This is to certify that the
dissertation entitled

AUTOCONFIGURATION AND SECURITY FOR
WIRELESS NETWORKS

presented by

HONGBO ZHOU

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Computer Science & Engineering

Major Professor's Signature

July 20, 2005

Date

AUTOCONFIGURATION AND SECURITY FOR WIRELESS NETWORKS

By

Hongbo Zhou

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

2005

ABSTRACT

Autoconfiguration and Security for Wireless Networks

By

Hongbo Zhou

A mobile ad-hoc network (MANET) is a temporary wireless network composed of mobile nodes without any infrastructure. It has many advantages over a hardwired network or a wireless LAN and can be applied to many scenarios where it is expensive or impossible to build an infrastructure, such as a temporary network in meeting rooms, airports, stadiums, battlefields, search and rescue, and "smart transportation". However, prior to the practical deployment of the MANET, there are some problems to solve, among which are autoconfiguration and security. We proposed Prophet Address Allocation to allocate IP addresses automatically, which is efficient in a large-scale MANET. Improved with security mechanisms, Secure Prophet Address Allocation is able to guarantee the uniqueness of address allocation in presence of many kinds of attacks. With the deployment of autoconfiguration, a mobile node will change its address more frequently, so IP address handoff scheme is proposed to save the communication overhead caused by the address change. With the introduction of autoconfiguration, the assumption under the security framework in the MANET is changed. Thus, a multiple-key cryptography-based distributed certificate authority is proposed to build a security infrastructure in the MANET. Similar to the MANET, a sensor network is an

infrastructureless wireless network consisting of a large number of sensor nodes, in which autoconfiguration is an important issue to be solved. We proposed a reactive ID assignment for sensor networks, which is tolerant to packet loss and invulnerable to specific kinds of attacks. The advantages of these schemes are supported by our theoretical analysis, simulation results and prototype implementation.

ACKNOWLEDGEMENTS

I would like to thank my advisors, Dr. Matt W. Mutka and Dr. Lionel M. Ni, for their continuous support and help throughout the research project. It was definitely my great pleasure to conduct this interesting and challenging research under their guidance. Their broad and profound knowledge and effective instruction have given me a great help.

I am also very grateful to my academic committee members, Dr. Li Xiao, Dr. Abdol H. Esfahanian, and Dr. Zhengfang Zhou for their valuable advice and inspiring comments.

My family also contributes to my dissertation. I would like to thank my wife Yuhui Wang for her support during my graduate studies, and my daughter Julina Zhou for the joy she brings to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Introduction and Motivation

1.1.1 Mobile Ad-hoc Networks

A Mobile Ad-hoc Network (MANET) [1] is a temporary wireless network composed of mobile nodes, in which an infrastructure is absent. Compared to a hardwired network and a wireless LAN, it has many unique characteristics [2]:

(1) There is no infrastructure in the MANET. If two nodes are within transmission range of each other, they can communicate directly; otherwise, the nodes in between must forward the packets for them. Thus, every node in the network must act as a router, and the MANET is a multi-hop wireless network;

(2) All the nodes in the MANET are free to move arbitrarily, so the network topology keeps changing;

(3) Every node relies on a battery to provide the power. Thus, any network protocol should take power consumption into consideration in their design;

(4) Compared to a host in a hardwired network, a node in the MANET is more exposed to different kinds of attacks such as eavesdropping, DoS attacks, and IP spoofing attacks, because the physical link is a broadcast channel, and there is no infrastructure in the MANET.

Due to the abundance of mobile devices, the speed and convenience of deployment, and the independence of network infrastructure, a MANET will find its application in the scenario where it is expensive or impossible to build an infrastructure, such as:

(1) Military use (e.g. a network in the battlefield)

(2) Search and rescue;

(3) Vehicle-to-vehicle communication in intelligent transportation [3];

(4) Temporary networks in meeting rooms, airports, stadiums, etc.;

(5) Personal Area Networks connecting cell phones, laptops, smart watches, and other wearable computers.

1.1.2 Sensor Networks

A sensor network consists of a large number of sensor nodes that are engaged in environmental monitoring and wireless communications, simultaneously. It is similar to a MANET since both are infrastructureless multi-hop wireless networks. However, they are different in the architectures and data communication schemes. In a sensor network, the communication is data-centric instead of address-centric, which means the user is interested in the location and the data collected by the sensor node, but does not care about the address of the sensor node. Moreover, since the payload length in the data packet is usually small, it wastes bandwidth and power if the data are encapsulated in a TCP/UDP/IP packet. Thus, customized network protocols are adopted instead of TCP/IP to save the communication overhead and energy consumption, such as directed diffusion [4] described in Chapter 2.

Due to the low cost of a sensor node and the convenience of deployment, a sensor network has many applications, such as battlefield surveillance, precision agriculture, and wildlife study [5].

1.2 Problem Statement

Generally speaking, a MANET is an IP-based network. Thus, IP address assignment to mobile devices is one of the most important network configuration parameters. A mobile device cannot participate in unicast communications until it is assigned a free IP address (and the corresponding subnet mask). Since the IP address is regarded as identification and used in routing and forwarding, the IP addresses allocated must be unique.

If a MANET is connected to a hardwired network by a gateway, all the nodes in the MANET should have the same network address for simplicity of routing among them and the hardwired nodes. In other words, their addresses should be either private addresses in IPv4 or with the same special prefix in IPv6. Thus a mobile node may initiate communications with a hardwired node with the aid of NAT. As for communications initiated by the latter, mobile IP may be necessary, which is beyond the scope of our research topic.

For small scale MANETs, it may be simple and efficient to allocate free IP addresses manually. However, the procedure becomes difficult and impractical for a large-scale open system where mobile nodes are free to join and leave. Much effort has been spent on routing protocols for MANET in recent years, such as OLSR [6], FSR [7], DSR [8], and AODV [9], while research on automatic configuration of IP addresses (autoconfiguration [10]) for MANET is relatively less. Although there is a Working Group in IETF called Zeroconf [11], it mainly focuses on the environments such as small or home office and embedded systems.

Automatic address allocation is more difficult in a MANET environment than that in hardwired networks due to instability of mobile nodes, low bandwidth of wireless links,

openness of MANET, and lack of central administration. Therefore, more overhead occurs to avoid address conflict compared to the protocols for hardwired networks, such as DHCP [12] and SAA [13]. However, since address allocation is the first step toward the practical application of the MANET, it is worth further research effort.

Before discussing address allocation issues, several scenarios are described to illustrate the difficulty of the problem. In the simplest scenario, a mobile node joins and then leaves a MANET once, such as nodes A and B illustrated in Fig. 1.1. An unused IP address is allocated on its arrival and becomes free on its departure.

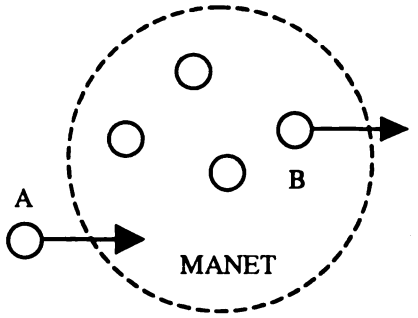


Figure 1.1 A node joins and leaves the MANET once

However, nodes are free to move arbitrarily during its session in the MANET. If one or more configured nodes go out of others' transmission range for a while, the network becomes partitioned as illustrated in Fig. 1.2 (a). When they approach each other, the partitions merge later. Because mobile nodes may not be aware of partitioning, they still use the previously allocated IP addresses. If a new node, say B, arrives at one partition and is assigned an IP address belonging to the other partition, say A's IP address, conflict happens when these two partitions merge as illustrated in Fig. 1.2 (b).

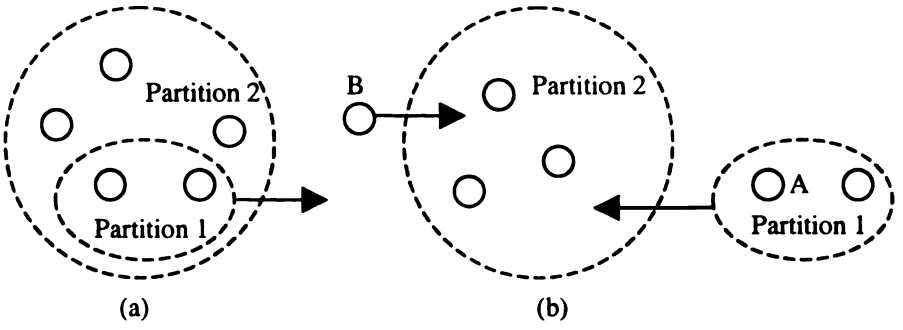


Figure 1.2 Network partitions and merges

Another scenario is when two separately configured MANETs merge, which is illustrated in Fig. 1.3. Because address allocation in one MANET is independent of the other, there may be some duplicate addresses in both of them. For example, node A in MANET 1 has the same IP address as node B in MANET 2. As a result, some (or all) nodes in one MANET may need to change their addresses.

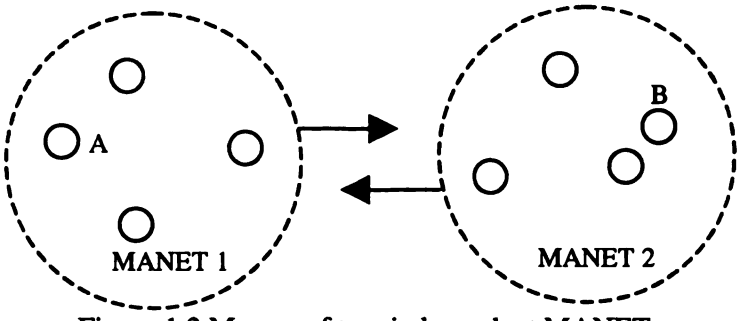


Figure 1.3 Merger of two independent MANETs

In another scenario, students are free to switch between a series of seminar rooms held at the same time. A mobile node leaves one MANET and then joins another MANET. This node could be regarded as the special case of the situation mentioned above because the single node could be viewed as a one-node partition.

The last scenario is fairly rare. Suppose there are two independent MANETs that are close to each other. A node in between decides to join a MANET nearby and functions as

a relay node, which leads to connection of the two MANETs. This is the same as merger of two independent MANETs.

In summary, a feasible autoconfiguration algorithm should handle the following three general scenarios:

Scenario A: A mobile node simply joins a MANET and then leaves it forever;

Scenario B: A MANET partitions and then the partitions merge later;

Scenario C: Two separately configured MANETs merge.

Moreover, the introduction of autoconfiguration in the MANET changes the underlying assumption in the security framework. When public cryptography is applied in the MANET, there must be a Certificate Authority to certify the binding of a node's public key and its IP address because the IP address is regarded as identification. In autoconfiguration, the identification itself is generated dynamically. Thus, the design of the security framework needs to take autoconfiguration into consideration.

Autoconfiguration is also important in the sensor network. The difference of between the autoconfiguration in a MANET and that in a sensor network is that in the former, every mobile node can initiate communication with each other, thus every node must have a globally unique ID. However, in a sensor network, locally unique IDs will suffice. For example, the directed diffusion communication paradigm assumes locally uniqueness of IDs. Thus, the three scenarios mentioned above do not need to be considered.

Although all the sensor nodes may be equipped with a locating device such as GPS, it is not adequate to simply use the location of a sensor node or the hash value of the location information as its ID because:

(1) The number of bits required to represent the location information in the address field may be large, and thus considerable power will be wasted;

(2) In a sensor network with high node density, the nodes that are close to each other may have the same location information due to the low resolution of the locating device;

(3) For the hash value of the location information, without comparison, it is still unknown if the hash values of adjacent nodes are different or not.

In summary, a specific autoconfiguration scheme for a sensor network is necessary.

1.3 Our Contribution

Our contribution to autoconfiguration and security in the MANET and sensor network can be summarized as follows:

(1) We proposed Prophet Address Allocation for the MANET, which outperforms other autoconfiguration schemes in terms of communication overhead and latency. It is efficient for a large-scale MANET;

(2) All the existing autoconfiguration algorithms assume a secure environment, and thus fail to work when there are malicious nodes present. Improved with security mechanisms, we proposed Secure Prophet Address Allocation to defeat many kinds of attacks on autoconfiguration;

(3) With the deployment of autoconfiguration, a mobile node will change its IP address more frequently and bring more communication overhead. We proposed IP address handoff scheme to save the communication overhead caused by address change;

(4) The autoconfiguration brings Sybil attacks to the MANET, which is fatal to prevalent threshold cryptography-based Distributed Certificate Authority (DCA) and impossible to defeat. We proposed multiple-key cryptography-based DCA scheme to implement a security framework in the MANET.

(5) We proposed a reactive ID assignment scheme for the sensor network, which is more efficient than proactive schemes and easy to integrate with directed diffusion communication paradigm. With additional mechanisms, it is tolerant to packet loss and invulnerable to specific attacks from malicious nodes.

In addition, we are studying the integration of autoconfiguration with other network protocols, such as routing protocols. The research is expected to motivate the research on autoconfiguration and security in the MANET and sensor network.

1.4 Structure of the Content

The rest of the dissertation is organized as follows. A literature review of existing autoconfiguration algorithms and security framework is outlined in Chapter 2. Chapter 3 describes Prophet Address Allocation for a large-scale MANET. Chapter 4 presents Secure Prophet Address Allocation for a MANET. The IP address handoff scheme is presented in Chapter 5. Chapter 6 describes multiple-key cryptography-based DCA scheme for the MANET. Chapter 7 presents the reactive ID assignment scheme for sensor networks. Chapter 8 concludes the dissertation and gives hint for the future work.

CHAPTER 2 RELATED WORK

This chapter introduces related work routing protocols, autoconfiguration, and security in a MANET and ID assignment in a sensor network.

2.1 Routing protocols for MANETs

Routing and forwarding are the most basic operations in the MANET. Many routing protocols are proposed for unicast, multicast, and broadcast communications.

2.1.1 Unicast routing protocols for MANETs

Generally speaking, unicast routing protocols can be divided into two classes: topology-based routing protocol and geographic-based routing protocols. Topology-based routing protocols can be further divided into the following categories according to their routing table management:

(1) Proactive routing protocol update routing tables periodically, such as OLSR and FSR;

(2) Reactive routing protocols build routing entries on demand, such as DSR and AODV;

(3) Hierarchical routing protocols combine both proactive and reactive operations, such as ZRP and LANMAR.

Geographic-based routing protocols assume that the source knows the location information of itself (with a locating device) and the destination (through location service), so the location information can be piggybacked in the IP packets, which are

forwarded to the nodes close to the destination hop by hop. Compared to topology-based routing protocols, geographic-based routing protocols seem to be good candidates for large-scale MANETs. Examples of geographic-based routing protocols include LAR and GRA.

(1) Optimized Link State Routing Protocol (OLSR, [6])

OLSR aims at large and dense MANETs. It is based on a Multipoint Relaying (MPR) flooding technique [10] to reduce the number of topology broadcast packets (The MPR flooding will be discussed in more detail in Subsection 2.1.4). The procedure of OLSR is:

(1) Every node broadcasts HELLO messages that contain one-hop neighbor information periodically. The TTL of HELLO messages is 1, so they are not forwarded by its neighbors. With the aid of HELLO messages, every node obtains local topology information;

(2) A node (also called selector) chooses a subset of its neighbors to act as multi-point relaying nodes for it based on the local topology information, which are specified in the periodic HELLO messages later. MPR nodes have two roles: 1) When the selector sends or forwards a broadcast packet, only its MPR nodes among all its neighbors forward the packet; 2) The MPR nodes periodically broadcast its selector list throughout the MANET (again, by means of MPR flooding). Thus every node in the network knows by which MPR nodes every other node could be reached. Note that 1) reduces the number of retransmissions of topology information broadcast, and 2) reduces the size of broadcast packet. As result, much more bandwidth is saved compared with original link state routing protocols.

(3) With global topology information stored and updated at every node, a shortest path from one node to every other node could be computed with Dijkstra's algorithm, which goes along a series of MPR node.

(2) Fisheye State Routing Protocol (FSR, [7])

FSR aims at large scale MANETs and MANETs with high mobility. The name comes from the special property of fish eyes. The fish gets a high-resolution picture about the object nearby, while the resolution decreases when the object moves farther. The fisheye state routing protocol adopts the same idea. The source only needs to know the general direction towards the destination far away. The intermediate nodes will correct the packet's movement on transit.

The procedure of FSR is:

(1) For a specific node, the whole network is divided into different scopes based on the distances (i.e., hops) of other nodes relative to it. For example, the nodes within distance of 2 hops are in the inner scope, and all the other nodes are in the outer scope;

(2) The link state updates are broadcast to the neighbors. However, the routing entries corresponding to the nodes in different scopes are sent at different frequencies: the routing entries towards the nodes in the inner scope are sent at the highest frequency, the other entries are sent at lower frequency. Therefore, the nodes nearby will receive more up-to-date link state updates, but the node far away may have less accurate link state information. The link state updates are not flooded throughout the MANET, but exchanged among neighboring nodes, which also provides the mechanism for neighbor discovery.

When the source needs a route toward a destination that is far away, first it uses the most recent link state information to compute the shortest path. Although the link state information may not be up-to-date, as the packet approaches the inner scope of the destination, the accuracy of the path increases and finally, the packet will arrive at the destination correctly.

To reduce routing traffic overhead further, link state information are propagated periodically only. The breakage of a link between two mobile nodes will not trigger the broadcast of link state update, because it happens frequently in the MANET.

(3) Dynamic source routing protocol (DSR, [8])

DSR aims to serve a MANET with up to two hundred mobile nodes. Unlike other unicast routing protocols, DSR does not maintain the routing table because it utilizes the source routing option in data packets. It uses Route Cache instead, which stores the complete list of IP addresses of the nodes along the path towards the destination.

The basic procedure of DSR is:

(1) Route discovery: If the source route entry towards a destination is not present in the route cache, a Route Request packet is broadcast throughout the MANET. Before the intermediate node forwards the packet, it appends its own IP address in a list in the request packet. When the destination receives the packet, the request packet has accumulated the path from the source to the destination. Then the destination performs another route discovery to find the route towards the source if the underlying MAC layer supports unidirectional links; otherwise, it just reverses the source route recorded in the request packet. In either way, a Route Reply packet that contains the route from the source to destination is sent

back to the source. After the procedure of route discovery, both the source and destination have the source route towards each other.

(2) Route maintenance: Unlike proactive routing protocols and AODV mentioned below, no periodic HELLO message is introduced in DSR. Every node along the path is responsible for the validity of the downstream link connecting itself and the next hop in the source route, which could be detected by MAC layer or DSR-specific software acknowledgement. If link breakage is found, the source of the route will be notified with a Route Error packet. The source node then re-initiates a route discovery procedure.

The route cache is widely adopted in DSR. For example, the intermediate nodes cache the route towards the destination and backward to the source. Moreover, because the data packet contains the source route in the header, the overhearing nodes are able to cache the route in its routing cache.

The route cache greatly reduce the routing overhead in the following aspects:

(1) During route discovery phase, if the intermediate node has the route towards the destination in its routing cache, it can answer with a route reply packet and send a gratuitous route reply about the source to the destination at the same time;

(2) Because DSR supports multi-paths, if the source receives a route error packet, it can use an alternative path store in the routing cache, thus saving the overhead of route discovery;

(3) If the intermediate node detects the downstream link breakage when forwarding a data packet, but it has another source route in its routing cache towards the

same destination, it forwards the packet along the new route, which is called packet salvaging.

(4) Ad hoc on-demand distance vector routing protocol (AODV, [9])

AODV is another reactive routing protocol, which consists the following procedures:

(1) Route discovery: If the route is not available in the routing table towards the destination, a RREQ (Route Request) packet is broadcast throughout the MANET with a search ring technique. On receipt of RREQ, the node creates a reverse routing entry towards the originator of RREQ, which is used to forward replies later. The destination or the intermediate node, which has a valid route towards the destination, answers with a RREP (Route Reply) unicast packet. On receipt of RREP, the reverse routing entry towards the originator of RREP is also created, similar to the processing of RREQ. Associated with each routing entry is a so-called precursor list, which is created at the same time. The precursor list contains the upstream nodes that use the node itself towards the same destinations.

(2) Route maintenance: Every node along an active route periodically broadcasts HELLO messages to its neighbors. If the node does not receive a HELLO message or a data packet from a neighbor for a while, the link between itself and the neighbor is considered to be broken. If the destination with this neighbor as the next hop is believed not to be far away (from the invalid routing entry), a local repair mechanism may be launched to rebuild the route towards the destination; otherwise, a REER (Route Error) packet is sent to the

neighbors in the precursor list associated with the routing entry to inform them of the link failure.

To ensure loop freedom in distance vector routing protocols, every node maintains a sequence number. The sequence number is sent with RREQ (for source) and RREP (for destination) and stored in the routing table. The larger the sequence number, the newer the route information.

(5) Zone Routing Protocol (ZRP, [14])

ZRP is a framework of hybrid routing protocol suites, which is made up the following modules:

(1) Intrazone routing protocol [15]

(2) Interzone routing protocol [16]

(3) Bordercast resolution protocol [17]

In ZRP, every node has a zone that is defined to be the nodes within the distance of n hops (n is a configurable parameter). Within the zone, intrazone routing protocol, which is a proactive protocol, is adopted to maintain the local topology. When the route between different zones is needed, Intrazone routing protocol, which is a reactive protocol, is used to find the path between the source and destination. Bordercast is an efficient broadcast technique that reduces the number of redundant forwarding in route discovery of interzone routing protocol.

The advantages of ZRP include:

(1) Because intrazone routing protocol is proactive, the route towards the node within the zone is available before it's needed, thus the delay and overhead of route discovery is avoided;

(2) Because the periodic broadcast of topology information is confined within the zone, the change of link status at one end of the network will not affect the other end of the network;

(3) The path between different zones is build on demand, which saves the overhead of periodic broadcast of topology information throughout the MANET, as what proactive routing protocols do;

(4) Proactive intrazone routing protocol helps the route maintenance of reactive interzone routing protocol. The broken link can be bypassed with the aid of local topology information, and route optimization can be achieved within the zone;

(5) Local topology information obtained from intrazone routing protocol helps efficient forwarding of broadcast packets, which transmits the packet from covered area of the network to uncovered part.

The radius of the zone is a configurable parameter. Different zones may have different radii. With properly configured zone radius, ZRP will outperform both proactive routing protocols and reactive routing protocols.

(6) Landmark ad hoc routing protocol (LANMAR, [18])

LANMAR is a hierarchical routing protocol for large scale MANETs with group mobility. It assumes that the network is composed of some logical groups (subnet), in which the members move together. Within the group, a "landmark" node is elected. A modified FSR routing scheme is utilized to broadcast the link state of the members within the group and landmark nodes across the groups proactively. Thus, every mobile node's routing table contains routing entries towards its group-mates and all the other groups'

landmarks. When the source wants to send data packets to the destination, the packets are forwarded towards the destination's landmark node. Once the packets enter the scope of the group, the packets will be delivered to the destinations directly.

To cope with the scenario of drifted nodes within a group, a path between the landmark node and the drifter is maintained to keep track of the drifters. To cope with an isolated node (whose group size is 1), either a reactive method is utilized to find the path to it, or some Home Agent are elected to record the path to it.

(7) Location-aided routing (LAR, [19])

The central point of LAR is the limited flooding of routing request packets in a small group of nodes that belong to a so-called request zone. Compared with other routing protocols such as AODV or DSR, in which routing packets are flooded throughout the network, LAR saves considerable bandwidth and leaves those mobile nodes that are not between the source and destination untouched.

To construct the request zone, the expected zone of the destination needs to be obtained first. Suppose both the average speed (say v) and the location of the destination at time t_0 (say L) are known to the source, the expected zone of the destination at time t_1 is the circle with center at L and radius of $v(t_1 - t_0)$.

Two different schemes are brought to construct the request zone: (1) a rectangular request zone that contains the location of source and the expected zone of the destination; or (2) the group of the nodes closer to the destination than the source.

The procedure of route discovery in LAR is:

(1) The source puts the location information of itself and the destination in the routing request packet;

(2) The routing request packet is broadcast within the request zone. In other words, the nodes within the request zone forward the message and the others discard the message;

(3) On receipt of the route request packet, the destination sends back a route reply packet that contains its current location;

(4) If LAR fails to find the route to the destination due to estimation error or other reasons, the routing protocol resorts to flooding routing messages throughout the MANET.

(8) Geographical Routing Algorithm (GRA, [20])

The assumption in GRA is that every node knows the position of itself, the destination and all its neighbors. Stored in the routing table at each node are entries (p_i, S_i), where p_i is the geographic position of some node and S_i is one of its neighbors, which means the S_i is closer to p_i than the node itself.

When source S wants to send data packets to destination D, S puts the geographical information of itself and D in the data packet, searches its routing table to find the nearest neighbor to D, and forwards the packet to that neighbor. When another node A receives it, it will forward it to the nearest neighbor to D, as S does. However, if node A finds itself closest to D than any neighbors, but it has no entry for D in the routing table, it initiates a route discovery procedure (e.g., Depth First Searching). The entire path from A to D will recorded in the route request packet when it arrives at D, so D can answers with an ACK to update the routing table at the nodes along the path from D to A. When A receives the ACK, the data packet flow can continue.

(9) Location service for Geographical-based routing protocol

Geographical-based routing protocols have three assumptions:

(1) The mobile node knows its location, which can be easily satisfied with the aid of outdoors and indoors locating devices (e.g., GPS);

(2) The mobile node knows the location of its neighbors, which can be achieved by periodical exchange of HELLO messages piggybacked with location information;

(3) The source knows the location of the destination in advance, which is the role of the location service.

There are four kinds of location services available:

(1) DREAM system

The most intuitive method was proposed as DREAM [21], in which mobile nodes broadcast their location information throughout the MANET periodically. As a result, the source knows the up-to-date location of the destination before data transmission. Although it may consume much bandwidth, it is very simple, robust, and easy to implement.

(2) Quorum system

The second location service proposed is Quorum system [22]. Quorum system originates from information replication in databases and distributed systems, and could be applied to location service. In quorum system, some mobile nodes are chosen to form a backbone network in the MANET. These backbone nodes are further divided into several quorums such that the intersection of every pair of quorums is non-empty.

Suppose the location updates of node A are sent to the nearest backbone node (say B) in a quorum and then spread into the whole quorum. The queries about A's location from

node C are sent to a backbone node (say D) in another quorum and then spread into the whole quorum. The node in both quorums (say E) will reply to the queries.

(3) Grid system

The Grid system [23] is a hierarchical location service. The area of the MANET in the Grid system is divided into many small 1-order squares. Every four adjacent 1-order squares form a bigger 2-order square, and so on. A mobile node keeps the other nodes in the same 1-order square informed of its up-to-date location. As to n-order square where n is greater than 1, every mobile node recruits one node in each of four (n-1)-order squares to keep its location information. Here comes the question: which node should be chosen? The answer is the node with the least greater node ID in each (n-1)-order square.

