting of 3 paths is found instead of 2.

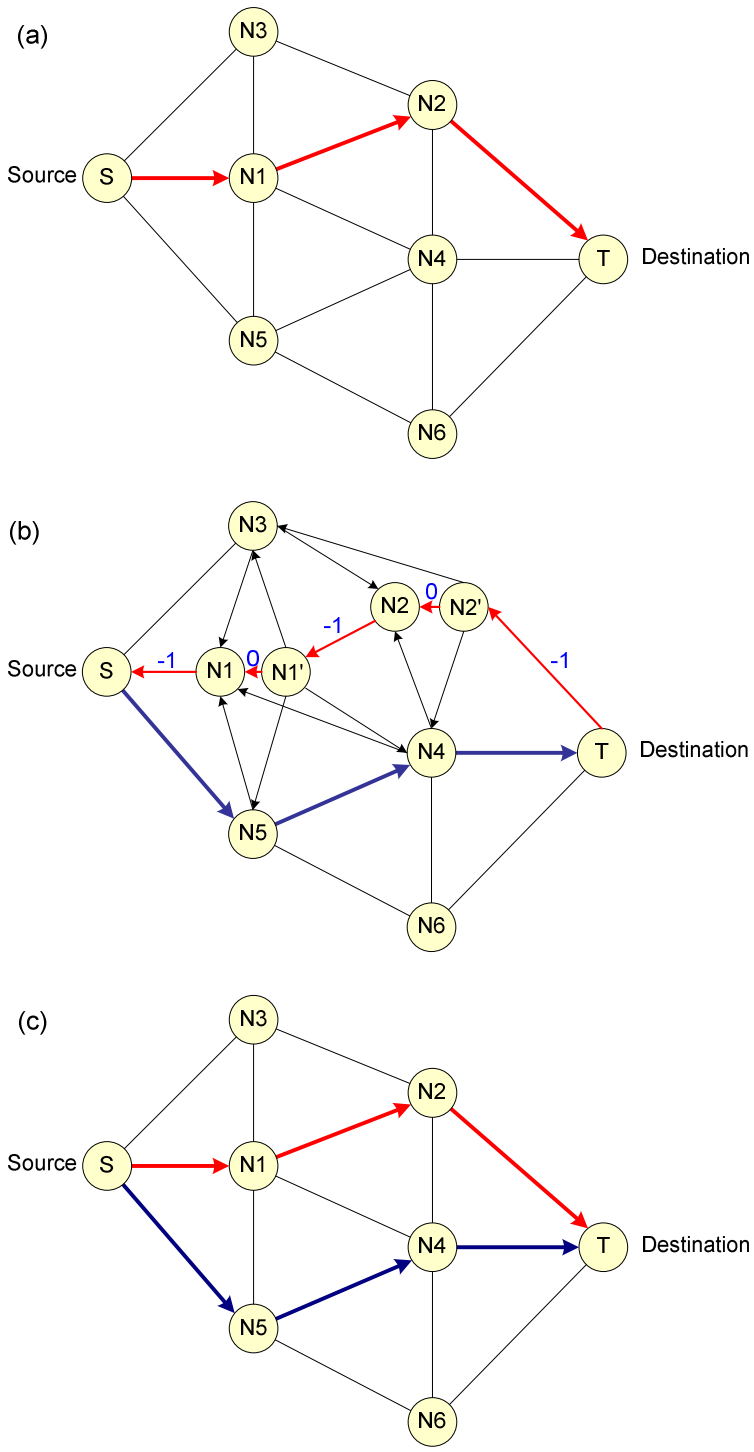


Figure 6-4. Finding the maximal node disjoint paths. (a) Iteration 1 – Modified Dijkstra algorithm. (b) Iteration 2 – Graph transformation and Modified Dijkstra algorithm. (c) Iteration 2 – resulting 2 paths. (d) Iteration 3 – Graph transformation and Modified Dijkstra algorithm. (e) Iteration 3 – Edge regrouping. (f) Iteration 3 – resulting 3 paths

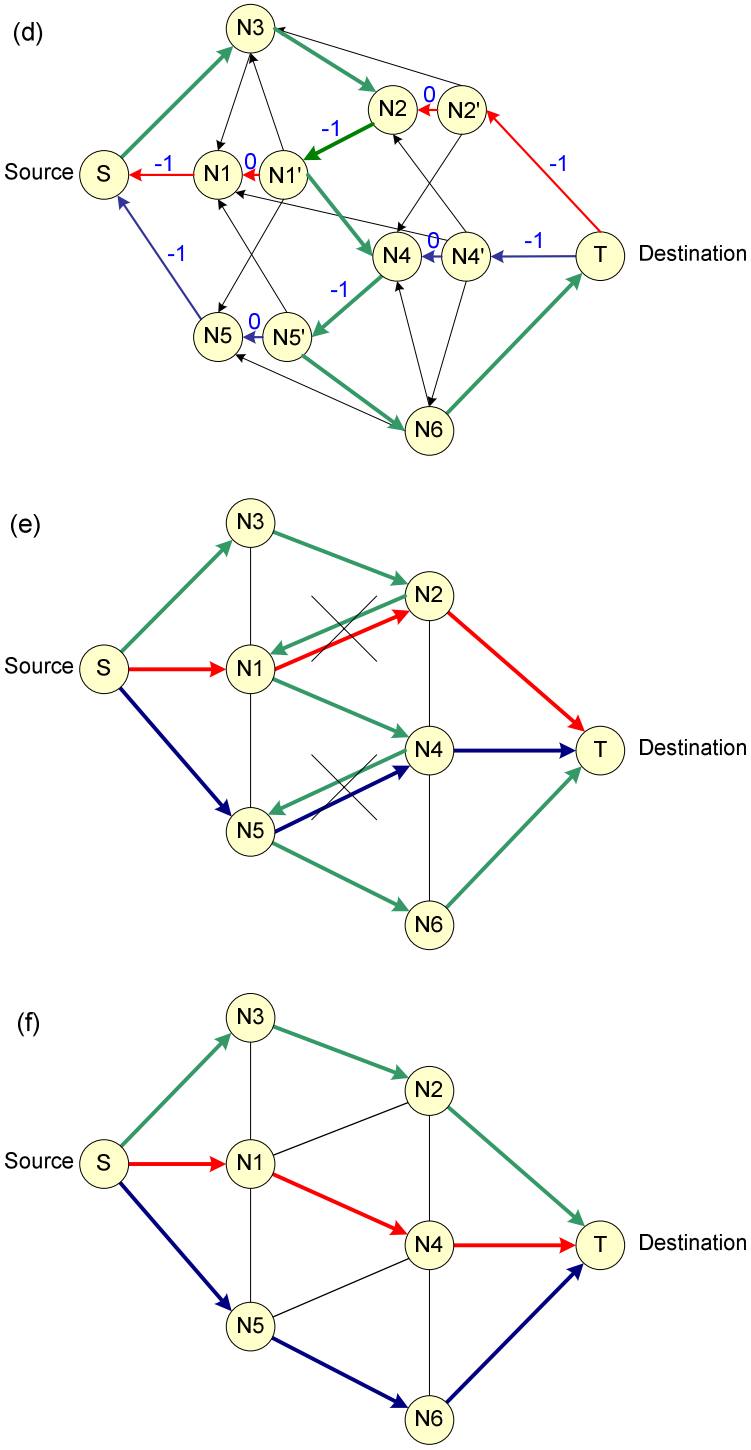


Figure 6-4. Continued.

Because the paths in the path set are changing in each iteration, we calculate the *Pmsg* after each iteration. If *Pmsg* is not improving (dropping) in the iteration, the path set found in the previous iteration yields the best security results. The paths finding algorithm stops.

6.4 Performance Evaluation

In this section we present the simulation results to show the effectiveness as well as the performance evaluation of SPREAD.

We simulate an ad hoc network with 100 nodes randomly deployed in a 1000m by 1000m area. The transmission range of each node is set equal in each simulation and varies in different simulations. The simulation results are averaged over 20 randomly deployed networks. To factor out the effect of routing protocols, in the simulation we assume the network topology is known. In each network, we find 1, 2, …, till maximal node-disjoint paths for each source-destination pair which is at least three hops away. Two sets of simulations are executed. In the first set, each node is assumed equally likely to be compromised with probability q_i=0.152. In the second set of simulation, each node is assigned a probability randomly: 10% of nodes with probability q_i=0.50, 30% of nodes with q_i=0.20, 40% of nodes with q_i=0.10, and 20% of nodes with q_i=0.01. In the first set, all the links are of same cost. In the second set, we use the proposed link cost function to define the link cost based on the node security level (q_i).

