ADDRESS ALLOCATION TO MOBILE AD HOC NETWORKS

by

Zeeshan Sakander

B.Sc. University of Manitoba, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Fall 2006

APPROVAL

Name: Zeeshan Sakander

Degree: Master of Science

Title of thesis: Address Allocation To Mobile Ad Hoc Networks

Examining Committee: Dr. Jian Pei
Chair

Dr. Jiangchuan Liu
Assistant Professor, Computing Science
Senior Supervisor

Dr. Qianping Gu
Professor, Computing Science
Supervisor

Dr. Jie Liang
Assistant Professor, Engineering Science
SFU Examiner

Date Approved: October 6th, 2006

Abstract

Addressing in MANETs is of significant importance, as a mobile device cannot participate in unicast communications until it is assigned a conflict-free IP address. All routing protocols assume nodes to be configured *a priori* with a unique IP address. Allocating addresses to mobile nodes is a fundamental and difficult problem. Unlike infrastructure based networks, MANETs support autonomous and spontaneous networking and therefore, should be capable of self-organization and self-configuration. We present a new IP based address allocation protocol for MANETs based on quadratic residues. Each node in the network is capable of assigning a unique IP address with low latency. Addresses are reclaimed automatically, as the quadratic residues lie in cycles. This significantly reduces communication overhead and bandwidth. Our approach also has support for network merging and partitioning. The proposed scheme has low communication overhead, even address distribution and low latency when applied to large scale MANETs.

Keywords: Address Allocation; Mobile Ad-Hoc Network; Node Joining; Quadratic Residue Protocol; Temporary Address; Permanent Address

Dedication

In memory of my late maternal Grandfather ***Chaudhry Nasir Ahmed****,*
who passed away on 30th November,1999

"Never Lose Hope in Life and Keep Trying!"

Acknowledgments

First and foremost, my grateful thanks go to Dr. Jiangchuan Liu, the supervisor of my Master's thesis. His invaluable and patient guidance, constructive feedback and support accompanied me in every stage of my thesis research.

My special thanks goes to my parents, brothers and sisters for their love and encouragement during my stay in Canada. Without their help and encouragement I would have not been able to pursue new goals in life. I would like to express my deepest feelings to my wife for allowing me the time and space to complete my research, and for her patience and assistance throughout my thesis.

I would also like to thank my friends from the University of Manitoba, who were a great help during my Bachelor's degree and to everyone who read my thesis.

Last but not least, I would like to thank my late grandfather for all that he taught me during his lifetime and the countless support he provided throughout his life.

Contents

List of Tables

List of Figures

List of Abbreviations

MANET	Mobile Ad Hoc Network	1
QR	Quadratic Residue	35
QNR	Quadratic Non Residue	35
PS	Pseudo Square	38
DAD	Duplicate Address Detection	5
NID	Network Identifier	7
XactID	Transaction Identity	24
ACK	Acknowledgement	20
NACK	Negative Acknowledgement	20
TTL	Time to Live	15
AddrReply	Address Reply Message	18
AddrConfig	Address Configure Message	28
ZRP	Zone Routing Protocol	51
AODV	Ad Hoc on Demand Distance Vector Routing	52

List of Symbols

a	..... Quadratic Residue Square
$p^{'}$	..... Verifies that p is a safe prime
$q^{'}$	..... Verifies that q is a safe prime
N	..... Product of two primes p * q
$\phi(n)$	..... Single network key
S	..... Node assigning the address to a new node
R	..... Wired Network Router connecting with MANET
k	..... Timeout period
Z	..... Destination node in wired network
b	..... Encryption exponent
S_0	..... Initial address of an address block

Chapter 1

Introduction

A Mobile Ad-Hoc Network (MANET) is an independent self organizing network in which each node functions as both an end host and a router. This form of wireless network is created by mobile nodes without any existing or fixed infrastructure. An ad-hoc network is a form of community network because it relies on the willingness of mobile hosts to forward and relay packets toward the destination. The formed network can be changed dynamically without the need of any system administrator. Ad hoc networks generally consist of hand held devices and laptop computers. These devices usually have limited transmission range, bandwidth and battery power.

The topology of a mobile ad-hoc network is typically highly dynamic because its nodes are free to move independently and randomly. The size of a MANET is constantly changing as nodes come in and out of the network range. A node is not part of a MANET until it is within the transmission range of an already configured node in the MANET. During the time a node is present in the MANET; it may or may not participate in communication or packet forwarding.

Nodes in the MANET need some form of identity before participating in any form of communication. Each end host in the MANET need to be uniquely addressed so that the packets can be relayed hop-by-hop and delivered ultimately to the destination. Routing protocols in MANET assume *a priori* that mobile nodes are configured with a valid (conflict free) IP address. Each node has a 48 bit MAC address at the link layer level. However, each end host needs some form of network address to successfully establish connection between two end hosts. This network address will uniquely identify each node present in the network. Using traditional IP-based address assignment, such as DHCP [8], is not possible because nodes in the MANET are highly mobile and a central authority is not always reachable. Mobile IP [6] is also not a solution because MANET nodes do not stay connected to a wired network all the time. Addressing thus becomes significant in *ad-hoc* wireless networks due to the absence of any centralized coordinator. An address allocation protocol is required to enable dynamic address assignment to all nodes in a MANET.

Ad-hoc networks have been used in military operations, shopping malls, disaster relief operations, conference rooms and peer to peer networks. Extensive research has been done on MANETs but still many problems and challenges remain unsolved. In MANETs, one must consider scalability limitations, communication overhead, bandwidth constraints, routing protocols, address assignment, power consumption, security concerns, and quality of service (QoS) mechanisms.

1.1 Motivation

Address allocation in MANETs presents special challenges not found in wired networks. MANET topologies are highly dynamic, and a central authority is not always reachable. Manual configuration is not always possible and there is no network administrator. Address allocation in MANETs must configure each node with a unique IP address before routing can begin. Upon arrival of a new node, a MANET must be able to select, allocate and assign a conflict free network layer IP address to an unconfigured node. The address configuration

process must be fast since a node cannot participate in communication until it has configured a unique IP address. A distributed address allocation protocol is required where each node in the MANET is capable of assigning a unique address.

Since ad-hoc networks are mobile, network partitions and mergers must be dealt with. MANETs can become separated into several different small partitions, resulting into no communication between them. These network partitions may or may not merge back together later. Two or more independent MANETs may be merged into one big network. MANETs have low bandwidth, limited transmission range and power. Therefore broadcast messages and extra communication overhead should be kept to a minimum. Nodes in the MANET may either switch off from ad-hoc mode rapidly or move away permanently from the MANET without providing any information. Our research attempts to provide a scalable and delay free network layer address assignment protocol which addresses all these issues with almost no additional overhead.

Current address allocation protocols proposed for MANETs aim to provide efficient address assignment in a dynamic network environment in order to enable correct communication in the network. These approaches have certain similarities, but they differ in their usage of centralization and decentralization, and explicit or implicit duplicate address detection. Some approaches are dependent on the routing protocol used in the MANET while others are independent of them. The performance of these approaches varies under different network conditions.

Our algorithm was based on generating a large pseudorandom number to bound the address space within 2^m where m is an integer. By doing this, we do not have to reclaim addresses, as shown in Figure 1.1 below. We can divide the address space into separate disjoint address pools. If m = 16, we would have $2^{16} - 2$ possible addresses. We can divide the address space into address blocks of size 2^{12} each. Thus, we could have up to 10 disjoint address blocks. Already configured nodes can allocate addresses to new nodes. Addresses would repeat automatically after the maximum address size has been reached. Individual partitions can be assigned unique addresses, but there may be inconsistencies in several network partitions.

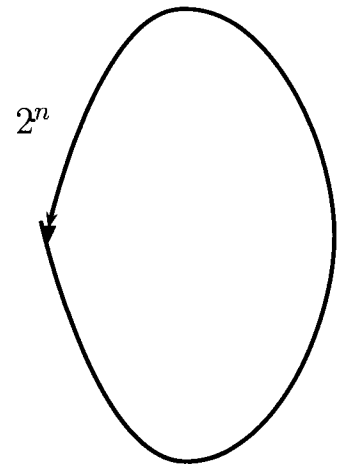


Figure 1.1: Random Address Space

Also, note that much duplication will occur when two independently configured MANET merge together. Suppose one MANET uses 2^m address and another 2^n, where $m < n$. We have 2^m possible duplicate addresses. One of the MANETs has to completely give up its addresses and acquire addresses from the other MANET. This will result in extremely high overhead cost and lot of unnecessary address changes as a result of network merger. Thus all the active connections would be broken, and therefore, we conclude that this is not a practical approach. Our research shows that the addressing function we propose eliminates all drawbacks that are present in the random address generation approach.

1.2 Problem Statement

The main task of an address allocation protocol is to manage the resource address space. It must be able to select, allocate and assign a unique network address to an unconfigured node. The address configuration process should be fast, since a node cannot participate in unicast communication before it owns a unique address. Since every address change may break transport layer connections, unnecessary address changes should be avoided. When nodes leave the network permanently the addresses assigned to those nodes have to be reclaimed and reused. When two different network partitions merge, duplicate addresses

must be detected and resolved quickly to prevent misrouting. Communication between network partitions is not possible. A node may not be even aware of becoming partitioned from the network. Also, address allocation protocol overhead should be kept to a minimum.

The following MANET related issues and scenarios need to be carefully handled for an address allocation protocol to be successful:

- Address space
- Initiator/ prophet node
- New node joining process
- Address leakage
- Distinguishing concurrent requests for address assignment
- Mobility of address assigner
- Mobility of address requester
- Network partitioning
  - Partitions merging back into the network
  - Partitions moving independently or merging with a new network
- Address reclamation
- Network partitioning and subsequent merging
- Detection/distinguishing between multiple MANETs
- Merging of independent networks/MANETs
- Duplicate Address Detection (DAD)
- Communication overhead, limited energy and low bandwidth

- Allocating address or forwarding Discover message
- MANET address block expansion

1.3 Contribution

In this thesis, we devise a new Address Allocation (AA) protocol for mobile ad-hoc networks. Our protocol provides solutions for realistic mobility scenarios in a MANET: network partitions and subsequent mergers, including two or more independent network mergers. We devise an addressing function that predicts future address space, calculates separate address blocks and their respective seeds. We guarantee disjoint address space among different address blocks. Our research shows that our addressing function guarantees address space reclamation automatically without maintaining any data structure of allocated and de-allocated addresses.