The grid is self-containing. It is independent of unicast routing protocols, which means location updates and location queries are forwarded based on location information as well.

(4) Home agent-based system

Another location service was proposed as home agent-based system [24]. In home agent-based location service, a node chooses the location scope where it first joins the MANET as its home agent. It periodically sends location update to all the mobile nodes in its home agent. Because the location of home agent is announced to all the other nodes in the beginning, all the following queries could be sent to its home agent and get the corresponding reply.

The drawbacks of this scheme are:

(1) Inefficiency: Suppose the node moves far away from its home agent, location updates have to go across the long distance;

(2) High requirements on memory: every node has to keep every other's home agent.

2.1.2 Secure unicast routing protocols

To defeat attacks at routing protocols, secure unicast routing protocols are proposed, including SAR, ARAN, and Ariadne.

(1) Security-aware routing protocol (SAR, [25])

SAR works like an ordinary reactive unicast routing protocol because it consists of route discovery and route maintenance, except that the routing metric is changed from the distance between mobile nodes to the "level of security". When the source broadcasts a route discovery packet (RREQ) throughout the MANET, only the intermediate nodes whose security levels are equal to or higher than that required in the packet can forward RREQ and RREP packets. Others will drop the routing packets. As a result, the path between the source and destination may not be the shortest, but is secure. The nodes that have lower security level or are compromised will be circumvented.

(2) Authenticated Routing for Ad hoc Networks (ARAN, [26])

ARAN originates from AODV, with the following differences:

(1) Every time a node receives a RREQ or RREP packet, it first validates the identity of the source with the source's corresponding public key, and then signs the message with its own private key.

(2) ARAN requires only the destination can sends back a RREP packet.

(3) ARAN stores a routing entry per source/destination pair in the routing table, which consumes more memory space.

ARAN requires key distribution in advance. Thus, a public/private key pair can be bound with an IP address.

(3) Ariadne [27]

Ariadne also utilizes a digital signature to sign routing messages to prevent malicious nodes from changing message contents or sending false messages. A message authentication code (MAC) is appended to the packet to identify the sender of the forwarder, in which the digital signature can be accomplished in the following three different ways:

(1) Every mobile node in the MANET has a public/private key pair which is certified from a CA;

(2) Every pair of source/destination has a shared secret key, if there is such a mechanism to set up these n(n+1)/2 keys;

(3) TESLA broadcast authentication protocol proposed by the authors. In this protocol, a sender chooses a random initial key value (K_N) and generates a one-way key chain with a one-way hash function H, such that $K_i = H[K_{i+1}] = H_{N-i}[K_N]$. The sender uses the keys in order of $K_1, K_2, \ldots, K_N$. At time t_i, the sender sends a packet signed with key K_i. After a while, it's the time that the packet is supposed to arrive at the destination, the sender broadcasts the key K_i. Thus the receiver can use this recently published key K_i to authenticate the keys published before (such as K_{i-1}), and the packets signed with K_i.

2.1.3 Multicast routing protocols for MANETs

To send a data packet to multiple receivers in the MANET simultaneously, the simplest method is to resort to broadcast. However, broadcast consumes considerable bandwidth and power, which should be avoided as much as possible.

There have been many multicast routing protocols proposed for MANET. According to their underlying routing fabrics, they could be divided into two groups: tree-based protocols and mesh-based protocols.

The tree-based protocols originate from their counterparts in hardwired networks. The group members and (possibly) some non-members form a shared multicast tree. When the sender sends out a data packet, the receiver receives it from its upstream node in the tree and forwards it along the downstream links in the tree. Because only the tree members participate in the packet transmission, a lot of bandwidth is saved compared to pure broadcast. Tree-based multicast routing protocols include MAODV, AMRoute, and AMRIS.

Mesh-based multicast routing protocols use a mesh instead of a shared multicast tree for packet delivery, which provides redundant links among group members. Compared with tree-based routing protocols, they may consume more bandwidth. However, they are more resilient to network dynamics. Mesh-based routing protocols include ODMRP and NSMP.

(1) Multicast Ad-hoc On-Demand Distance Vector Routing Protocol (MAODV [28])

AODV is extended to support multicast routing in the MANET. Every multicast group has a sequence number to indicate the freshness of the multicast routing information. Thus, one and only one group leader is elected to broadcast periodical GROUP HELLO messages throughout the MANET to maintain the sequence number. The group leader is by default the first node joining the group, but could also be another node when the first node leaves the group.

To support multicast transmission, a multicast tree is formed on-demand to include all the group members and some non-members, which are relay nodes. The process of building such a tree is similar to the route discovery procedure in unicast routing: every time when a node wants to join a multicast group or to send a data packet to a multicast destination (while it does not have the proper routing entry), a RREQ message is broadcast throughout the MANET. The nodes in the multicast tree for this group send back a RREP message. The nodes forwarding RREQ and RREP record the path backwards to the source of packet, as they will do in unicast routing. On receipt of multiple RREP packets, the node chooses one branch of the multicast tree and connects to it, thus a loop is avoided.

When a link breakage is detected due to node movement, the node that is farther away from the group leader initiates local repair. Again, it broadcasts a RREQ message and waits for RREP from the group leader. By this means the tree is reconstructed to accommodate the topological change.

(2) Adhoc Multicast Routing Protocol (AMRoute [29])

AMRoute builds a user-multicast tree, in which only the group members are included. Because non-members are not included in the tree, the links in the tree are virtual links. In other words, they are in fact multi-hop IP-in-IP tunnels. Thus, AMRoute depends on the underlying unicast routing protocol to deal with network dynamics, although it has no favorite unicast routing protocols.

Like MAODV, there is only one logical core in the multicast tree, which is responsible for group member maintenance and multicast tree creation.

The multicast operation in AMRoute consists of the two steps:

Mesh creation

In the very beginning, a group member is the core for its own 1-node mesh and begins to broadcast JOIN_REQ messages periodically. When a group member (which is a core currently) receives such a message from another core, it answers with a JOIN_ACK message, which means the two cores find each other. Thus, the two 1-node meshes merge, which leads to only one member elected to be the core for the new mesh. A bi-directional tunnel is built between these two nodes at the same time. As a result, a mesh forms from a scratch that includes all the group members.

Tree creation

The core of the mesh broadcasts periodic TREE_CREATE messages throughout the mesh (along the tunnels). On receipt of this message, a group member chooses one mesh link from which it receives the message to be the tree link and ignores the other duplicate messages. A TREE_CREATE_NAK message is sent back along the ignored the mesh links to prune the mesh links from the multicast tree. Depending on the mobility pattern and bandwidth, an ACK-based scheme could be used instead to indicate the mesh link to be tree link.

(3) Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS, [30])

In AMRIS every node is assigned an id-number. The source of the multicast session has the smallest id. As to other group members, their ids increase with their distance from the source. To build a delivery tree, the source generates its own msm-id and then broadcasts a NEW-SESSION message throughout the MANET. During that time, every

node chooses its own msm-id, which is larger than the one contained in the message and forwards its new msm-id. Thus, every node has an msm-id.

When a node wants to join the multicast session, it chooses the neighbor that has the smaller msm-id as its parent and sends it a JOIN-REQ message. If the neighbor is in the tree (if the tree has been built), it answers with a JOIN-ACK message, which means the joining is successful; otherwise (when it is the first time to build the tree), the neighbor forwards JOIN-REQ to its own neighbors and waits for the reply, which is repeated until the JOIN-REQ arrives at an on-tree node or the source. As a result, a delivery tree rooted from the source is formed to include all the group members and some relay non-members.

Every group member broadcasts a one-hop beacon message to maintain link availability. If a link is broken, the node with a larger msm-id tries to reconstruct the branch. If it's within one-hop distance of another group member, it will re-join the delivery tree after it receives the beacon message from its on-tree neighbor; otherwise, it broadcasts a JOINE-REQ message.

(4) On-Demand Multicast Routing Protocol (ODMRP, [31])

ODMRP is a reactive multicast routing protocol. The source establishes and maintains group membership and multicast mesh on demand if it needs to send data packets to the multicast group, which is somewhat similar to MAODV. However, it builds a mesh instead of tree for packet transmission. A set of nodes, which are called the forwarding group, participate in forwarding data packets among group members.

When a source node needs the route to a multicast group, it begins to periodically broadcast a JOIN REQUEST message, which is forwarded by all the nodes in the

MANET. When a group member receives such a message, it records the IP address of the node upstream to be the next hop for the source, and broadcasts a JOIN TABLE to its neighbor. On receipt of the JOIN TABLE, the neighbor node examines the table to see if it is the next hop for the source in one entry. If the answer is positive, the node sets itself to be a forwarding node and broadcasts its own JOIN TABLE to its neighbors as well. Thus the JOIN TABLE is sent back until is reaches the source. At that time the forwarding group is formed and the route is built. From then on the data packets can be delivered to the receivers properly.

The other notable properties about ODMRP are:

(1) All the states in ODMRP are soft states, which are refreshed by the control messages mentioned above or data packets, which achieves higher robustness;

(2) ODMRP is not only a multicast routing protocol, but also provides unicast routing capability.

(5) Neighbor Supporting Ad hoc Multicast Routing Protocol (NSMP, [32])

In NSMP, the source, relaying nodes, and the receivers are designated as forwarding nodes, which form a multicast mesh. All the nodes that are adjacent to at least one forwarding node are designated as neighbor nodes.

When a source needs the route to other group members, a route discovery procedure is initiated to build the mesh: a FLOOD_REQ message is forwarded by all the nodes and a REP message is sent back by every group members, which works similarly with ODMRP.

The difference between ODMRP and NSMP is that the source node in the former periodically broadcasts route request packet throughout the network for purpose of group

and route maintenance, while the latter limits the scope of broadcast of such packets to the set of forwarding nodes and neighbor nodes after the mesh is built. Due to node locality, link breakage can be easily repaired with scoped broadcasting. Thus, considerable bandwidth is saved compared to ODMRP. In case of network partition and new neighbors joining, NSMP resorts to global flooding.

2.1.4 Broadcast routing protocols for MANETs

Broadcast is inevitable in MANETs for the following reasons:

(1) Proactive routing protocols such as OLSR;

(2) Route discovery in reactive routing protocols such as DSR and AODV;

(3) Conflict-detection and best-effort address allocation (described in Section 2.2);

(4) Service discovery or service advertisement;

(5) Some multicast communication may resort to broadcast;

(6) Unicast communication has to resort to broadcast due to very high mobility.

The simplest method is to receive and then forward the data packet for every node in the MANET. A list of Flood Packet Identifier (FPI) [33], which is based on source IP address, packet ID and fragment offset (for IPv4), is maintained to distinguish between duplicate data packets. Therefore, every node forwards the data packet only once. However, this simple method itself still consumes much bandwidth and power.

(1) Multipoint Relaying (MPR [34])

As we mentioned before, OLSR bases the transmission of topology broadcast packet on Multipoint Relaying flooding. In MPR flooding, every node keeps on broadcasting

one-hop neighbor information in periodic HELLO messages to all its neighbors. Thus every node knows local topology about one-hop neighbors and two-hop neighbors.

When a node broadcasts, all its one-hop neighbors will receive the packet. To save the bandwidth in the further forwarding, MPR flooding selects a subset of one-hop neighbors as relays that can reach all the two-hop neighbors. These selected neighbors are called MPRs (multipoint relay). When those MPRs forward, they again choose their own MPRs to forward the packet. This procedure is repeated until all the nodes receive the packet.

The less the MPRs, the more bandwidth it saves. However, it is a NP-complete problem to determine the minimum number of MPRs. The authors proved that the well-known NP-complete problem of Dominating Set Problem can be reduced to it, and gives a heuristic algorithm to solve the problem:

(1) Choose those one-hop neighbors that are only neighbors of some two-hop neighbors;

(2) While there are still two-hop neighbors that cannot be reached, choose the one-hop neighbor that can cover the maximum number of remaining two-hop neighbors.

(2) Scalable Broadcast Algorithm (SBA, [35])

SBA also utilizes 2-hop neighbor information. Suppose node B receives a broadcast packet from node A. Because node B is a 1-hop neighbor of node A and it has 2-hop neighbor information, node B knows all 1-hop neighbors of node A must have received the packet at the same time. If all node B's 1-hop neighbors are not node A's 1-hop neighbors, node B will forward this packet to its own uncovered 1-hop neighbors.

Node B will usually wait for a random delay before the re-transmission. If node B receives a rebroadcast packet from another neighbor during that time, it will determine

again if it has any neighbors that have not received the packet so far. If the answer is positive and the delay expires, it will re-transmit the packet.

Although both MRP and SBA utilize 2-hop neighbor information to reduce unnecessary retransmission of broadcast packets, they are different in making decision on choosing the subset of the neighbors to forward. In MPR, the sender uses a sophisticated algorithm to actively choose some neighbors to be the relays, and informs them in the periodical exchanged HELLO messages; while in SBA, the receivers make local decision whether it should forward the packet or not.

2.2 Autoconfiguration for MANETs

Autoconfiguration is the first step towards the practical application of the MANETs. Several solutions have been suggested and studied by other researchers, which can be divided into the following three categories.

2.2.1 Conflict-detection allocation

The conflict-detection allocation adopts a "trial and error" policy to find a free IP address for a new mobile node in the MANET. The new node chooses an IP address tentatively, and requests for approval from all the configured nodes in the MANET. If the conflict is found by veto from a node with the same IP address, the procedure is repeated until there is no duplicate address. At that time the node uses the latest chosen IP address as its "permanent" address. One of the conflict-detection allocation algorithms is the protocol proposed in [36]. Other examples include IPv6 autoconfiguration for MANET proposed in [37] and the scheme in [38].

The procedure above is defined as strong DAD (Duplicate Address Detection) in [39], which is able to handle scenario *A* easily, without any solution for Scenarios *B* and *C*. The so-called weak DAD is proposed in [39], which aims to handle network merger. It favors proactive routing protocols and requires little modification to routing protocols.

2.2.2 Conflict-free allocation

The conflict-free allocation assigns an unused IP address to a new node, which could be achieved by the assumption that the nodes taking part in allocation have disjoint address pools. Thus they could be sure that the allocated addresses are different. Dynamic Configuration and Distribution Protocol (DCDP) [40] is a conflict-free allocation algorithm, which was originally proposed for autoconfiguaration in hardwired networks. Every time when a new mobile node joins, an address pool is divided into halves between it and a configured node.

One advantage of conflict-free allocation is that it still works in Scenario *B*. Even if the network becomes partitioned, the nodes in different partitions still have different address pools. Thus the addresses allocated are different as well. When the partitions become connected, no further work is necessary. As to Scenario *C*, it is very likely that there are conflicts if the configuration of two MANETs begins with the same reserved address range.

A similar idea is proposed in [41], which tried to solve the issue of networks' partition and merger.

2.2.3 Best-effort allocation

In this approach, the nodes responsible for allocation try to assign an unused IP address to a new node as far as they know. At the same time the new node uses conflict detection to guarantee that it is a free IP address.

An example of best-effort allocation is Distributed Dynamic Host Configuration Protocol (DDHCP) proposed in [42]. DDHCP maintains a global allocation state, which means all mobile nodes are tracked, so it is known which IP addresses have been used and which addresses are still free. When a new node joins the MANET, one of its neighbors could choose a free address for it. The reason why it still bothers to detect conflict is that the same free IP address in the global address pool could be assigned to two or more new nodes arriving at almost the same time.

One advantage of DDHCP is that it works well with proactive routing protocols, since every mobile node broadcasts periodically. Another advantage is that it takes into account network partition and merger. A partition ID is generated by the node with the lowest IP address and broadcast throughout the partition periodically. Thus, the partition and merger may be detected by partition ID (with the aid of periodic exchange of HELLO messages). When partitions become connected, conflict detection and resolution is initiated.

2.3 Handoff Schemes

With the introduction of autoconfiguration in the MANET, a node may change its address due to address conflict, which will bring communication overhead. There have been several schemes proposed for handoff operations.

2.3.1 Mobile IP

Mobile IP intends to provide basic support for mobile hosts in a LAN [43]. According to the scheme, a mobile host is assigned a permanent home address that is bound with its home agent. When it becomes connected to a foreign network, it receives a temporary care-of address and other information (e.g., the subnet mask and default router) from the foreign agent. The mobile host registers its current care-of address at its home agent, which then builds a tunnel between itself and the foreign agent. When another host initiates communication with the mobile host, it usually gets the mobile host's home address from DNS query and sends the packets to the home address. The packets will be then forwarded to the mobile host's care-of address by the home agent through the IP tunnel.

Mobile IP is efficient for IP address handoff in a LAN that has an infrastructure. However, because the nodes in the MANET are mobile and instable, none of them can be designated as the home agent or foreign agent for another node. Thus, it cannot be applied in the MANET.

2.3.2 Tunneling mechanism

In addition to autoconfiguration in the MANET, the scheme in [38] proposed a solution for the maintenance of communication states after address changes. The node (say node A) that changes its IP address notifies the other end (say node B) with a special Address Error (AERR) message. From then on, they communicate with each other through an IP-in-IP tunnel: the outer IP header contains the A's new address, while the inner IP header contains A's old address. Unlike the IP tunneling in Mobile IP, the

communicating nodes A and B are also the end points of the tunnel: the source encapsulates the packet that is decapsulated at the destination.

This approach is able to preserve communication states at both ends, but it does not solve all the problems. Moreover, it brings a "DoS" problem that will be discussed in Section 5.4.

2.4 Security Framework for MANETs

Threshold cryptography was originally proposed for the DCA in hardwired networks ([44] - [45]), which is based upon public key cryptography. The public key of the DCA is known to all the users, while the secret key is divided into many secret shares that are stored among the servers. For the (k, n)-threshold cryptography, there are n servers, each of which has a unique secret share. When a client node wants its message signed by the servers, it sends the message to each server, which applies its secret share in computation of the partial signature. With the partial signatures from at least k servers, the DCA server group can construct a valid signature that can be verified with the well-known public key.

Compared with the traditional centralized CA scheme, the threshold cryptography-based DCA has the following advantages:

(1) The secret shares have no explicit relations except that they are all part of the secret key, which means that one share cannot be deduced from another share. Even if one server is compromised, the attacker still has no information about other shares. The attacker has no choice but to compromise at least k servers to find out the secret key;

(2) As long as there are at least k servers that apply their shares in the signing procedures correctly, the valid signature can be generated. Thus, the threshold-based

scheme is tolerable to some missing or faulty servers, which makes it especially suitable for the MANET where a server node may leave or shut down without notice;

(3) To further improve the security of the scheme, the shares can be refreshed periodically among the servers [46]. The share before the refresh operation and that after the refresh operation has no relation, which means that even if one share is leaked, it will become useless after the refresh interval. Thus, a mobile adversary is challenged to compromise at least k servers in a short time[1]. Although the secret shares are changed, the secret key is always the same, which means the corresponding well-known public key is the same. Therefore, the refresh operation is transparent to client nodes.

Threshold cryptography was introduced into the MANET in [47]. On receipt of a request from a client node, each server generates a partial signature with its share and sends it to a special node that is designated as a combinator. The combinator collects all the responses from servers and calculates the signature for the client node. The scheme was applied to a large-scale MANET in [48], in which the nodes are divided into many clusters. The cluster heads form the server group and provide certificate service to cluster members.

Due to its invulnerability to mobile adversaries and tolerance to instable nodes, the threshold-based DCA scheme becomes the "de facto" standard for certificate authority service in the MANET.

---

[1] Of course, we assume that the server node is not totally controlled by the attacker. Otherwise, the attacker will know all the server's shares at different times.

2.5 ID Assignment in Sensor Networks

The scheme proposed in [49] utilized a proactive conflict detection method for a general sensor network, including a mobile sensor network, and a stationary sensor network with new members joining. When a node boots up, it first chooses a random physical address and then announces it with periodic broadcasts of HELLO messages with the interval of 10 seconds. All the nodes record the source address of the HELLO message in a neighbor table, which is included in the subsequent HELLO messages. Therefore, every node will have 2-hop neighbor information, which is utilized to resolve address conflicts among 2-hop neighbors. If a node finds that one of its neighbors chooses a duplicate address, it will notify this neighbor to change the address. To further decrease the average address field length, the scheme encodes the physical address using Huffman coding.

In [50], the scheme is modified to specifically suit stationary sensor networks in which periodical broadcasts of HELLO messages are replaced by a fixed number of broadcasts, with 4 cycles and the interval of 8 seconds recommended. The address is encoded as a Huffman code as well.

To implement Huffman coding, the user of the sensor network must first run a simulation with the expectant node density to compute the Huffman code table. The code table is then input into every node for coding and decoding of the physical address in the packet header for data communications. However, according to the simulation results, only 0.3 to 0.8 bit is saved for one physical address on average with Huffman coding compared with fixed-length format. In the case where the actual density of nodes is not

the same as expected, the authors admitted that no benefit in codeword length would be achieved.

CHAPTER 3 PROPHET ADDRESS ALLOCATION

In this chapter, the algorithm for Prophet Address Allocation is described in Section 3.1. The theoretical analysis in Section 3.2 shows that our algorithm is better than other schemes in terms of communication overhead, latency, and scalability for a large-scale MANET, which is supported by simulation results in Section 3.3. Section 3.4 concludes this chapter.

3.1 Algorithm

IP address autoconfiguration is the same as assignment of different numbers from an integer range, say R, to different nodes. Conflict-detection allocation and best-effort allocation use random guesses and then make sure there is no duplicate by means of broadcast of conflict detection. Conflict-free allocation partitions R into several disjoint subsets $R_1, R_2, \ldots, R_m$ and chooses a random subset to divide between different nodes.

The idea included in these algorithms is that every mobile node obtains an unused IP address randomly on its own. Unless a node announces its IP address throughout the MANET, it cannot be known to others that this IP address is occupied. What if all the IP addresses that have been allocated and are going to be allocated are known to every participating node in advance? Broadcast could be avoided while conflict is still detectable.

3.1.1 Prophet allocation

Suppose we may obtain an integer sequence consisting of numbers in R by a function, say $f(n)$, which is stateful. The initial state of $f(n)$ is called the *seed*. Different seeds lead to different sequences with the state of $f(n)$ updated at the same time. The sequences of $f(n)$ satisfy the following two properties (if R is large enough):

(1) The interval between two occurrences of the same number in a sequence is extremely long;

(2) The probability of more than one occurrence of the same number in a limited number of different sequences initiated by different seeds during some interval is extremely low.

Thus we could derive an IP address autoconfiguration algorithm from the aforementioned sequence generation:

(1) The first node in the MANET, say A, chooses a random number as its IP address and uses a random state value or a default state value as the seed for its $f(n)$;

(2) When a new node, say B, approaches A and asks A for a free IP address, A uses $f(n)$ to obtain another integer, say n_2, and a state value, and provides them to B. Node A updates its state accordingly;

(3) Node B uses n_2 generated by A as its IP address and the state value obtained from node A as the seed for its $f(n)$;

(4) Now node A and node B are both able to assign IP addresses to other new nodes.

The communication between node A and node B may be accomplished by means of one-hop broadcast since B does not have an IP address yet. However, it still saves much communication overhead compared with multi-hop broadcast needed in conflict detection.

The algorithm is illustrated as an example in Fig. 3.1. Suppose every node is represented by a 2-tuple: (*address*, *state of f(n)*). Here R is [1,8], $f(n)$ is ($address \times state \times 11$) mod 7 and the effective address range is [1,6]. In Fig. 3.1, node A is the first node in the MANET and uses a random number of 3 as its IP address and seed. When node B joins, node A gets 1 (=(3×3×11) mod 7). Node A changes its state of $f(n)$ to 1 and assigns 1 to B. When C approaches A and D approaches B, they receive 5 (=(3×1×11) mod 7) and 4 (=(1×1×11) mod 7) from A and B, respectively. In the third round of allocation, a conflict will happen. Note that 4 out of 6 addresses are allocated without conflict in the first 2 rounds of allocation, and the allocation later leads to a conflict. The reason of conflict is due to a small range of R.

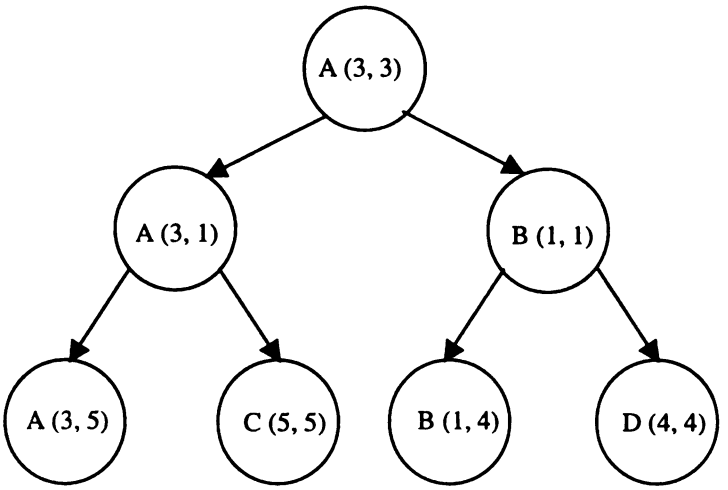


Figure 3.1 An example of prophet allocation

In the beginning of allocation, node A chooses the seed for the whole MANET and the sequences may be computed locally. Therefore, node A is a prophet in the MANET, which means it knows in advance which addresses are going to be allocated. Thus, we call this algorithm *prophet allocation*.

Because the potential conflict in the allocation may be known at node A in the beginning, it is able to launch local conflict detection before allocation. If there are many duplicate numbers in the sequences, node A could choose another seed to generate other sequences until there are fewer conflicts. Those duplicate numbers could be marked in the beginning of allocation.

Address reclamation is unnecessary for prophet allocation because the same number will reoccur in the sequence. Nevertheless, the minimal interval between two occurrences in the sequences is extremely long. When a node is assigned an old address, say n, the previous node with the same address of n has likely already left the MANET.

3.1.2 Mechanism for network partition and merge

Prophet allocation is able to solve the problem of network partition and merger of a MANET easily. As for Scenario *B*, because the sequences are different even if the MANET becomes partitioned, the newly allocated addresses are still different among the partitions. Therefore, there is no conflict if the partitions become merged later.

With regard to Scenario *C*, we borrow the idea of partition ID in DDHCP with a little modification. Here we designate the first node in the MANET to generate the network ID (NID) using a random number, which is propagated to new nodes during the course of allocation. Because NID is a random number, if the number of bits for NID is large enough, two MANETs will have different NIDs. Since some reactive routing protocols

(e.g., AODV [9]) require periodic exchange of HELLO messages between neighboring nodes, if NID is piggybacked in HELLO messages, the merger of two separate MANETs may be easily detected.