TR(m)	200	250
Node degree	10.3	15.4
Diameter	9	6.8

Table 6-1. Network parameters of the simulated ad hoc networks.

Table 6-1 gives some basic parameters of the network topology of the simulated ad hoc networks. We see that ad hoc networks typically have dense connectivity that allows the exploitation of multipath routing techniques.

Figure 6-5 shows the probability that multiple paths are found in the simulated network. It is observed that the probability that multiple node disjoint paths exist in an ad hoc network is pretty high. Since our SPREAD scheme depends on the availability of multiple paths, the existence of such multiple paths justifies the feasibility of our scheme.

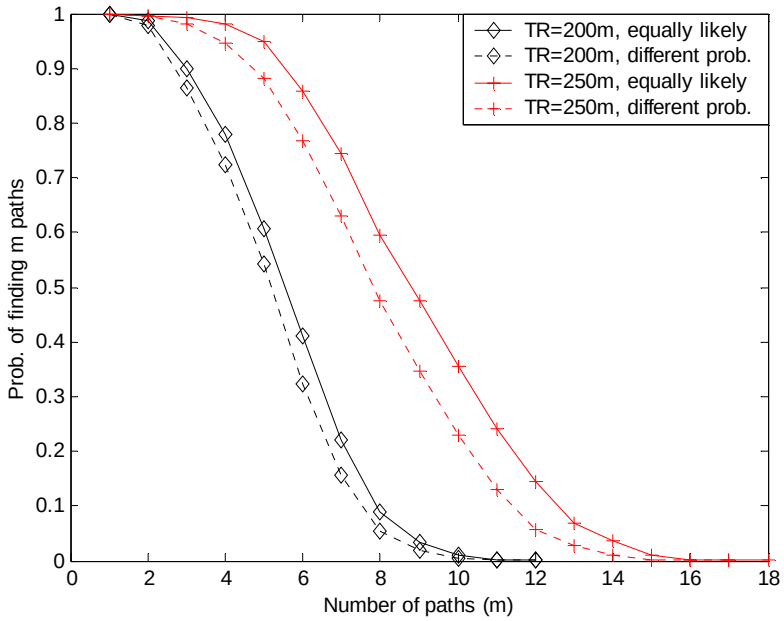


Figure 6-5. Capability of path finding.

In fact, if we run the maximal node disjoint paths finding algorithm purely for finding the maximum number of paths without considering the security property of the path set, the number of paths found in both sets would be identical. This implies that the maximum number of paths the algorithm is able to find is independent of the link costs; it

solely depends on the network topology although the actual paths found might be different for different link costs. In our simulation, we stop augmenting the path set when the security property of the found paths does not improve. Table 6-2 gives the probability that the path finding algorithm stops for this reason before finding the maximal number of paths. It indicates that when the nodes are of equal security level, the number of paths plays the most significant role. Basically, the more the paths, the more secure. However, if nodes are of different security levels (probabilities), the security of each path will have more impact on the overall security of the path set. It is not necessarily true that more paths imply higher security. This also explains that in Figure 6 the number of paths selected in simulation set 2 is less than that in simulation set 1.

TR(m)	200	250
Simulation Set 1	0.45%	0.33%
Simulation Set 2	22.7%	38.8%

Table 6-2 The path finding algorithm stops before finding the maximum number of paths

Figure 6-6 shows the probability that the message is compromised when multiple paths are used. Here, we consider the case that the message is compromised due to the node compromising. This probability is the probability for colluded attacks. One message is considered compromised when at least one compromised node is located on each of the paths selected to deliver this message. This probability for individual attack is zero because no single node is able to relay all the necessary shares due to the spreading of the shares. Noticing the logarithmic scale of the probability, we observe that the probability drops quickly (actually exponentially fast) with the increase of the number of paths used.

This result verifies the effectiveness of our SPREAD idea. We also noticed that when nodes are with different security level, our algorithm tends to select more secure paths that further decrease this probability significantly. The discontinuity of the figure indicates that no compromised message is found in our simulation (over 50,000 messages for TR=200m and over 40,000 messages for TR=250m).

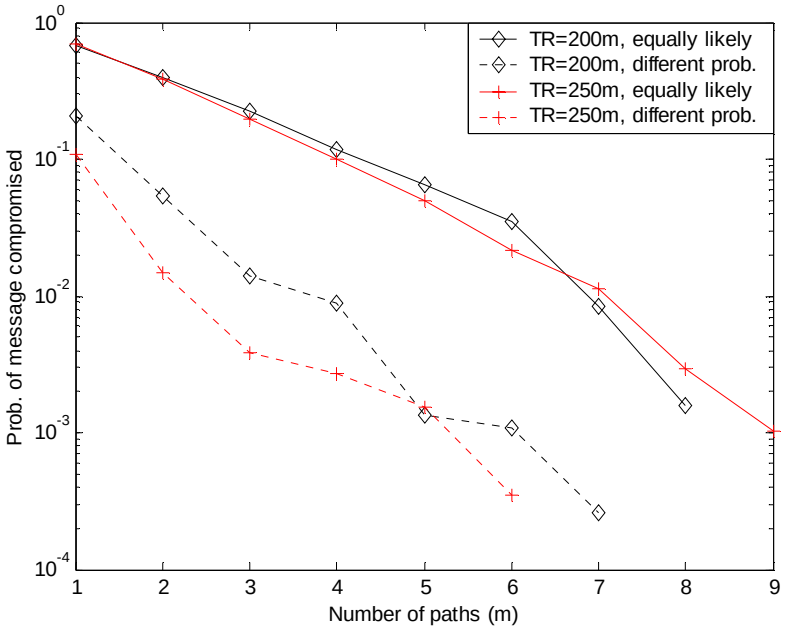


Figure 6-6. Message compromise probability.

Figure 6-7 shows the probability that message is eavesdropped when multiple paths are used. Since the wireless channel is a broadcast channel, anyone sits within the transmission range of a transmitting node is able to eavesdrop (overhear) the node's transmission. This figure actually presents the probability for individual attack. The probability for colluded attack is pretty high (almost 1) because in our simulation, we have about 15 compromised nodes among the totally 100 nodes. It is observed that, with

the increase of the number of paths, this probability decreases. However, the decrease becomes less significant when more paths are used. In fact, there is a lower bound of this probability because anyone sits within the transmission range of the source node would be able to overhear all the shares. Of course, this probability is the one that an adversary might overhear a message, it does not mean that the message can be compromised because the message shares are encrypted as well. Again, this verifies that the SPREAD idea makes it harder for an enemy to collect enough data to break the secret.

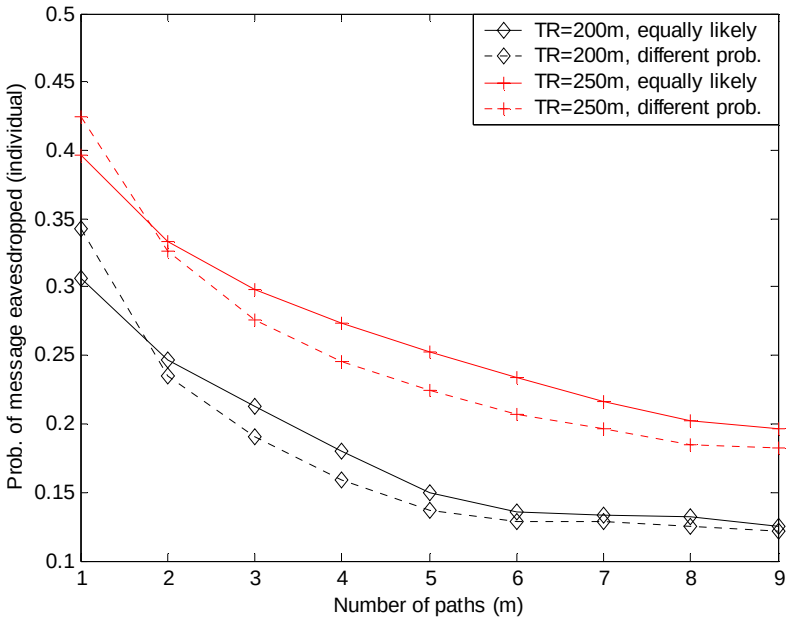


Figure 6-7. Message eavesdropping probability.