We prevent address leaks and manage the address block to prevent exhaustion of the address space. Moreover, we distinguish among multiple concurrent address requests and replies by different nodes in the MANET. Our protocol minimize duplicated address assignment in a MANET. Our protocol has a solution for IPv4 networks, 32-bit network layer address and it can also be easily modified for IPv6 networks. Our solution is highly scalable and is capable of configuring millions of nodes in the MANET. All these problems are addressed with minimum overhead. We provide a fast and convenient way of carefully choosing the address space. Our technique is not dependent on the underlying routing protocol and can be used with either reactive, proactive, or hybrid routing protocols.

We further extend our technique by providing a solution for hybrid network connectivity; that is to connect an independent mobile ad hoc network with an external wired network or internet. Our approach manages ongoing connections and route packets ultimately to their destination while the MANET moves away temporarily from the transmission range of a

wired network.

Our address allocation model for MANET is movited by Prophet Address Allocation [13]. However, our model guarantees a unique address space, which better handles address leakage, distinguishes concurrent requests and provides solutions for hybrid networks connectivity and routing. Also, nodes in the MANET with the smaller network identifier (NID) do not have to break all ongoing connections as they do in [13]. We do not have to run duplicate address detection (DAD) mechanism each time a single node joins from another network or during every network merger. Moreover, we also show that future address allocation in the merged network will be conflict-free and explain how to handle the address space function after a network merger, something which is not done in [13].

Lastly, we design an algorithm for the initiator node to calculate a definite future address space, distinct address blocks and their respective range. Our address allocation mechanism potentially provides security for military operations, search and rescue activities, conference rooms and disaster relief MANETs.

1.4 Thesis Outline

First, in Chapter 2, we provide a survey of different address allocation techniques that have been proposed for MANETs. Next in Chapter 3, we describe the node joining process and initiator node network address space computation. In Chapter 4, we present a detailed explanation of our address generation function. Next in Chapter 5 and 6, we discuss two proposed address allocation protocols. Chapter 7 provides the original and optimized address generation algorithms. In Chapter 8, we discuss hybrid networks. Chapters 9 and 10 discuss a quick way to find length of address block and security in MANETs. Lastly, in Chapter 11, we summarize our results and discuss future research directions.

Chapter 2

Background and Related Work

In the past 10 to 15 years, much research has been done on ad-hoc routing protocols. All the routing protocols presented in that times require each node to have its own unique IP address to transmit packets. Address allocation protocols for hardwired networks [8, 6] are not feasible in MANETs.

We can classify the address allocation protocols into three categories:

1. Stateful protocols
2. Stateless protocols
3. Hybrid protocols

Stateless approaches use conflict detection mechanism for allocating addresses to new nodes. A new node randomly chooses an address from a prescribed address block and then verifies

its uniqueness by either querying that address to receive a reply or by performing duplicate address detection. No address allocation tables are maintained. An unconfigured node self-assigns an address randomly and verifies its uniqueness with duplicate address detection (DAD).

According to [29], recent address allocation protocols can be divided into stateless and stateful approaches. In a stateless approach [3], a node chooses an IP address and performs DAD to check its uniqueness. Another stateless approach is Weak DAD [18], it allows two nodes to have the same IP address in the MANET but misrouting is prevented by associating a unique key with each node. This key can be a randomly generated large number, or the MAC address of a node. In stateful allocation, nodes need to synchronize with each other in order to maintain the state of allocated addresses.

In stateful approaches, state is maintained of both used and free addresses. It assumes the existence of one or more central authorities, and therefore, central and/or distributed allocation tables are used. If distributed tables are used, then the state of all the tables is synchronized periodically. Free addresses are the potential addresses to be assigned to new nodes upon arrival, and used addresses are previously assigned addresses. A method to calculate address range or address blocks is required.

Stateful approaches can be further divided into leader based-approaches, distributed common address space approaches and distributed disjoint address space approaches. In leader-based approaches [11] the workload on a leader is too heavy and a node may not be always reachable in the MANET. In MANETconf [22], address assignment is based on a distributed mutual exclusion algorithm that treats IP addresses as a shared resource. As in MANETconf, the dynamic address configuration protocol [3] also requires nodes to perform DAD during address assignment, which increases the latency of nodes joining the MANET. Complete synchronization is required between all nodes to avoid duplicate addresses.

A central allocation table contains assigned addresses, their corresponding MAC addresses

and their lifetimes. The central authority periodically floods verify packets to the network. Already-configured nodes reply with confirmation messages to refresh their lifetimes in the allocation table. If a reply from an address is not received, a unicast message is sent to that address. If still no reply is received, the address is deallocated from the allocation table and released. New nodes request an address by replying with an address request message. Dynamic Host Configuration Protocol (DHCP)[8] is not feasible because a server may not be permanently reachable by all nodes. In DHCP, there is also a single point of failure.

In distributed common allocation table approaches every node is allowed to select an address for an unconfigured node. Since the address space is shared by all the nodes, a tight synchronization procedure is required to avoid duplicate address assignment. Consider what happens when a network is partitioned and then the partitions merge back together later. In the individual partitions, nodes are assigned unique addresses, but when the partitions merge there are duplicate addresses. It is almost impossible to maintain a correct state in all the nodes of the MANET. Nodes may move away or switch off from ad-hoc mode rapidly, thus causing inconsistencies.

In multiple disjoint allocation table approaches, the address space is split among all the network nodes. As no two nodes share the same address pool, the initiator does not need to ask other nodes for permission to assign a specific address. A new node also receives half of the address space of its assigning node. Note that the address space can soon be exhausted.

Multiple disjoint allocation table and common distributed allocation function approaches can be grouped into one category known as conflict free address approaches. Here, an address is assigned from a disjoint address space and thus no confirmation of its uniqueness is required. In Prophet address allocation there is a low probability that a number will not repeat in a certain sequence, but it is not guaranteed, and as a result it may lead to unknown address duplication. Prophet address allocation [13] uses a distributed function to allocate unique addresses. Each node uses a distributed allocation function, which provides a disjoint address space. Addresses do not need to be reclaimed because numbers eventually repeat. During network merging, the partition with smaller network identifier (NID) changes the

IP addresses of all its nodes, no matter if duplication occurs or not. Although Prophet has the benefits of lower communication overhead, nodes break all ongoing connections with the smaller NID. When two large networks merge, the impact of connection loss is significant. Also [13] does not explain what happens to the allocation function after the network merges.

In Buddy protocol [20], nodes maintain disjoint allocation tables to be able to autonomously assign addresses to new nodes. The address space is split among the nodes and each node has a disjoint set of addresses. If a node leaves the network abruptly, part of the address space is not available for future allocation. To solve this problem, a synchronization procedure requires all nodes to periodically flood their allocation table to the whole network. The solution of Tayal and Patnaiks [4] has several limitations. First, the scheme is reactive as the nodes are required to report before leaving. Whenever there are no addresses available, to recycle addresses of departing nodes, the scheme requires i) polling to determine if any nodes have left and ii) broadcasting to assign the available addresses to existing nodes. The above situation leads to long delays when a new node joins the network.

Dynamic Address Allocation Protocol (DAAP) [16, 2] is a stateful protocol where every node gets half of the address space of its neighboring node. This scheme does not provide solution for network partitions, mergers and packet losses.

Another approach [19] uses the leader to assign IP addresses to all the nodes in the MANET. The MAC address of the initiator node is taken as the identifier of the network. The problem is that there are instances of duplicated MAC addresses. Therefore, we can have instances where two separate networks are established by nodes having the same MAC address. In this case, we cannot identify the merger of two networks. Secondly, the initiator can switch off from the ad-hoc mode and switch on later, thus forming a new network with the same MAC identifier as before. This can also happen when an initiator moves away from MANET and forms a new network. Thus, we may have circumstances of multiple networks with the same unique identifier.

All the stateless approaches and common distributed allocation table approaches can be categorized as conflict avoidance approaches. Addresses are assigned from a pre-authorized or calculated address pool. Then permission is granted by all the nodes for the address allocation, or all the nodes in the MANET are informed of the allocated address to purge it from the pending address table entry.

In [30], a ring based approach is used. Numbers are repeated in a ring but it assumes that when a node leaves the network, it informs its neighbor before switching off from ad-hoc mode which will not always be the case. Also when two networks merge, we will have duplicate addresses amounting to the size of smaller network. There are other stateless approaches like [26, 27]. There is also a non-IP-based variable size addressing scheme DART [14], but in this thesis we will concentrate on the IP-based address allocation.

Chapter 3

Node Joining

Node joining is a major and fundamental part of the address allocation mechanism in mobile ad-hoc networks. When a node comes within the range of a MANET, one of the two scenarios needs to be considered:

- The node is just entering Ad-Hoc Mode and it is not configured with a network-layer IP address.
- The node was part of another MANET and hence is configured with an IP address in that network, but due to its mobility it was partitioned from its home network and is joining a new MANET.