There are two methods to cope with Scenario *C*. The simpler method is that when mobile nodes detect the merger of two independent MANETs, the nodes in one MANET, say MANET 1 (for example, MANET 1 has a smaller NID), choose to discard their current IP addresses and acquire new addresses and NID from their neighbors in the other MANET (say MANET 2), which propagates from the intersection of the two MANETs until all the nodes in MANET 1 acquire their new addresses. Thus, the overhead of local conflict detection and conflict resolution is saved at the cost of breaking on-going communication and routing fabrics in MANET 1. This is especially suitable for the situation of a merger of a MANET with a one-node partition, which will be aware that it has no neighbors with the same NID and will decide to change its IP address.

If both networks have many members, the method above will bring too much overhead, so we can resort to the second method. If we specify that the seed for the MANET be carried in the HELLO messages as well, and that the conflicting nodes in one network (say, MANET 1 that has a smaller NID) change their address, the node in MANET 1 that detects merger is able to find potential address conflicts between two MANETs locally by applying $f(n)$ on the two seed values for MANETs and initiates conflict resolution if necessary. The possibly conflicting addresses are contained in the message that is broadcast to MANET 1. If a node in MANET 1 has the IP address contained in the list, it changes its address accordingly, which is similar to the method above: the nodes in MANET 1 acquire their new IP address from MANET 2. However,

only the conflicting nodes in MANET 1 change their addresses. The remaining nodes keep their old addresses, but use the states generated within MANET 2 in the following allocations. If several nodes detect the merger at the same time, they could initiate conflict resolution independently, or random delay is introduced to save repeated work. The larger NID will be the NID of the merged network.

3.1.3 Design of f(n)

The stateful function *f(n)* should be carefully designed. In the example in Fig. 3.1, we used primes to scatter the numbers in the sequence. In a real design, *f(n)* is closely related to address range as well. For IPv4, class C private addresses of 192.168.0/24 are not large enough for dozens of mobile nodes in the MANET because of the high probability of collision. Class A private addresses of 10/8 and Class B private addresses of 172.16/12 will be suitable. As to IPv6, there is no need for such a concern because of its huge address range.

It is difficult to find such an *f(n)* that exactly satisfies the two properties mentioned before. However, an *f(n)* that approximately satisfies the properties is easy to design. One such *f*(n) we suggest is based on the fundamental theory in arithmetic: every positive integer may be expressed uniquely as a product of primes, apart from the rearrangement of terms. The canonical form of a positive number n is $n = \prod_{i=1}^{k} p_i^{e_i}$, where the primes p_i satisfy $p_1 < p_2 < \ldots < p_k$ and the exponents are non-negative integers. Apparently, if k-tuples $(e_1, e_2, \ldots, e_k)$ have different e_i $(i = 1, \ldots, k)$, there will be different n. Our idea is to generate different k-tuples.

Suppose $k = 4$. The first node obtains a random address of a and an initial state of ($\underline{0}$, 0, 0, 0). Fig. 3.2 shows the procedure of generating new states and updating old states. A node is represented by (*address*, (e_1, e_2, e_3, e_4)), with *address* = ($a + 2^{e1}3^{e2}5^{e3}7^{e4}$) mod *range* + 1 (with the exception of the first node). The parameters sent from the allocator to the new node include: (1) seed value (a); (2) the exponential array (e_1, e_2, e_3, e_4); (3) the index of increasing exponential (the underlined element). The rules of state generation and update during the allocation are: (1) the increasing exponential (the underlined element in the 4-tuple) of the allocator is increases by 1; (2) the state of a new node is copied from the allocator, but the index of the increasing exponential shifts to the right (as the underline moves to the right).

k may be much larger in real applications. Thus, our algorithm requires an array of primes and exponents only and nothing else. However, the array of exponents need not be stored in nodes or carried in the messages between neighboring nodes during allocation. Our simulation also shows that optimization is achievable in the computation of addresses.

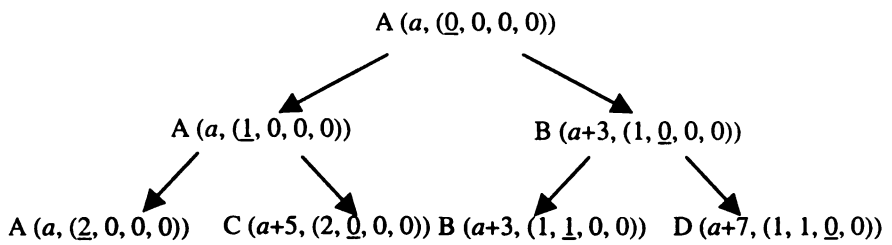


Figure 3.2 Generation and update of states in f(n)

There will be infinite different numbers generated by $f(n)$ in theory. However, given a small range of addresses, there might be duplicate numbers. The possibility of duplicate addresses is negligible for a small number of nodes or using class A private addresses.

3.1.4 Protocol

Fig. 3.3 depicts the state transitions of a mobile node during its session in the MANET.

The protocol is as follows:

(1) When a mobile node switches to the ad-hoc mode, it begins periodic broadcast of state request packets, and changes from the UN-INITIALIZED state to the WAITING state. Note that only one-hop broadcast is necessary. Its MAC address may also be carried in the request packets, which is used by the responder to build a unicast reply;

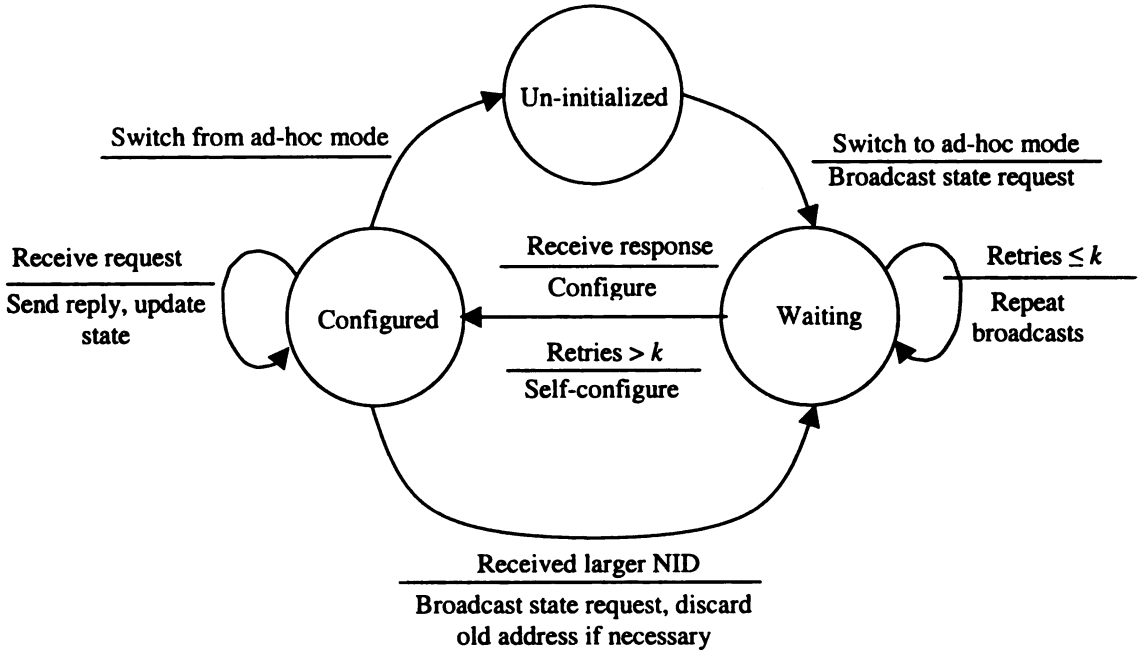


Figure 3.3 The finite state machine for prophet allocation

(2) The mobile node stays in the WAITING state and repeats state request for less than or equal to k times;

(3) If the mobile node receives a reply during that time, it configures itself with the IP address, initial state value, and NID contained in the reply, and changes to the CONFIGURED state;

(4) Otherwise, it chooses itself an IP address and NID randomly and a default state value as its initial state value and changes to the CONFIGURED state;

(5) During the CONFIGURED state, the mobile node repeats broadcasting HELLO messages, sends back replies on receipt of state request packets from other nodes, and updates its own state accordingly;

(6) If the mobile node receives a HELLO message with a larger NID, it discards its current IP address if necessary[2] and begins to broadcast state request packets, and re-enters the WAITING state;

(7) When the mobile node ends its session in the MANET, it switches out of the ad-hoc mode and changes to the UN-INITIALIZED state.

3.2 Performance Analysis

In this section, evaluation metrics for allocation performance are first defined. Then theoretical analysis of all four kinds of solutions is presented.

3.2.1 Metrics for performance evaluation

Distributed operation: A specific node in a MANET cannot be trusted as a configuration server as the one in DHCP because of its mobility, limited transmission

---

[2] The node discards its current IP address depending on which method is in use (please refer to subsection B).

range, and power supply. Failure of any number of nodes should not prevent autoconfiguration from working. Therefore, the algorithm must be distributed.

Correctness: All three scenarios discussed in Section 1.2 need to be considered. No two or more nodes with the same address could coexist for a long time. Conflict resolution should be initiated as quickly as possible if necessary.

Complexity: Taking into account limited computation power and memory capacity of mobile nodes, the solution should be as simple as possible. The solution may consist of several modules: allocation, conflict detection, state maintenance, etc. The complexity of each module should be carefully considered.

Communication overhead: Does the solution require broadcast in a MANET? Or does the solution only incur communication between neighboring nodes? Broadcast consumes too much bandwidth, which should be avoided as much as possible. Periodic broadcast is surely unacceptable.

Evenness: If the allocated addresses of most mobile nodes are clustered in a subset of the whole address range, the address distribution is uneven, which also means the probability of conflict is high. Thus, conflict detection may be launched several times and will lead to high communication overhead. Otherwise, if the distribution is even, the probability of conflict is low, which results in low communication overhead.

Latency: The time between the point when a node initiates autoconfiguration and the one when it is assigned a free IP address is referred as latency. The shorter the latency, the better. Broadcast leads to longer latency, while local communication results in shorter latency.

Scalability: The bandwidth consumed by broadcast is positively related to the number of nodes in the MANET. The latency is proportional to the diameter of the network, which is also positively related to the number of nodes. Therefore, if multi-hop broadcast is required in autoconfiguration, it has poor scalability. If most of communications happen locally, it has excellent scalability.

All of these metrics are closely related. The more even the distribution and the lower communication overhead, the shorter the latency and the better scalability. In other words, evenness and communication overhead are more important than the other metrics.

3.2.2 Performance comparison

Table 3.1 presents a comparison of the aforementioned methods. The first four rows are a characteristics summary of the four allocation algorithms. The last five rows focus on the qualitative evaluation of their performance.

Table 3.1 Characteristics and performance comparison

	Conflict detection	**Conflict free**	**Best effort**	**Prophet**
Network organization	Flat / Hierachical	Flat	Flat / Hierachical	Flat
State maintanence	Stateless	Partially stateful	Stateful	Stateful
Address conflict	Yes	No	Yes	No
Address reclamation	Unneeded	Needed	Needed	Unneeded
Complexity	Low	High	High	Low
Communication overhead	$O((n+l)\times k)$	$O(2l/n)$	$O((n+l)\times k)$	$O(2l/n)$
Evenness of distribution	Even	Possibly uneven	Even	Even
Latency	$O(2\times t\times d\times k)$	$O(2t)$	$O(2\times t\times d\times k)$	$O(2t)$
Scalability	Small	Medium / Small	Small	High

Conflict-detection allocation is the simplest method. No state is maintained. No address reclamation is needed. However, broadcast adopted in conflict detection leads to high communication overhead, high latency, and small scalability. For example, suppose the number of mobile nodes is n, the number of links is l, the average transmission time between two adjacent nodes is t, the network diameter is d (in terms of nodes), and the retry time is k. If there is no address conflict, the number of packets needed in conflict detection is at least $(n+l)\times k$, and the time spent is $2\times t\times d\times k$. Otherwise, the communication overhead will be more and the latency will be longer. The distribution of addresses is even because it uses random guess. Therefore, the probability of conflict is rare with a large address range and small number of mobile nodes.

Conflict-free allocation is simple in address assignment itself. However, a difficult problem arises in the management of the address pool. If a mobile node notifies others before it leaves or shuts down gracefully, it could release its IP address and address pool. However, if it leaves the MANET silently or shuts down abruptly, it will take away its IP address and address pool from the whole address range, which cannot be used by others. Thus, a mechanism for address reclamation is necessary, which is far more difficult and complicated than allocation. As other performance metrics, because most communication happens between neighboring nodes, it has low communication overhead, low latency, and medium scalability. For example, the packets needed are one-hop broadcast messages, which are proportional to the average number of degrees, i.e., $2l/n$. The latency is proportional to the round-trip time between two adjacent nodes, i.e., $2t$. However, the distribution of addresses depends on the allocation pattern, which is also important for determining its scalability. For example, if new nodes keep requesting the same

configured node for address pools, the size of the address pool will decrease exponentially. Thus the scalability worsens. This could be remedied by balancing the address pools among the configured nodes, which makes the management of address pools more difficult.

The performance of best-effort allocation is expected to be almost the same as that of conflict-detection allocation: high communication overhead, even distribution, high latency, and low scalability. However, because global state is maintained, the complexity is higher due to overhead incurred by state management and synchronization.

From the analysis above, we can arrive at the conclusion that the allocation algorithm must satisfy the following properties to achieve low latency and high scalability:

(1) Local communication (which means low communication overhead);

(2) Random assignment (which leads to even distribution).

In prophet allocation, when a new node joins the MANET, it just asks for one of its configured neighbors for its IP address and initial state. Thus, the first property is satisfied. With a carefully designed $f(n)$, the numbers in sequences may be distributed evenly in the integer range, and hence the second property may be satisfied. Thus the performance in communication overhead and latency of prophet allocation is expected to be almost the same as that of conflict-free allocation, while the complexity of the former is much lower than that of the latter, and the distribution of the former is even. As a result, the prophet allocation is suitable for large scale MANETs.

3.3 Simulation

According to our analysis in the last section, the performance of best-effort allocation is similar to that of conflict-detection allocation. Simulation of the former has been done in [42]. Therefore, we chose to implement the conflict-detection allocation proposed in [36] together with prophet allocation to compare their performance.

The simulation was done on ns-2 (version 2.1b8a) with CMU extension for ad hoc networks [51]. Statistics about communication overhead and latency in Scenario *A* and Scenario *B* were collected to show that prophet allocation outperforms conflict-detection allocation and best-effort allocation for large scale MANETs.

3.3.1 Simulation parameters

The random waypoint mobility model was adopted in the simulation [52]. After a node pauses for several seconds, a random destination point is chosen. Then the node moves towards that point at a maximum speed of 5 m/s, which is repeated until the end of simulation. The pause time is 10 seconds for 50 and 100 nodes, and 20 seconds for 150, 200 and 250 nodes, respectively. Different area sizes are also introduced to demonstrate the effect of density of nodes on the performance. For example, scenario files of 800×800, 1000×1000, and 1200×1200 were simulated for 100 and 150 nodes, scenario files of 1000×1000, 1200×1200, and 1300×1300 were tested for 200 nodes. The final results are the average of the results obtained with all the area sizes.

During the simulation, mobile nodes join the MANET every 30 seconds (for 50, 100 and 150 nodes) or 10 seconds (for 200 and 250 nodes) in the order of node ID. Because we aim to investigate the performance of large scale MANETs, no node departure is

introduced in the simulation. Another reason is that the number of nodes has no effect on the correctness of the algorithms.

We used DSR as the ad hoc routing protocol during the simulation. Both conflict-detection allocation and prophet allocation have no assumptions on the underlying routing protocols, because multi-hop broadcast and one-hop broadcast were implemented without the aid of routing protocols.

3.3.2 Simulation verification

To verify correctness of the implementation of allocation simulation, we first ran the simulation for 3, 4 and 5 nodes separately. The area size was chosen to make all the nodes connected in the topology. The simulation results are equal to our analysis, which shows that multi-hop broadcast and one-hop broadcast were correctly implemented in conflict-detection allocation (CDA for short in the diagrams) and prophet allocation (PA for short in the diagrams), respectively. The number of received packets at each node for 3-node simulation is illustrated in Fig. 3.4.

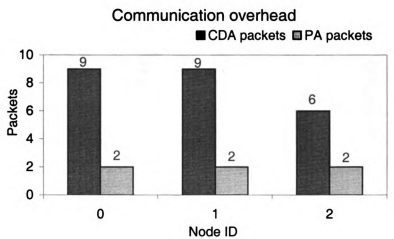


Figure 3.4 Received packets at each node for 3-node simulation

3.3.3 Communication overhead

Because every successfully received packet, either unicast packet or broadcast packet, must have consumed bandwidth (and power as well), we use it as the evaluation metric for communication overhead.

Fig. 3.5 shows the total number of packets received in 50-node simulation with different area sizes. The number of packets generated in conflict-detection allocation is 51.71 times of that in prophet allocation on average. As the density of nodes decreases, the communication overhead of conflict-detection allocation decreases because the link number decreases, and the network becomes partitioned during the simulation. The communication overhead of prophet allocation decreases because the neighboring nodes become fewer.

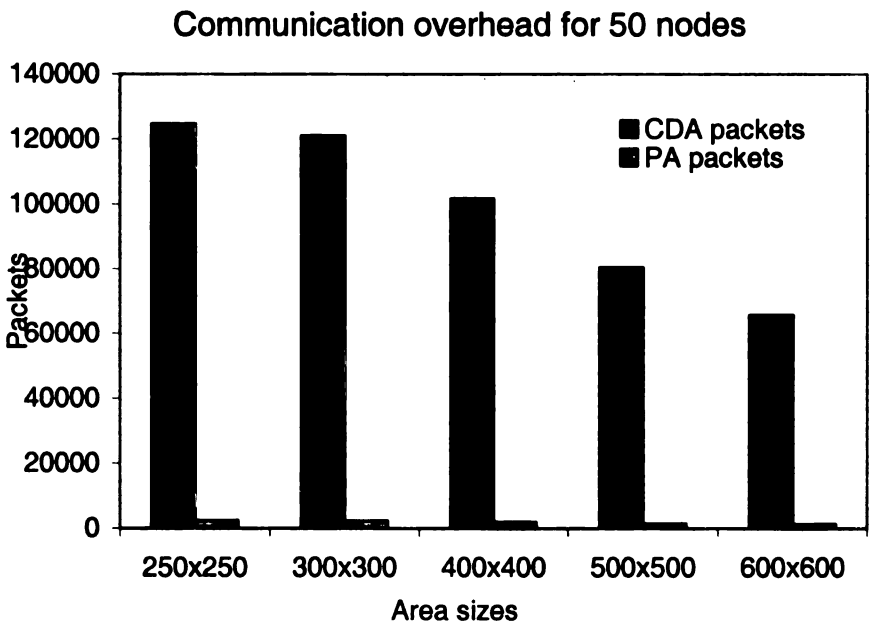


Figure 3.5 Communication overhead for 50 nodes

Fig. 3.6 shows the ratio of packets generated in conflict-detection allocation to those in prophet allocation for 50, 100, 150, 200, and 250 nodes, in contrast with a linear line. According to the diagram, the ratio of communication overhead in conflict-detection allocation to prophet allocation is approximately proportional to the number of nodes in the MANET, which means the more nodes, the more gain in communication overhead in prophet allocation.

3.3.4 Latency

During the simulation, the nodes participating in the conflict-detection allocation tried a maximum of 3 times for broadcast of duplicate address detection packets. While in prophet allocation, except for the first node, every node tried infinitely to broadcast state request packets until it received a state reply from its configured neighbor. The intervals for both are set to be the same[3], so we need only to compare their retry times.

Fig. 3.7 shows the average retry times in a 50-node simulation within different sizes of areas. Most nodes receive their responses during the first round of state request. As the node density decreases, the retry time increases.

---

[3] Of course, the interval for multi-hop broadcast in CDA should be much longer than that for one-hop broadcast in PA; however, they are difficult to compute in advance because of the dynamic topology.

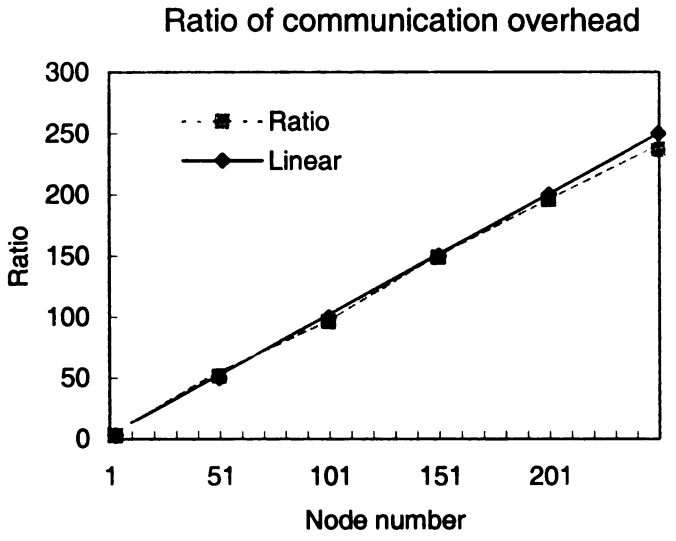


Figure 3.6 Ratio of communication overhead of CDA to PA

Fig. 3.8 shows the relationship of retry times and the node number. According to the diagram, the average retry time for prophet allocation fluctuates around 1.5 independently of the number of mobile nodes in the MANET. The retry time for 150 nodes is the highest because the node density is the lowest in the simulation, which means the nodes have to try many times in the beginning. Taken into account that the round-trip time between neighboring nodes is independent of network size, the latency for large scale MANETs is nearly the same as small scale MANETs, while the latency in conflict-detection allocation increases for large scale MANETs.

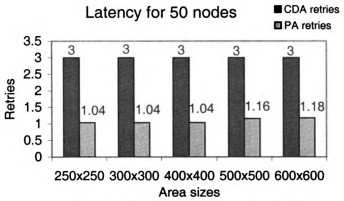


Figure 3.7 Latency for 50 nodes

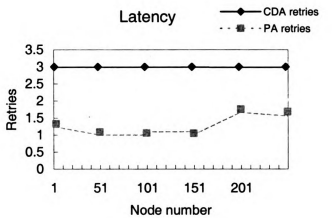


Figure 3.8 Latency for different node numbers

3.4 Summary

Based on studies of scenarios in IP address allocation and several allocation algorithms proposed by other researchers, we proposed prophet allocation for large scale MANETs, which achieves low complexity, low communication, even distribution, and low latency. Both theoretical analysis and simulation results were conducted to

demonstrate the superiority of prophet allocation over three other known methods. With a little more effort, prophet allocation is able to handle all the three scenarios efficiently.

CHAPTER 4 SECURE PROPHET ADDRESS ALLOCATION

There are have been several autoconfiguration schemes proposed for the MANET. All of them are based upon the assumption that the environment is secure. None of these approaches takes any security mechanism into consideration. When applied in real situations in which malicious nodes may exist, these autoconfiguration schemes will fail to function properly: either no new nodes will be allowed to join the MANET or there will be duplicate addresses in the network, which contradicts their original intention.

Most research effort on security in the MANET has been spent on secure routing protocols ([25] — [27]), and key management ([47] – [48], [53]), whereas the study on secure autoconfiguration in the MANET has not been explored. Note that the schemes for secure routing protocols may not be appropriate in secure autoconfiguration because the new node does not have a valid IP address before completion of autoconfiguration. It has to rely on multi-hop and one-hop broadcast, which is unrelated to unicast routing. It is the same reason why key management becomes difficult in such a scenario because most schemes for key management are based on the bindings of public keys with valid IP addresses.

Secure autoconfiguration is more difficult than autoconfiguration itself because the scheme must be invulnerable to different kinds of attacks. Furthermore, mobility adds more complexity to its design. In addition to the scenarios described in Chapter 3, there is one more scenario that needs consideration:

Scenario D: A more difficult scenario is that a MANET has a malicious cut-vertex, as illustrated in Fig. 4.1. The cut-vertex node means that the network will become disconnected without this node. Because all the traffic between the nodes in different parts has to go through node M, the autoconfiguration may fail if node M drops some particular packets or modifies their contents.

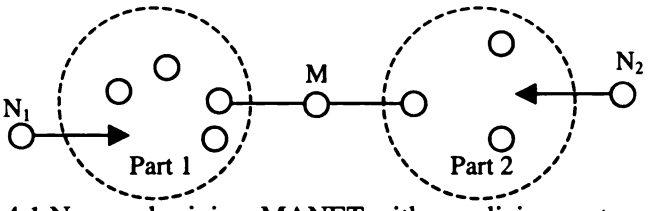


Figure 4.1 New nodes join a MANET with a malicious cut-vertex

To the best of our knowledge, this is the first effort on secure autoconfiguration in the MANET. The misbehaviors of all the autoconfiguration schemes in the presence of malicious nodes are analyzed in Section 4.1. Based on the authors' previous work, an extended algorithm, namely secure prophet address allocation, is proposed in Section 4.2. The proposed autoconfiguration algorithm is able to survive IP spoofing attacks and other attacks, as analyzed and exemplified in Section 4.3, whereas other schemes fail. The analysis is supported by the simulation results in Section 4.4. Section 4.5 concludes this chapter.

4.1 Misbehaviors in Insecure Environments

All existing autoconfiguration algorithms for MANETs assume secure environments. They cannot function properly in the case of attacks from malicious nodes. This section classifies the attacks and demonstrates the algorithms' misbehaviors.

4.1.1 Attacks at autoconfiguration

Generally speaking, the procedures of all the autoconfiguration schemes can be divided into two phases:

Phase 1: Address request

The new node wants a spare IP address. It either chooses one for itself and waits for approval from the allocator(s), or asks the allocator(s) for one.

Phase 2: Address configuration

The new node receives responses from the allocators, either positive or negative. In the former case, the new node can assign itself the IP address and other parameters (such as netmask); otherwise, it repeats phase 1.

Attacks can be launched during both phases.

During phase 1, a malicious node can impersonate a new node and continue requesting free IP addresses. Suppose the IP address range is small. Many IP addresses will be wasted in stateful allocation, which leads to a scarcity of spare IP addresses. As a result, the prospective new nodes will be denied the ability to join the MANET, which is actually a DoS attack. Both best-effort allocation [42] and conflict-free allocation [40] are vulnerable to DoS attacks because either the spare IP address or the address pools will be exhausted on receipt of multiple address requests.

However, a DoS attack can be easily defeated. One insufficient solution is that the IP address range is set to be very large to deter this attack. Perhaps a better solution is that the allocator, which becomes aware of the address scarcity, initiates a "clean up" procedure to reclaim those wasted addresses. The "clean up" procedure is not specified in detail in [36] - [38]. However, it is expected to be both time-consuming and bandwidth-

consuming according to the scheme in [41], which may undermine normal communications.