Figure 6-8 shows the bandwidth overhead calculated on a per-hop basis when multiple paths are used compared with the single minimum-hop path case. We can see that using multipath does consume more network bandwidth because longer paths are

used. However, this is the tradeoff. For security critical applications, the network efficiency might not be a major concern.

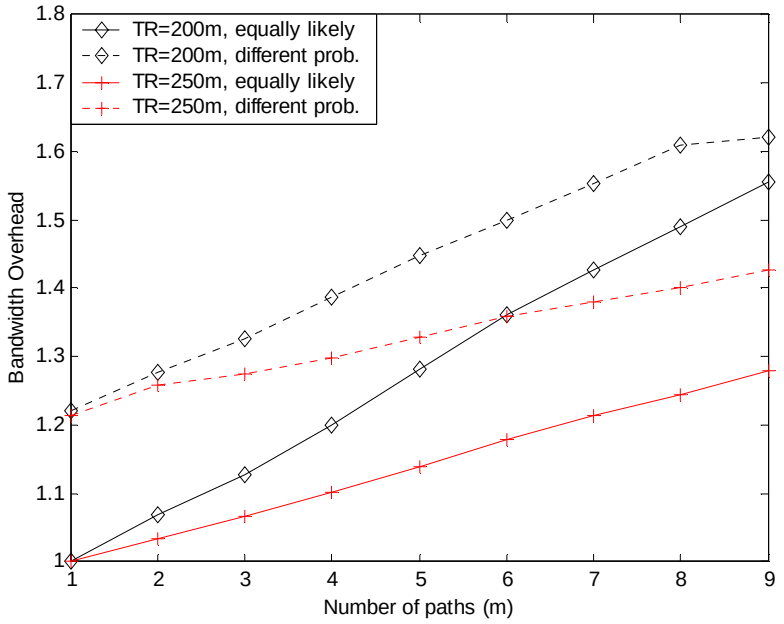


Figure 6-8. Bandwidth overhead.

6.5 Summary

Multipath routing is one of the most important technical components in our SPREAD scheme. In this chapter we present the strategy we proposed to implement the multipath routing in an ad hoc network. Because the on-demand and source routing type of routing protocols have been proven to be effective and efficient in an ad hoc network, our strategy takes advantage of this fact and make use of the link cache organization to separate the multipath routing function into two sub-functions (sub-layers), a lower layer routing protocol that is able to discover the part view of the network topology, and a path set optimization algorithm that selects the optimal paths based on the discovered partial

view of the network. Like the layered concept in any network, the two sub-layers are made transparent and independent to each other. Although they rely on each other passing the necessary information, the detailed techniques implemented in each layer can be made independent. This strategy allows us to take advantage of any multipath routing protocols that have been developed in the current literature and then optimize the path set for our SPREAD scheme.

We then develop a security related link cost function and a maximal paths finding algorithm to optimize the path set for SPREAD. The multiple paths with the desired property, i.e., as many as possible independent paths and at the same time as secure as possible, can be found effectively. The simulation results justify the feasibility of the approach and show the effectiveness of SPREAD in terms of significantly reduced message compromise probability.

CHAPTER 7
SPREAD IN WIRED NETWORKS

In the previous chapters, we described the basic idea and the system architecture of the SPREAD idea. We also investigated the implementation of the SPREAD idea in mobile ad hoc networks. However, SPREAD is a general idea to improve the security by combining the secret sharing and multipath routing. Its applicability is not limited to mobile ad hoc networks. The SPREAD idea can also be applied in wired networks to enhance the data confidentiality. The major difference between the deployment of SPREAD idea in the MANET and the wired networks lies in the routing protocols. In this chapter, we expand the SPREAD idea into wired networks (e.g., the Internet), by presenting a distributed multipath routing algorithm and examining its performance in terms of paths fining capability and security improvement.

7.1 Multipath Routing in Wired Networks

With the emergence and popularity of applications such as world wide E-commence and Virtual Private Network (VPN), more and more important and confidential information are transmitted over the public Internet. There is a more pervasive need to protect the privacy and integrity of the transmitted messages. Although the SPREAD idea is originally proposed for a MANET environment where it finds its full applicability and effectiveness, the deployment of the SPREAD idea is not limited to a MANET. The basic idea, the system architecture, as well as the research results regarding the secret sharing system and share allocation we presented in the first four chapters, are full applicable for a wired network. However, the topology of a wired network (Internet)

is more stable and the available network bandwidth is more compared with a MANET. The broadcasting type of route discovery is not necessary a good solution in wired network. In addition, a MANET is usually deployed temporarily for a specific reason and would last only for a certain time period and at a limited scale, while the current Internet is well developed and has millions of hosts connected. The scalability and the distribution of the routing protocol are of more concern in the Internet environment. In this chapter, we expand the deployment of the SPREAD idea into wired networks by investigating the multipath routing in wired networks.

The routing protocols used in today's Internet are destination-based single shortest path algorithms. In between a single source-destination pair, normally the same shortest path will be used. The existing Internet routing protocols provide very limited multiple paths routing capability. Only when there exist multiple paths and those paths are of the same (or varies within certain range of) cost, the packets will be forwarded via multiple paths to the same destination, and this is mainly done for load balancing, congestion control, and reliability.

Obviously, when transmitting a secret message across the network, if all the message shares take the same path, a hacker can intercept the whole message at any intermediate node on that path. The expected security from secret sharing scheme is actually lost. To enforce the distribution of the secret while they are transmitted across the network, we need to transmit the multiple shares via multiple independent paths so that, except the destination node, no intermediate node can collect T or more shares.

How to find the desired multiple paths then is the key implementation issue in the deployment of the SPREAD idea in wired networks. A closely related topic is the long

studied k-shortest path problem. Generally speaking, the *k*-shortest paths problem is to list the *k* paths connecting a given source-destination pair in the digraph with minimum total cost. Effective algorithms have been developed to enumerate the *k* shortest paths [72,73]. However, the paths found by the *k*-shortest path algorithms tend to share many links. The lack of independence limits the effectiveness of providing the security for the message shares. Algorithms that overcome the problem of path independence are ones that find disjoint paths between nodes. Ogier, et al. proposed a distributed algorithm for finding two disjoint paths of minimum total cost from each node to a destination [74]. The algorithm was further applied for Quality of Service (QoS) routing in connection-oriented networks that support calls with multiple QoS requirement [33]. Sidhu, et al. proposed a distributed algorithm that finds multiple disjoint paths to a destination [75]. Later they proposed a congestion control scheme based on this algorithm to alleviate congestion in networks under light to moderate loading [35]. Other researchers have studied the capability of multipath routing in aggregating bandwidth, reducing blocking probability, and increase the fault tolerance, etc. [15,32,36], which we had introduced in Chapter 1 of this dissertation.

The previous work mainly focused on the Quality of Service (QoS) issue of connection-oriented networks and the impact of the multipath routing on network capacity. Based on the relationship among the paths, there are two types of multipath: multi-service paths and multi-option paths. The first path type, multi-service paths, denotes paths between nodes that have different characteristics. For example, one path is for low delay service and another path for high throughput service. Since applications may have different QoS requirements, multi-service paths allow applications to choose

paths that best fit their communication demands. The second path type, multi-option paths, denotes the scenario that multiple paths between nodes are provided for the same type of service. For example, an algorithm may provide four multi-option paths for the high bandwidth service. Based on the ways the multiple paths are used, there are two prototypical multipath usage modes: using paths concurrently or once at a time. The concurrent multipath routing schemes include two modes: redundant and non-redundant. Redundant mode allows a certain number of paths to be fault while the whole system is still able to route the information flow continuously. In SPREAD scheme, by combining with the secret sharing scheme, we apply the multipath routing to a new dimension -- to improve network security. The maximal number of independent and concurrent paths is desirable in this approach. In next section, we present a distributed multipath routing algorithm that is able to find multiple disjoint paths between any source-destination pair. To maintain the independence of the paths, the source routing technique is used in this scheme.