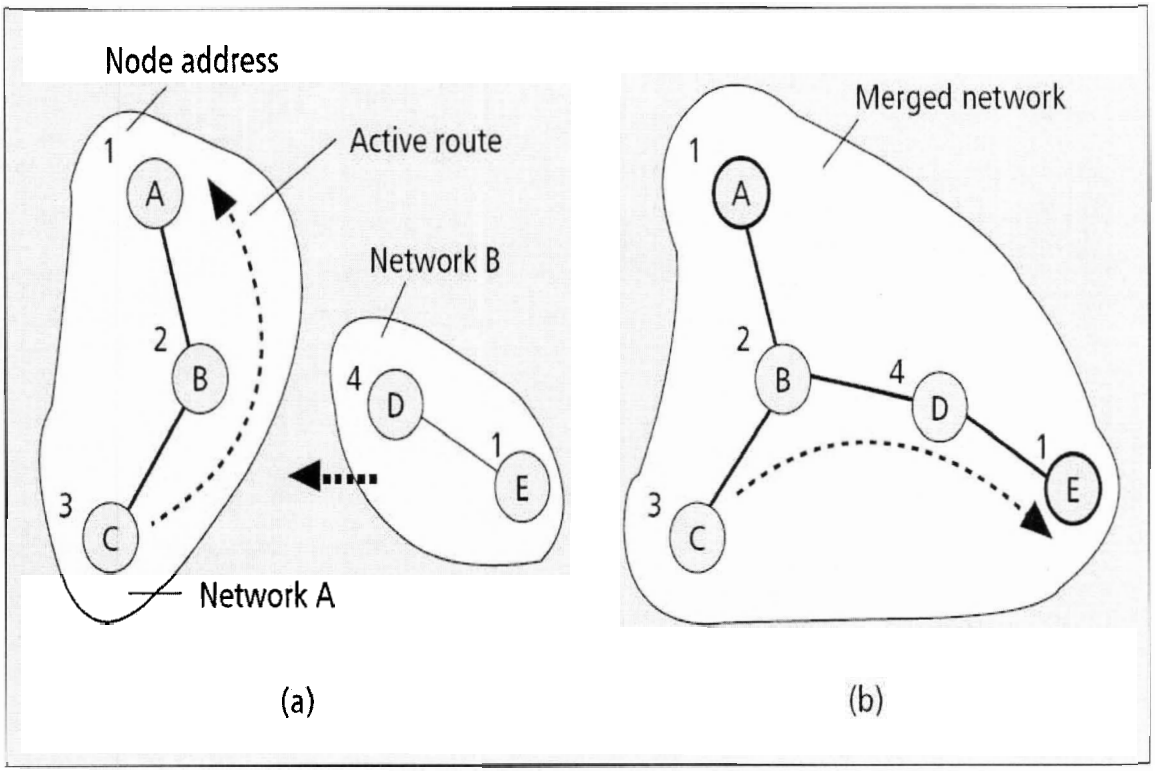


Figure 3.1: Independent Networks

3.1 Joining a configured MANET

When a node joins a MANET, it is provided with a unique (conflict free) address without requiring permission from the remaining nodes in the MANET. There is no need to run a DAD mechanism nor to maintain state information on already assigned addresses and available potential addresses. Techniques requiring DAD to run at the time of address assignment such as [3], induce delay, lead to flooding and thus incur extra communication overhead. For example, if the one hop round-trip time of a MANET containing 100 nodes

is 0.20 seconds, then the delay caused is 20 seconds if a positive response is received during first iteration of DAD from the entire network. In case of message loss, DAD needs to be run again and it would further take 20-25 seconds to complete. DAD is executed at least 3 times in [3] to get a valid conflict-free IP address. Therefore, in our example, it will take approximately 60 seconds (1 minute).

If an address conflict is found during third phase of DAD, the whole process is repeated again. Therefore, address assignment delay is doubled and it will take approximately 2 minutes if DAD is completed successfully during this iteration. Note that delay or latency caused due to DAD is directly related to the number of nodes in the MANET and it increases with each duplicate address found. Clearly, running DAD is very expensive, especially in MANETs where nodes have limited bandwidth and power.

In the problem above, a solution is required which guarantees conflict-free IP addresses by making no (or minimum) use of DAD. This way we can quickly configure a node with a valid address which is ready to be used.

3.2 Single Node

A new (joining) node upon its arrival in the MANET does not hold any address and therefore, needs to be configured with an initial IP address. When a node x switches on to ad-hoc mode it will send a Discover message. Note that this message needs to be only one hop broadcast, and so we can fix its Time-to-live (TTL) value to be 1, assuring it is broadcast only one hop. A neighboring node with the capability of assigning a valid address will respond with an IP address in the address reply message. Now x would get a conflict-free IP address without querying other nodes in the MANET or running a Duplicate Address Detection mechanism to ensure its uniqueness. This saves lot of extra communication overhead and delay. We should avoid extra overhead due to depletion of battery power in mobile nodes, mobility and low bandwidth.

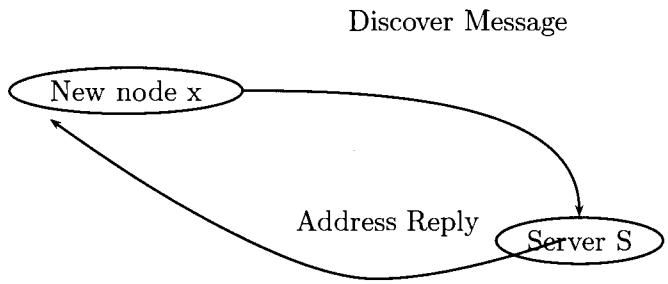


Figure 3.2: Node Joining

Here is how the protocol works:

1. Requester sends a Discover message
2. Server/Initiator responds with an IP address

At this point some special handling of the address assignment is necessary. The initiator provided the requester with a valid IP address. On the other hand, we are not sure whether the requester has yet configured itself with the address it received. Due to low battery power and high mobility, there is a high probability that the requester either switched off or moved away from the MANET transmission range. As a result, the requester did not configure itself with the address sent in the address reply message. We will later discuss in detail how to handle this scenario.

3.2.1 Forwarding Address Request

When a new node x sends one hop broadcast of Discover message, a neighboring node upon receiving the message will reply with a valid address. There might be a situation where a

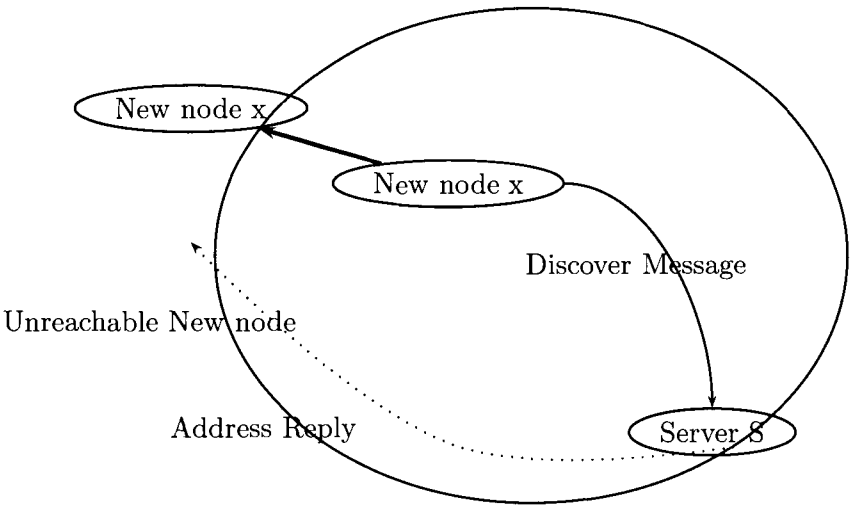


Figure 3.3: Address Reply Unreachable

neighboring node is either not capable of assigning a new address or it ran out of its allotted address space. In either case an already configured node will forward the Discover message to a known server address, otherwise it will increase the TTL field of the Discover message by one. This way the packet will be broadcast to the one-hop neighbors of the configured node. An assigning node, upon receiving the Discover message will send a unicast reply including a valid IP address to the originator. That is the node requesting a new address. The process is shown in Figure 3.4 below.

If the requester will not confirm the use of the address sent in the address reply message, there is no way that a server will know yet if the address is in use or not.

3.3 Concurrent Requests

If two or more joining nodes concurrently send Discover messages to the same neighbor or server at the same time, their requests would be indistinguishable. Assigning nodes will

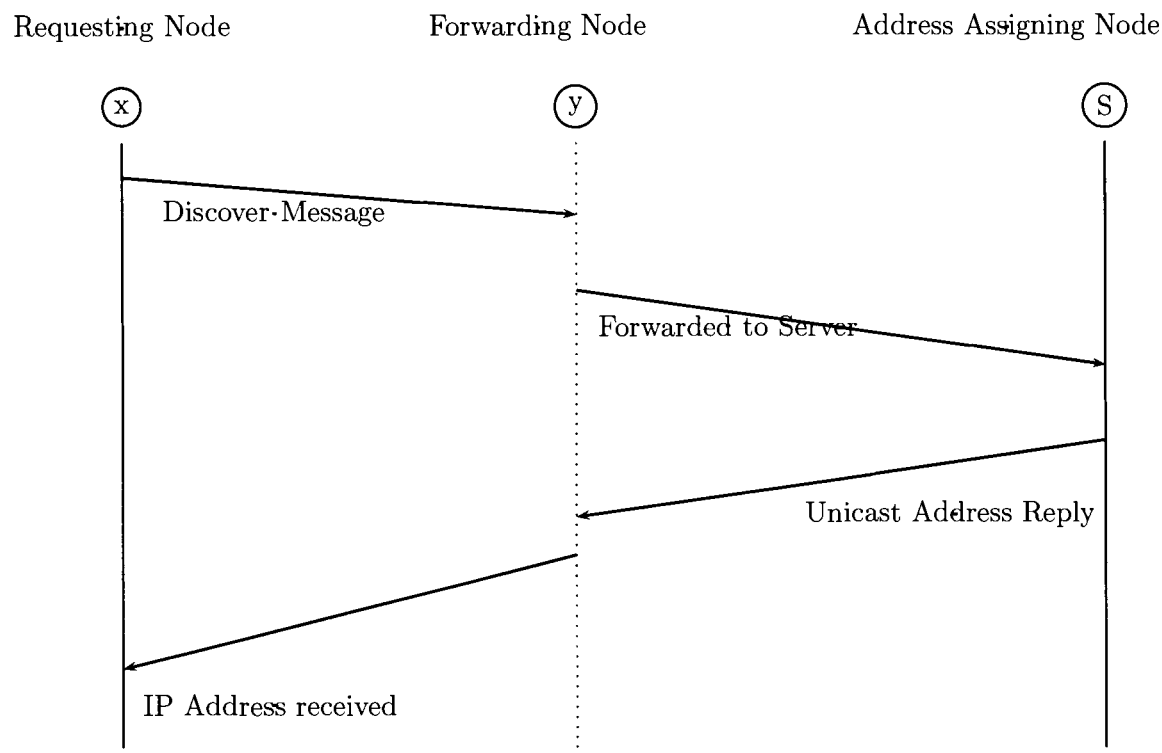


Figure 3.4: Forwarding Request

have no means of distinguishing between the two joining nodes, and as a result both new nodes could select the same address. Almost all of the address allocation protocols proposed for MANETs does not cover this scenario. Consider two nodes, x and y, that simultaneously join a MANET and each send Discover messages to the server/assigning node. If the server S sends address 1 to node x, one of the following scenarios will occur:

a. The Address Reply (AddrReply) message never reaches x, but it reaches y, and therefore y configures itself with the address in the message. This behavior is not tolerable. Suppose next time x again sends the Discover message but another new node z gets the address. If this process happens repeatedly, we might actually have a node which

will starve, because it waits too long to acquire an address. Therefore, the address allocation protocol must be fair and no node should starve.

b. Both x and y configure themselves with the same address. This behavior is not tolerable because it leads to address duplication. We will see the consequences of address duplication in detail later.

c. The AddrReply message never reaches x nor y. This behavior is not tolerable because this causes address leak, and thus the address space will get depleted.

d. Only x gets the AddrReply message and configures itself with the valid IP address. This is the ideal behavior as the message never reaches y.