During phase 2, a configured malicious node can act as an allocator and always send back a negative response, which keeps the new node repeating phase 1; or it sends back a false positive response, which leads to address conflicts in the MANET. All existing allocation schemes are unable to detect and prevent this attack.

For example, in a conflict-detection allocation ([36] – [38]), the new node chooses a random address (say x) and broadcasts a conflict detection packet throughout the MANET. Any veto from a node will prevent it from using this address. Even if the chosen address is not in use, the malicious node can impersonate the node that has occupied the same IP address and keep replying with vetoes, which leads to the failure of allocation, as illustrated in Fig. 4.2.

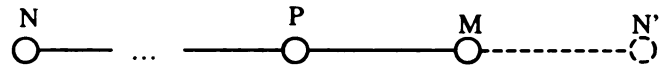


Figure 4.2 An example of an IP spoofing attack

In Fig. 4.2, N represents the new node, and M represents a malicious node. Node P is a neighbor of node M. Although node P may be aware that it has no neighbor with the address of x (by means of neighbor detection), it still thinks that the veto message is forwarded by node M from another node (node N'). Moreover, M can declare that the veto originates from another innocent node if there is no signature scheme applied to the message. Any allocation algorithms that utilize the conflict detection mechanism cannot defeat this *IP spoofing* attack.

In the best-effort allocation, a malicious allocator can always give the new node an occupied address, which leads to repeated broadcasts of conflict detection packets

throughout the MANET and the rejection of the new node. Moreover, this attack is difficult to be differentiated from the case when the global allocation states are not properly synchronized among all the mobile nodes because of the unreliable broadcast transmission. In conflict-free allocation, a malicious allocator can give a non-disjoint address pool to the new node, which affects the allocation to prospective new members and causes a huge problem for the MANET. This attack can be named the "*state pollution*" attack.

4.1.2 Difficulties to prevent IP spoofing attacks

IP spoofing attacks are fatal to the allocation approaches that utilize a conflict detection mechanism. However, this attack is easy to launch because the malicious node can construct a whole IP packet including the IP header, so it can put a fake source IP address in the response. In Linux, the programmer can even build up a data link layer packet, which may contain a fake source MAC address as well.

IP spoofing is difficult to detect and prevent even with the aid of secured switches and routers in hardwired networks, let alone in MANETs where an infrastructure is absent. There are some methods proposed to detect IP spoofing attacks. However, these methods can be easily circumvented.

One method that seems promising is a cryptographic method such as using a digital signature. For example, the bindings of the IP address and public key are stored in the Certificate Authority (CA) in the MANET and the public keys are certified by the CA, like the scheme used in [47] [48]. However, this scheme may not fit the scenario where nodes are free to join and leave and have dynamic assigned IP addresses. The public-key management proposed in [53] is not secure enough because the malicious node can

generate a public/private key pair for the fake IP address and then certify the public key with its own.

Actually, CA and cryptographic schemes cannot be utilized in secure autoconfiguration because the CA itself needs a valid IP address first. Thus, to obtain a valid address for CA, we need another CA to validate the control messages signed by the allocators, which forms a circle[4].

Even if there is a CA in the MANET, the allocation algorithm with conflict detection scheme cannot prevent IP spoofing attacks, which can be illustrated in the following example. Suppose a new node requests the address of x in the conflict detection packet. The new node cannot use x in communications unless it is sure that x is not occupied by others. Once the malicious node receives the conflict detection packet, it immediately changes its address to x temporarily, generates a public/private key pair and registers the key pair with address x at CA, and then sends the veto signed with the public key bound with the fake IP source address back to the new node.

The reason that CA is difficult to be applied in secure autoconfiguration is that it regards the IP address as a node's identifier and binds the public/private key pair with the IP address. However, this kind of identifiers is allocated dynamically in the scenarios of autoconfiguration. Moreover, there seem no other versatile candidates for identifiers. For

---

[4] Of course we can specify that the first node in the MANET as the CA server and let it choose a random address. Because it is the first node, no conflict detection is necessary, and thus there will be no IP spoofing attacks against it. However, this is not the general case, and a centralized CA is not desirable in the MANET. We can also place CA server(s) in the MANET and specify the fixed address(es), which is beyond our topic on autoconfiguration.

example, the physical address is not appropriate due to the following reasons: (1) most wireless NICs are detachable so the users may change the wireless NIC during the session of the MANET; (2) the user is able to modify the MAC address of the wireless NIC; (3) some wireless NICs are manufactured from small companies that may not allocate the addresses according to IEEE standard, so the uniqueness of MAC addresses cannot be guaranteed. Although the users are most unlikely to change the CPU during the middle of communications, the built-in identifiers in some CPUs are not proper either, because in a heterogeneous MANET, not all CPUs have identifiers. Moreover, the built-in identifiers in CPUs are hidden from outsiders and it is still easy for a malicious node to forge a fake CPU identifier.

4.1.3 Misbehaviors of prophet address allocation

The original prophet address allocation scheme can defeat a DoS attack easily, as addressed below in (1). However, it cannot detect the IP spoofing attack and the "state pollution" attack:

(1) A malicious node pretends to be a new node, and keeps requesting free IP addresses.

According to the original scheme, a new distinct address will be generated at each request, which still looks like a normal allocation, except that the number of nodes in the MANET seems to be more than the actual number. However, according to the properties of the integer sequences generated with the partition function, the wasted addresses will be reused later. Thus, allocation will not be disabled in the presence of such DoS attacks.

(2) A malicious node refrains from sending a response.

As long as the new node has a good neighbor, the new node can receive a response and configure itself with the parameters provided by the good neighbor.

Even if all the neighbors are malicious and keep silent on receipt of the request, the new node may continue moving around and it will receive a response eventually.

(3) A malicious node occupies the channel and prevents good neighbors from sending responses, which is related to fair channel sharing in the MAC layer. This kind of attack has been well studied and hence is not covered here ([54]).

(4) A malicious node answers with invalid parameters.

In the original design of prophet address allocation, the following parameters must be contained in the response (without optimization): 1) the seed value for the whole MANET; 2) the exponential array; 3) the index of the increasing exponential; 4) The source address of the responder (implicit), as illustrated in Fig. 3.2. These parameters are not sufficient to validate the correctness of the allocation because the index array keeps changing during the allocations.

One of the most difficult problems is that the malicious node does not update its state and tries to assign the previously allocated address to a new node, which is the "state pollution" attack and is very difficult to detect.

Likewise, because the seed value (a) is known to all the nodes, a malicious node can generate a seemingly reasonable index array that in fact belongs to another node and allocate the generated address to the new node, which is the IP spoofing attack.

In summary, IP spoofing attacks and "state pollution" attacks cannot be defeated without an extension to the original prophet address allocation scheme, which is described in the next section.

4.2 Secure Prophet Address Allocation

In this section, we first provide reasonable assumptions required for our scheme to be survivable in the presence of malicious nodes, and then describe the extension added to the original scheme with different scenarios in mind. The last subsection discusses the performance of secure prophet address allocation.

4.2.1 Assumptions

We assume that all the configured neighbors of the new node are not malicious nodes. The assumption can be easily satisfied because all the nodes are able to move arbitrarily in the mobile ad-hoc networks. Even if several malicious nodes conspire and encompass a new node for a period, other good nodes can enter the radio transmission range of the new node unexpectedly and become its neighbors.

In the case that the MANET is constructed from scratch, we could derive from the assumption that the first node initiating the MANET must be a good node because the second node has only one neighbor.

From the discussion in Section 4.1, we can see that the conflict-detection allocation and best-effort allocation will fail even if the assumption is satisfied, because one malicious node in the network can launch IP spoofing attacks against new members. For the conflict-free allocation in which the new node can collect all the replies from its neighbors, the assumption is also necessary for it to survive "state pollution" attacks.

Another implicit assumption is that there is an interval between two new nodes joining the MANET, which is utilized to simplify the description of the secure allocation. The minimum value of the interval is the traversal time along the diameter of the network,

which is defined to be 0.4 second for a medium-scale network with the diameter of 10 nodes according to [9]. In Subsection 4.3.1, we provide solutions to remove this assumption.

4.2.2 Authenticity of the seed value

In a secure environment, a new node can immediately configure itself with the parameters contained in the first response. However, in an insecure environment, this response may come from a malicious node, so the new node must wait for a period to collect as many responses as its neighbors to validate them.

With all the responses, the new node needs to find the correct seed value (a) first. Because the seed value is chosen by the first node in the network and kept constant during allocation, the seed value in all the replies should be the same.

If the malicious neighbors conspire and put a false seed value in their replies, according to the above-mentioned assumption and the fact that a good neighbor always provides the correct seed value, the new node will get different seed values. If the malicious nodes put different NIDs in their replies, the new node may think it is on the edges of two networks and can randomly decide which network it is going to join. If the malicious nodes put the correct NIDs, it seems to the new node that two different networks unfortunately choose the same random NID because two seed values usually mean two networks. In the latter case, the new node can first choose one network randomly to join and then choose a new NID for its network. The remaining work is the conflict resolution occurred on merger of two networks described in subsection 4.3.2 .

4.2.3 Extension

In the original prophet allocation, there is an implicit assumption that all the nodes update their internal states correctly. In an insecure environment, a malicious node can stop updating its state or forge a false state, which renders difficulties in determining the authenticity of the other parameters in the reply. However, we can change the implicit assumption to an explicit rule by broadcasting an announcement. If a node does not obey the rule, its reply will be discarded by the new node; if it obeys the rule, it will guarantee that no duplicate address is generated.

Thus, we require that the reply contain two more parameters:

(1) The initial exponential array, which is the exponential array that the allocator (new node's parent node) received from its own allocator (the new node's grandparent node, except for the first node in the MANET) and was used to compute its own address. Thus, the source address can be validated[5].

(2) A new state variable: priority. The priority indicates the freshness of the state contained in the reply. Every time the node updates its state, the priority is increased by 1. The larger the number is, the fresher the state will be. This number should be equal in all the good configured nodes, and synchronized by multi-hop broadcast of the announcements.

In summary, the reply contains the following parameters:

(1) The seed value for the whole MANET (a);

(2) The exponential array ($e[1..n]$, may be fake);

---

[5] The source address is not truly "validated". Please refer to the discussions in Section 5.

(3) The index of the increasing exponential (c)[6];

(4) The source address of the responder (x, may be fake).

(5) The initial exponential array ($i[1..n]$, may be fake);

(6) Priority (p).

The relationships among these state variables are

$$x = f(a, i[1..n]) \tag{1}$$

$$e[j] = \begin{cases} i[j], j < c \\ p, j = c \\ i[j] = 0, j > c \end{cases} \tag{2}$$

Equation 1 means that the allocator's source address can be computed with the seed value and the initial exponential array as input to the stateful partition function. Equation 2 means that every time the state is updated (the increasing exponential is increased by 1), the priority is increased by 1.

4.2.4 Authenticity of the priority

The correctness of secure prophet address allocation is based upon the authenticity of the priority. Because the priority keeps changing and is synchronized by multi-hop broadcast that is unreliable, even a good node may miss an ACK packet containing the up-to-date priority. Thus, it will fail to update its state, and hence its reply will be rejected by the new node. As a result, none of the replies has the same priority value in the worst case. To solve this problem, we can designate the new node choose the highest one. We

[6] The index c may be fake, but it does not matter, as explained in Section 5.

also require that all the nodes monitor the forwarding of the ACK packets to add reliability to broadcasts.

4.2.5 Protocol

Fig. 4.3 depicts the state transition of a mobile node during its session in the MANET with secure prophet address allocation in Scenario *A* (other scenarios will be discussed in Subsection 4.3.2).

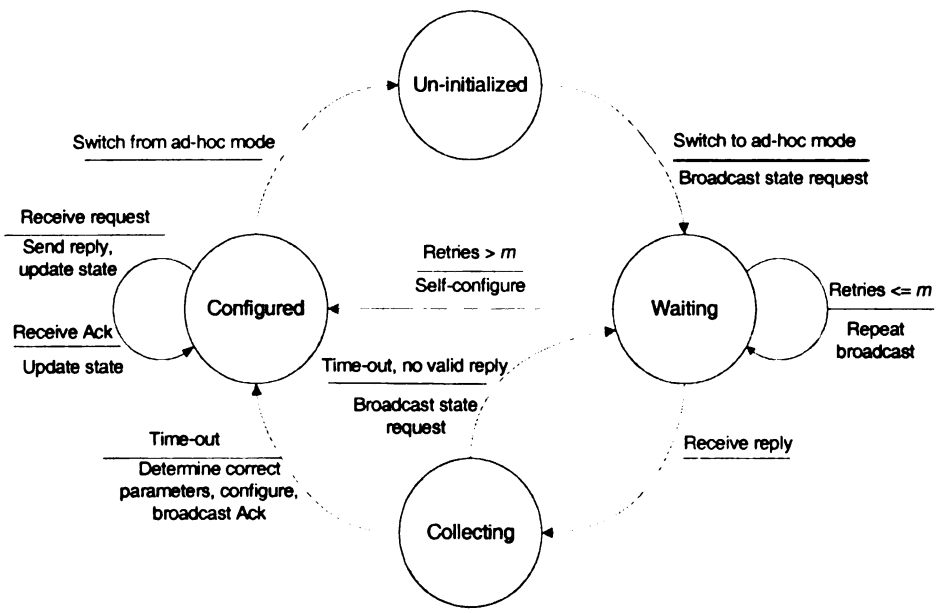


Figure 4.3 The finite state machine for secure prophet address allocation

The new protocol works as follows:

(1) When the node switches into the ad-hoc mode, it begins to broadcast state request packets periodically, and changes from the UN-INITIALIZED state to the WAITING state.

(2) The mobile node stays in the WAITING state and repeats state requests for less than or equal to m times;

(3) If it receives a reply during that time, it changes from the WAITING state to the COLLECTING state, and stays in the latter to collect as many responses as possible;

(4) When the COLLECTING state times out, the node validates all the replies that it has received: first, the new node determines the correct seed value (a) and priority (p) for the whole MANET. If there are different seed values, the new node chooses one randomly; if there are different priority values in the remaining replies, it chooses the highest value among all the replies, and then discards all the replies with different seed values. For the remaining replies, it first determines the relationship of parameters according to Equation (1), and discards all the replies that fail the test. It then determines the correctness of the index of increasing exponential according to Equation (2), and discards all the replies with incorrect parameters. For the remaining replies, it chooses one randomly and uses the seed value, initial exponential array and index contained in the reply and the highest priority value to generate its address, and then broadcasts an ACK packet with the priority value plus 1, and enters the CONFIGURED state.

(5) If there are no valid replies, it starts to broadcast state request packets again and returns to the WAITING state;

(6) If the node fails to receive any (valid) replies for m times, it chooses itself an IP address and NID randomly and a default state value as its initial state value, and changes to the CONFIGURED state;

(7) Within the CONFIGURED state, the mobile node repeats broadcasting HELLO messages, sends back replies on receipt of state request packets from other nodes, and updates its own state accordingly;

(8) When the configured node receives an ACK packet with a larger priority, it updates its state according to the new priority value;

(9) When the mobile node ends its session in the MANET, it switches out of the ad-hoc mode and changes to the UN-INITIALIZED state.

The parameter m and the value for timeout could be preset heuristically and adjusted according to the parameters such as node density, bandwidth, and packet loss rate.

4.2.6 Performance

With the introduction of ACK packets, the performance of secure prophet address allocation is degraded to the level of conflict-detection allocation and best-effort allocation. In a practical deployment of a MANET, users have to make a decision on the trade-off between overhead and security. It seems a feasible option that the level of security is a user-configurable parameter reflecting real circumstances, and that the autoconfiguration algorithm switches between the insecure scheme and the secure scheme accordingly.

4.3 Invulnerability Analysis

The invulnerability of the secure prophet address allocation is analyzed with some typical examples in this section.

4.3.1 Invulnerability in Scenario A

With the aforementioned modifications, the secure prophet address allocation can function correctly in Scenario *A*. We use the simple network illustrated in Fig. 4.4 as an example.

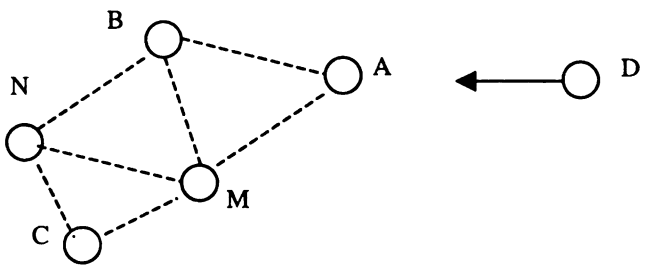


Figure 4.4 A simple MANET

In Fig. 4.4, node N is a new node and is joining the MANET, node D is another new node that is going to join the network. Node M is a malicious node and all the others are good ones. Node A is the first node in the network and 2 hops away from node N. The exponential array has 4 elements, as that in Fig. 3.2, and the address range is large for simplicity of explanation.

(1) IP spoofing attack

Suppose that nodes B, C, and M join the network and obtain their addresses from node A successively. Their states before node N's arrival are given in Table 4.1.

Because node A is 2 hops away from the new node, node M can impersonate node A when sending a reply to node N[7]. Because node A has the address of *a*, the initial

[7] Suppose that node A does not care about receiving a packet with its address as the source address from its wireless NIC here.

exponential array contained in the reply from node N must be (0, 0, 0, 0); otherwise, the parameters cannot pass the test of Equation (1) in Subsection 4.2.3.

Table 4.1 The state of nodes in example 1

Node	Address	Initial array	Current array	Priority
A	a	(0, 0, 0, 0)	(4, 0, 0, 0)	4
B	a+3	(1, 0, 0, 0)	(1, 4, 0, 0)	4
C	a+5	(2, 0, 0, 0)	(2, 4, 0, 0)	4
M	a+9	(3, 0, 0, 0)	(3, 4, 0, 0)	4

The priority values in good neighbors B and C are synchronized to 4, which is the current largest value. To make sure the reply may be chosen by node N, node M must set priority to at least 4 in the reply (here we use priority of 4, which is similar for other higher priority values). Thus, if the reply from node M is chosen by node N, it will get the address of a + 17. In the original prophet address allocation, when node D approaches node A later, node A will use the same state parameters in allocation, which means that node D will get the same address of a + 17. Therefore, duplicate addresses are generated with IP spoofing attacks. However, in secure prophet allocation, node N will flood an ACK packet containing the priority of 5 throughout the network. Once node A receives the ACK packet with a larger priority value, it will change its priority to 5. Eventually, node D will get the address of a + 33 and thus there will be no duplicate addresses. In summary, with the introduction of the priority value and the ACK packet, node A can skip to the next state if it is impersonated.

If node D joins the network before the flooding of the ACK packet, there will still be duplicate addresses. We assume that there is an interval between two new nodes joining the network. This assumption can be removed with the following methods. If the average arrival rate of new nodes is below a threshold value, the new node waits for a random

delay before initiation of normal secure allocation; otherwise[8], in addition to the random delay before allocation, every new node caches all the "correct" replies that passed Equations (1) and (2) in Subsection 4.2.3, chooses one tentatively, puts its initial exponential array in the ACK packet that it broadcasts, and waits for some time to finalize the allocation. Due to Equation (1), the initial array must correspond with the address of the new node. If the new node receives an ACK packet with the same initial array during the second waiting time, it chooses parameters from another cached reply randomly to configure itself and finalize the configuration. If it is the only "correct" reply, it repeats address allocation again. Otherwise, if there is no other ACK packet, it finalizes the choice.

For example, nodes N and D join the network at the same time and choose the same random delay in Fig. 4.4. If node N chooses the reply from node M and node D chooses the reply from node A, both of them will get the same address of $a + 17$ and the same initial array of (4, 0, 0, 0). If they both put the initial array and priority of 5 in the ACK packet and wait for some time a bit longer than 0.4 second, node N will receive the ACK packet from node D and vice versa. Node N then ignores the previous reply from node M, chooses the reply from either node B or node C. Suppose it chooses the reply from node B, it will get the address of $a + 163$ and initial array of (1,4,0,0). Because the priority value of node B is changed to 5 automatically, so the next allocated address from node B will be $a + 487$, which means there will be no duplicate addresses allocated from node B.

---

[8] The average arrival rate should be less than a maximum value, for example, 1/0.4 sec (= 2.5/sec) for a medium-scale MANET, which means that 150 nodes join the network in one minute. Otherwise, according to queuing theory, there will be infinite new nodes waiting to be configured if new nodes keep arriving.

If node D has only one reply from node A, it repeats allocation until it encounters more than one neighbor in the dense network during movement.

With the above-mentioned scheme, node M cannot prevent new nodes from joining the network by means of broadcasting false ACK packets. For example, after allocation to node N, node M will know that its reply is chosen by node N from the ACK packet. If it immediately broadcasts an ACK packet with the same address and initial array, node N will choose a "correct" reply from another neighbor. If the allocation happens far away from node M, node M has no idea of the number of neighbors of the new node. Even though node M can forge an ACK packet, the new node can still configure itself properly.

(2) "State pollution" attack

Suppose that nodes B and M obtain addresses from node A successively, and then node C acquires address from node M and these allocations are normal. Their states before node N's arrival are given in Table 4.2.

Table 4.2 The state of nodes in example 2

Node	Address	Initial array	Current array	Priority
A	a	(0, 0, 0, 0)	(4, 0, 0, 0)	4
B	a+3	(1, 0, 0, 0)	(1, 4, 0, 0)	4
C	a+109	(2, 3, 0, 0)	(2, 3, 4, 0)	4
M	a+5	(2, 0, 0, 0)	(2, 3, 0, 0)	3

Suppose node M does not update its priority value from 3 to 4 in the reply to node N. In the original prophet address allocation, node N will also obtain the address of a + 109, which conflicts with node C. However, in the secure prophet allocation, since the reply from node M has smaller priority value, the reply will be ignored by node N. In summary, the malicious node is forced to update its state after allocation.

(3) Combination of IP spoofing attack and "state pollution" attack

Suppose that nodes B, C, and M join the network and obtain addresses from node A successively. Their states before node N's arrival are given in Table 4.1.

Node M wants to impersonate node A in the allocation to node N, so it uses the address of a and the initial array of (0, 0, 0, 0). If it refrains from updating the priority value from 3 to 4, its reply will be discarded. Thus, it must put the priority of at least 4 in the reply. If it modifies the index of the increasing exponential of c from 0 to 1, which means the current array is (0, 4, 0, 0) instead of (4, 0, 0, 0). According to Equation (2), there should be no 0 on the left of the underlined value in the array, as exemplified in Fig. 4.2, so the reply will be discarded. Even if the reply is adopted by node N, node N will obtain the address of a + 82, the initial array of (0, 4, 0, 0), and index of 2, which will generate different exponential arrays such as (0, 4, 5, 0), (0, 4, 6, 0), and so on.

Another example is that node M impersonates a non-existing node with the address of a + 33, which is supposed to be the second address that node A is going to generate. To convince new node N the source address, node M must put the initial array of (5, 0, 0, 0) in the reply. If node M sets the index c to be 0, then it impersonates node A, which was discussed before. If it sets index c to be 1, then node N will get the address of a + 2593, the initial array of (5, 4, 0, 0), and the index of 2, which will generate exponential arrays such as (5, 4, 5, 0), (5, 4, 6, 0), and so on. When node D approaches node A after node N joins the network, node D will get the address of a + 33 and initial array of (5, 0, 0, 0) and index of 1. It will generate arrays of (5, 5, 0, 0), (5, 6, 0, 0), and so on. In summary, there are no duplicate exponential arrays, and thus no duplicate addresses.

(4) A malicious neighbor impersonates many non-existing neighbors (Sybil attack)

Suppose that nodes B, C, and M join the network and obtain addresses from node A successively. Their states before node N's arrival are given in Table 4.1.

Node M wants to impersonate more than 2 non-existing neighbors, so it appears that several malicious nodes conspire together, which is called a Sybil attack [55]. If the forged replies contain different seed value, it is the same case discussed before. If the malicious node puts different priority values in different forged replies, then the highest value provided from both good neighbors and bad neighbors will be chosen. If the malicious node impersonates different non-existing nodes and put different state parameters, the new node can still generate unique IP addresses. For example, node M sends three different replies, as illustrated in Table 4.3:

Table 4.3 The state parameters in forged replies from M

Message	Address	Initial array	Current array	Priority
1	a+33	(5, 0, 0, 0)	(5, 4, 0, 0)	4
2	a+73	(3, 2, 0, 0)	(3, 2, 5, 0)	5
3	a+163	(1, 3, 0, 0)	(1, 3, 5, 0)	5

Among all the replies, node N will choose 5 as the current priority. Since all the source addresses and the corresponding initial arrays satisfy Equation (1), any of the initial arrays could be used in generation of the new address. Suppose node N uses the initial array of (5,0,0,0) and new priority value of 5, it will get the address of a + 7777. After allocation, the current priority value will be increased to 6, which means that the malicious node cannot use the same initial array of (5,0,0,0) and priority of 5 to generate duplicate addresses.

(5) Other attacks

There are other kinds of attacks. For example, if a malicious node keeps broadcasting ACK packets, many states will be wasted. However, the allocation scheme still works.

Although frequent multi-hop broadcasts will degrade the performance of a MANET, this attack is not related to secure autoconfiguration itself.

The new node may be a malicious node and may refrain from sending back an ACK packet. However, its behavior can be monitored by all the neighbors. We can specify that all the neighbors start a timer and wait a random interval for an ACK packet. When the timer expires, the neighbor broadcasts a gratuitous ACK packet containing its current priority value. To abide by the rules utilized to remove the assumption on the arrival interval, the neighbor puts in the gratuitous ACK packet the current array it generates for the new node (which is the initial array for the new node) and the current arrays contained in the overheard reply packets from other neighbors to the new node.

4.3.2 Mechanism for merger of two networks

As to the case of the merger of two independent MANETs, the two networks may have duplicate addresses even without malicious nodes, so conflict resolution mechanism proposed in the original Prophet address allocation is still necessary: on receipt of HELLO messages with a different NID and seed values, a node on the edge of two networks computes the addresses allocated in both networks and notifies possible conflicting nodes in one network to change their addresses. All the nodes in one network acquire new states from the other network in the following allocation. They can utilize the mechanism in IP address handoff scheme in next chapter to repair routing fabrics and preserve communication states.

However, to prevent the communication overhead in address handoff scheme caused by false HELLO messages broadcast from malicious nodes, the node detecting the merger in one network can choose another random seed value that has no conflicting

address with the seed value of the other network, and broadcasts the new seed value throughout its network. All the nodes in its network just add the offset value to their current addresses and obtain new addresses. Thus, it is possible to update routing tables and construct NAT tables without additional communication overhead.