7.2 Multipath Routing Algorithm

7.2.1 Notations

To facilitate the presentation of our algorithm, we first give some notation. A network is modeled as an undirected graph, $G=(V,E)$, with a finite set of nodes, V, and a finite set of links, E. A link in E, connecting a pair of nodes, x and y, in V, is denoted by (x,y). The cost associated with the link is represented by $c(x,y)$. We assume that the cost metric here represents the security levels of each link (node) as defined in chapter 6 section 2. The cost $c(x,y)$ is a positive number which is same in both directions.

There is an arbitrary node, t, in V, called the destination node. A path p from s to t is a sequence $(s,i,j,\ldots,k,t)$ of nodes of G such that, each link of $(s,i),(i,j),\ldots,(k,t)$ is in E.

The cost of the path, denoted as *c(p)*, is defined as the sum of its link costs. The path identifier of a path *p=(s,x,...,y,t)* is defined as *pid(p)=y*. A shortest path tree, *SPT*, rooted at destination *t,* is a subgraph of *G* such that the length of every path from each node *x* to *t* is minimized. Links in the tree are called *tree* links while the links not in the tree are called *nontree* links. Let *SPT(x,t)* denote the unique path from *x* to *t* in *SPT*. If *y* is on the path *SPT(x,t), y* is defined to be *downtree* of *x*, and *x* is *uptree* of *y*. For a node *x*, if there exist a link *(x,y), y* is called a *neighbor* of *x*. Let *nbr(x)* denote the set of neighbors of *x*. If *y∈nbr(x)* and *y* is uptree of *x*, *y* is a *uptree neighbor* of *x*. Let *upnbr(x)* denote the set of uptree neighbors of *x*. If *y∈nbr(x)* and *y* is downtree of *x*, *y* is *parent* of *x*. Let *parent(x)* denote the parent of *x*. If *y∈nbr(x)* and *(x,y)* is a nontree link, *y* is called a *horizontal* neighbor of *x*.

As we assume a big enough buffer at the destination so that the delay variation among shares could be absorbed, in the algorithm, we relax one of the desired properties that the paths should be of roughly equal length. In stead we set a constant parameter *COST_BOUND* to avoid too high cost path. A *(s,t*) path with cost greater than *COST_BOUND* c(SPT(s,t))* will not be accepted.

7.7.2 Distributed Algorithm

The distributed multipath routing algorithm described here is able to find for each node *x* a set of disjoint path to destination node *t*. Initially, each node *x* in graph *G* has an empty path set *Px*, which is used to store the disjoint paths from *x* to *t*, and an empty path set *Cx*, which is used to store candidate paths also from *x* to *t* but not disjoint from paths in *Px*. The elements in *Px* and *Cx* have the same structure *{pid,cst,path}*, where *path* is the node sequence of the path; *pid* is the path identifier; and *cst* is the cost of the path.

Here we define *Px* and *Cx* as ordered sets, with all the elements ordered by *cst* in increasing order. The order of the elements is always kept when elements are added or removed.

We assume that each node has the knowledge of its parent, uptree neighbors, horizontal neighbors and the costs to them. We also assume that no topological changes occur during the path finding procedure.

Two types of messages are used to advertise the path information. The general form of the message is *msg{mtype,nid,pid,cst,path},* where *mtype* is the message type, either 0 or 1; *nid* is the identifier of the node sending the message; *path* is node sequence of the path to be advertised; *pid* is the path identifier of the path; *cst* is the cost of the path. The two types of message are of same structure but differ in the rule the messages are propagated.

The first phase of the algorithm is the propagation of the type-0 messages. The destination node *t* initializing the algorithm by sending *msg{0,t,∅,0,(t)}* to each of its neighbors. Each intermediate node *x*, upon receiving a type-0 message *msg{0,nid,pid,cst,path},* learns about a new path *q=(x)+path*. If the message is from its parent, *q* is the shortest path *SPT(x,t)* and is added to path set *Px*. Meanwhile, node *x* further propagates the type-0 message by sending *msg{0,x,(pid==∅)?x:pid,cst+c(x,parent(x)),q}* messages to all its uptree neighbors and horizontal neighbors. If the message is from horizontal neighbor, *q* is an alternative path to destination *t* and thus included into candidate path set *Cx*. The type-0 messages received from the horizontal neighbors are not propagated further. The propagation of

type-0 messages terminates at the leaf nodes of *SPT* after the leaf nodes have sent and received type-0 message on their horizontal neighbors.

Once having received the type-0 messages from its parent and all its horizontal neighbor(s), node *x* checks its candidate path set *Cx* for disjoint paths. For each path *p* in *Cx*, node *x* includes *p* into *Px* if *p* is disjoint from any other path in *Px*. When *p* is included into *Px*, it is removed from *Cx*.

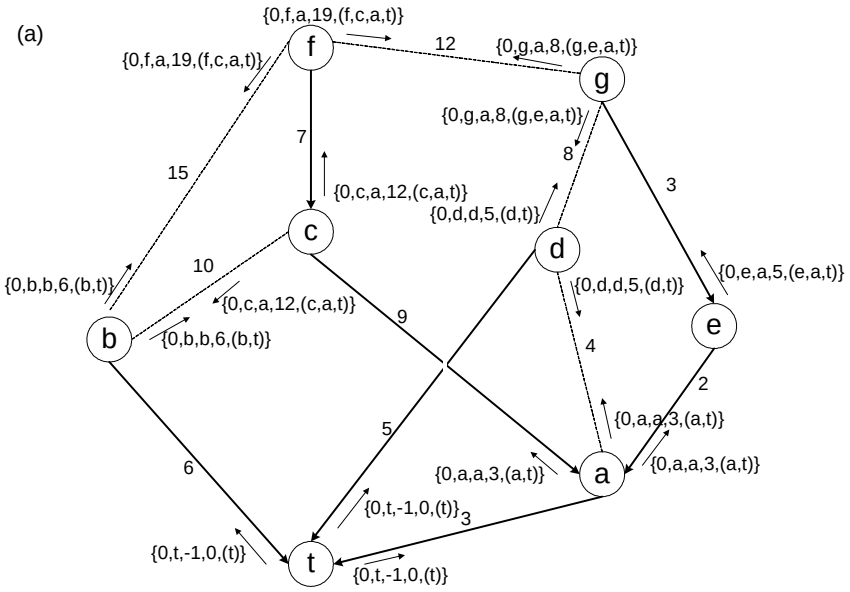
The second phase of the algorithm is to find more disjoint paths by exchange type-1 messages. The propagation of type-1 messages is initiated independently by each node at which alternative disjoint path(s) is found at the end of phase 1 (*SPT(x,t)* is not considered as an alternative path). For each alternative path *p*, node *x* form a type-1 message *msg{1,x,p.pid,p.cst,p)* and send it to all its type-1 eligible neighbors. A node *v* is defined as a type-1 eligible neighbor of *x* for path *p* if it satisfies the following conditions: (1) $v \in nbr(x)$; (2) $v \notin p$; (3) $v \notin upnbr(x)$.

Upon receiving a type-1 message *msg{1,nid,pid,cst,path},* node *x* learns about a new path *q=(x)+path*. Due to the distribution of the computation, node *x* may receive type-1 messages before its phase 1's path finding procedure finishes. If this is the case, node *x* will buffer the received type-1 messages until the phase 1 processing finishes. Node *x* includes *q* in *Px* if *q* is disjoint from any other path in *Px* of lower cost. If *x* includes *q* in *Px*, it excludes from *Px* every path that intersects with *q*. The path(s) removed from *Px,* and *q*, when not included in *Px*, are considered to be included in *Cx*. Node *x* includes path *q=(x,v,...,y,t)* in *Cx* if *q* is the cheapest path through *v* or it is the cheapest path with *pid=y* in *Cx*. Node *x* checks *Cx* for possible disjoint path(s) whenever there is a path removed from *Px*.

Whenever a new path *p* is added to *Px*, node *x* forms a type-1 message *msg{1,x,p.pid,p.cst,p.path}* and send it to all its type-1 eligible neighbors.

The propagation of type-1 messages terminates when no disjoint path is added to any path set *Px* or no node can send a type-1 message to any of its neighbors. At this time, each node has found a set of disjoint paths to the destination node *t*.