We would not allow situation in (b) to occur at all, because it leads to address duplication and as a result a DAD mechanism would need to be run. We know that DAD is an expensive operation and it requires broadcasting. Once a duplicate address is detected one of the involved nodes has to change its IP address and hence all ongoing connections with this node are broken as a result of the address change. We will see later how to prevent this from happening.

If situation in (c) occurred, it would cause an address leak. To prevent such leaks, we will introduce a transaction identity number XactID in the address request, reply and configure messages. XactID has two fields: a current transaction ID and a previous transaction ID. A node randomly generates a XactID number at the time of sending the initial address request. This would be entered in the current XactID field and the previous XactID value would be set to null. If an address request is sent again, the requesting node will increment its current XactID field. This time the previous XactID field will be set to the value of the current XactID sent in the previous Discover message. This way the assigning node keeps track of new Discover messages sent by the same node, and the assigning node will assume that the address reply message did not reach the requester, and it would again send a unicast AddrReply message to the requester.

For case (a), suppose y gets IP address 2 that was originally destined for x. Node x will time-out and resend the Discover message to S. In this case, S will resend the address reply containing IP address 2 back to x. At this time, we will have duplicate addresses. We will see later how we can prevent this from occurring.

We extend our node joining technique to ensure that an assigning node not only provides a conflict-free address, but also has the capability to distinguish between the concurrent requests of two or more joining nodes in the same region. To avoid address leaks and duplication, we should always ask a joining node to confirm its IP address after configuration. This ensures successful address assignment and avoids duplication.

3.3.1 MAC Address Solution

One solution is to append the 48-bit MAC address of each joining node in the Discover message to distinguish between the two nodes. This way the server would be able to distinguish between two concurrent requests. The server will know exactly which node has accepted the address and will send an acknowledgement (ACK) message to that node. If both x and y accept the same address simultaneously, the server will receive two configure messages. The server would see two different MAC addresses and recognize that two nodes have configured themselves with the same IP address. To avoid address duplication, the server will send an ACK to the node for which the address was initially destined and negative acknowledgement (NACK) to the other node. A node upon receiving a NACK will have to apply for a new address. To avoid delays associated with address assignment after a NACK message has been sent to a node, we piggyback a new address along with a NACK message. Thus, a node gets a valid IP address to configure itself at the time of receiving the NACK.

Here is how the protocol works:

a. Requester sends a Discover message

b. Server/Initiator responds with an IP address

c. Requester will confirm the use of address with a Configure message

d.
- NACK message sent, go back to step c
- Initiator will respond with an ACK message confirming the configured address

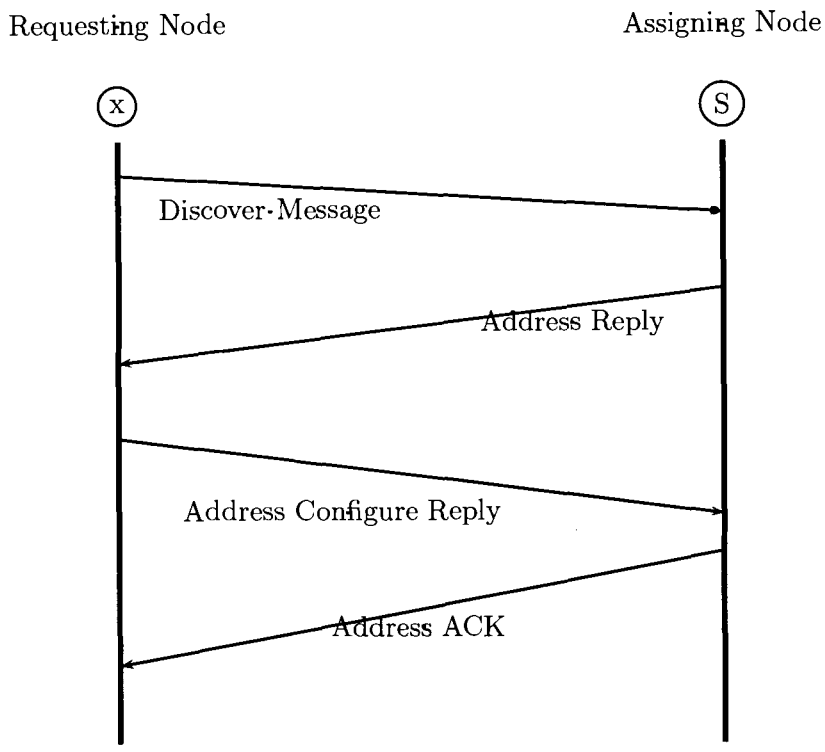


Figure 3.5: Address Acquiring Process

There are many known circumstances where two nodes may have the same MAC address [22] [1]. Due to MAC address duplication, two requests would be indistinguishable. Therefore, to avoid this situation, we let a node choose a temporary address from a Quadratic Non Residue (QNR) modulo N, and a transaction ID. This will ensure that a temporary

address is not a Quadratic Residue (QR). Since it is not part of our address allocation function because it is not a Quadratic Residue. There are $1/2\ \phi(n)$ available QNR modulo N and they are easily computable. We will explain later how to quickly get a QNR.

This is better than just using 0.0.0.0 address to send a Discover message as in a DHCP [8] server to get an IP address. Multiple hosts joining at the same time will have the same initial requesting address, and this may lead to many duplicate addresses and hence a need to run DAD. Another option could be to choose a random number as an IP address but then this random number may be a network layer address of an already configured node in the MANET.

3.3.2 Modified Protocol

A joining node must know N in order to compute a QNR. This process is very quick. Here is the modified version of the protocol.

Requester joins the MANET, and listens for a periodic routing hello message. We will piggy back the value of N in the periodic routing *hello messages.* Requester computes a QNR and configures itself temporarily with this address. At this point, requester will not participate in any communication other than receiving and sending address request and response messages.

a. It then sends a Discover message.

b. If the neighbor is a Server, it replies with an IP address, otherwise it forwards the request all the way to the initiator node.

c. Initiator replies with a valid address.

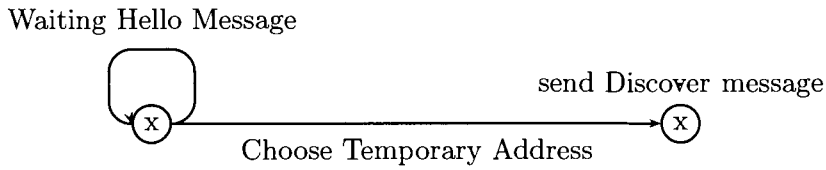


Figure 3.6: Temporary Address

d. Requestor configures itself with the IP address and sends a configure message to the initiator.

e. Initiator will reply with an ACK

3.4 Multiple Address Assigning Nodes

In this case, a node sends a *Discover* message and receives two or more IP addresses through multiple address reply messages. A simple scenario can be where one address reply is received from each neighboring node. Hereafter, it is the choice of the requester to accept any one of these addresses. A node, after selecting an IP address (say 2), will send a configure message to the corresponding assigner.

3.4.1 Free Addresses

We can have the requester send a unicast message to the assigner of its configured address. This way, all remaining assigners will know that the provided address has not been chosen by the requester. Therefore, all nodes other than the assigner of 2 will know that the sent IP address is still free and they can assign this address to a new node.

Field	Description
Address	Allocated IP address sent in Address Reply to new node
Temp Address	Node's Temporary Address that is a QNR
MAC Address	MAC Address of the Requesting Node
Xact ID	Transaction Id generated by the requester
Timer	Timer count wait to receive Address Configure Message
Status	To keep record of configured message and ACK sent

Table 3.1: Pending Address Data Structure

If we avoid sending a unicast Address Chosen message to all servers, then we need to send a one or two hop broadcast message of the selected address. On the other hand, each server can put the sent address in the pending address table 3.1 and associate a timer with each entry. Each address in the pending table has a timeout period. A server will wait for the timeout period to expire 3 times before assigning this address to another node. This ensures that duplicate address assignment is prevented.

3.4.2 Lost Address Reply

If an AddrReply message is lost, the requester re-sends the Discover message by incrementing the current Transaction Id (XactID) field to a new value and setting the previous XactID field equal to the current XactID value of the last address request message sent by the same node. The server upon receiving this message will check the temporary address, MAC address and XactID of the node and will locate the last entry by keeping a pointer from the most recent XactID to the previous XactID. It resends the AddrReply message and adds the new message to the table by updating the pointer to point to the current XactID of the Discover message. Tracking of address requests received from a node is kept by keeping a pointer from the current XactID to the previous XactID. This process is repeated all the way to the very first Discover message received by that node.

3.4.3 Assigner Partitioned before Receiving Configure Message

Address assigning nodes like the requesting nodes are also mobile. The assigning nodes may move away from the MANET during the address assignment process. Consider what happens if the assigner becomes partitioned from the network and, as a result, does not receive the address configure message. Nodes are unaware of getting partitioned from the MANET. If the address is not taken by any node before the network is partitioned, it will not cause any problem. But, if the address is taken by a node just before the network becomes partitioned, we will have duplicate addresses if the partitions merge back together later. The assigner will assume this address to be free and will reassign it to a new node in its own partition.

Assigner End

This will not happen if we let the assigner send an address ACK message to the configured node. Upon receiving the configure message the assigner will remove the address from the pending table entry and send an address ACK message back to the requester. The requester will not take part in communication until it receives positive address ACK from the assigner.