4.4 Simulation Experiments

According to our analysis, conflict-detection allocation and best-effort allocation will fail to function in the presence of IP spoofing attacks[9]. Therefore, they were ignored in our simulation. To validate the survivability of secure prophet address allocation, we only need to compare it with the original prophet address allocation to see if it achieves uniqueness of IP address allocation in the presence of different kinds of attacks.

The simulation was done on ns-2 (version 2.27) with the CMU extension for ad hoc networks. Statistics about the number of duplicate address pairs in Scenario *A* were collected to show that the secure prophet allocation is able to survive IP spoofing attacks, "state pollution" attacks, and Sybil attacks in insecure environments. According to our analysis above, if the secure scheme works in Scenario *A*, it will also work in Scenarios *B* and *C*.

4.4.1 Simulation setup and verification

A simple simulation of 9 nodes in a 3×3 grid is conducted to verify the correctness of the implementation, as illustrated in Fig. 4.5.

---

[9] There will be repetitious flooding of conflict detection packets throughout the MANET and no new nodes will be allowed to join the network.

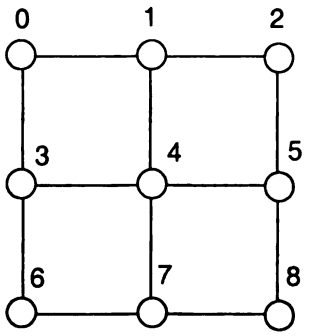


Figure 4.5 A simple simulation

In Fig. 4.5, the node 4 is specified as a malicious node, all the others as good ones. They join the network in the ascending order of their IDs.

To simplify the simulation of IP spoofing attacks, the malicious node "borrows" the state of its allocation, i.e. node 3. During the simulation of prophet address allocation, node 5 gets the same state as node 6, which results in duplicate addresses between nodes 5 and 6. During the simulation of secure prophet allocation, node 3 skips to the next state after the ACK packet is flooded throughout the network, and thus nodes 5 and 6 have different addresses.

For the simulation of "state pollution" attacks in prophet address allocation, the malicious node refrains from updating its state after allocation, which results in duplicate addresses between nodes 5 and 7. For the simulation in secure prophet allocation, in addition to "state pollution" attacks, we also implemented Sybil attacks in which the malicious node impersonates a random number of non-existent nodes and sends randomly forged reply messages. However, with the introduction of ACK packet in secure allocation, all the nodes in the network still have unique addresses.

4.4.2 Simulation results

The survivability of both allocation schemes is compared in a simulation of 50 nodes moving around randomly in the area of 1500×1500 with random waypoint mobility model. The size of the area is chosen to make the network is densely connected to test scenario *A*.

Table 4.4 shows the number of duplicate address pairs with IP spoofing attacks. Table 4.5 shows the number of duplicate address pairs with "state pollution" attacks and Sybil attacks. The number of malicious nodes varies during each run of simulation. Generally speaking, the number of duplicate addresses generated in prophet address allocation increases with the number of malicious nodes. However, no duplicate addresses are generated in secure allocation.

Table 4.4 The number of duplicate address pairs with IP spoofing attacks

Percentage of malicious nodes	Prophet address allocation	Secure prophet allocation
10%	4	0
20%	5	0
25%	3	0
33%	3	0
50%	13	0

Table 4.5 The number of duplicate address pairs with "state pollution" attacks and Sybil attacks

Percentage of malicious nodes	Prophet address allocation	Secure prophet allocation
10%	0	0
20%	1	0
25%	3	0
33%	9	0
50%	14	0

4.5 Summary

Secure autoconfiguration assures uniqueness of address allocation in the MANET in insecure environments, which is the first step towards secure MANETs in wide applications. However, most research effort has been focused on secure routing protocols, secure communications, and key management, whereas secure autoconfiguration has been neglected. Nevertheless, the latter is still difficult because we cannot simply combine cryptographic methods with pre-existing address allocation schemes due to the reasons in Subsection 4.1.2.

This is the first effort to propose a secure autoconfiguration for MANETs. Based on studies of insecure scenarios, categories of attack schemes, and our previous work, we extended our prophet address allocation so that it survives DoS attacks, IP spoofing attacks, "state pollution" attacks, and Sybil attacks. Thus, new nodes will be able to join the MANET without being assigned duplicate addresses in insecure environments. In addition to the simplicity of the algorithm, secure prophet address allocation is especially suitable for the environments in which a CA does not exist or a pre-existing trust relationship among nodes cannot be built. Both theoretical analysis and simulations were conducted to demonstrate the survivability of the proposed scheme.

CHAPTER 5 IP ADDRESS HANDOFF IN THE MANET

There is a problem associated with IP address assignment of a mobile node is that the IP address may change during its session in the MANET. IP address change is not a serious problem in hardwired networks because the IP address of a host is either statically configured or dynamically allocated by a DHCP server. It usually does not change its IP address during a session unless it reboots. However, because the nodes in the MANET are free to move arbitrarily, IP address change happens more frequently when applied with autoconfiguration, global connectivity, and hierarchical addressing schemes.

There are several scenarios in which a mobile node will change its IP address:

(1) Merger of two partitions of a network

If some mobile nodes in the MANET move out of the transmission range of the other nodes, the network becomes partitioned as illustrated in Fig. 5.1(a). Because these nodes may not be aware of the partition, they may still use the previous allocated addresses. If the IP address of a node (say node A) in one partition is allocated to the new node (say node B) in the other partition, address conflict occurs when these two partitions become connected, as illustrated in Fig. 5.1 (b). One example is when some attendants leave a meeting room for a short period and then return during a presentation session. The prophet address allocation is insensitive to this scenario, while the nodes in one partition may need to change their addresses with DDHCP [42].

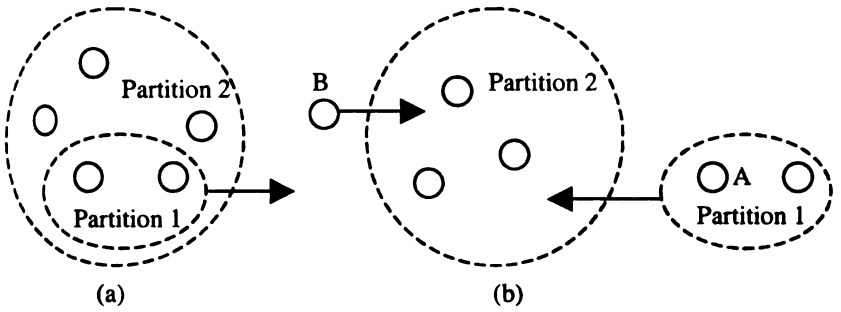


Figure 5.1 A network is partitioned and then merged later

(2) Merger of two independent MANETs

The second scenario is that two independently configured MANETs merge. Because these two networks are autoconfigured separately, there may be some duplicate addresses in both networks, such as node A in MANET 1 and node B in MANET 2 in Fig. 5.2. Thus, one needs to change its address due to the merger.

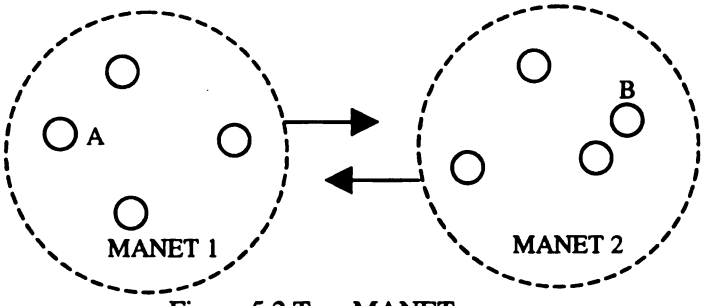


Figure 5.2 Two MANETs merge

(3) Merger of a MANET with a LAN

The third scenario is that a MANET merges with a LAN that has an "ad-hoc" mode Access Point (AP)[10]. The mobile nodes (such as node R in Fig. 5.3) that are within transmission range of the AP of the LAN may want to use the configuration information

[10] For example, the AP from Ericsson is able to support both ad hoc and infrastructure modes simultaneously.

(e.g., a free IP address in the LAN and the default router) broadcast by the AP to configure itself and function as a relay node. As a result, the MANET becomes connected to the Internet [56] [57]. Furthermore, if the MANET and the LAN use the same private address range, there may be duplicate addresses in both the MANET and LAN, such as node A and node B in Fig. 5.3. Because the hardwired host in the LAN may not be willing to release its address, the node in the MANET will have to change its address.

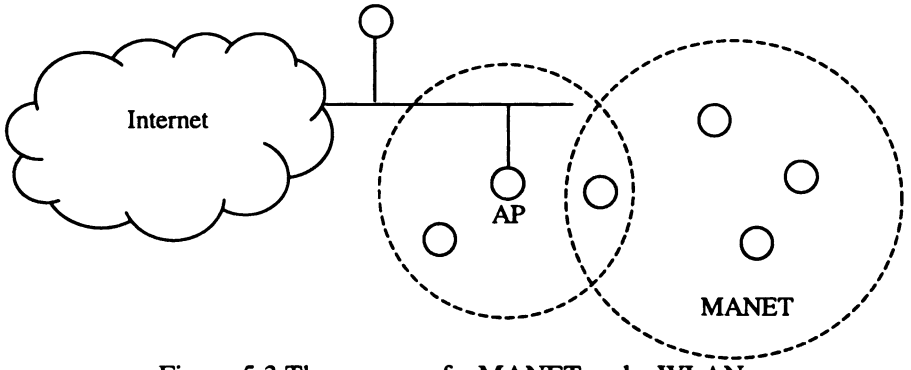


Figure 5.3 The merger of a MANET and a WLAN

(4) A MANET with a hierarchical addressing scheme

In the network where a hierarchical addressing scheme is deployed [58], the mobile nodes are divided into different clusters, each of which has a unique subnet address. When a mobile node (such as node A in Fig. 5.4) moves from one cluster to another cluster, it will change its address to one with the corresponding subnet address.

Although many autoconfiguration algorithms for MANETs have been proposed to allocate a mobile node a free IP address on its arrival in the MANET without an address conflict, few consider the issue of IP address change. To the best of our knowledge, this is the first effort to present a systematic solution for IP address handoff in the MANET.

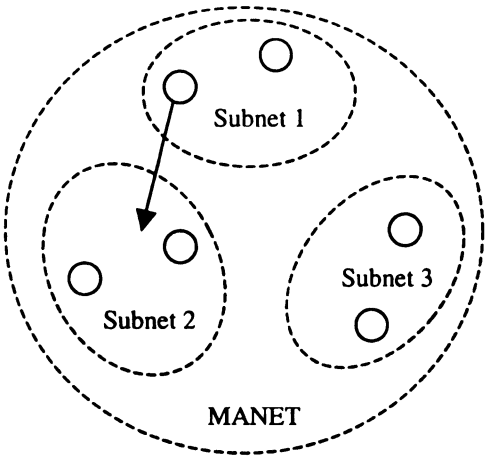


Figure 5.4 A MANET with hierarchical addressing scheme

This chapter is structured as follows. Section 5.1 discusses the overhead caused by IP address change of a mobile node during its session in the MANET. The solutions to remedy the overhead due to broken routing fabrics and on-going communications are presented in Section 5.2 and Section 5.3, respectively. Section 5.4 gives analytical evaluation of the performance and discusses its limitations, other scenarios and challenges to key management in MANETs. A prototype implementation and test are introduced in Section 5.5. Section 5.6 concludes this chapter.

5.1 Motivation

There are two major issues resulting from IP address change of a mobile node in the MANET:

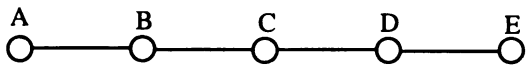
5.1.1 Broken routing fabrics

Unlike the hosts connected to the edge networks, all the nodes in a large-scale MANET have to function as routers (i.e., multi-hop routing). If a node changes its IP

address, all the routing entries that point to the node as the downstream next hop will be obsolete.

Fig. 5.5 gives an example of a simple MANET composed of 5 nodes in a chain with AODV[11] as the routing protocol. Suppose that the IP address of node C is x. The routing tables at node B and node D are also shown in Fig. 5.5. When node C changes its address from x to y, all the routing entries in node B and D will be invalid.

According to the specification of AODV, after two HELLO message intervals (i.e., 2 seconds), nodes B and D will detect downstream link breakage. If the destination of data packets is within the distance of certain hops, nodes B and D may initiate scoped broadcasts to rebuild the path between B and D; otherwise, they will send a Route Error (RERR) packet back to nodes A and E respectively, which triggers flooding of route rediscovery packets.



Dest	Next
D	C (x)
E	C (x)

Dest	Next
B	C (x)
A	C (x)

Figure 5.5 A MANET of 5 nodes in a chain

In a more complicated topology, node C will have many neighbors and will be on many active paths. Thus, much overhead will incur for route maintenance.

[11] The routing protocols that support multiple paths, such as DSR, are insensitive to the IP address change of a node along the path.

5.1.2 Broken on-going communications

While referring to Fig. 5.5, suppose that node A is communicating with node E. If node E changes its address from u to v, the on-going communications between A and E will be broken, which does not meet the requirement of real-time multimedia applications.

Because the address of u will not exist any longer (if the address change is not caused by address conflict), the local repair mechanism of AODV will fail eventually. As a result, the overhead of route rediscovery is inevitable. Even if node A initiates route rediscovery, it will not find the destination, unless the DNS scheme proposed in [59] is combined with the reactive routing protocol.

An ad-hoc approach is that node E resumes the connection actively. However, because node E may be a server of an application (e.g., the user of node E is running a FTP server so that other participants of the meeting can download documents from him), it is not responsible for the initiation of communication. Furthermore, the application may run in the background or the user of node E may not be aware of the broken communication. Thus, we need a better solution.

IP address change may compromise privacy as well. One example is the case of the merger of two MANETs. Suppose that node C is running a VoIP application with node B in MANET 2, as illustrated in Fig. 5.2. Because node B has the same address as node A in MANET 1, node B will change its address after the merger (for example, MANET 2 has a smaller NID). If node A keeps its address, the route maintenance procedure will rebuild the path between node C and node A. Thus, the voice data packets destined for node B will be redirected to node A. Suppose that node A is also running the VoIP

application simultaneously, and that they use the same UDP ports. If the voice data packets are not encrypted, node C will talk with node A for a while until it realizes that it is speaking with a wrong party in the middle of the communication.

5.2 Solutions to Broken Routing Fabrics

Unlike the link breakage caused by node movement, the overhead of broken routing fabrics is due to IP address change of a node. Because the node may still be within the transmission range of its neighbors, and it is aware of the address change, a simple solution can be implemented to remedy this kind of overhead[12].

We assume AODV as the routing protocol. Suppose that node C changes its address from x to y, as illustrated in Fig. 5.5. Node C can notify all its neighbors (such as nodes B and E) of the address change. A new routing control packet, namely Route Shift packet, can be introduced into AODV scheme. The packet is a one-hop broadcast packet that contains the source's old address and new address. On receipt of the packet, the neighbors change the next hop from x to y in all the routing entries with node C as the next hop.

However, this solution is vulnerable to IP spoofing attacks, which are difficult to detect and prevent in MANETs that have no infrastructure. With the introduction of the Route Shift packet, a malicious node can impersonate another node and broadcast the packet to undermine the routing fabrics. We need a way to identify the source of the packet.

---

[12] If the address change happens with link breakage simultaneously, the broken routing fabrics will be repaired with route maintenance.

One solution is to use a cryptographic method such as a digital signature, in which node C signs the Route Shift packet with its private key. All its neighbors contact the Certificate Authority (CA) to get the certificate for node C's public key and validate the Route Shift packet. Although this method can defeat IP spoofing attacks, it brings delay and communication overhead.

Our solution is that node C chooses a random number for its current address x, and puts the hash value of the number in the Route Request packet and Route Reply packet[13] in transit, and periodical HELLO messages. All its neighbors store the hash value in either their neighbor tables or routing tables. When node A changes its address, it puts the random number in the Route Shift packet. Because the packet is a one-hop broadcast packet, which will be received by its neighbors simultaneously, it is not vulnerable to the "man-in-the-middle" attack. If the hash value of the number contained in the Route Shift packet is equal to the stored hash value, the neighbors will be sure of the source of the packet. This method depends on the complexity of the hash function. MD5 [60] is such a good candidate that it is very difficult to determine the number from its hash value.

5.3 Solutions to Broken Communications

The second type of overhead due to IP address change is the broken communications between the source and destination. We first provide the reasonable assumptions, and then describe the schemes for route rebuilding and communication states preservation.

---

[13] These routing control packets are not destined for node C itself.

5.3.1 Assumptions

We assume that the IP layer of the mobile node supports more than one IP address. Because we can bind at least two IP addresses with a NIC in most mainstream operating systems (e.g., Unix and Windows), we can assume that it is the same for mobile devices.

With two IP addresses bound to the same interface, we specify that the new address as primary address and the old address as secondary address, and designate that the node use the primary address in the outgoing IP packets. Therefore, the node can still receive the packets destined for the old address for a short period, but the old address will not be used in the following new connections.

In order not to trigger RERR packets sent from the neighbors of the changing node, we also need to extend the HELLO message to contain both the primary and secondary addresses for several intervals. However, to prevent the data packets destined for another node whose primary address is the same as its secondary address to be forwarded to it, the node must not reply to the Route Request (RREQ) packet for its old address.

The second assumption is that the underlying links are bi-directional. Because most MAC layers deployed in MANETs confirm to 802.11 standard, our assumption can be easily satisfied.

5.3.2 Route rebuilding

Suppose that node A is communicating with node B and node A changes its address (say, from x to y) during the communication. Although node A can still receive the packets destined for its old address of x for a short period with the mechanism proposed above, we expect that the following communications be based upon the new address of y.

However, because the new address has not been seen before, a broadcast of RREQ[14] from the other end is necessary to build the path towards it.

To save the overhead of route rediscovery, we resort to a gratuitous Route Reply (RREP) packet. Because the path to node B may be still valid, node A can send a RREP packet with its new address to node B, which generates valid routing entries backwards to A in the routing tables in the nodes along the path because underlying links are bi-directional (our assumption). The routing entries towards the old address of x will expire eventually.

5.3.3 Communication preservation

The most important problem in handoff processing is the preservation of communication states at end points. This is because the checksum in the transport layer is computed based upon the source and destination IP addresses and the transport layer will check the corresponding header in the IP packet against the communication states before delivering the data to the upper layer.

Suppose that node A is communicating with node B, and that node A changes its address from x to y. We adopt an NAT mechanism running on both nodes to preserve the communication states:

(1) At node A, the new destination address of y in the incoming packets is modified to the old address of x prior to delivering it to the transport layer, and the old source address in outgoing packets is modified to the new address before sending it to the link layer;

---

[14] The RREQ may be combined with a name query message.

(2) At node B, the new source address of y in the incoming packets is modified to the old address of x prior to delivering it to the transport layer, and the old destination address in the outgoing packets is modified to the new address before sending it to the link layer.

Although the packets from node A to node B can contain A's old source address of x, which does not affect forwarding policy and the routing fabrics, we still perform NAT on it for purpose of correct reporting of Route Error packets if the path from A to B becomes invalid due to node mobility.

Compared with the tunneling mechanism proposed in [38], our approach has the following advantages:

(1) The overhead of a second IP header is saved. Although the length of the IP header is only 20 bytes in IPv4 or 40 bytes in IPv6 (without any options), it may lead to fragmentation/defragmentation that is time-consuming;

(2) Because only one address in the IP header is modified in NAT, it will be faster when applied with the improved computation of IP checksum [61]

(3) The tunneling scheme brings a "DoS" problem as illustrated in Fig. 5.6. Suppose that node A has been communicating with node B before node A changes its address from x to y due to the address conflict with node C. An IP tunnel is built between B and A to redirect packets destined for A's old address of x to its new address of y. If node C begins to communicate with B, the packets from B to C will also be forwarded to A through the tunnel, which means node C will never get any replies. Moreover, suppose that node B runs a TCP server application and listens at a well-known port. When node C initiates

a connection from the same client port as node A, the connection between node A and node B will be reset due to the incorrect sequence number and TCP flags. Although the IP-in-IP tunnel scheme is simple, it cannot solve these problems.

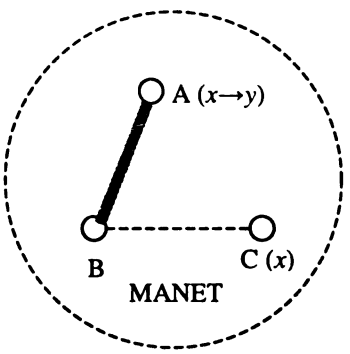


Figure 5.6 A "DoS" problem caused by IP tunneling

To overcome these problems, our scheme extends NAT to utilize both port numbers and sequence numbers[15] to distinguish different connections at node B, which can be explained with an example below. Suppose that node B is a web server, and that node A is fetching web pages from B when it changes its address from *x* to *y*. An NAT table, such as Table 5.1, is built at node B to store the required information.

Table 5.1 NAT table at node B

1	2	3	4	5	6
Old remote address	**New remote address**	**Local port**	**Remote port**	**Remote sequence number**	**Next remote sequence number**
x	y	80	2030	228743	22884312
...	...	...	...	...	...

The procedure at node B works as the following:

[15] Only a TCP header has a sequence number. For a UDP header, the sequence number can be regarded as 0, which is meaningless.

(1) When node B receives a packet, before it delivers the packet to the transport layer, it checks the columns 2, 3, and 4 in the NAT table with the corresponding fields in the packet. If there is a match, there are three possibilities: a) the sequence number in the TCP header is equal to field 5 in the entry, which means the packet is either a retransmitted packet from node A or the first packet from node A since the entry was inserted into the NAT table. The new source address is modified to be the old address in field 1, and the next remote sequence number column is the sum of the remote sequence number and the payload length; b) if the sequence number in the packet is equal to field 6 in the entry, which means the packet is a new packet from node A, the new address in the packet is modified to be the old address, the value in field 6 is copied to filed 5, and field 6 is increased by the payload length; c) if the sequence number is not equal to either, which means the packet comes from node C that has x as its primary address, the packet is discarded silently. In all the other cases, the packet is delivered to the upper layer intact.

(2) When node B has a packet for the destination address of x, before the packet is sent to the link layer, node B compares columns 1, 3, 4 and 6 in the NAT table and the corresponding fields in the packet to find a match[16]. If there is such an entry, as the one in Table 5.1, the old destination address of x is changed to the new address of y, together with re-computation of the IP checksum; otherwise, the packet is sent intact.

---

[16] The acknowledge number in the packet is compared with the next remote sequence number in the NAT table.

For UDP communications, because there is no sequence number in the UDP header, only the first four fields in the NAT table are used.

Compared with node B, the processing at node A is simpler. Another NAT table, such as Table 5.2 is used. If the local port number in the packet is equal to one in the table, the old address is modified to the new address in the outgoing packets, and vice versa for incoming packets.

Table 5.2 NAT table at node A

Old address	New address	Port number
x	y	2030
x	y	...

Because node A changes its own address, it is trivial for it to insert the entry when it has a packet to send. The problem remains that how the entry is inserted at node B's NAT table. A special message, Address Change Message (ACM), can be sent from node A to B indicating the creation of the NAT entry, which includes the old address, new address, protocol (UDP or TCP), local port, remote port, and sequence number (TCP only). The next remote sequence number in B's NAT table is initialized as zero, and will be filled by the following TCP data packets. To save communication overhead further, the message can be combined with the gratuitous RREP packet mentioned in route rebuilding.

The ACM packet must be sent before any data packets for node B. If node A is going to send a packet immediately after the address change, the data packet can be buffered before sending of the ACM packet. If node A has nothing to send immediately after address change, it waits for data packets from node B. Although node B has not been informed of A's address change, according to our assumption, the route from node B to

node A with the old address will be valid for a short period. Thus, the data packet can still arrive at node A, which triggers an ACM packet sent to B.

To remove the entry, the TCP flag of FIN can be examined. For UDP entries, an expiration time can be associated with each of them. The entry is refreshed with UDP data packets and removed when it expires. The timeout method could also be utilized for TCP entries in case that the FIN packet is lost on transit or the other end shuts down abruptly.

To prevent IP spoofing attacks, the ACM packet must be signed with node A's private key. Therefore, the contents can be validated with A's public key at node B.

5.4 Performance Evaluation and Discussion

This section analyzes the performance, limitations, other scenarios, and challenges of the handoff scheme.

5.4.1 Performance analysis

Suppose that there are n nodes in the MANET with l links. The average degree of a node is d (= l/n). Node A, which changes its IP address, has k connections with m nodes and on p active paths. The average length of the path is q. In a MANET where the mobile nodes are distributed evenly, we can assume that $l >> n >> q$, $n >> d$, and $n > m$;

As analyzed before, there are two kinds of overheads:

(1) Overhead of broken routing fabrics

Without the handoff scheme, the active paths going through node A will become invalid. As a result, at least one end of each path will initiate route rediscovery if the local repair mechanism is not utilized, which means there will be at least p times of flooding

throughout the MANET. If no flooding optimization is adopted, the same packet will be forwarded by all the nodes once. Thus, there will be $2l$ packets for one flooding. As a result, there will be at least $2pl$ packets caused by broken routing fabrics.

With the introduction of the Route Shift packet, which is a one-hop broadcast packet, only d packets are generated.

(2) Overhead of broken communications

Without the handoff scheme, the m nodes that communicate with node A will perform route rediscovery. Therefore, $2ml$ packets will be generated to find the path towards node A.

With the introduction of a gratuitous RREP packet and the ACM packet, only $(m+k)q$ unicast packets are necessary to rebuild the route and insert NAT table entries. If RREP and ACM are combined, the number of packets is decreased to kq.

5.4.2 Limitations

If node A wants to communicate with node C as in Fig. 5.6, because C's primary address of x has been bound with A's NIC as the secondary address, all the packets from A to C will be regarded as local packets. Thus, they cannot communicate with each other until node A does not use the secondary address any longer.

As discussed in the section above, to distinguish TCP connections from node A (that changes its address) and node C (that has the same primary address), node B utilizes both port numbers and sequence number in the TCP header. If node C connects to node B's well-known port from the same port number after node A changes its address, the connection will be rejected. If the client port is dynamically allocated, as most application programs do, node A and node C may have different local ports.

On the contrary, node B may connect to C's well-known port after it has connected with A's same port (for example, node B is browsing A and Cs' homepages at the same time) and A has changed its address. If the client port is dynamically allocated, B will have different local port numbers, which will not be affected by our scheme. If node B binds the local port in the program, because the acknowledgement number in the connection with node A is usually different from the one in the connection with node C, the communication will not be affected either. However, if the acknowledgement number is the same, its NAT module will redirect the connection.