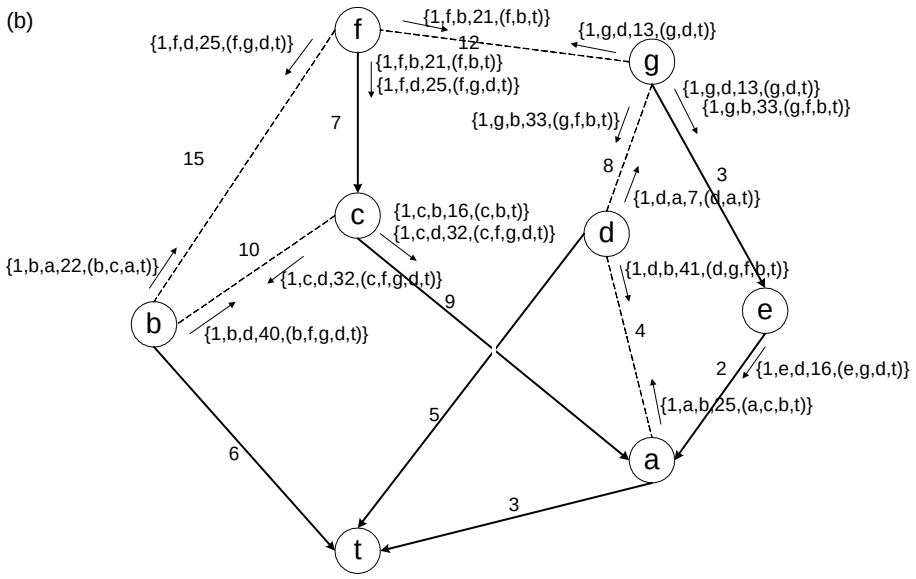
7.2.3 An Example



At the end of phase 1, Px at each node:
Pa={a,3,(a,t)},{d,9,(a,d,t)}
Pb={b,6,(b,t)},{a,22,(b,c,a,t)}
Pc={a,12,(c,a,t)},{b,16,(c,b,t)}
Pd={d,5,(d,t)},{a,7,(d,a,t)}
Pe={a,5,(e,a,t)}
Pf={a,19,(f,c,a,t)},{b,21,(f,b,t)}
Pg={a,8,(g,e,a,t)},{d,13,(g,d,t)}

Figure 7-1. Inllustration of the multipath routing algorithm. (a) type-0 message exchange and paths found in phase 1. (b) type-1 message exchange and paths found in phase 2

Figure 7-1 shows an example to illustrate how the algorithm works. Graph *G* represents an arbitrary network of 8 nodes and 12 links. The shortest path tree *SPT,*



At the end of phase 2, Px at each node:
Pa={a,3,(a,t)},{d,9,(a,d,t)},*{b,25,(a,c,b,t)}*
Pb={b,6,(b,t)},{a,22,(b,c,a,t)},*{d,40,(b,f,g,d,t)}*
Pc={a,12,(c,a,t)},{b,16,(c,b,t)},*{d,32,(c,f,g,d,t)}*
Pd={d,5,(d,t)},{a,7,(d,a,t)},*{b,41,(d,g,f,b,t)}*
Pe={a,5,(e,a,t)},*{d,16,(e,g,d,t)}*
Pf={a,19,(f,c,a,t)},{b,21,(f,b,t)},*{d,25,(f,g,d,t)}*
Pg={a,8,(g,e,a,t)},{d,13,(g,d,t)},*{b,33,(g,f,b,t)}*

Figure 7-1. Continued

rooted at an arbitrary destination node *t*, superimposes on *G*. Directed solid lines mark the tree links with arrows pointing from uptree to downdree. The nontree links are marked by dotted lines.

Figure 7-1(a) illustrates the exchange of type-0 messages. It is observed that by the end of phase 1, each node finds its shortest path to the destination node, which consists of only tree links. Each node also finds alternative disjoint path(s), which contains just one nontree link. Figure 7-1(b) illustrates the exchange of the type-1 messages. By exchanging type-1 messages, each node expands its disjoint path set by adding paths consisting of more than one nontree links. The disjoint paths found for each node at the end of phase 1 and phase 2 are listed respectively.

7.3 Performance Evaluation

7.3.1 Path Finding Capability

We conduct a simulation study based on our algorithm and compare it with the algorithm proposed in [75]. The simulation is performed on 8 arbitrarily chosen networks. The first network is the one shown as the example in this chapter. The rest 7 are all 20-node networks but with different parameters, as shown in the table. The notation *(M,D,d)* represents a network of *M* nodes with diameter *D* and average node degree *d*. All the links in the networks are assigned arbitrary link cost. The algorithm in [75] is referred to as A while ours as B. Notation U refers to the situation where our algorithm is performed on the same network but all the links are considered as of unit cost (i.e., we use the hop count as the cost metric).

It is observed from the simulation results that our algorithm is able to find more disjoint paths compared with algorithm A, which is most desirable in our approach. However, our algorithm reaches this by exchanging more messages, as we expect. Column 5 shows the average number of messages required per path. Although our algorithm needs slightly more messages per path, the value in such range of 2, 3 or 4 messages per path is still quite acceptable.

In general, the number of disjoint paths exist in the network is solely dependent on the network topology, while the number of disjoint paths found by the algorithm also depends on the link cost assignment. Unreasonable link cost assignment may decrease the number of paths found. It is observed from the simulation results that the algorithm tends to find more disjoint paths when we treat all the links as of unit cost, corresponding to the cost structure independent of traffic. This is a useful approach when we need as many as possible disjoint paths while the cost of the link is not a major concern. For example, if

we need to deliver a message to a destination with high security, we may tolerate certain delay, therefore, we would use as many disjoint paths as possible to decrease the probability of interception and recovery of the original message to be secured.

Network (M,D,d)		Total # Path	Total # Msg	Msg / Path	Path/ Node Pair
(8,3,3)	*A*	138	180	1.30	2.46
	B	151	259	1.72	2.70
	U	149	251	1.68	2.66
(20,7,2)	*A*	422	448	1.06	1.11
	B	422	448	1.06	1.11
	U	422	448	1.06	1.11
(20,7,3)	*A*	775	1184	1.53	2.04
	B	788	1327	1.68	2.07
	U	845	1398	1.65	2.22
(20,6,3)	*A*	730	1117	1.53	1.92
	B	738	1301	1.76	1.94
	U	794	1401	1.76	2.09
(20,5.3)	*A*	714	1115	1.56	1.88
	B	747	1316	1.76	1.97
	U	783	1370	1.75	2.06
(20,5,4)	*A*	926	1788	1.93	2.44
	B	1101	2774	2.52	2.90
	U	1124	2850	2.54	2.96
(20,4,4)	*A*	1003	1799	1.79	2.64
	B	1130	3063	2.71	2.97
	U	1173	3041	2.59	3.09
(20,4,5)	*A*	1091	2390	2.19	2.87
	B	1288	4661	3.62	3.39
	U	1416	4922	3.48	3.73

Table 7-1. Simulation results of multipath routing protocol

7.3.2 Performance on Security Improvement

In this section we present the simulation results to show the effectiveness of the SPREAD scheme in terms of the enhancement to the data confidentiality. Five types of 20-node networks, with the node degree equals to 4, 5, 6, 7, 8 respectively, are evaluated.

For each type of network, the simulation results are averaged over 20 random networks which are generated using a random graph generator based on Waxman's generator [76]. Two sets of simulation are executed. In the first set, each node is assumed to be independently and equally likely compromised with probability 0.152. In the second set, we assume nodes with different probability be compromised. The probability that a node might be compromised is selected from 4 values: 10% of nodes with probability 0.50 being compromised, 30% of nodes with probability 0.20, 40% of nodes with probability 0.10, and 20% of nodes with probability 0.01. For both sets, we use the proposed link cost function to define the link cost based on the node security level so that the most secure paths are selected. For each network, we run the multipath routing algorithm to find the maximal most secure paths between any source-destination pair which is not directly connected. Then we calculate the message interception ratio when different number of paths ($m=1,2,3,4,5$) is used.

Number of Nodes	**20**	**20**	**20**	**20**	**20**
Node Degree	**4**	**5**	**6**	**7**	**8**
Average Diameter	**4.25**	**3.80**	**3.40**	**3.15**	**3.00**

Table 7-2. Network parameters of the simulated networks

Table 7-2 summarizes some parameters of the simulated networks, including the node degree and the average network diameter.