If the assigner gets partitioned without receiving the AddrConfig message, it will keep the address in the pending table entry, and would timeout 3 times before assigning this address again to a new node. Note that the server timeout period should be long enough and it should periodically check the pending table entry. If the assigner quickly merges back with the network, it may receive the configure message and send a unicast ACK to the requester.

Requester End

Now, we will see what would happen at the requester side. If the new node configures the address and sends an AddrConfig message it will wait for an ACK message before taking part in any communication. If no positive ACK is received, it will timeout and resend the configure message, repeating the process for 3 times to account for packet loss. If after three retries no ACK is received the node can imagine that the assigner became partitioned or switched off from ad hoc mode. Hereafter, the node can either hold the address or choose another address. To avoid facing the previous problem of network partitions being merged together later, node x will choose another address and send an address configure message.

Protocol

Instead of the joining node sending a one hop broadcast of *Discover* message, it will listen for the periodic hello messages sent by already configured nodes. Each hello message will contain the senders IP address, NID and N.

New nodes upon joining the MANET will compute a QNR modulo N . It will select that value as its temporary address, generate a random XactID and send these values along with its MAC address in the unicast *Discover* message.

The assigner will then send a unicast reply back to node x's temporary address having the MAC address equal to node x MAC address. After successfully receiving the address reply message, the requester will then send an address configure message to the server. Upon receiving this message, the corresponding entry from the server address pending table is freed, and an address acknowledgment (ACK) is sent to the requester.

Note that there is a trade-off between providing multiple addresses and a single

address. When a node is only offered a single address, but no address ACK message is received, it will cause a delay in getting another valid address.

On the contrary, when a node is offered multiple addresses, the failure of the assigner to provide a positive address ACK would not lead to additional delay in acquiring a valid address. The requester can always choose another address from a list of offered addresses and it will send an AddrConfig message to the corresponding assigner without any further delay.

3.5 Mobility Scenarios of Assigner and Requester

3.5.1 Assigner Mobility

After Sending Address Reply

If S moved away from the MANET after sending an address but the message is lost or not received by x, then:

a. For x it is the same as no reply received from S.

b. But S has devoted one address to x.

c. Node x will query another server R with a Discover message and will repeat the address confirmation process.

Suppose S became partitioned after offering an address(say 2) to x. S will continue assigning new addresses in its own partition. It may or may not merge back with the original network later. For consistency it is important that S does not remove an allocated address from its pending table entry until an address configure (AddrConfig) message is received from the requester after address allocation.

If S either moved away or switched off without sending an AddrReply message, x will not know which one of the two happened. Also, it is unrealistic to expect that a node will inform the MANET about its departure. Therefore, x will query another node with a Discover message.

Partitions before Acknowledging Configure Message

If S moved away after x received an IP address but before sending ACK of the AddrConfig message:

The AddrConfig message may or may not have reached the assigner. If it did not reach it, then the assigner will assign the address to a new node after partitioning. This partition may merge back later and cause address duplication.

If the AddrConfig message reached S and it unaware of being partitioned from the original network send an ACK to the requester, S will not know if the ACK reached the requester or not. The requester will treat it as no ACK received, and will repeat the process for the second time before sending a Discover message to another server node.

3.5.2 Requester Mobility

Partitions before Receiving Address Reply

The requester will never get a valid address. It may join another MANET upon which it will acquire a new address.

If it merges back into the original partition quickly it may send the Discover message again to node S. Node S will know that x has incremented its XactID, and therefore it did not receive an old Discover message. S will resend the same address again as indicated by the pending table entry. The remaining address configuration process continues until x is configured.

If node x merges after a long time away, it might have became an initiator itself, and thus started its own network. It may be the only node in its network or it may have several other nodes in its network. We will treat this scenario same as network merging. We will discuss network merging and its impact in detail later.

Partitions after Receiving Address Reply

The requester will get a valid IP address to configure itself if the partition occurs after it receives the AddrReply message. Then requester will be unaware of getting partitioned and it will send an AddrConfig message to S which clearly never reaches the network. It might resend the address request query 3 times. If there still is no response/ACK is received, it starts listening for the hello message.

If in this time the node returns to its original network or joins another MANET, it will hear a hello message and can get a valid address by performing the address acquisition process. Otherwise, it will know that it is the only node in its MANET and then it computes the seeds, address blocks and assigns itself an IP address. If the node becomes partitioned after receiving the ACK, it is just a simple network partition. We will discuss network partitioning in detail later.

3.6 Initiator/Prophet Node in MANET

If an initial node i does not hear a hello message it sends a Discover message and turns on the timer to receive a reply. If it timeout after a suitably chosen interval, it will repeat the same process for about two more times to accommodate packet loss and/or hidden terminal problems. If still no response is received, it timeout for the third time and then calculates the MANET address space, future addresses, address range and network ID. Meanwhile, if another node j, which is out of the transmission range of node i, timeout for the third time after sending Discover message, then it will also compute the address range, future addresses and NID. Now there exist two independent MANETs which are out of the transmission range of each other. After few seconds both nodes i and j may come within radio frequency range of each other. This scenario can be treated as a network merger according to the previous literature or we can allow two different networks to assign unique network addresses based on the requesting nodes distance from the assigning node.

In previous approaches, one node has to give up its address and obtain an address from another node or randomly assign itself an IP address and detect its uniqueness by performing DAD. In stateless approaches this does not cause a significant delay because an initial node does not perform the load of acting like a prophet node and doing all the future address calculations. In stateful approaches, an initiator giving up its address would have incurred the unnecessary cost of calculating the address blocks.

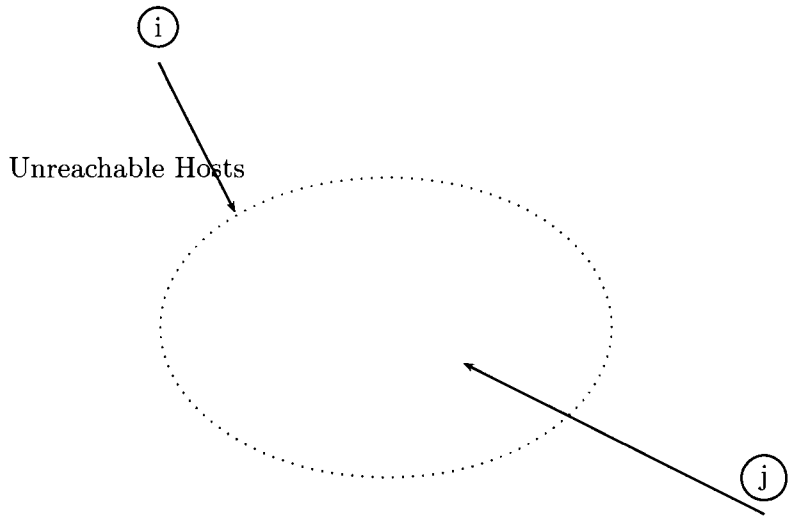


Figure 3.7: Outside Range

Consider the case where two MANETs contain hundred of nodes. In both cases, each node in the two MANETs must perform DAD after the network merger and before assigning new addresses in the merged MANET. If we have 500 nodes in the merged network with a one-hop round trip time of 0.10 seconds, then DAD takes 500 seconds (8 minutes 20 seconds) to run. But we know that to account for lost messages and unreachable hosts we have to perform it three times, thus costing us approximately 1500 seconds. One can easily see that DAD is an extremely expensive operation and should be avoided as much as possible.

Therefore, we present a new approach. We let two different network addresses exist in the MANET and therefore avoid using DAD as much as possible. Due to limited battery power and low bandwidth, we make minimum use of flooding and broadcasting. In our approach, we allow two different initiators to assign unique network addresses as long as they are not in conflict with each other. This is how it works; two initiators will choose different values for p, q, and N and hence their addresses and address blocks will be different than others. Note that even if the two initiators choose the same p, q and N (which is very

unlikely) they will differ in the network ID part of their IP addresses and hence they can assign conflict-free addresses. As long as there is no conflict, we can benefit from assigning addresses and still be highly scalable. We let two different network addresses exist in the bigger MANET and are still able to assign unique, conflict-free addresses without any routing or node identifying problem. This technique is similar to wired networks where a router has two different network addresses attached to it on different interfaces.

Now consider what can happen if we let both initiator nodes exist and hold their respective addresses.

a) They choose same Network Prefix/ ID

b) They choose different Network Prefix/ID

c) They choose same $N = p * q$

d) They choose different p and q, resulting in a different N

If (b) and (d) happens, there will be no duplication in IP addresses.

If (a) and (c) happens, one of the network address generators (initiators) has to drop its address and future addresses and acquire an IP address from another initiator.

If (b) and (c) happens, the IP addresses will still be distinguishable because of the different network bits of each IP address.

If (a) and (d) happens, the network ID would be same, but the Quadratic Residue

mod N would be different in two cases, but there still would be some duplication. Therefore, we calculate all the QRs of both initiator nodes. If duplication exceeds 30 percent then an initiator has to drop its address pool. Otherwise, duplicated addresses are purged out from the future address space by declaring them dirty and maintaining a dirty table. In the former case an initiator has to drop its address and acquire an IP address from another node.

Chapter 4

Address Generation

Our address allocation protocol makes use of Quadratic Residue (QR) [9] to assign addresses to nodes in the MANET. In this chapter, we will first describe what QR is and then we will explain how our addressing function will make use of QR to assign conflict-free unique addresses to new nodes in the MANET.

4.1 Quadratic Residue

We propose an address allocation mechanism using the Quadratic Residue (QR). Let p be an odd prime and $a \in Z_p^*$ (where Z_p^* is a set of invertible elements in p such that their inverse exists) then a is Quadratic Residue (QR) square if,

$$x^2 \equiv a \ (mod\ p) \tag{4.1}$$

has a solution and a Quadratic Non Residue (QNR) otherwise.

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

Table 4.1: Quadratic Residue Modulo 11

For example, the Quadratic Residues in Table 4.1 are

$$QR = 1, 4, 9, 5, 3$$

and the Quadratic non Residues

$$QNR = 2, 6, 7, 8, 10$$

If we calculate QR from 1 it gives back 1 repeatedly. If we start at any other point, for example at 4 or 9, we get the next QR as shown in the Figure 4.1.