With regard to UDP communication, because there is no sequence number in the UDP header, if nodes A and C use the same local client port number, the packets for C will be redirected to A. Furthermore, because there are no connection flags, UDP communication between nodes A and B will be forced to break if there has not been any data exchange for a long period after creation of the NAT entry.

Another limitation is that if there is no data exchange between node A and B for a while immediately after A changes its address, the path from B to A will be invalid due to node mobility. Thus, a route rediscovery combined with name query from node B is inevitable. However, this problem can be solved if node A actively sends an ACM packet for each pre-existing connection.

5.4.3 Multiple address changes

Our scheme can be applied to multiple address changes without any significant modifications.

Suppose that the source node changes its IP address twice during the session, without loss of generality. After the first address change, the NAT tables have been built at both

the source and destination to change the secondary address in the data packets to primary address and back forth. When the second address change happens, because the IP layer supports more than one IP address, the old primary address can be "pushed back" to be the second secondary address.

The solution for broken routing fabrics can be utilized as usual. As to broken on-going communications, if the communication was initiated with the second secondary address (between the first and the second address change), new NAT entries will be created. However, for the communications initiated with the first secondary address (before the first address change), it must be associated with the up-to-date primary address. To solve this problem, noting that the primary address is mainly used for purpose of routing, we just need to modify the new address field of corresponding entries at both the source and destination.

Thus, for all the existing NAT entries at the source node, when a second address change happens, a special flag can be set for each entry in addition to modification of new address field (Table 5.2). When an outgoing data packet has a matching local port number, an ACM message is sent to the destination, and the flag can be cleared. On receipt of this second ACM message, the destination modifies its NAT table to reflect the change.

5.4.4 Address changes at both source and destination

If both of the source and destination change their IP addresses simultaneously, because their old addresses are bound with them as secondary addresses, the ACM messages or data packets from either node with the old destination addresses will still reach the other end successfully within a short time interval.

However, once the ACM message from the other end is received, a second NAT table is created, in addition to the first table caused by its own IP address change, as the tables illustrated both in Table 5.1 and in Table 5.2. Therefore, every data packet will undergo NAT twice[17]:

(1) The outgoing data packets will be performed source NAT with Table 5.2, and then destination NAT with Table 5.1;

(2) The incoming data packets will be performed source NAT with Table 5.1, and then destination NAT with Table 5.2.

Thus, the communication states at both ends could be preserved in spite of both address changes.

5.4.5 Challenge to key management

Address handoff brings challenges to the key management in MANETs because most existing schemes assume that the IP address of a node is fixed, and thus a public/private key pair is bound with an IP address. When applied with autoconfiguration, their assumptions will be invalid for the following two reasons:

(1) The addresses of the nodes are dynamically assigned when they join the MANET.

(2) A node may change its address during its session in the MANET due to address conflict.

Suppose there is a CA in the MANET (either centralized or distributed). When the mobile node joins the network and is assigned with an IP address, it must register the binding of its public key with its current IP address with the CA. When it changes its

---

[17] The order of source NAT and destination NAT is not important.

address, it needs to register itself again with the CA to get a new certificate. However, since the binding of its previous address and its public key may not expire at the CA, its request will be rejected by the CA even it is really the owner of the public key, because it seems that two different nodes have the same public key in the eyes of the CA.

The approach proposed in Section 5.3 will solve this problem: we can specify that the node generate another random number every time when it is assigned an IP address. If the length of the random number is long enough, two nodes will have different random numbers. During the process of registration, the hash value of the random number is included in the messages. When the node changes its address, the random value associated with the previous address is included in the registration message and encrypted with the other end's public key or the secret key that they have agreed upon. Thus, the source of the messages can be identified.

5.5 Prototype Implementation

A prototype of the handoff scheme is implemented to test the preservation of communication states in a LAN, as illustrated in Fig. 5.7. After a TCP connection was built between the client of a laptop and the server of a desktop, the client changes its IP address. Although the client is working in the infrastructure mode, it should be the same in the ad-hoc mode. The application is a simple string echo program, in which the server echoes the string typed at the client's terminal.

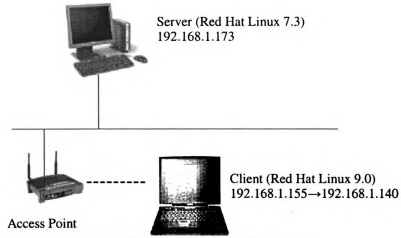


Figure 5.7 The testbed of handoff scheme

The handoff scheme is implemented as hooks by means of netfilter [62]. The NAT processing of outgoing packets is performed at NF_IP_LOCAL_OUT, while the NAT processing of incoming packets is performed at NF_IP_PRE_ROUTING.

The code illustrated in Fig. 5.8 shows the procedure of NAT processing of outgoing packets at the client. The processing of incoming packets is similar. To simplify the test, we did not use the aforementioned NAT tables in the prototype because there is only one TCP connection. In a real implementation, a hash table may be utilized to expedite lookup of NAT tables. Other optimizations are also desirable. For example, IP addresses and port numbers should be stored in network order, and the fast computation of IP checksum should be used.

The test is done in the following steps:

(1) The client initiates a connection to the server;

(2) After a while, the IP address of the client is changed from 192.168.1.155 to 192.168.1.140 at the wireless NIC (eth1);

(3) The old address of 192.168.1.155 is bound with the client's wireless NIC as an alias (eth1:0)[18];

```
#define OLD_ADDRESS 0xC0A8019B // 192.168.1.155
#define NEW_ADDRESS 0xC0A8018C // 192.168.1.140
static unsigned int handoff_NAT_out(unsigned int hook, struct sk_buff **pskb, const struct net_device
*indev, const struct net_device *outdev, int (*okfn)(struct sk_buff *)
  {
    struct tcphdr* th;
    // Whether we should perform NAT or not
    if ((*pskb)->nh.iph->saddr == htonl(OLD_ADDRESS) && (*pskb)->nh.iph->protocol == 6)
    {
      th = (struct tcphdr*)((char*)(*pskb)->nh.iph + (*pskb)->nh.iph->ihl*4);
      if (th->dest == htons(10000))
      {
        // Change (source) address
        (*pskb)->nh.iph->saddr = (DWORD)htonl(NEW_ADDRESS);
        // Recompute IP checksum
        (*pskb)->nh.iph->check = 0;
        (*pskb)->nh.iph->check = in_checksum((WORD*)((*pskb)->nh.iph),    (*pskb)->nh.iph-
>ihl*4);
      }
    }
    return NF_ACCEPT;
  }
```

Figure 5.8 NAT processing of outgoing packets at changing node

[18] This step is necessary because otherwise the outgoing packets will be dropped at the client's IP layer.

(4) Install hooks at both the client and server;

(5) Continue typing strings at the client's terminal, which are shown on the server's terminal.

We used ethereal running on the client to capture all the TCP packets during the test, which are shown in Fig. 5.9. In Fig. 5.9, the first three lines are the control packets for TCP handshaking procedure between the client (192.168.1.155) and the server (192.168.1.173). The lines 4-7 are the TCP data packets before the client's address change. Starting from line 8, the client's address is changed from 192.168.1.155 to 192.168.1.140, so the source address of all the following outgoing packets is the new address. However, for the incoming data packets, because the new destination address has been modified by the hook, so the destination address of the captured incoming packets is still shown as the old address. With ethereal running on the server side, the outgoing packets' destination address field still contains the new address of the client.

5.6 Summary

With the deployment of autoconfiguration, global connectivity, and hierarchical addressing scheme (in large scale networks), mobile nodes in MANETs may change their IP addresses more frequently than fixed hosts in hardwired networks. However, the issue of address change has not been carefully studied. Based upon the analysis of the overhead caused by address change, we introduced the Route Shift packet, Address Change Message, and NAT scheme to solve the problem. Thus, the overhead caused by broken routing fabrics is saved. Moreover, both ends can continue their TCP/UDP communications, unaware of the address change. The method could be applied to the

scenario that the node changes its IP address more that once, and that both the source and destination change their addresses simultaneously, without any significant modifications.

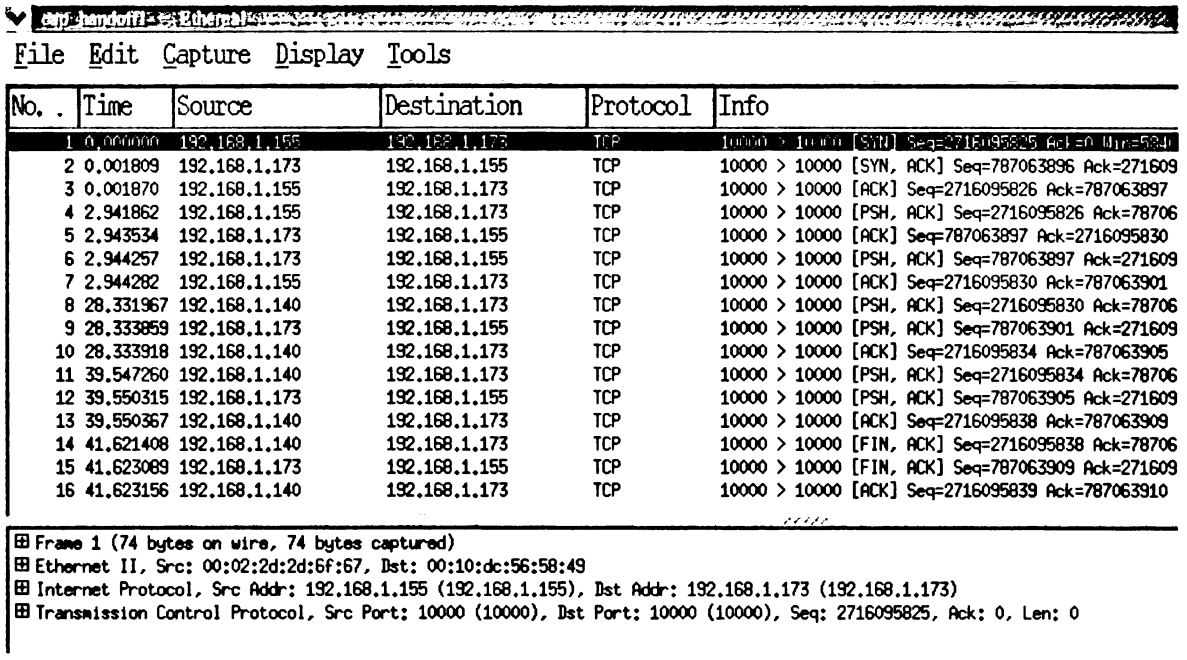


Figure 5.9 The captured TCP packets during the test

CHAPTER 6 MULTIPLE-KEY CRYPTOGRAPHY-BASED DISTRIBUTED CERTIFICATE AUTHORITY

One characteristic of the MANET is limited physical security. Thus, public key cryptography is applied in the MANET to improve security. Every node in the MANET has a unique public/private key pair. During data communications initialization, both ends can exchange their public keys to establish the secret key for the subsequent data encryption. From then on, they can switch to symmetric cryptography that is faster than public key cryptography.

Although the scheme is simple to implement, it is vulnerable to "man-in-the-middle" attacks unless there is a certificate authority (CA) in the network. With a CA being present, each node registers the binding of its public key and IP address in the CA, and acquires the certificate from the CA as a proof of the binding when it enters the MANET. Thus, "man-in-the-middle" attacks and IP spoofing attacks can be defeated.

In hardwired networks, the CA can be a centralized server, which is impractical in the MANET. Because every node in the network is mobile and power-limited, none is reliable. Therefore, most research effort is spent on building a distributed Certificate Authority (DCA) in the MANET.

The chapter is structured as follows: Section 6.1 describes the vulnerability of threshold cryptography-based DCA scheme in the presence of Sybil attacks. A new

scheme based on multiple-key cryptography, namely MC-DCA scheme, is proposed in Section 6.2. It is invulnerable to Sybil attacks, and achieves lower communication overhead and moderate latency in comparison to a threshold-based scheme, which is supported by the simulation results in Section 6.3. Section 6.4 concludes the chapter.

6.1 Vulnerability of Threshold Cryptography-based DCA

The threshold-based DCA was originally proposed for hardwired networks, in which the administrators of the server hosts can ascertain others' identities and trust each other. In the MANET where Sybil attacks may be present, the threshold scheme may be compromised.

The Sybil attack refers to the attack from a malicious node that impersonates many identities in the network when cooperation is necessary to provide the service. The attack on the threshold scheme is illustrated in Fig. 6.1.

In Fig. 6.1, nodes A, B, and C are good nodes; node M is a malicious node. All of them are forming the (k, n)-threshold DCA server group. Node M impersonates non-existent nodes M_1, M_2, ..., and M_{k-1}. It will know the value of k during the parameter negotiation procedures and forge k-1 identities as needed; or it prepares m identities beforehand and persuades other server nodes that m+1 should be large enough for parameter k, hopefully leading to the value of k that is less than or equal to m+1. In either way, once all the nodes agree on the parameters and exchange their shares, node M will have enough secret shares to construct valid signatures.

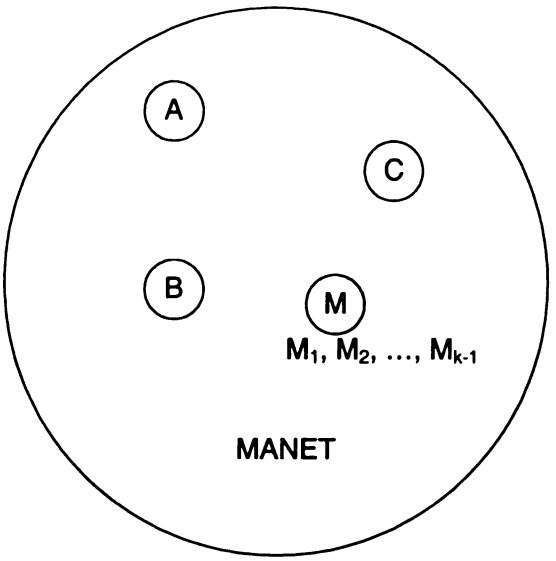


Figure 6.1 Sybil attack on threshold-based DCA scheme

The Sybil attack is fatal to the threshold scheme, but there is no efficient way to defeat it because it is difficult to bind a single identity with one node in the MANET. The CA scheme regards the IP address of the node as its identity, which can be easily forged by a malicious node, especially in the presence of autoconfiguration schemes that are utilized to assign IP addresses to nodes automatically in an open system[19]. Similarly, it is not difficult for a malicious node to have many hardware addresses. Even if the CPUs in some nodes have unique built-in identifiers, the identifiers are hidden from outsiders. The malicious node can still forge the built-in identifiers easily.

6.2 Multiple-key Cryptography-based DCA

To defeat Sybil attacks, a new scheme based on multiple-key cryptography, namely MC-DCA scheme, is described in this section.

---

[19] In a closed system where identities are assigned from a central authority, the problem is trivial.

6.2.1 Assumption

We assume that a MANET is an open system where nodes are free to join and leave, and thus there is no single party of trust.

6.2.2 Multiple-key cryptography

Multiple-key cryptography was proposed in [63], which is based on public key cryptography. In the traditional public key cryptography, there are two keys: k_1 and k_2. The message encrypted/signed with one key can be decrypted/verified with the other key. Either one can be the public key; and the other is the private key. In [63], the concept is extended to multiple keys: k_1, k_2, ..., and k_n. the message encrypted/signed with one subset of keys can only be decrypted/verified with all the other keys. For example, suppose that there are 4 keys: k_1, k_2, k_3, and k_4. The message encrypted with k_1 and k_3 can only be decrypted with k_2 and k_4. With only either k_2 or k_4, the message cannot be decrypted. Obviously, if one key is chosen to be the public key, all the others must be private keys, which is the way that multiple-key cryptography is applied in DCA.

Suppose that there are n servers. A central authority (i.e., the owner of the scheme in [63]) divides the secret key into n shares and stores one share at one server. If a client wants its message signed by the DCA, it sends the message to one of the servers. Each server signs the message with its share in turn, and the last server sends the signature to the client.

In multiple-key cryptography, since every server node is required to generate the valid signature, even if a malicious node has many identities in the DCA, it cannot forge

signatures when there are good server nodes. The underlying assumption is that good nodes are encouraged to join the DCA group, just as good citizens are encouraged to provide service to others in the human society.

However, since we assume that there is no single trusted party in the MANET, the original multiple-key cryptography cannot be applied to the DCA in the MANET directly. Some modifications are necessary.

6.2.3 Algorithm

We utilize the distributed algorithm in [64] to choose secret shares and calculate the public key for server nodes. Suppose that there are n servers. In [64], all the server nodes agree on three parameters: two large primes p and q such that q divides p-1, and g that is a generator of G_q (G_q is the unique subgroup of Z^*_p of order q). These three parameters are parts of the public key of the DCA and known to all the users in the network. Server node i chooses its secret share x_i, and computes the public part $h_i = g^{x_i}$. The private key of the DCA is the sum of x_i ($\sum_{i=1}^{n} x_i$), and the public key is the product of h_i ($\prod_{i=1}^{n} h_i$). The next steps in [64] are the procedures for share refreshing, which are utilized in the threshold scheme and thus unrelated to our multiple-key scheme.

According to the algorithm in [64], if server node i is not the same node of server node j, and server node i does not leak its secret share, server node j has no idea about i's secret share. As a result, a subset of the server nodes cannot forge signatures as the whole DCA group. All of the server nodes have to cooperate to generate valid signatures.

If server node i leaves, the DCA is down because no one else knows its secret share. Even if it knows it is going to leave, it cannot transfer its secret share to another node, say node k, and designate node k to take its place in the DCA, because node i cannot ascertain if node k is another identity of a server node (say, server node j) or not. It is possible that server node j is a malicious node and that it impersonates all the other server nodes simultaneously except server node i. If it also impersonates node k and replaces server node i, it will have all the secret shares and be able to forge valid signatures.

To account for a missing server node, a version number associated with the public key is introduced in the MC-DCA scheme. Once a server node is detected to have left the network, new nodes will be invited to join the server group. The old server nodes may keep their secret shares and public parts, while the new server nodes choose their own secret shares and calculate new public parts. As a result, a new public/private key pair for the DCA is generated with the version number increased by one. The clients store all the public keys, and verify the certificate issued from the DCA using the public key with the same version number. This scheme has a potential security advantage in comparison with the threshold scheme because the private keys are different with different version numbers, and the previous private keys cannot be recovered. If one public/private key pair of the DCA is compromised in a rare event, the certificates issued with the previous version numbers are still valid.

Notice that the partial signature signed with the secret share can be verified with the corresponding public part, the faulty server node can be easily pinpointed. The client just stores the public key of the DCA, while the server nodes store each other's public part. If the client receives a signature that cannot be verified with the DCA's public key, it can

bring all the partial signatures to the server group to find out the faulty server node and exclude it from the DCA group.

To further improve the security of the MC-DCA scheme, all the server nodes can choose new secret shares and calculate the public parts once a while. The old secret shares are discarded, and the version number of the resulting public key is increased by one. If no client applies for a certificate during the next interval, the version number can be kept to be the same even if the secret shares and public key change. For example, suppose that the time interval is t. At time $3t$, the version number is 4 after the server nodes choose new secret shares and have new public keys. At time $4t$, they choose new secret shares again. If there is no application for certificates between $3t$ and $4t$, the version number is still 4.

To limit the number of DCA's public keys stored at the client in the MC-DCA scheme, once the increase in the version number of the public key reaches a special value, all the client nodes whose certificates are signed with previous public keys can renew their certificates at the DCA, and all the nodes remove these obsolete public keys from their repositories. For example, suppose that the special value is 20. When the version number of the public keys increases from 1 to 21, all the clients with the certificates signed with the public keys from version 1 to 11 renew their certificates with the public key of version 21. After the renew procedure, the public keys with version 1 to 11 can be safely removed from all the nodes. The aforementioned method could save much communication overhead in the renew procedure because some client nodes with old certificates may have already left the MANET, just as some server nodes.

6.2.4 DCA group membership management

To maintain the registration table of bindings of IP addresses and public keys in the MC-DCA scheme, the server nodes need to track each other with the aid of periodic HELLO messages. Otherwise, if all the server nodes leave without notice, the registration information will be lost, which will lead to high communication overhead in the subsequent registrations. For example, suppose that client node A registers its public key x at the DCA and obtains the certificate. After all the previous server nodes leave and new DCA group forms, another client node, node B, tries to register the same public key of x at the DCA. Without the previous registration information, the new DCA group has to inquire all the client nodes to determine the uniqueness of the public key.

Although the threshold scheme is tolerable to some missing server nodes, it has stricter requirements on group membership maintenance than the MC-DCA scheme. For example, suppose that (3, 5)-threshold cryptography is adopted. If three server nodes leave the MANET without notice, the remaining two server nodes cannot recover the secret key of the DCA. Although it can utilize the concept of the version number of public keys in the MC-DCA scheme, there would be no advantage of the tolerance to missing server nodes mentioned before. Thus, the periodic HELLO messages are indispensable. In the previous example, once a server node is detected to have left the MANET, the remaining four server nodes need to invite new server nodes to join the DCA server group[20]. The mechanism of periodic HELLO messages is not mentioned in

[20] The DCA server group may choose to wait until there are three server nodes remaining in the network in the threshold scheme. However, it cannot be predicted that one of the remaining server nodes leaves before a new server node joins.

[47], but the cluster head's periodic beacon messages with intervals of 10 seconds to 30 seconds are utilized in [48], which can be regarded as a kind of periodic HELLO message.

The difference between the MC-DCA scheme and the threshold scheme in group membership maintenance is that the minimum number of the remaining server nodes in the former scheme is one, while the minimum number in the latter scheme is the threshold value.

6.2.5 Procedures

The procedures of the MC-DCA scheme work as follows:

(1) The first client node in the MANET needs a DCA service, so it broadcasts an INVITE message to initiate an invitation to all the other nodes in the MANET;

(2) On receipt of the INVITE message, each node decides itself if it participates in the service or not, to satisfy the assumption of the multiple-key cryptography-based scheme. If it decides to join the server group, it makes an announcement with a broadcast of PARTCP message, including its own public key. All the other nodes will record it as a server node;

(3) All the server nodes agree on the parameters of p, q, and g, and each chooses its secret share and calculates the public part. Then they exchange their public parts, and compute the public key by multiplying all of them. One of the server nodes makes an announcement of the public key with version number 1;

(4) The client sends to each server node its public key, IP address, and other related information encrypted with the server nodes' own public keys to request a certificate in a REQUEST message;

(5) On receipt of the REQUEST message from a client node, each server node calculates the partial signature with its secret share, signs the partial signature with its own private key, and sends a REPLY message to the client;

(6) The client verifies the signature with the public key of the DCA after it combines all the partial signatures together. If the verification fails, it brings all the signed partial signatures to the DCA group to pinpoint which server node is malfunctioning and to exclude it from the group;

(7) Each server node chooses a secret share and calculates the public part periodically, and updates the version number if necessary;

(8) The server nodes send each other periodic HELLO messages to track the membership of the DCA group and exchange their renewed public parts. If one server node is detected to have left the MANET and they just received a request from a client node, or the number of remaining server nodes is less than a threshold value (e.g., 3), an invitation is initiated, with steps (2) and (3) repeated, except that the old server nodes may choose to keep their previous secret shares and public parts. The version number is increased by 1;

(9) If a client node detects a server node is missing, it can either initiate an invitation proactively or wait reactively until the server nodes detect it and broadcast an invitation in step (8);

(10) If the number of DCA's public keys reaches the capacity of the client's repository, the client renews its certificate if necessary and updates the repository.

There may be some optimizations. For example, when a client node receives an invitation just before it is going to send a request to the DCA, it can delay the request for a while. Another example is that if the replies from some server nodes are lost due to network congestion, the client can send the request to these server nodes a second time.

6.3 Performance Evaluation

This section compares both schemes' performance with simulation.

6.3.1 Simulation setup

The simulations were run with 50 nodes in an 800×800 m^2 topology area. All the nodes move with the random waypoint mobility model. The maximum speed is 10.0 m/sec and the minimum speed is 2.0 m/sec. The pause time is 10.0 seconds.

In the simulation, to determine if a node is willing to participate in the DCA server group, each node chooses a random value uniformly distributed in the range of [0, 1]. The nodes that choose a random value less than a threshold value are eligible to be the server nodes. In the simulation of both schemes, the threshold values are 0.1, so that around 10% of the nodes will be the server nodes, with the minimum number of server node being 3. The simulation time is 600 seconds. All the nodes have a lifetime and the time to request certificates pre-set, both of which are evenly distributed in the range of 0 and 2 times the simulation time (i.e., 1200 seconds).

In both schemes, all the server nodes send each other periodic HELLO messages. If the HELLO messages from a server node are not received for 3 intervals, it is regarded as departed. The interval in the threshold scheme is set to be 10 seconds (a longer interval will lead to the number of server nodes being less than the minimum value). The share

refreshing is also contained in the HELLO messages. In the MC-DCA scheme, because the reactive invitation method in step (9) in Subsection 6.2.5 is adopted, the interval for HELLO messages is 5 seconds to expedite the invitation process. The client node's repository has a capacity of 40 public keys.

6.3.2 Simulation results

The simulations were run 4 times. The sum of control messages in each category received at all the nodes for one simulation is illustrated in Fig. 6.2. Because the group membership maintenance in the threshold scheme is stricter than that in the MC-DCA scheme, more communication overhead occurs to invite new members.

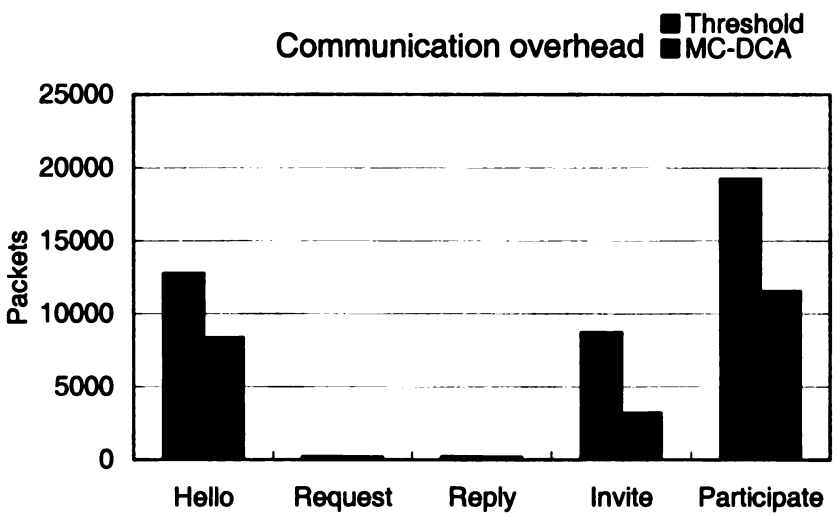


Figure 6.2 The number of packets in each category

Fig. 6.3 compares the sum of all the control messages received for four simulations. Generally speaking, the number of packets received in the MC-DCA scheme is around one-half of that in the threshold scheme.