Figure 7-2 shows the probability that multiple node disjoint paths are found by the distributed multipath routing algorithm in the simulated networks. We observe that for a well connected network (e.g., node degree equal to or larger than 4), the chances to find 2

or more paths are pretty high, in both simulation sets. Since our SPREAD scheme depends on the availability of multiple paths, the existence of such multiple paths and the capability of finding such paths justify the feasibility of our scheme.

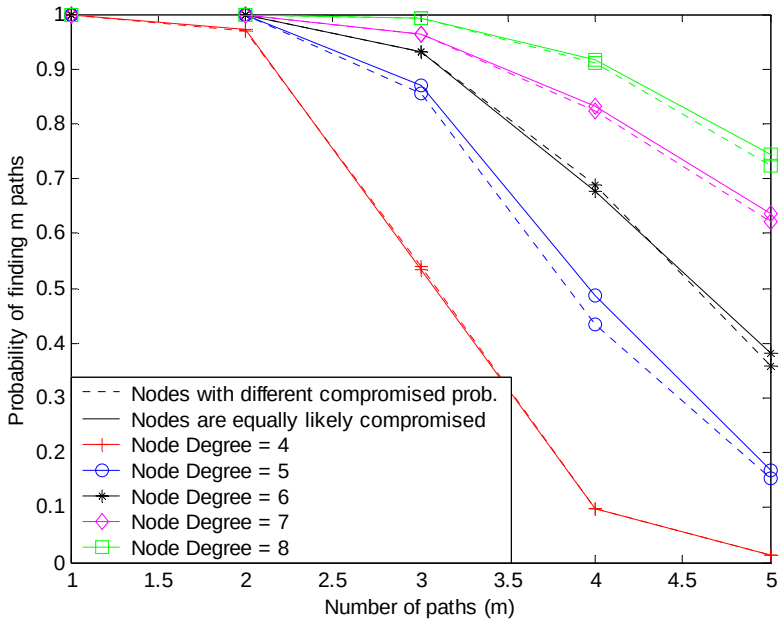
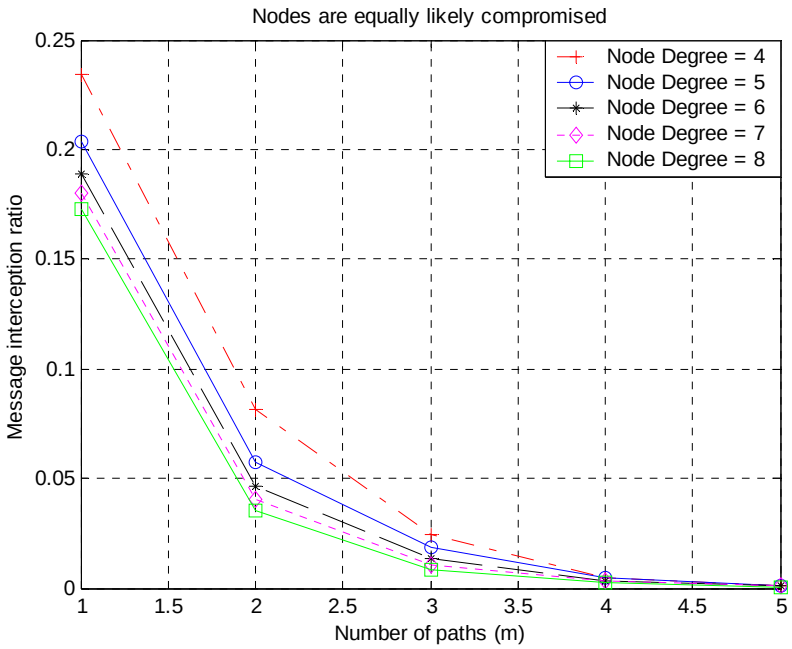


Figure 7-2. Probability of finding multiple paths.

Figure 7-3 shows the probability that the message might be compromised when different number of paths is used. Figure 7-3(a) is for simulation set 1 (i.e., each node is equally likely compromised). Figure 7-3(b) is for the simulation set 2 in which nodes have different probability being compromised. It is clear that, in both sets, the message interception ratio drops significantly, actually exponentially, with the increasing of the number of paths used. This result verifies the effectiveness of our SPREAD idea. Notice that the message interception ratio here is the additional security achieved on top of the cryptographic scheme. The adversary still needs to decrypt the message after intercepting all the necessary shares.

(a)



(b)

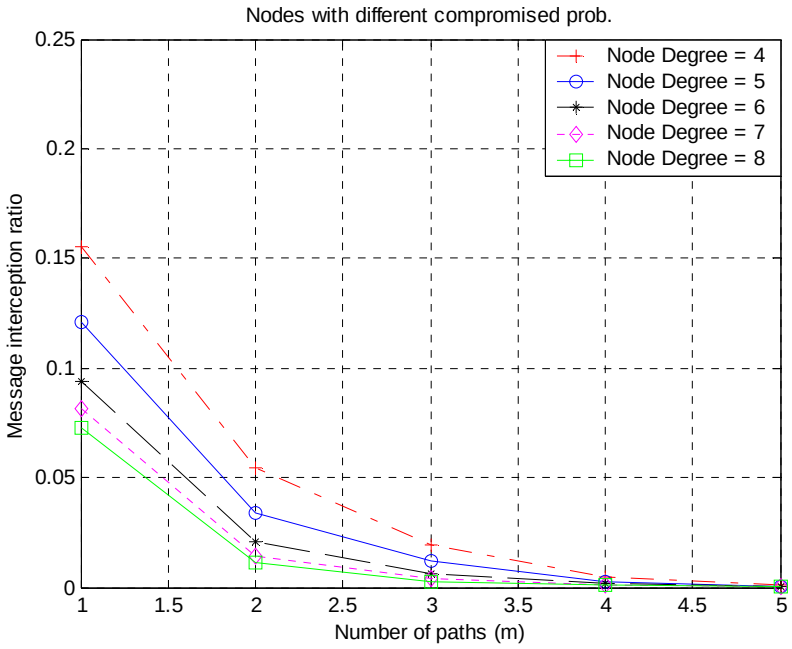


Figure 7-3. Message interception probability. (a) Nodes are equally likely compromised. (b) Nodes are compromised with different probabilities

The average node compromised probability in simulation set 1 and 2 is made equal. However, we observe significant differences in the achieved message interception probabilities in figure 7-3(a) and 7-3(b). The simulation set 2 actually achieves better security than simulation set 1. This is because when nodes are equally likely to be compromised, the multipath routing algorithm actually finds the minimum hop paths. While when nodes have different security levels, by incorporating the security link cost function proposed in this paper, the multipath algorithm will find paths according to their security levels. Those paths are more secure thus achieve better security. This trend is clearer in the networks where node degree is higher, which basically provide more choices from which the routing algorithm can select more secure paths.

7.4 Summary

Multipath routing algorithms and protocols are a key implementation issue in SPREAD. In this chapter, investigate the deployment of SPREAD in wired network by developing a distributed multipath routing algorithm suitable for a wired network. The algorithm takes path independence, path quantity, as well as path security level into consideration. The simulation shows that with comparably low complexity, the proposed algorithm is able to find, for each source-destination pair in the network, a set of disjoint paths. The proposed algorithm is compared with another disjoint path finding algorithm and the result shows that our algorithm has better performance in terms of desired properties by SPREAD. The simulation results also show that the significant improved security can be achieved by SPREAD.

CHAPTER 8
CONCLUSIONS

An ad hoc network is a self-configurable, self-organizing, infrastructureless multi-hop mobile wireless network. The concept of an ad hoc network allows a whole set of networking applications that can benefit from the mobility and dynamics inherent with ad hoc networks. However, it is also this very appealing feature that poses new challenges in design and implementation of an ad hoc network. Many traditional networking technologies and solutions will not be fully applicable in an ad hoc network. Some kind of enhancement or modification is necessary to make them working well in an ad hoc environment.