The number of invertible elements, such that their inverse exist in modulo p are $p-1$. The total number of Quadratic Residues in p is equal to $\frac{(p-1)}{2}$. Half of the numbers are QRs and other half are QNRs. Euler's theorem states that a is a QR if and only if:

$$a \in QR \ mod \ p \iff a^{\frac{(p-1)}{2}} \equiv 1 \ (mod\, p) \tag{4.2}$$

Euler's Theorem yields a polynomial time algorithm for QR by using the square and multiply algorithm [9] for exponentiation modulo p. This gives us a cost of $O(log\ p)^3$. Another efficient way to find a QR is through Jacobi symbols, see [9]. It says that a is a

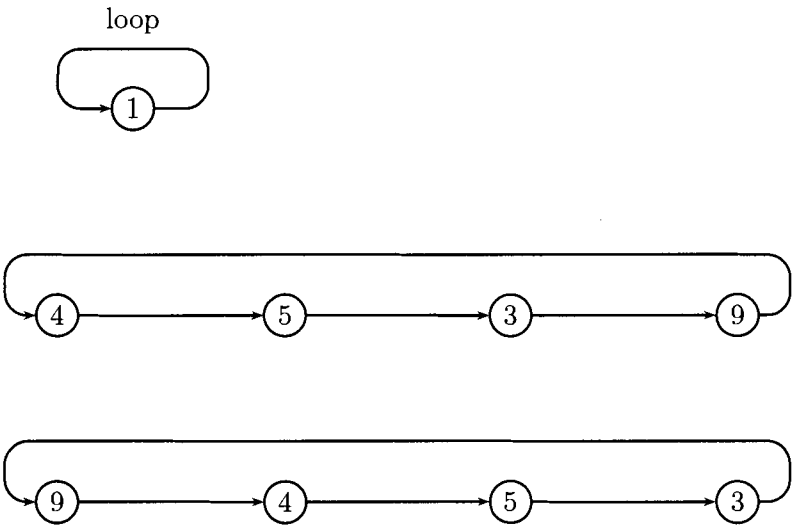


Figure 4.1: QR Cycle

QR if the Jacobi symbol of $a \ mod \ p \equiv 1$ and QNR if it is -1. The Jacobi symbol can be computed in polynomial time $O(log \ p)^2$.

$$a \in QR \iff (\frac{a}{p}) = 1 \tag{4.3}$$

and a Quadratic Non Residue if the Jacobi symbol is -1

$$a \in QNR \iff (\frac{a}{p}) = -1 \tag{4.4}$$

Now, we try to take the QR of a composite integer which is a product of two primes, where each prime p has $\frac{p-1}{2}$ QR in it. Let two primes be p and q, $p = 7$ and $q = 5$ then $N = 35$.

There are 3 Quadratic Residues in 7 = 1, 4, and 2

There are 2 Quadratic Residues in 5 = 1, and 4

a	QR mod p	QR mod q
1	1	1
4	4	4
11	4	1
16	2	1
29	1	4
9	2	4

Table 4.2: QR Modulo N

In the product of p and q the number of QRs = 3 * 2 =6

The set of QRs in N is $\{1, 4, 11, 16, 29, 9\}$. This set does not contain 2 because 2 is a QR mod 7, but it is not a QR mod q.

Note that only 1 and 4 are QR in both p and q, and they are also QR in N. In addition to that 11,16,29 and 9 are QR in N because both of them are QR mod 7 as well as QR mod 5. In the above Table 4.2 all values of a represent the QRs present in mod N. Then we see that each value is also a QR mod p and QR mod q. Now let's find out what type of cycles, if any, are present in QR mod N.

The cycles in N=35 are not distinct as in the earlier case when we considered the example of $p = 11$. In figure 4.2, the square of 16 mod N is 11 and the square of 9 mod N is also 11. This means two different paths have a vertex in common and one of those is acyclic. QR 4, 9, and 29 do not lie on any cycle. The cycle we get is one of length 1 with QR 1 as always, another cycle of length 2 involving QR 16 and 11. We will later discuss how we can avoid this case. What we do not want is to have two different numbers when applied to our addressing function $S_{i+1} = S_i{}^2 \ mod \ N$ generating the same next number. This would mean that there is a chance of duplicate addresses and inconsistent states. If each cycle is considered as a separate address block then we might get duplicated addresses. What we want is to generate address blocks (cycles) which are totally independent of each other and disjoint(i.e. such that there are no two distinct QRs that, when applied to addressing

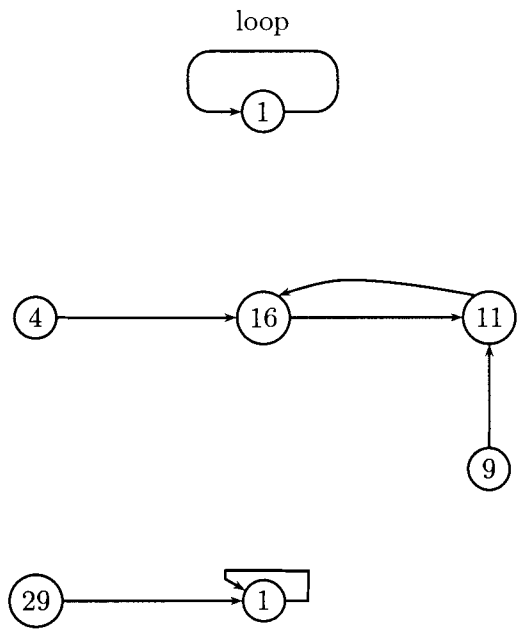


Figure 4.2: QR Cycle in N

function, generate the same next QR).

4.2 Address Space

Suppose we have two odd primes p and q , and $N = pq$. The number of invertible elements in N called $\phi(N)$ is the product of $(p-1) \times (q-1)$. The number of QRs modulo N is:

$$QR \in Z_N = \frac{1}{4} \times (\phi(N)) \tag{4.5}$$

and the number of pseudo squares (PS) is also

$$PS \in Z_N = \frac{1}{4} \times (\phi(N)) \tag{4.6}$$

Pseudo squares are explained later. The number of QNR in mod N is:

$$QNR \in Z_N = \frac{1}{2} \times (\phi(N)) \tag{4.7}$$

where:

$$\phi(N) = (p-1) \times (q-1)$$

We want the QR mod N to occur in distinct cycles. A QR lying in one cycle would not be present in any other cycle modulo N. The initial QR of a cycle acts as a seed S_0 to generate a sequence of numbers until a seed repeats again. The period or interval is the gap between the first and second occurrence of a same number in a sequence. We are interested in long intervals. The interval before a QR repeats again is a length of the cycle. In Figure 4.3, S_0 is a seed (i.e. that it is the first QR which generates a sequence of QRs). This sequence repeats in cycles and we denote the last QR before an interval ends by S_π. Now our interval will start at the first QR S_0 and end at S_π. The occurrence of different numbers before an interval ends will be called the length of a cycle.

After S_π, the seed repeats again and the remaining sequence is repeated exactly in the same order as it occurred before. This forms a cycle which repeats each time in exactly the same manner. Given a seed S_0 which is a quadratic residue modulo N, the sequence S_i of quadratic residues can be found by successive squaring mod N. This gives us the cost of $O(log\ N)^2$.

$$S_{i+1} = (S_i \times S_i) \ mod \ N \tag{4.8}$$

The address allocation function f is dependent on the seed S_0. The function takes as input the seed and output is $f(s_0) \in Z_p$, the Quadratic Residue next in the sequence.

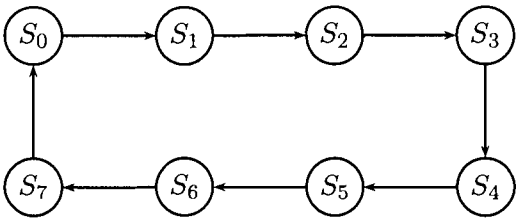


Figure 4.3: Quadratic Residue Sequence

For explanation purpose we will give a small example. Let $p = 23$ and $q = 7$, and thus $N = p \times q$, $\phi(n) = (p-1) \times (q-1) = 132$. The total number of QR mod N is 33 and there are five cycles in total as shown in Table 4.4 below, one cycle of length one, one cycle of length two and three cycles of length ten. Note that if we use large primes, we will get cycles of long lengths.

All the sequences in Table 4.4 are generated from a set of seeds and each seed repeats again at the end. This indicates that the interval or sequence is finished. These sequences occur in cycles, for example Figure 4.4 shows a cycle of length ten with seed $S_0 = 2$. QRs lying in these long cycles are used to allocate addresses in our protocol. Note that each address, after reaching the maximum cycle length, would repeat in an identical manner as

1, 2, 4, 8, 9, 16, 25, 29, 32, 36, 39, 50, 58, 64, 71, 72, 78, 81, 85, 93, 95, 100, 116, 121, 123, 127,128, 141, 142,144,151, 156 There are 33 quadratic residues

Table 4.3: Quadratic Residue Modulo 161

Seed S_0	*Sequence* S_i
1	1
93	116, 93
2	4, 16, 95, 9, 81, 121, 151, 100, 18, 2
8	64, 71, 50, 85, 141, 78, 127, 29, 36, 8
25	142, 39, 72, 32, 58, 144, 128, 123, 156, 25

Table 4.4: Seed Generated Sequence

before. The logic behind this is when a number(address) repeats in a cycle, the node which was previously assigned this address would have left the MANET. The cycle of length ten is actually a small cycle used in our example for explanation purposes. If primes of 12-bit or 13-bit are taken to compute N, we will get long cycles of length more than one million. As a result, we can successfully re-assign this conflict free address to a new node.

In this way the address space is reclaimed automatically and each node can be assigned a unique address. The range or length of a cycle corresponds to a state. If we

S_0	Cycle Length (Range)
1	1
93	2
2	10
8	10
25	10

Table 4.5: Range of Cycles

have ten different cycle lengths, we will have ten different states. This means that in Table 4.5 there are three states as there are three different cycles each of length ten. Note that each of these cycles belong to a unique seed. The intersection of any two different cycles is always a null set.

An example is shown in Figure 4.4 below, where the initial QR 2 acts as a Seed and the next QR is obtained by squaring a current QR and reducing it modulo N.