Fig. 6.4 illustrates the average latency in the MC-DCA scheme, which is defined to be the interval between the time when a node initiates the request and the time when it gets its certificate. Theoretically, the maximum latency is around 3 times the HELLO message interval (i.e., 15 seconds). Due to node mobility, network congestion, and random timeout mechanism in the simulation, a few nodes may have longer latency.

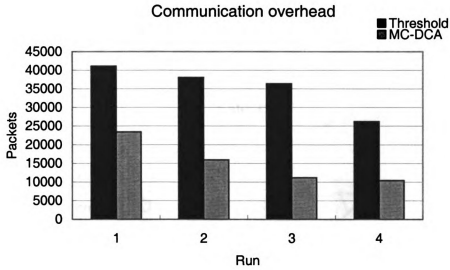


Figure 6.3Communication overhead

6.4 Summary

With the invulnerability to mobile adversaries and tolerance to missing or faulty server nodes, the threshold cryptography-based DCA scheme is regarded as an ideal candidate for a MANET where the nodes are instable. However, during the formation of the DCA, if a malicious node initiates Sybil attacks in which it impersonates multiple identities, it may acquire enough secret shares to forge valid signatures. Since it is not difficult for a

malicious node to forge multiple identities, the security of the threshold cryptography-based DCA scheme can be easily compromised.

To defeat Sybil attacks, a new scheme based on multiple-key cryptography, namely the MC-DCA scheme, is proposed. In the scheme, every server node is required to generate signatures. Thus, as long as there are good server nodes, even if a malicious node has many identities, it cannot forge signatures of the DCA. Compared to the threshold scheme, it also achieves lower communication overhead and moderate latency, which is supported by simulation results.

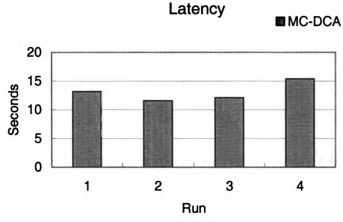


Figure 6.4 The latency in MC-DCA scheme

CHAPTER 7 REACTIVE ID ASSIGNMENT FOR SENSOR NETWORKS

A sensor network consists of many sensor nodes that are monitoring the environment and sending information to the user. Since sensor nodes are usually small and deployed at unfriendly places, they must rely on a battery that provides limited power supply. Therefore, power consumption is among the most important issues in the design of the architecture and networking protocols. Most research effort has been spent on improving efficiency of data communications and lengthening sensor's lifetime.

Proactive ID assignment proposed in [49] [50] assigns locally unique IDs to sensors immediately after they start up. So the IDs are ready even there may be no data communications for some time. They utilize periodical broadcasts of HELLO messages to solve ID conflicts, which consumes much power. We need a more efficient approach.

7.1 Reactive ID Assignment

To preserve as much power as possible and solve the conflicts, we can combine ID conflict resolution and data communications together. The ID conflict resolution procedures are delayed until data communications are necessary, which is called reactive ID assignment scheme.

In this section, the assumptions for our reactive ID assignment scheme are given first, and then the procedures are described in detail with an example. The last two subsections discuss the impacts of packet loss and security mechanisms.

7.1.1 Assumption

Although the schemes in [49] [50] recommended Huffman coded addresses for sensor nodes, we still adhere to fixed-length IDs due to the following reasons:

(1) The nodes in a sensor network are usually manufactured in batches. Although the average length of Huffman-coded address is less than the size of a fixed address format, it would be much easier for designers to allocate the fixed-length field for the MAC address in the physical layer in advance;

(2) The a priori Huffman code table may not be optimal for the nodes in a sensor network, and cannot be calculated in some cases (e.g., the sensor nodes may be dropped from airplanes or missiles);

(3) The optimal Huffman coding can save only 0.3 to 0.8 bit for one address, and it will become non-optimal as some sensor nodes die with time and new nodes join the network; and

(4) The Huffman code table must be stored in the memory of the sensor node during all its lifetime.

We define 1-hop uniqueness as address uniqueness among direct neighbors, and 2-hop uniqueness as address uniqueness among 2-hop neighbors. The assumption for 1-hop uniqueness is that the number of nodes in the largest complete sub-graph in the sensor network should be less than the range of the addresses (or the range of addresses minus 1, if a special address is designated as the broadcast address). The assumption for 2-hop uniqueness is that the maximum sum of the number of 1-hop neighbors and the number of 2-hop neighbors should be less than the range of the address.

With the expectant average node density (d) and transmission range (r), the designer can choose the length for the address field (l) deliberately to satisfy the assumptions:

(1) To satisfy 1-hop uniqueness, the address range should be greater than the number of nodes within the transmission range of one node, which means $l > \log_2(\pi dr^2)$;

(2) To satisfy 2-hop uniqueness, the address range should be greater than the number of nodes within a circular area of two times the transmission range, which means $l > \log_2(4\pi dr^2)$.

Without any information about the node density, the designer should choose the address range large enough conservatively.

If the nodes in a sensor network are deployed too densely, the assumptions may be violated. However, if the power of the sending packets can be adjusted, which is a desirable attribute for sensor nodes, the sending nodes can lower the power to satisfy the assumption. If the sending power cannot be adjusted, a "virtual link breakage" method can be utilized, which is described in Subsection 7.1.5.

We also assume that during the flooding of an interest message in the directed diffusion paradigm, every node broadcasts only once, which is the basic optimization in flooding [33].

7.1.2 Scheme

We can use an example to illustrate how to combine directed diffusion with conflict resolution in reactive ID assignment. For the small network in Fig. 7.1, suppose that node A is the sink, node B is the source. Nodes A and B choose the same random address of

a_1, nodes C and D choose the same random address of a_2. There are duplicate addresses among direct neighbors A and B, and among 2-hop neighbors C and D.

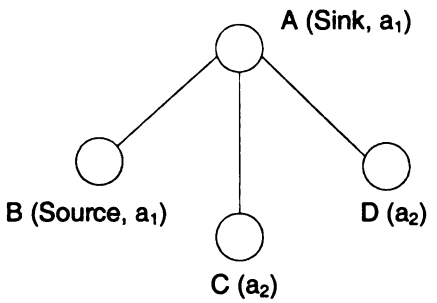


Figure 7.1 A small sensor network

Because all the addresses are randomly chosen and no communication has occurred yet, a node is not sure if its address is 1-hop unique, so it can use the address only temporarily. When the sink node broadcasts an interest message, the node can eliminate duplicate addresses among its direct neighbors because the receiver can choose another address randomly if it receives a packet with the same address. In the example in Fig. 7.1, once the sink node A broadcasts the interest message, node B must change its address (to a_3, for example). This applies to other nodes when the interest packet is forwarded.

We can also utilize forwarding to eliminate duplicate addresses among 2-hop neighbors according to the assumption that every node forwards the interest message only once. After nodes C and D forward the message, node A will receive the same message with the same source address twice. Thus, node A will be aware that there are two direct neighbors with the same address. Node A can unicast a special control message (RESOLVE message) to notify them that there exists a 2-hop conflict. On receipt of the RESOLVE message from node A, both node C and node D need to change to another random address.

If nodes C and D unfortunately choose the same address again, or choose the new address of node B (a_3), there will still be a 2-hop conflict[21]. To prevent the conflict, we require that nodes C and D broadcast an announcement of their changes (CHANGE message), which can be collected and checked by their neighbors.

Because the locally unique ID is used in the construction of the path between the sink and source, a mechanism is necessary to identify the origin of the change message. As illustrated in Fig. 7.2, if node B records node A (with the address of x) as its next hop back to the sink and then it finds that there are two 1-hop neighbors with the same address of x, it notifies them to change. Node A may change to the address of y, node C to z. On receipt of both change messages, which new address should be used as the next hop for node B?

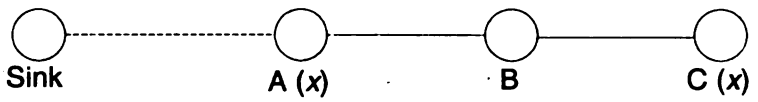


Figure 7.2 An example of 2-hop conflict

There are two methods to solve this problem:

(1) We can designate that every node choose a random number in addition to its random address. If the length for the random number is long enough (e.g., 16 bits), the probability that two neighboring nodes choose the same random number will be very low. If the random number is piggybacked in the broadcast of the interest message and change message, a node is differentiated among its neighbors. For a stationary sensor network, the random number is needed only once for the first broadcast/ forwarding of the interest message. In case that new

[21] They will not change to node A's address since they are aware that the address of a_1 is already occupied.

nodes may join the stationary network later or a node misses copies of the first interest message, a node will receive a message without the piggybacked random number as the first message. The node can just drop the message and broadcast a special control message requesting its neighbors' random number for once. Thus, the overhead caused by the random number in the interest message is trivial. However, it will bring too much communication overhead in a mobile sensor network since every interest message must include the random number.

(2) The alternative for mobile sensor networks is to use a hop count field that is usually found in routing messages in a MANET. As the interest message passes a node, the hop count field is increased by 1, which is also recorded in the node. The hop count field is also included in the change announcement message. Therefore, if the hop count in the change message is equal to the receiver's hop count minus 1, then the next hop address is updated if it is the same as the old address contained in the change message. We can limit the size for the hop count field to only 4 bits, and utilize the modulo operation. For example, in Fig. 7.2, if the hop count is 15 for node A, 0 for node B, and 1 for node C, on receipt of both change messages, node B will update its next hop address to *y* because $0 = (15 + 1) \bmod 16$. If the hop counts for both nodes A and C are 2, and it is 3 for node B, either one could be the next hop for node B because neither of them uses node B as the next hop. With the hop count field, the neighbors can be differentiated, and the possibility of the path loop is minimized, as analyzed with simplification below.

In Fig. 7.3, node A is the upstream hop for node B back towards the sink node. Node A has the hop count of i, and thus node B's hop count is i+1. To form a loop on the path between node B and the sink node, one of node B's neighbor, say node C, should have the hop count of i, and the path from its upstream node (say, node E) to the sink node goes through node B, as the dotted line in Fig. 7.3. In summary, node C must lose all the interest messages forwarded from its neighbors whose hop counts are not i-1, such as node B or node D, while it must hear the interest message from one of the neighbors whose hop counts are i-1, such as node E.

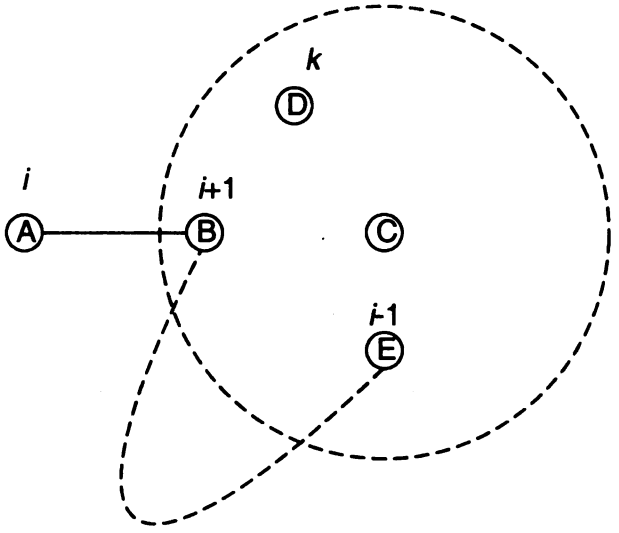


Figure 7.3 Path loop

Given that node B has a hop count of i+1, we need to calculate the probability that node C has a hop count of i with the assumptions that node C has not received any interest message yet, and that node B has not forwarded the interest message. Suppose that the transmission range is r, the average node density is d, and the packet loss rate is q. The total number of node C's neighbors is $\pi r^2 d$. If the hop counts are distributed evenly for all node C's neighbors that have received the interest message, the number of

node C's neighbors whose hop counts are not i-1 is $15\pi r^2 d/16$, while the number of remaining neighbors is $\pi r^2 d/16$. The probability that node C has a hop count of i is $q^{15\pi r^2 d/16} \times (1-q) \times \pi r^2 d/16$, which is the probability for one neighbor of node B. Considering that node B has $\pi r^2 d$ neighbors (we should exclude node A because the event that node A has the hop count of i has already happened, but it does not matter much), the probability of the loop is $p = q^{15\pi r^2 d/16} \times (1-q) \times (\pi r^2 d)^2/16$.

If we plug in some values, such as $q = 0.1$, $r = 250$ m, and vary d from the minimum density[22] of 1/40000 $/m^2$ to the maximum density of 0.1 $/m^2$, the highest values of p is 3.4×10^{-5} when the node density is the minimum value. Similarly, if we use $r = 10$ m and vary d from the minimum density of 1/81 to the maximum of 1, the highest probability is 1.91×10^{-4} when the node density is 1/81. Taken into consideration of high node density, lower packet loss rate, and the low probability that the path from node E must also go through node B, the possibility of path loop is negligible.

The reactive ID assignment scheme can be applied to mobile sensor networks as well. As the sensor nodes move around, there will still be local ID conflicts after previous conflict resolution. However, as long as there is no data communication, the ID conflict brings no harm to the sensor network, and it will be resolved during the next data communication.

The differences between the reactive ID assignment and the proactive method are:

---

[22] When all the sensor nodes are deployed in a grid with the interval of 200 m, the sensor network has nearly the lowest density. The density is $n^2/[40000(n-1)^2]$, which is 1/40000 when n approaches infinity.

(1) The ID conflict resolution is postponed until data communication is initiated in the reactive scheme; and

(2) The 1-hop neighbor table is not included in any control messages in the reactive scheme, which achieves shorter message length and less power consumption;

The performance of the two schemes is compared in Section 7.2.

7.1.3 Procedures

In summary, the procedure works as follows:

(1) In the beginning, every node chooses a random ID;

(2) The sink node broadcasts an INTEREST message;

(3) All the neighbor nodes record the sender's ID. If the sender's ID is the same as its own, it chooses another one randomly, and broadcasts a CHANGE message (this solves 1-hop conflict);

(4) The neighbor waits for a random delay and rebroadcasts the INTEREST message;

(5) If a node receives an INTEREST message with the same source ID more than once, it puts the ID in a RESOLVE message and broadcasts to its neighbors (this solves 2-hop conflict)[23];

(6) If a node receives a RESOLVE message containing its ID, it chooses another one randomly (because it records all the 1-hop neighbors' IDs, so it will not lead

[23] To be more exact, it should be a unicast message received by more than one recipient. However, during the simulation, we use broadcast instead.

to 1-hop conflict), and broadcasts a CHANGE message (to avoid further potential 2-hop conflict);

(7) After the intended source node receives the INTEREST message, it unicasts a REPLY message back to the sink (every node records the sender's ID of the first copy of the INTEREST message as the next hop back to the sink);

(8) On receipt of a CHANGE message, a node updates its next hop back to the sink, if necessary.

7.1.4 The impacts of packet loss

Since there are only three kinds of packets utilized in reactive ID conflict resolution, it is easy to analyze the impacts of packet loss.

(1) INTEREST message

In case that a copy of the INTEREST message is lost, some conflicts may still exist. However, they will not prevent the transmission of reply messages, and will be resolved during the next data communication, as illustrated in Fig. 7.4.

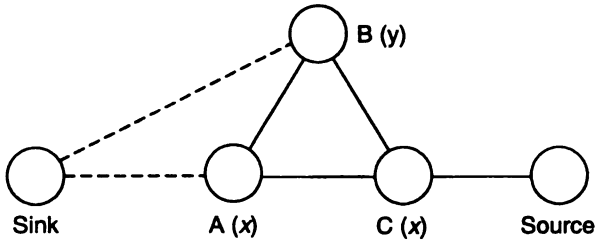


Figure 7.4 An example of packet loss

In Fig. 7.4, there are a 1-hop conflict for nodes A and C, and a 2-hop conflict for node B. There are two cases for node B: it either has a separate path back towards the sink, or node A is the next hop along the path from node B to the sink. In either case, nodes A and C miss each other's INTEREST message, and thus node C can only hear the INTEREST

message from node B and record node B as its upstream node. Once node C forwards the REPLY message from the source to node B, node B can send it back to the sink properly in the first case. In the second case, the REPLY message forwarded by node B will be received by both nodes A and C. Node A can send it back to the sink, node C just drops the duplicate REPLY message and waits for the next ID conflict resolution.

(2) RESOLVE message

In the case of the broadcast of the RESOLVE message, if one of the conflicting neighbors does not receive it, there will be no conflict. If neither/none receives it, then it is similar to the loss of INTEREST message as mentioned above.

(3) CHANGE message

If a copy of the CHANGE message is lost, the path back towards the sink will be broken in the worst case. Thus, the changing node needs to keep its old address for some time, and notifies the sender of a reply message about the change if the reply message contains its old address.

More mechanisms can be included to improve robustness of the scheme. For example, if a node finds there is a 2-hop conflict between its next hop and another neighbor, or it overhears a RESOLVE message that contains its next hop's address, it can set a timer waiting for receipt of a CHANGE message from its next hop. If the timer expires before it receives the CHANGE message, it can query the next hop for its new address.

7.1.5 Security mechanisms

Due to the limited computing power and power supply, it is difficult to implement traditional security mechanisms that are computation intensive in a sensor network. This subsection discusses some typical attack schemes from a malicious node and the

countermeasures in the reactive ID assignment scheme. Note that the authenticity and integrity of INTEREST and REPLY messages are related to secure data communications in the sensor network, and thus beyond our topic. We only focus on the RESOLVE and CHANGE messages introduced in the reactive ID conflict resolution scheme. We denote the malicious node as node M, all the other nodes are normal, as illustrated in Fig. 7.5.

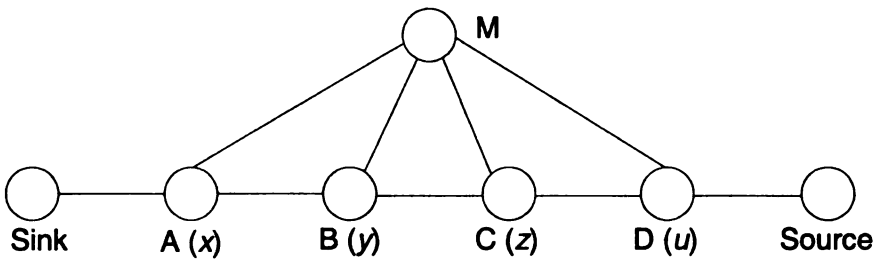


Figure 7.5 A part of the sensor network

(1) Spoofing attacks

Spoofing attacks can undermine the path from the source back towards the sink, as illustrated in Fig. 7.5.

In Fig. 7.5, node B records node A with address of x as its next hop back towards the sink, and node C records node B with address of y as its next hop. Node M happens to know it and broadcasts a false CHANGE message. It impersonates node A and claims that its address of x is changed to z. On receipt of the CHANGE message, node B changes its next hop to the address of z, which forms a loop.

Since the random value or hop count value for node A is known to all its neighbors, it is easy for node M to launch this kind of attack. We can refer to the solution in [15] and designate that the hash value of random number be piggybacked in the broadcast of INTEREST messages and stored in its neighbors. The random number itself must be included in the CHANGE message. If the stored hash value is equal to the hash value of the random number, the origin of the CHANGE message can be authenticated. The node

that broadcasts the CHANGE message should choose a new random value and put the hash value in the CHANGE message for the purpose of the subsequent authentication.

(2) DoS attacks

The RESOLVE message can be utilized by the malicious node, node M, to initiate DoS attacks. For example, on receipt of a RESOLVE message, node B needs to change its address if its address is included in the message. Since the address is used for communications between neighboring nodes only, an address change will not cause any problem by itself. However, if the address range is small, node B will soon find that there is no choice left for its address because it seems that all the addresses have been occupied by its 2-hop neighbors.

One solution is to increase the address range, which leads to increase in communication overhead and can be easily circumvented. Another solution is to lower node B's power for receiving packets because in normal cases (where there is no attacks) it means that the node density is high. When the power is low enough, the link between nodes B and M will be broken, and thus the RESOLVE message will not be received. If node M keeps broadcasting false RESOLVE messages, it may be isolated by other nodes, with the side effect that node B may also be isolated. A better solution is that when the number of occupied addresses is greater than or equal to a threshold value of the address range, node B regards the link between itself and node M as "broken" virtually, which means that it ignores any messages from node M, and refrains from sending any messages (except for broadcast messages) to node M. Because node B is still connected to nodes A and C, it will not be isolated.

(3) DoS and spoofing attacks

If node M impersonates node C in sending false RESOLVE messages, we need to apply the same approach to counter spoofing attacks as mentioned above to authenticate the origin of a RESOLVE message.

7.2 Simulation

The simulations for both our reactive ID assignment scheme and the schemes proposed in [49] [50] (the former aims at general sensor networks and is denoted as proactive-1, the latter aims at stationary sensor networks and is denoted as proactive-2) are implemented to compare their performance in *ns*–2 (version 2.27) with CMU extension for ad hoc networks. For a stationary sensor network, the nodes are placed in a grid. For a mobile sensor network, the random waypoint mobility model is adopted in the simulation.

7.2.1 Simulation verification

A simple simulation scenario is run to verify the correctness of the implementation, which has 15 nodes in a 5×3 grid, as illustrated in Fig. 7.6. The numbers shown in the figure are the unique IDs (UID) of the nodes, which are used for analysis only and should not appear in reality. The distance between two nodes is 200 meters so that a node in the middle of the network has 4 direct neighbors. The size for the address is 4 bits.

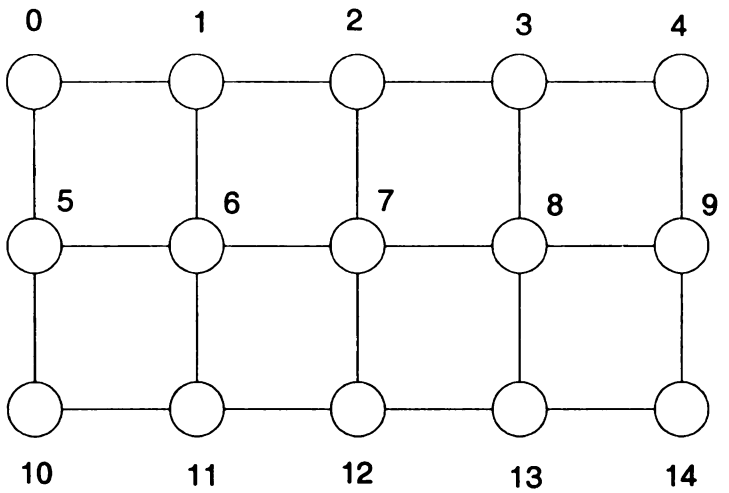


Figure 7.6 A sensor network in 5 3 grid

In the beginning, all the nodes choose a random ID, which is placed in the parentheses following its UID, as in Fig. 7.7. After initialization, there are two cases of 1-hop conflicts (between nodes 3 and 8, nodes 4 and 9), and one case of 2-hop conflict (between nodes 6 and 10).

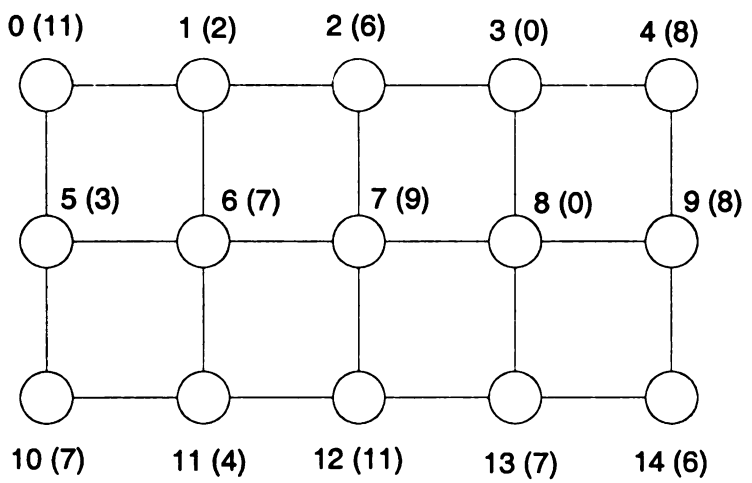


Figure 7.7 Every node chooses a random ID

Once node 0 broadcasts an INTEREST message for destination node 9, all the nodes forward the message. After node 3 receives the INTEREST message from node 8, it knows there is a 1-hop conflict. Node 3 first changes its ID to 6 randomly, which conflicts with node 2's ID. Because node 3 has recorded node 2's ID, it then changes its

ID to 13 randomly, which is appropriate. Similarly, node 8 changes its ID from 0 to 2^{24}. After node 4 broadcasts, node 9 changes its ID from 8 to 0, then to 1. Node 3 receives the INTEREST message from a node with ID of 7 twice, so it can conclude that there is a case of 2-hop ID conflict. It then broadcasts a RESOLVE message to its neighbors. On receipt of the RESOLVE message, both nodes 6 and 10 change their IDs. All these changes result in the IDs illustrated in Fig. 7.8.

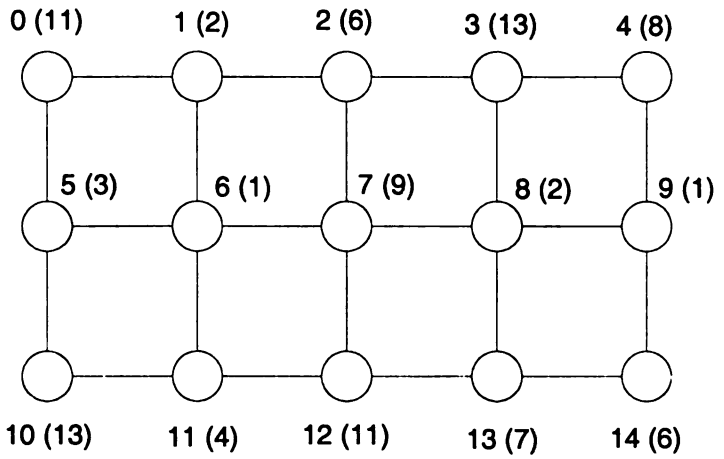


Figure 7.8 Every node has a locally unique ID

Fig. 7.9 shows the number of packets received at each node for two broadcasts of INTEREST messages[25]. The payload packets include both INTEREST messages and REPLY messages. The overhead packets include both CHANGE and RESOLVE messages. Compared with the number of payload packets, the number of overhead packets is small. Because 1-hop and 2-hop conflicts are resolved during the first

---

[24] Because nodes 3 and 8 broadcast almost simultaneously, they both change their IDs. If there is an interval between their broadcasts, only one needs to change.

[25] We use the number of received packet as the metric for communication overhead because each received packet consumes receiver's power.

broadcast, and all the nodes are stationary, there is no increase on overhead during the second broadcast.