In this dissertation, we address the security solutions in an ad hoc network. We propose and study a Secure enhancement Protocol for REliable dAta Delivery (SPREAD) to enhance the data confidentiality service in an ad hoc network. Combining with the secret sharing principle, the SPREAD scheme takes a multipath routing approach to distribute a secret into network while transmitting data across an insecure network. Three major design issues of the implementation of SPREAD are elaborated in this dissertation. Our investigation and simulation show that by SPREADing traffic among multiple paths (nodes) in the network, SPREAD is an effective way to enhance the data confidentiality (in terms of the message compromise probability), particularly against the compromised nodes problem in an ad hoc network. In addition, the SPREAD scheme can also be designed to improve the reliability of the transmitted data, against the unstable wireless link problem and frequent topological changes. However, to this end, a more efficient

MAC layer protocol that is able to resolve the “route coupling” problem [38] is desirable in order to fully enjoy the performance benefit from multipath routing.

A few remarks are in order. First, the SPREAD scheme only considers the security when messages are transmitted across the network, assuming the source and destination are trusted. Secondly, the SPREAD scheme cannot address the confidentiality alone, it only statistically enhances such service. For example, it is still possible for adversaries to compromise all the shares (e.g., by collusion). Finally, the SPREAD can be made adaptive in the sense that the source node could make final decision whether a message is delivered at certain time instant according to the security level and the availability of multiple paths. Moreover, the chosen set of multiple paths may be changed from time to time to avoid any potential capture of those multiple shares by adversaries.

The work described in this dissertation is theoretical in the sense that it only presents the primitive prototype needed to implement the SPREAD idea. A lot more design and implementation details need to be resolved to make SPREAD an actual operational protocol. For example, at which layer should we define the SPREAD protocol, the application layer or the network layer? Except the source routing technique, can we support the multipath routing in a distributed fashion? What is the proper interface with other protocols, such as TCP/UDP? How do we incorporate synchronization, flow control or error control in the simultaneous multipath routing? And so on. In addition, as the MANET networking technology is still under active research, we do not yet know what is a ``typical" application scenario, neither do we have the practical knowledge about its behavioral dynamics from actual deployments. Once these information become available, many solutions will likely need to be modified.

LIST OF REFERENCES

1. W. Lou, Y. Fang, “A survey on wireless security in mobile ad hoc networks: challenges and available solutions”, in Ad Hoc Wireless Networking, Kluwer, May 2003

2. L. Zhou and Z. J. Haas, “Securing ad hoc networks”, IEEE Network Magazine, 13(6):24-30, November/December 1999

3. J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, “Providing robust and ubiquitous security support for manet”, 9th International Conference on Network Protocols (ICNP’01), Riverside, CA, November, 2001

4. J-P. Hubaux, L. Buttyan and S. Capkun, “The quest for security in mobile ad hoc networks”, The ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHOC'01), Long Beach, CA, October 2001

5. Y.-C. Hu, D. B. Johnson and A. Perrig, “SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks”, 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA'02), Callicoon, NY, June 2002

6. Y.-C. Hu, A. Perrig and D. B. Johnson, “Ariadne : a secure on-demand routing protocol for ad hoc networks”, the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom’02), Atlanta, GA, September 2002.

7. P. Papadimitratos and Z. J. Haas, “Secure routing for mobile ad hoc networks”, Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS’02), San Antonio, TX, January 2002

8. H. Yang, X. Meng and S. Lu, “Self-organized network-layer security in mobile ad hoc networks”, ACM Workshop on Wireless Security (WiSe'02), Atlanta, GA, September 2002.

9. S. Marti, T. Giuli, K. Lai and M. Baker, “Mitigating routing misbehavior in mobile ad hoc networks”, the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom’00), Boston, MA, August 2000.

10. L. Buttyan and J.-P. Hubaux, “Enforcing service availability in mobile ad hoc networks”, The ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHOC’00), Boston, MA, August 2000

11. S. Buchegger and J.-Y. Le Boudec, “Performance analysis of the CONFIDENT protocol”, The ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHOC’02), Lausanne, Switzerland, June 2002.

12. Y. Zhang, W. Lee and Y. Huang, “Intrusion detection techniques for mobile wireless networks”, ACM Wireless Networks Journal, 9(5):545-556, September 2003.

13. N. Borisov, I. Goldberg, D. Wagner, “Intercepting mobile communications: the insecurity of 802.11”, Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking (MobiCom’01), Rome, Italy, July 2001,

14. G. J. Simmons, “An Introduction to Shared Secret and/or Shared Control Schemes and The Application”, in Contemporary Cryptology: The Science of Information Integrity, IEEE Press, pp.441-497, 1992.

15. E. Gustafsson, G. Karlsson, “A literature survey on traffic dispersion”, IEEE Networks, 11(2):28-36, Mar/Apr 1997.

16. A. Tsirigos, Z. J. Haas, “Multipath routing in mobile ad hoc networks or how to route in the presence of frequent topology changes”, IEEE Military Communications Conference (Milcom’01), McLean, VA, October 2001

17. A. Tsirigos, Z. J. Haas, “Multipath routing in the presence of frequent topological changes”, IEEE Communication Magazine, 39(11):132-138, November 2001

18. A. Nasipuri, R. Castaneda, S. R. Das, “Performance of multipath routing for on-demand protocols in mobile ad hoc networks”, Mobile Networks and Applications, 6(4):339-349, 2001

19. D. Ganesan, R. Govindan, S. Shenker, D. Estrin, “Highly-resilient, energy-efficient multipath routing in wireless sensor networks,” Mobile Computing and Communications Review, 5(4):10-24, 2002

20. D. B. Johnson, D. A. Maltz, Y-C. Hu, J. G. Jetcheva, The dynamic source routing protocol for mobile ad hoc networks, *IETF Internet Draf*t, draft-ietf-manet-dsr-06.txt, Nov 2001

21. W. Diffie and M. Hellman, “New direction in cryptography”, IEEE Transaction on Information Theory, 22(6):644-654, November 1976

22. W. Stallings, Cryptography and Network Security: Principles and Practice, 2nd edition, Prentice Hall, 1999

23. K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields and E. M. Belding-Royer, “A secure routing protocol for ad hoc networks”, the 10th IEEE International Conference on Network Protocols (ICNP’02), Paris, France, November 2002

24. A. Perrig, R. Canetti, D. Tygar, D. Song, “Efficient authentication and signature of multicast streams over lossy channels”, IEEE Symposium on Security and Privacy, Oakland, CA, May 2000

25. A. Weimerskirch and G. Thonet, “A distributed light-weight authentication model for ad-hoc networks”, Lecture Notes in Computer Science, No. 2288, pp.341-354, 2002

26. M. D. Corner and B. D. Noble, “Zero-interaction authentication”, the 8th ACM International Conference on Mobile Computing and Networking (MobiCom'02), pp. 1-11, Atlanta, GA, September 2002

27. S. Jiang, N. H. Vaidya and W. Zhao, “A dynamic mix method for wireless ad hoc networks”, IEEE Military Communications Conference (Milcom'01), pp. 873-877, McLean, VA, October 2001.

28. S. Jiang, N. Vaidya and W. Zhao, “Preventing traffic analysis in packet radio networks”, DARPA Information Survivability Conference & Exposition II (DISCEX'01), vol.2, pp. 163-158, Anaheim, CA, June 2001

29. N. F. Maxemchuk, “Dispersity routing”, International Conference on Communications (ICC ’75), pp.41.10-41.13, San Francisco, CA, June 1975.

30. N. F. Maxemchuk, “Dispersity routing in high speed networks”, Computer Networks and ISDN Systems, 25(6):645-661, 1993.

31. N. F. Maxemchuk, “Dispersity routing on ATM networks”, IEEE INFOCOM’93, vol.1, pp.347-57, San Francisco, CA, Mar 1993.

32. Johnny Chen, “New Approaches to Routing for Large-Scale Data Networks”, Ph.D Dissertation, Rice University, 1999

33. N. Taft-Plotkin, B. Bellur, and R. Ogier, “Quality-of-Service Routing Using Maximally Disjoint Paths”, 1999 Seventh International Workshop on Quality of Service, London, UK, June 1999

34. T. T. Lee, S. C. Liew, “Parallel Communication for ATM Network Control and Management”, IEEE Globecom’93, vol.1, pp.442-446, Houston, TX, December 1993

35. D. Sidhu, S. Abdallah, R. Nair, “Congestion Control in High Speed Networks via Alternative Path Routing”, Journal of High Speed Networks, 2(2):129-144, 1992.