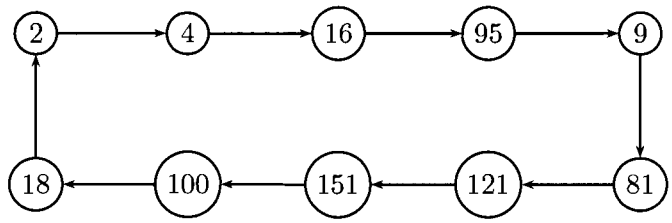


Figure 4.4: Seed Generated Quadratic Residue cycle

Chapter 5

Distributed Protocol

When a node switches on to ad-hoc mode it starts the timer and waits to listen for the periodic routing *hello message*. If it does not hear any *hello messages*, it times out and sends a Discover message. If it does not receive a reply it retries for a suitable number of times, K. It then assumes that it is the first node in the network and chooses two primes p and q, which are congruent to 3 mod 4. Primes of this form are chosen because their square root is easy to calculate. Then it computes $N = p \times q$ and $\phi(n) = (p-1) \times (q-1)$. The Prophet/Initiator node calculates, the total number of QR, the number of distinct cycles, length of each long cycle (address block). If length of the long cycle is small, the initiator can choose two other primes, compute N and length of the address blocks. This process is repeated until a node gets long cycles. Later we will describe a method of quickly and efficiently computing the long cycle length. The initiator (first node) configures itself with an IP address, keeps the seed value S_0 of each distinct long cycle and its corresponding range.

When a new node joins the network, it sends a Discover message. An already configured node provides the address as described in the node joining process. Along with

an IP address a new node also receives a set of seeds S_0, and the corresponding range of each seed. Therefore, a new node gets an IP address, state value(seed) and range. Now the new node, along with participating in communication, is also capable of assigning a unique conflict free IP address without taking permission from any other node in the MANET. Each state represents the sequence of addresses in that cycle. A newly joined node, after configuration, will calculate the next address by squaring the current quadratic residue and reducing it modulo N. Each cycle of QR is disjoint, therefore a node using a cycle x knows that a cycle y would not have any address in common with x. Therefore, the probability of duplicate address assignment is zero. This greatly reduces the delay associated with address assignment; where in other techniques a node needs to run DAD or has to take permission from remaining nodes in the MANET to perform address assignment. This not only decreases the latency and delay, but also reduces the communication overhead and saves bandwidth.

5.1 Address Reclamation

Now the questions of what happens after reaching maximum length of an address block, and how to keep track which nodes have left and which are still in the MANET arise. In other words, how to reclaim the address space in order to continue assigning unique addresses to new nodes upon arrival. QR repeating in cyclic form guarantees that there is no extra overhead spent for reclaiming addresses or address space. It is important that primes are chosen carefully such that the union of two different cycles is always a null set. Therefore, our address space will be disjoint and all the addresses in it would repeat automatically in the same sequence as before. However, we have to make sure initially that the interval of each address cycle is long enough so that when an address repeats, the node which was assigned this address before would have already left the MANET. This is the duty of the prophet (initiator) node to compute the address space and perform initial address calculations.

5.2 Maximum Address Size

After the maximum address range has been reached by a node, it has two options: either to start re-assigning the addresses as they repeat or increment its state value by choosing an unused seed. If the re-assignment of the first address will lead to duplication the node will increment its state value to another seed. To confirm the seed is not in use by another node in the MANET, it floods the network with a *NewSeed* message and waits for the reply from all other nodes. If it receives one negative acknowledgement (NACK), it chooses another seed and repeats the process for K times. This is done to account for delay or message loss, in order to confirm that it will not use a seed already in use. If it does not receive any NACK after K retries it assumes that the seed is free and will use it in the future to assign addresses to new nodes.

5.3 Network Partitioning

If a network becomes partitioned, nodes in each partition can still continue assigning unique address. As each QR cycle is disjoint, no address duplication will occur. Should partitions merge back together later, address uniqueness is guaranteed. This way we handle network partitioning without incurring any extra communication overhead.

Consider two independent MANET as in Figure 5.1. All of the nodes in each MANET have unique IP addresses. Now we will see what happens when two independent networks merge. Network ID (NID) is piggy-backed in *"hello messages"*. A network merger is detected when a node hears a *"hello messages"* with a different NID. In Figure 5.1 there are two independent MANET. Node D is communicating without any problem with node A having IP address a. Note that in the second MANET there is node K which also has an IP address a. As these nodes belong to different networks, there is no address

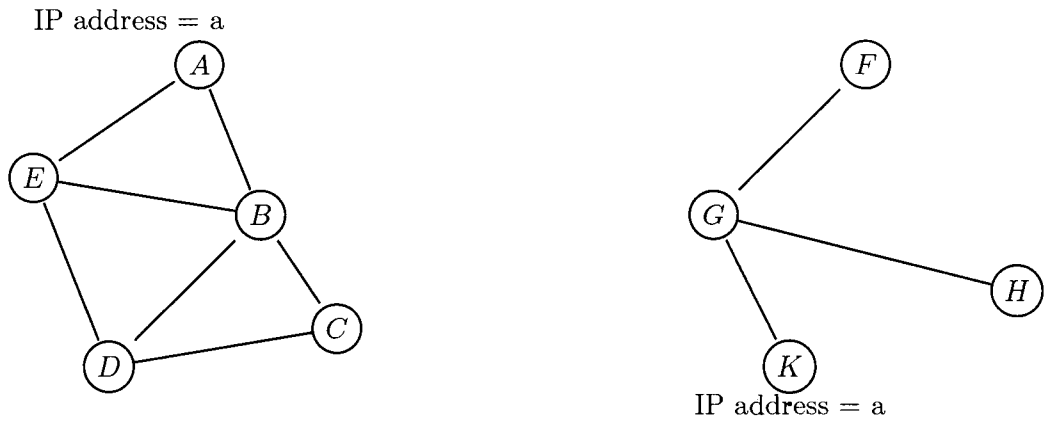


Figure 5.1: Independent MANET

duplication or communication problem. Let us consider what happens when these two independent networks come close to each other and get connected to form one big MANET. Misrouting can occur because of duplicate addresses as shown in Figure 5.2 below. A merger of these two networks causes packets from node D to be misrouted to node K instead of A. Node A and K have the same IP address and therefore we have duplicate addresses in the MANET. As a result packets which were meant to be routed to node A can be misrouted to node K. We have to solve this duplicate address problem and/or prevent misrouting after network merger.

5.4 Network Merger

We have two ways of handling a network merger. One makes use of DAD and other is not dependent on DAD.

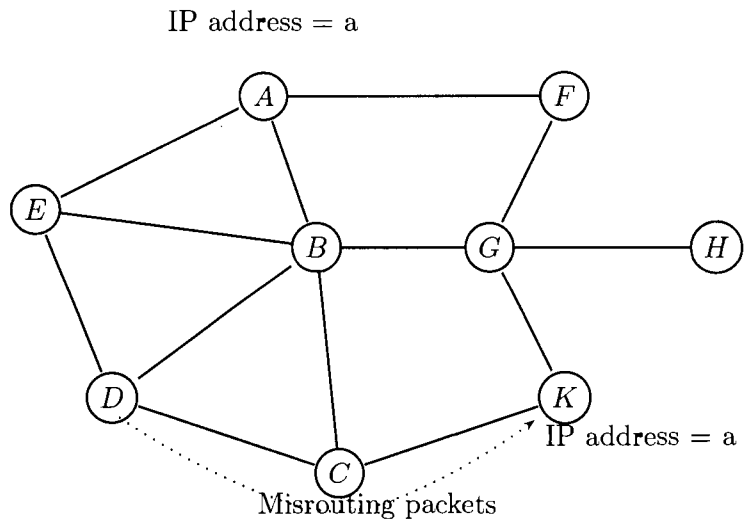


Figure 5.2: Network Merger

5.4.1 Independent of DAD

In the first solution there is no need to invoke an explicit procedure upon merging of the two independent MANETs. We can associate the respective $\phi(N)$ values of each MANET to be the key of that MANET. The idea is similar to Weak DAD [18], but we associate a single network key with all nodes of one network instead of using a separate key for each node.

The logic is to distinguish between the nodes having the same IP addresses and those belonging to two separate networks during network merging. We will show that this does not increase communication overhead and we can still prevent misrouting. When we choose two large primes, the $\phi(N)$ value for each network would be different than for the others

with a high probability. Two nodes can even have the same address but misrouting can be prevented by associating $\phi(N)$ with the nodes IP address to identify each node. In this way, we can easily distinguish between two nodes holding the same IP address but having different key ($\phi(N)$).

In Table 5.1 below, we have shown a routing table at node D. Two nodes A and K hold the same IP address a as shown in Figure 5.2 above but different keys. In Table 5.1 node A is shown by IP_A and it has the key $\phi(N)$ of the first network. Node K is shown by IP_K and it has the key $\phi(N)$ of second network. These keys are assigned at the time node acquires its address. Note that all of first network nodes share the same key, as do the nodes in network two. Nodes from different networks do not share the same key. As in Figure 5.2 above, if IP_A is equal to IP_K, we can still distinguish in these two nodes by their respective keys and misrouting is prevented. This handles network merging problems with almost zero overhead and no connections are broken due to IP address changes.

This process is extremely efficient. Nodes hold the key of the network in which they were assigned an IP address. The disjoint cycles of QR make sure that there is no duplicate address in a single network. Duplication can only occur when we have identical QR in two merging networks. Addresses are unique in each partition, and therefore, each MANET associates only one key to all the nodes that belong to that MANET. Hence one key would be enough to distinguish between two nodes having the same IP address but belong to different MANETs.

The distributed approach described above does not depend on the underlying routing protocol. It can be integrated with pro-active[24] or reactive [5, 7] routing protocols for MANETs.