As the address range increases, the communication overhead decreases. When the size for the address field is set to 5 bits in the simulation, there is no communication overhead because the randomly chosen IDs satisfy 2-hop uniqueness.

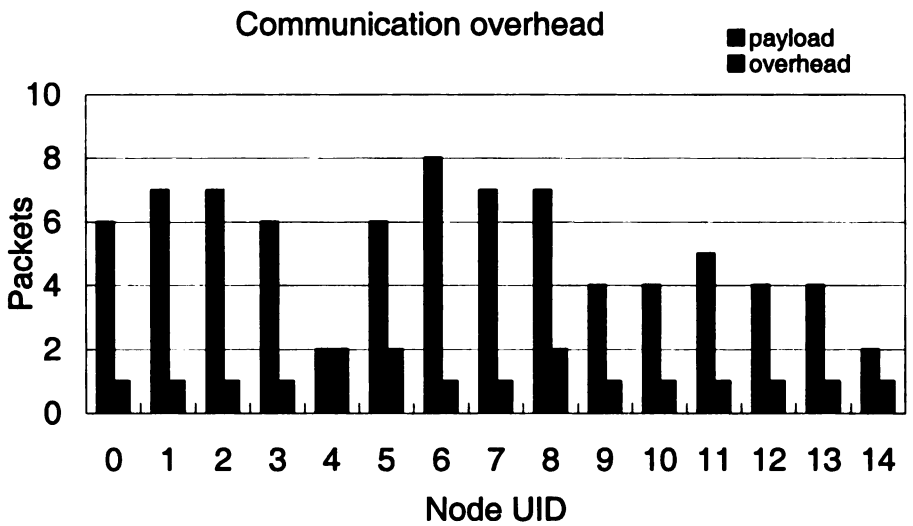


Figure 7.9 The number of packets received at each node for 2 broadcasts

7.2.2 Simulation of mobile sensor networks

The scenario for the simulation of a mobile sensor network contains 15 nodes moving around inside a square of 500×500 m^2 with random waypoint mobility model. Although a sensor node has very limited mobility nowadays, it will be possible in the near future that a sensor node is combined with a UAV or a robot, which makes it capable of moving arbitrarily.

The simulation is running for 180 seconds. The maximum speed is 10 m/s, the minimum speed is 2 m/s. The pause time is 0 second, which means all the nodes keep moving. The size for the address is 5 bits because the node density of the network is high.

In proactive scheme, the sensor nodes broadcast HELLO messages periodically with the interval of 10 seconds. In the reactive ID assignment scheme, at simulation time of 3.0 second, all the nodes broadcast a HELLO message so that they can record their neighbor information[26]. The neighbor table is shown in Table 7.1.

Table 7.1 The neighbor table before ID conflict resolution

Row	Node (field 1)	1-hop neighbors (field 2)
1	0 (23)	1 (2) 6 (8) 7 (10) 8 (12) 9 (18) 12 (2) 14 (8)
2	1 (2)	0 (23) 4 (16) 7 (10) 8 (12) 9 (18) 10 (13) 12 (2) 14 (8)
3	2 (28)	(23) 1 (2) 6 (8) 7 (10) 8 (12) 9 (18) 10 (13) 11 (12) 12 (2) 13 (27) 14 (8)
4	3 (15)	4 (16) 6 (8) 7 (10) 11 (12) 12 (2) 13 (27)
5	4 (16)	1 (2) 3 (15) 6 (8) 7 (10) 8 (12) 10 (13) 11 (12) 12 (2) 13 (27)
6	5 (0)	0 (23) 1 (2) 4 (16) 6 (8) 7 (10) 8 (12) 9 (18) 11 (12) 12 (2) 13 (27) 14 (8)
7	6 (8)	0 (23) 3 (15) 4 (16) 7 (10) 11 (12) 12 (2) 13 (27)
8	7 (10)	0 (23) 1 (2) 3 (15) 4 (16) 6 (8) 8 (12) 10 (13) 11 (12) 12 (2) 13 (27) 14 (8)
9	8 (12)	0 (23) 1 (2) 4 (16) 7 (10) 9 (18) 10 (13) 12 (2) 14 (8)
10	9 (18)	0 (23) 1 (2) 8 (12) 14 (8)
11	10 (13)	1 (2) 4 (16) 7 (10) 8 (12) 12 (2) 14 (8)
12	11 (12)	3 (15) 4 (16) 6 (8) 7 (10) 12 (2) 13 (27)
13	12 (2)	0 (23) 1 (2) 3 (15) 4 (16) 6 (8) 7 (10) 8 (12) 10 (13) 11 (12) 13 (27) 14 (8)
14	13 (27)	3 (15) 4 (16) 6 (8) 7 (10) 11 (12) 12 (2)
15	14 (8)	0 (23) 1 (2) 7 (10) 8 (12) 9 (18) 10 (13) 12 (2)

In Table 7.1, every node is represented with its unique ID followed by its local ID in parentheses. For example, in Row 1, node 0 has the ID of 23. Its direct neighbors include node 1 (with ID of 2), node 6 (with ID of 8), etc. The neighbor table is not symmetric due to a periodical purge operation on the neighbor table. In the neighbor table, if one node in field 2 has the same ID as the node in field 1, there is a 1-hop conflict, such as node 2 and

[26] The Hello message is not used in the reactive ID assignment scheme. It is only used in the simulation for the purpose of analysis.

node 12 in Row 2. If two nodes in field 2 have the same ID, there is a 2-hop conflict, such as node 8 and node 11 in Row 5.

After reactive ID conflict resolution at simulation time of 6.0 second, the neighbor table at the time of 8.0 second is shown in Table 7.2, which shows that 1-hop conflicts and 2-hop conflicts are resolved.

Table 7.2 The neighbor table after ID conflict resolution

Row	Node (field 1)	1-hop neighbors (field 2)
1	0 (23)	1 (6) 2 (28) 5 (0) 6 (14) 8 (4) 9 (18) 12 (24) 14 (25)
2	1 (6)	0 (23) 2 (28) 4 (16) 5 (0) 6 (14) 8 (4) 9 (18) 10 (13) 12 (24) 14 (25)
3	2 (28)	0 (23) 1 (6) 4 (16) 5 (0) 6 (14) 8 (4) 9 (18) 10 (13) 11 (17) 12 (24) 14 (25)
4	3 (15)	4 (16) 6 (14) 11 (17) 12 (24)
5	4 (16)	1 (6) 2 (28) 3 (15) 5 (0) 6 (14) 8 (4) 10 (13) 11 (17) 12 (24) 14 (25)
6	5 (0)	0 (23) 1 (6) 2 (28) 4 (16) 6 (14) 8 (4) 9 (18) 11 (17) 12 (24) 14 (25)
7	6 (14)	0 (23) 1 (6) 2 (28) 3 (15) 4 (16) 5 (0) 8 (4) 11 (17) 12 (24) 14 (25)
8	7 (10)	1 (6) 2 (28) 3 (15) 4 (16) 5 (0) 6 (14) 8 (4) 10 (13) 11 (17) 12 (24) 14 (25)
9	8 (4)	0 (23) 1 (6) 2 (28) 4 (16) 5 (0) 6 (14) 9 (18) 10 (13) 12 (24) 14 (25)
10	9 (18)	0 (23) 1 (6) 2 (28) 5 (0) 8 (4) 14 (25)
11	10 (13)	1 (6) 2 (28) 4 (16) 8 (4) 12 (24) 14 (25)
12	11 (17)	2 (28) 3 (15) 4 (16) 5 (0) 6 (14) 12 (24) 13 (27)
13	12 (24)	0 (23) 1 (6) 2 (28) 3 (15) 4 (16) 5 (0) 6 (14) 8 (4) 10 (13) 11 (17) 14 (25)
14	13 (27)	2 (28) 3 (15) 4 (16) 5 (0) 6 (14) 11 (17) 12 (24)
15	14 (25)	0 (23) 1 (6) 2 (28) 4 (16) 5 (0) 6 (14) 8 (4) 9 (18) 10 (13) 12 (24)

The communication overhead is illustrated in Fig. 7.10, with the number of packets received in the proactive scheme being far greater than that in the reactive scheme. The reason why the communication overhead is high is due to the high density of the network. Compared to the communication overhead due to periodical broadcasts, the reactive

scheme with a 4-bit hop count field in an INTEREST message introduced in subsection 3.2 still saves much power, as long as the frequency of broadcasting interest messages is less than that of Hello messages (6 messages/minute).

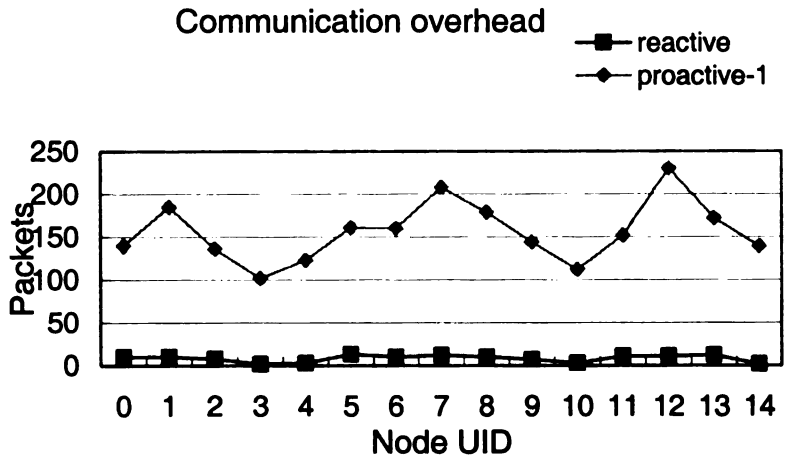


Figure 7.10 Communication overhead for simulation of mobile scenario

7.2.3 Simulation of large-scale stationary sensor networks with high node density

The simulations of large-scale stationary sensor networks with high node density were done for 100 (10×10), 225 (15×15), and 324 (18×18) nodes deployed in a grid. The length for the address field is 8 bits. The transmission range is 250 meters, while the nodes are placed in the interval of 80 meters in the grid. Thus, one node has a minimum number of 10 1-hop neighbors and a maximum number of 28 1-hop neighbors, which is confirmed by the simulation results.

In the proactive scheme, every node begins to broadcasts periodic HELLO messages containing its neighbor table. The interval of HELLO messages is 8 seconds, according to [13]. 2 seconds after the last broadcast, the neighbor table that is similar to Table 7.1 is printed out for analysis. In the reactive scheme, each node broadcasts a HELLO message

in the end of the simulation to build the neighbor table for analysis. A Perl script is utilized to locate 1-hop and 2-hop conflicts in the neighbor table.

Fig. 7.11 shows the sum of received control packets at all the nodes for each simulation. Notice that the 4 cycles recommended for the proactive scheme are not adequate to eliminate 1-hop or 2-hop conflicts with high node density for 225 nodes and 324 nodes. The simulations show that 13 cycles are minimum for 225 nodes, and that 15 cycles are minimum for 324 nodes. The reason is that although each node waits for a random delay before broadcasting HELLO messages, the HELLO messages are still lost at some neighbors due to high node density. According the simulation results, even for a stationary sensor network, the reactive scheme achieves much lower communication overhead than the proactive scheme.

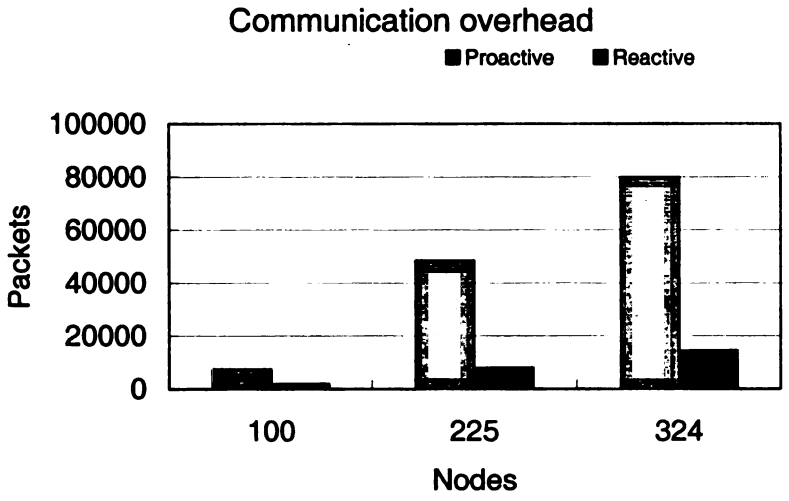


Figure 7.11 Number of received control packets

In the simulations of both mobile sensor networks and stationary scenarios, the number of control packets received at each node is far fewer for the reactive scheme than proactive schemes. Thus, both bandwidth and power are saved in the reactive scheme. Furthermore, no neighbor information is carried in the control messages, which leads to

shorter length and less power consumption for each individual control message. In summary, longer lifetime can be achieved with the reactive scheme.

7.3 Conclusion

Due to the differences between a MANET and a sensor network, the pre-existing autoconfiguration algorithms for the former cannot be simply applied to the latter. However, a mechanism is still necessary to assign locally unique addresses to sensor nodes efficiently. Compared with proactive schemes, a reactive ID assignment approach is proposed to accomplish the goal and preserve more power by means of delaying ID conflict resolution until necessary. It has no requirement on a priori unique IDs of the sensor nodes, and is easy to integrate with the directed diffusion communication paradigm. The effect of packet loss and potential attacks on the scheme are also studied with countermeasures provided.

CHAPTER 8 CONCLUSION REMARKS

8.1 Summary of the Work

The study on autoconfiguration is needed for the practical application of the MANET because IP address allocation is the first step of network configuration before its application. Existing autoconfiguration schemes are inefficient in the term of communication overhead and latency in a large-scale network, vulnerable to all kinds of attacks, and ignorant of the communication overhead caused by IP address change. In this research project, we proposed an innovative autoconfiguration algorithm to compute the IP addresses locally, which saves communication overhead and latency, and thus has better scalability. To defeat many kinds of attacks on autoconfiguration, a secure autoconfiguration scheme is proposed to be valid even if there are malicious nodes in the network. In the presence of autoconfiguration, global connectivity, and hierarchical addressing scheme, a mobile node is likely to change its IP address due to network merger or mobility. To save the communication overhead caused by address change, an IP address handoff scheme is proposed to repair broken routing fabrics and preserve communication states. Security concern is another important factor in deployment of MANETs. Because the autoconfiguration changes some underlying assumptions in the traditional security framework for a hardwired network, we also proposed a multiple-key cryptography-based distributed certificate authority scheme to improve security in the MANET. We also proposed a reactive ID assignment scheme to allocate locally unique

IDs for sensor nodes, which is more efficient than proactive methods in a large-scale dense sensor network or a mobile sensor network.

Some of research work has been published in [65] – [68].

8.2 Future Work

The future work on autoconfiguration and security includes:

(1) The re-design of the partition function in Prophet Address Allocation

The design of the partition function is the most important issue in Prophet Address Allocation. In the current design, some states are not utilized in the generation of addresses, such as those with the exponential array of (0, 1, 0, ...) and (0, 0, 1, 0, ...). A new scheme is necessary to utilize these states to further simplify the computation and decrease the probability of conflicts.

(2) The study on more complicated attack schemes on autoconfiguration

Our Secure Prophet Address Allocation can defeat DoS attack, IP spoofing attack, "state pollution" attack, and Sybil attack. In these cases, only one malicious node initiates the attack. There are more complicated attack schemes when two or more malicious nodes conspire. We need to take these scenarios into consideration in the improvement of secure autoconfiguration.

(3) The integration of autoconfiguration with other network protocols

Autoconfiguration generates IP addresses that are regarded as nodes' identities and used in routing and forwarding. Thus, autoconfiguration is closely related to other network protocols, such as routing protocols. The cross-design of the autoconfiguration with another network protocol may improve the efficiency of both protocols.

BIBLIOGRAPHY

[1] David B. Johnson, "Routing in Ad Hoc Networks of Mobile Hosts," Proceeding of the IEEE Workshop on Mobile Computing Systems and Applications, December 1994

[2] S. Corson, J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations," RFC 2501, available at http://www.ietf.org/rfc/rfc2501.txt

[3] P. Varaiya, "Smart Cars on Smart Roads: Problem of Control," IEEE Trans. Auto. Control, vol. 38, no. 2, 1993

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," Proceedings of MOBICOM 2000, Boston, MA, August 2000

[5] J. Kumagai, "Life of birds," IEEE Spectrum, Vol. 41, Issue 4, pp. 42-49, April 2004

[6] T. Clausen, P. Jacquet, A. Laouiti, et al., "Optimized Link State Routing Protocol," draft-ieft-manet-olsr-06.txt, September 2002 (work in progress)

[7] M. Gerla, X. Hong, and G. Pei, "Fisheye State Routing Protocol (FSR) for Ad Hod Networks," draft-ietf-manet-fsr-03.txt, June 2002 (work in progress)

[8] D. Johnson, D. Maltz, Y. Hu, and J. Jetcheva, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)," draft-ietf-manet-dsr-10.txt, July 2004, (work in progress)

[9] C. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector (AODV) Routing," Network Working Group RFC 3561, July 2003

[10] T. Narten, E. Nordmark, and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)," Network Working Group RFC 2461, December 1998

[11] Zero Configuration Networking, http://www.ietf.org/html.charters/zeroconf-charter.html

[12] R. Droms, "Dynamic Host Configuration Protocol," Network Working Group RFC 2131, March 1997

[13] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration," Network Working Group RFC 2462, December 1998

[14] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," draft-ietf-manet-zone-zrp-04.txt, July, 2002 (work in progress)

[15] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Intrazone Routing Protocol (IARP) for Ad Hoc Networks," draft-ietf-manet-zone-iarp-02.txt, July, 2002 (work in progress)

[16] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Interzone Routing Protocol (IERP) for Ad Hoc Networks," draft-ietf-manet-zone-ierp-02.txt, July, 2002 (work in progress)

[17] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Bordercast Resolution Protocol (BRP) for Ad Hoc Networks," draft-ietf-manet-zone-brp-02.txt, July, 2002 (work in progress)

[18] G. Pei, M. Gerla, and X. Hong, "LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility," Proceedings of IEEE/ACM MOBIHOC 2000, Boston, MA, August 2000

[19] Young-Bae Ko, and Nitin H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," MOBICOM'98, 1998

[20] R. Jain., A. Puri, and R. Sengupta, "Geographical Routing Using Partial Information for Wireless Ad Hoc Networks," IEEE Personal Communication, February 2001, Vol. 8, No 1, pp 48-57

[21] S. Basagni, et al., "A Distance Routing Effect Algorithm for Mobility, (Dream)," Proceedings of 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MOBICOM'98, Dallas, TX, USA, 1998, pp. 76-84

[22] Z. J. Haas and B. Liang, "Ad Hoc Mobility Management with Uniform Quorum Systems," IEEE/ACM Transaction on Networking, vol. 7, no. 2, Apr. 1999, pp. 228-40

[23] J. Li, et al., "A Scalable Location Service for Geographic Ad Hoc Routing," Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking, Boston, MA, 2000, pp. 120–30

[24] Ivan Stojmenovic, "Home Agent Based Location Update and Destination Search Schemes in Ad Hoc Wireless Networks," Technical Report TR-99-10, Computer Science, SITE, Univ. Ottawa, Sept. 1999

[25] Seung Yi, Prasad Naldurg, and Robin Kravets, "A Security-Aware Routing Protocol for Wireless Ad Hoc Networks," The 6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2002), 2002

[26] Bridget Dahill, Brian N. Levine, Elizabeth Royer, and Clay Shields, "A Secure Routing Protocol for Ad Hoc Networks," Proceedings of the 10th Conference on Network Protocols (ICNP), November 2002

[27] Yih-Chen Hu, Adrian Perrig, and David B. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks," Proceedings of MOBICOM'02, Atlanta, GA, September 2002

[28] E. M. Royer and C. E. Perkins, "Multicast Operation of the Ad hoc On-Demand Distance Vector Routing Protocol," Proceedings of IEEE MOBICOM'99, Seattle, WA, August 1999, pp. 207-218

[29] M. Liu, R. R. Talpade, and A. McAuley, "AMRoute: Adhoc Multicast Routing Protocol," Technical Report CSHCN TR 99-1, University of Maryland, 1999

[30] C. W. Wu and Y. C. Tay, "AMRIS: AMR with Increasing Sequence Numbers: a Multicast Protocols for Ad Hoc Wireless Networks," Proceedings in IEEE MILCOM'99, Atlantic City, Nov 1999

[31] S-J. Lee, M. Gerla, and C-C. Chiang, "On-Demand Multicast Routing Protocol," Proceedings of IEEE WCNC'99, New Orleans, LA, Sep. 1999

[32] S. Lee, and C. Kim, "Neighbor Supporting Ad hoc Multicast Routing Protocol," Proceedings of First Annual Workshop on Mobile Ad Hoc Network & Computing, MOBIHOC 2000, Boston, pages 37-50, August 2000

[33] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das, "IP Flooding in Ad hoc Mobile Networks," draft-ietf-manet-bcast-00.txt, November 2001 (expired)

[34] Amir Qayyum, Laurent Viennot, and Anis Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks," 35th Annual Hawaii International Conference on System Sciences (HICSS'2002), January 2002

[35] W. Peng and X. Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," Proceedings of MOBIHOC'00, Boston, MA, August 2000

[36] C. Perkins, J. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "IP Address Autoconfiguration for Ad Hoc Networks," draft-ietf-manet-autoconf-01.txt, November 2001 (work in progress)

[37] K. Weniger and M. Zitterbart, "IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks," Proceedings of European Wireless 2002, Florence, Italy, February 2002

[38] J.-H. Jeong, H.-W. Cha, J.-S. Park, and H.-J. Kim, "Ad hoc IP address autoconfiguration," draft-jeong-adhoc-ip-addr-autoconf-00.txt, May 2003 (work in progress)

[39] N. Vaidya, "Duplicate Address Detection in Mobile Ad Hoc Networks," Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'02), Lausanne, Switzerland, June 2002

[40] A. Misra, S. Das, A. McAuley, and S. K. Das, "Autoconfiguration, Registration, and Mobility Management for Pervasive Computing," IEEE Personal Communication System Magazine, Vol. 8, pp. 24-31, August 2001

[41] M. Mohsin and R. Prakash, "IP Address Assignment in a Mobile Ad Hoc Network," Proceedings of MILCOM 2002, Anaheim, CA, October 2002

[42] S. Nesargi and R. Prakash, "MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network," Proceedings of the 21st Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2002), New York, NY, June 2002

[43] C. Perkins (editor), "IP mobility support," Network Working Group RFC 2002, October 1996

[44] A. Shamir, "How to share a secret," Communications of the ACM, Vol.22, pp. 612 - 613, November 1979

[45] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," Proceedings of Advances in Cryptography (Crypto 89), Lecture Notes in Computer Science, Vol. 435, Springer-Verlag, pp. 307 – 315, 1989

[46] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive public key and signature systems," ACM Conference on Computer and Communication Security, Zürich, December 1996

[47] L. Zhou and Z. J. Haas, "Securing ad hoc networks," IEEE Network, Vol. 13, No. 6, pp. 24-30, November/December 1999

[48] M. Bechler, H.-J. Hof, D. Kraft, F. Pählke, and L. Wolf, "A cluster-based security architecture for ad hoc networks," Proceedings of the 23rd Conference of IEEE Communication Society (INFOCOM 2004), Hong Kong, China, March 2004

[49] C. Schurgers, G. Kulkarni, and M. B. Srivastava, "Distributed Assignment of Encoded MAC Addresses in Sensor Networks," Proceedings of MOBIHOC 2001, Long Beach, CA, October 2001

[50] C. Schurgers, G. Kulkarni, and M. B. Srivastava, "Distributed On-demand Address Assignment in Wireless Sensor Networks," IEEE Transactions on Parallel and Distributed Systems, Vol.13, No.10, pp. 1056-1065, October 2002

[51] K. Fall and K. Varadhan (editors), The ns Manual - the VINT Project, http://www.isi.edu/nsnam/ns/ns-documentation.html, December 2003

[52] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad Hoc Routing Protocols," Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 85–97, October 1998

[53] S. Čapkun, L. Buttyán, and J. P. Hubaux, "Self-organized public-key management for ad hoc networks," IEEE Transactions on Mobile Computing, Vol.2, No. 1, January-March 2003

[54] N. Abramson, "The ALOHA system-another alternative for computer communications," In Proceedings of the Fall 1970 AFIPS Computer Conference, pp. 281-285, November 1970

[55] J. Couceur, "The sybil attack," Proceedings of the 1st Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, March 2002

[56] R. Wakikawa, J. T. Malinen, C. E. Perkins, A. Nilsson, and A. J. Tuominen, "Global connectivity for IPv6 mobile ad hoc Networks," draft-wakikawa-manet-globalv6-02.txt, November 2002 (work in progress)

[57] E. M. Belding-Royer, Y. Sun, and C. E. Perkins, "Global connectivity for IPv4 mobile ad hoc networks," draft-royer-manet-globalv4-00.txt, November 2001 (work in progress)

[58] G. Pei and M. Gerla, "Mobility management for hierarchical wireless networks," Mobile Networks and Application (MONET), Vol. 6, No. 4, pp 331-337, August 2001

[59] P. Engelstad and G. Egeland, "Name resolution in on-demand MANETS and external IP networks," draft-engelstad-manet-name-resoltuion-00.txt, February 2003 (work in progress)

[60] R. Rivest, "MD5 message-digest algorithm," Network Working Group RFC 1321, April 1992

[61] J. Touch and B. Parham, "Computation of the Internet checksum via incremental update," Network Working Group RFC 1624, May 1994

[62] R. Russell and H. Welte, "Linux netfilter hacking HOWTO," http://www.netfilter.org/documentation, July 2002

[63] C. Boyd, "Some applications of multiple key ciphers," Proceedings of Advances in Cryptography (Eurocrypt'88), Lecture Notes in Computer Science, Springer-Verlag, pp. 455 – 467, 1988

[64] T. P. Pedersen, "A threshold cryptosystem without a trusted party," Proceedings of Advances in Cryptology (Eurocrypt'91), Lecture Notes in Computer Science, Vol. 547, Springer-Verlag, pp. 522-526, 1991

[65] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet address allocation for large scale MANETs," Proceedings of the 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2003), San Francisco, CA, April 2003

[66] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet address allocation for large scale MANETs," Ad Hoc Networks Journal, Vol. 1, Issue 4, pp 423-434, November 2003

[67] H. Zhou, M. W. Mutka, and L. M. Ni, "IP address handoff in the MANET," Proceedings of the 23rd Conference of IEEE Communication Society (INFOCOM 2004), Hong Kong, China, March 2004

[68] H. Zhou, M. W. Mutka, and L. M. Ni, "Multiple-key Cryptography-based Distributed Certificate Authority in Mobile Ad-hoc Networks," to appear in the Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2005)