36. A. Banerjea, “On the Use of Dispersity Routing for Fault Tolerant Realtime Channels”, European Transactions on Telecommunications, 8(4):393-407, July/August 1997

37. S-J. Lee, M. Gerla, “Split multipath routing with maximally disjoint paths in ad hoc networks”, International Conference on Communications (ICC’01), Helsinki, Finland, June 2001.

38. M.R. Pearlman, Z.J. Haas, P. Sholander, S. S. Tabrizi, “On the impact of alternate path routing for load balancing in mobile ad hoc networks”, The ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHOC’00), Boston, MA, August 2000.

39. K. Wu, J. Harms, “Performance study of a multipath routing method for wireless mobile ad hoc networks”, 9th international symposium on modeling, analysis and simulation of computer and telecommunication system (MASCOTS’01), Cincinnati, Ohio, August 2001.

40. J. Yang, S. Papavassiliou, “Improving network security by multipath traffic dispersion”, IEEE Military Communications Conference (Milcom’01), McLean, VA, October 2001

41. P. Papadimitratos, Z.J. Haas, E. G. Sirer, “Path set selection in mobile ad hoc networks”, The ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc’2002), EPFL Lausanne, Switzerland, June 2002,

42. A. Shamir, “How to Share a Secret”, Communications of the ACM, 22(11):612-613, November 1979

43. G. R. Blakley, “Safeguarding Cryptographic Keys”, Proc. AFIPS 1979 National Computer Conference , vol.48, pp.313-317, New York, NY, June 1979

44. T.-C. Wu, T.-S. Wu, “Cheating detection and cheater identification in secret sharing schemes”, IEE proc. Computers and Digital Techniques, 142(5):367-369, September 1995

45. W. Lou, Y. Fang, “Predictive caching strategy for on-demand routing protocols in ad hoc networks”, Wireless Networks, 8(6):671-679, November 2002

46. Bruce Schneier, Applied Cryptography, 2nd edition, Chapter 23, John Wiley & Sons, 1996

47. T. Cormen, C. Leiserson, R. Rivest, C Stein, Introduction to algorithms, 2nd edition, the MIT Press, 2001

48. C. Charnes, J. Pieprzyk, R. Safavi-Naini, “Conditional Secure Secret Sharing Schemes with Disenrollment Capability”, 2nd ACM Conference on Computer and Communications Security, pp89-95, Fairfax, Virginia, USA, November 1994

49. C. E. Perkins, P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers”, Computer Communication Review, 24(4):234-244, October 1994.

50. S. Murphy, J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks", ACM Mobile Networks and Applications, Special issue on Routing in Mobile Communication Networks, 1(4):183-197, October 1996.

51. C. E. Perkins, E. M. Belding-Royer, S. R. Das, Ad hoc on-demand distance vector (AODV) routing, IETF Internet draft, draft-ietf-manet-aodv-09.txt, November 2001

52. E. M. Royer, C-K Toh, "A review of current routing protocols for ad hoc mobile wireless networks", IEEE Personal Communications, 6(2):46-55, April 1999

53. J. Broch, D. Maltz, D. Johnson, Y-C. Hu, J. Jetcheva, "A performance comparison of multi-hop wireless ad hoc network routing protocol", The 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'98), pp. 85-97, Dallas, TX, October 1998

54. P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, M. Degermark, "Scenario-based performance analysis of routing protocols for mobile ad hoc networks", The 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'99), pp.195-206, Seattle, WA, August 1999

55. S. R. Das, etc., "Comparative performance evaluation of routing protocols for mobile ad hoc networks", The 7th International Conference on Computer Communication and Networks (IC3N), pp. 153-161, Lafayette, LA, October 1998

56. D. A. Maltz, J. Broch, J. Jetcheva, D. B. Johnson, "The effects of on-demand behavior in routing protocols for multihop wireless ad hoc networks", IEEE Journal on Selected Areas in Communications, 17(8):1439-1453, August 1999

57. M. K. Marina, S. R. Das, "Performance of routing caching strategies in dynamic source routing", 2001 International conference on distributed computing systems (ICDCS), Phoenix, AZ, April 2001

58. Y-C. Hu, D. B. Johnson, "Caching strategies in on-demand routing protocols for wireless ad hoc networks", The 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'00), Boston, MA, August 2000

59. C. E. Perkins, E. M. Royer, S. R. Das, M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks", IEEE Personal Communications, 8(1):16-28, February 2001

60. M. Takai, L. Bajaj, R.Ahuja, R. Bagrodia, M. Gerla, "GloMoSim: a scalable network simulation environment", Technical Report 990027, UCLA, Computer science department, 1999

61. M. Takai, J. Mertin, R. Bagrodia, "Effects of wireless physical layer modeling in mobile ad hoc networks", ACM Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'01), Long Beach, CA, October 2001

62. Z. Ye, S. V. Krishnamurthy, S. K. Tripathi, “A framework for reliable routing in mobile ad hoc networks”, IEEE INFOCOM’03, vol. 1, pp. 270-280, San Francisco, CA, March-April 2003

63. V. D. Park, M. S. Corson, “A highly adaptive distributed routing algorithm for mobile wireless networks”, IEEE INFOCOM’97, pp. 1405-1413, Kobe, Japan, April 1997

64. E. Ayanoglu, I. Chih-Lin, R. D. Gitlin, J. E. Mazo, “Diversity coding for transparent self-healing and fault-tolerant communication networks”, IEEE Transaction on Communications, 41(11):1677-1686, November 1993

65. N. Gogate, S. S. Panwar, “Supporting video/image applications in a mobile multihop radio environment using route diversity”, IEEE International Conference on Communications (ICC’99), Vancouver, Canada, June 1999

66. Z. J. Haas, M. R. Pearlman, “Improving the performance of query-based routing protocols through diversity injection”, IEEE Wireless Communications and Networking Conference (WCNC’99), New Orleans, LA, September 1999

67. C. Irvine, T. Levin, “Quality of security service”, Proc. of New Security Paradigms Workshop 2000, Cork, Ireland, September 2000

68. S. Yi, P. Naldurg, R. Kravets, “Security-aware ad-hoc routing for wireless networks”, Report no. UIUCDCS-R-2001-2241, Department of computer science, University of Illinois at Urbana-Champaign, August 2001

69. S. Chen and K. Nahrstedt, “An overview of quality of service routing for next-generation high-speed networks: problems and solutions”, *IEEE Networks*, 12(6):64-79, November/December 1998

70. R. Bhandari, Survivable Networks – Algorithms for diverse routing, Kluwer Academic Publisher, 1999.

71. R. Bhandari, “Optimal diversity routing in telecommunication fiber networks”, IEEE INFOCOM’94, pp. 1498-1508, Toronto, Canada, June 1994

72. J. Y. Yen, “Finding the k shortest loopless paths in a network”, Management Science, 17(1971):712-716, 1971

73. D. Eppstein, “Finding the k shortest paths”, SIAM J. Computing 28(2):652-673, 1999

74. R. Ogier, V. Rutenburg, N. Shacham, “Distributed algorithms for computing shortest pairs of disjoint paths”, IEEE Transaction on Information Theory, 39(2):443-455, March 1993

75. D. Sidhu, R. Nair, S. Abdallah, “Finding disjoint paths in networks”, *Proc. of ACM SIGCOMM’91*, pp. 43-51, Zurich, Switzerland, September1991

76. B. M. Waxman, “Routing of multipoint connections,” IEEE Journal on Selected Areas in Communications, 6(9):1617-1622, December 1988

BIOGRAPHICAL SKETCH

Wenjing Lou received her B.E. degree and M.E. degree in Computer Science and Engineering from Xi'an Jiaotong University, China, in 1993 and 1996 respectively. She received the M.A.Sc. degree in Computer Communications from Nanyang Technological University, Singapore, in 1998. From December 1997 to July 1999, she worked as a Research Engineer in Network Technology Research Center, Nanyang Technological University. From August 1999 to August 2000, she was a Ph.D. student in the department of Electrical and Computer Engineering at New Jersey Institute of Technology. From August 2000 to August 2003, she worked toward a Ph.D. degree in Computer Engineering at the University of Florida. Her research interests include wireless ad hoc networks, routing protocols in both wired and wireless networks, network security, and traffic management in ATM networks. Wenjing Lou is a member of Tau Beta Pi and a student member of the IEEE.