Destination	Key	Next Hop
IP_A	$\phi(n) = 1$	IP_E
IP_B	$\phi(n) = 1$	IP_B
IP_E	$\phi(n) = 1$	IP_E
IP_C	$\phi(n) = 1$	IP_E
IP_H	$\phi(n) = 2$	IP_E
IP_K	$\phi(n) = 2$	IP_E
IP_F	$\phi(n) = 2$	IP_B
IP_G	$\phi(n) = 2$	IP_E

Table 5.1: Network Key Identifier

5.4.2 DAD Dependent

In the second approach we run DAD to remove duplicate addresses. In both methods, a network merger is detected when a node hears a hello message with a different NID. It assumes that this is the network merger. The smaller NID is chosen to be the new NID of the merged network in both cases. In our approach two nodes from the intersection of the MANETs exchange N and the set of seeds S_0 of their individual MANETs. Now these nodes will only check for the conflicted QR values by generating a sequence of S_i of each state. If two values are found to be the same, duplicate addresses are detected. For example, if the same QRs are found in both MANETs then they are in conflict with each other. All such conflicted addresses are checked and if two nodes hold the same IP address, then one of these nodes has to give up its address and acquire a new IP address. In our approach in order to break minimum ongoing connections, we will pick the node with fewer TCP connections to change its IP address. This process is repeated for all duplicate addresses until there are no remaining duplicate addresses in the merged network.

Chapter 6

Clustering Approach

Address assignment in this method is slightly different than in the previous approach. Here an initial node chooses two safe primes, P' and Q' of 12 bits each and computes N(product of two safe primes) . A safe prime P' is defined to be a prime of the form $P' = 2 \times p + 1$, where p is also a prime. Hence, length of N is 24 bits. The idea is to fix the first 8 bits for network address and remaining 24 bits for QR values. Therefore, an initial node chooses an 8-bit NID. It computes a 32-bit IP address, by combining the 8-bit prefix for NID to the 24-bit value of the QR. When a new node joins the network, it computes an IP address by combining the 8-bit prefix with next QR in cycle to get an IPv4 32-bit IP address. Using this approach, we don't need to piggy back the NID in hello messages.

A node can learn about the NID from the 8-bit prefix of an IP address. Here we use 12-bit primes, but we can also use 13 or 14 bit primes and fix the network address to 6 bits or 4 bits, respectively.

A clustering mechanism can be implemented with the Zone Routing Protocol (ZRP)

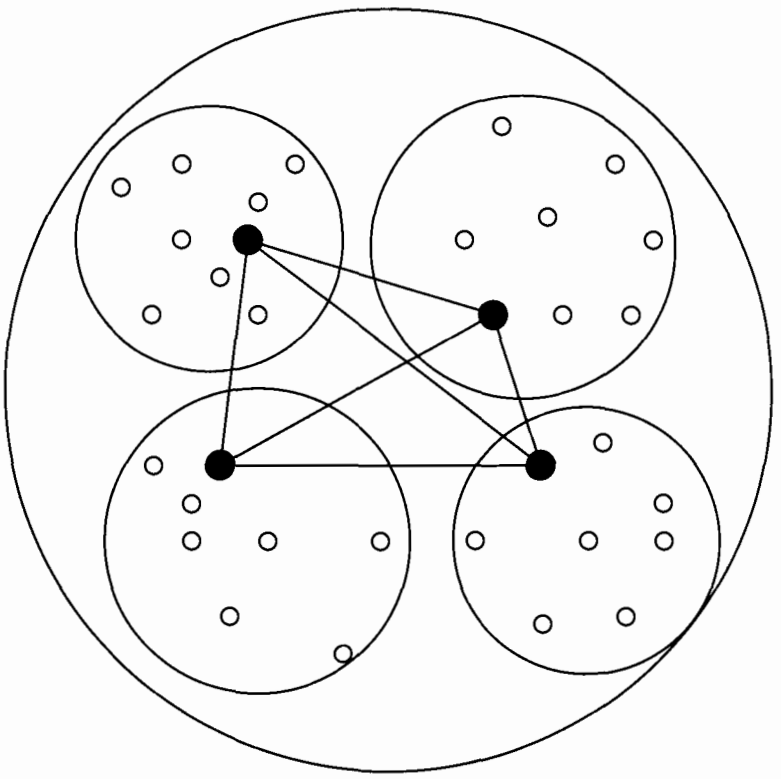


Figure 6.1: MANET Clustering Allocation

[32]. Nodes are arranged in clusters, and each cluster has exactly one cluster head, shown by a big black circle in Figure 6.1 above. All the remaining circles represents nodes in the MANET. Note that there are lines joining all cluster heads in the MANET. This way, cluster heads easily exchange information from one cluster to another. Cluster heads maintain routes to other clusters through cluster head information. Whenever a new cluster head is selected, its information is exchanged with all other clusters in the MANET. As a result, each cluster head updates its entries. Cluster heads are responsible for assigning addresses to new nodes in the cluster. Each cluster head gets a disjoint set of state value (S_0) seeds, range (length of cycles) and an IP address. Now the cluster head for each cluster is capable of assigning a unique IP address without consulting other clusters. This reduces latency in address assignment. Addresses are reclaimed automatically, as the address blocks repeat in cycles.

The range (interval before a QR repeats) has to be long. When an address repeats, the node which was assigned this address previously would have already left the MANET. Intra-group routing within a cluster is done with proactive routing [24], and inter-group (between clusters) routing is done in an on-demand basis, if needed, according to [5]. If a cluster gets partitioned from the MANET, it can still hold its address and keep assigning new addresses to newly arrived nodes. If it merges back some time later, there will be no duplicated addresses in the MANET because each cluster is using a disjoint set of numbers. When two independent MANET merge we get a new large MANET. A network merger is detected when a node receives a message from a node with a different prefix. If the two networks have the same QR, their IP address would be different; either because of the different prefixes length used or the different prefix numbers. In this case we will not have a duplicate address because the prefix is different.

Hence, no extra communication overhead is required for the network merger. If $phi(N)$ is the same, then the smaller partition becomes part of the bigger partition by giving up its addresses, and acquiring new IP addresses from the bigger partition after the merger. This approach can also be implemented without using a hybrid protocol with either OLSR [24] or AODV [5]. Note that we can use the same approach, without using clusters, in a somewhat decentralized manner where several nodes are capable of assigning addresses while others are not.

In the case where a cluster head leaves, we have to first detect that the cluster head has left the cluster and then choose a new cluster head. When a new node joins the MANET, it sends a *Discover* message in the joining cluster. If it does not receive a response from the cluster head, it times out and retries this process for a suitable K times. Then it sends a Head Clean message in the cluster. At this time the two-hop neighbors of the cluster head try to make contact with the cluster head node. If it's not reachable they send a *Choose message* upon which the node which was last assigned the address successfully acts as a new Cluster Head.

This node sends a *New Cluster Head* message within the cluster and assigns the

address to the new node by squaring its QR value and then reducing it modulo N. Address space is reclaimed and unique address can still be assigned without keeping track of all the address that have been assigned by the previous cluster head. This is because the next addresses are calculated from the last assigned address and remaining address would automatically repeat in cycles.

Chapter 7

Algorithm and Result

Here is the algorithm for finding Quadratic Residue cycles. We take two primes, p and q and compute N. We use Jacobi symbols to compute QR because their computational complexity $O(log\ p)^2$ is less than the Euler criterion of $O(log\ p)^3$. The Jacobi symbol of N is,

$$Jacobi(x, N) = Jacobi(x, p) \times Jacobi(x, q) \tag{7.1}$$

If the Jacobi symbol of (x, N) =1, then x can be either a QR or a pseudo square(PS). If it is a QR then the Jacobi symbol of $(x, p) = 1$ and $(x, q) = 1$. If it is pseudo square then Jacobi symbol of $(x, p) = -1$ and $(x, q) = -1$. We only need to check the Jacobi symbol for N and p. We implemented our algorithm in Maple and Java.

In our algorithm Figure 7.1 if x is a QR and is not in the list T, then we take it as a seed and compute the sequence of QRs until we hit the last element before the seed repeats again in the cycle. At that time the cycle is finished. List C keeps track of QR

```
QRCycles(n::integer,p::integer,q::integer)

    T:={ };   x;
    for  i from 1 to n − 1 do
        C := { };  x := i;
        if ( jacobi( x , n) = 1 and jacobi( x , p) = 1) then
            while not member(x, T) do
                C:= C union x;
                T:= T union C;
                x:= x^2 mod n;
            od;
            Period:=nops( C);
            if (nops(C) >= 1) then print(Period); fi;
        fi;
    od;

end:
```

Figure 7.1: Quadratic Residue Algorithm

in the current cycle along with its period (length of cycle) and list T contains all the QRs modulo N.

In our first experiment, we ran a test for our clustering approach by choosing two safe primes p and q of 12 bits each, and then computing $\phi(N)$ and N before computing the QR cycles to generate addresses. In the first experiment we chose

p= 2207

q=3467

N= p $\times$ q = 7,651,669

$\phi(N) = 7,645,996$

Total number of QR = 1/4 $\times$ $\phi(N) = 1,911499$

Cycle Length	Number of Cycles	Number of QR
1	1	1
29	38	1,102
1,732	1	1,732
50,228	38	1,908664

Table 7.1: Safe Prime Experiment Result

We get 38 long cycles of length 50,228 each. This means we can have 38 clusters and each cluster can configure 50,228 nodes. Using these long cycles we can configure close to two million nodes. Note that 99.85 percent of the QRs are in the long cycles. As the number of addresses we will get from these long cycles are 1,908,664. A cycle length of 50,228 is long enough to ensure that when the number repeats again, the node which was assigned this address previously would have left the MANET.

If some clusters get partitioned from the original MANET, they can still assign

unique addresses to newly joining nodes and if these partitions merge back together there would be no duplication. As N is 24 bits long, we can fix the 8-bit prefix for the network address. When a node hears a message from a node with a different prefix, it assumes a network merger has occurred and runs a network merging algorithm described before. Safe primes here were 12 bits each but we can also choose bigger safe primes of 16 bits each and get an N of 32 bits. In that case, we would be able to configure billion of nodes and we can piggyback the NID in hello messages generated by the first node in the MANET.

In another experiment, we chose two doubly safe primes instead of just safe primes to see how big a cycle (address block) we can get. We noticed that the length of the cycles we get is huge. This results in fewer distinct cycles (address blocks) because the length of a cycle is extremely big. The two doubly safe primes are 13 bits each:

$p = 4799$

$q = 4919$

Cycle Length	Number of Cycles	Number of QR
1	1	1
1,199	2	2,398
2,458	1	2,458
2,947,142	2	5894284

Table 7.2: Doubly Safe Prime Experiment Result

We get two long cycles of length 2,947,142 each. This ensures that the interval before a number (address) repeats again is quite large. When an address in a cycle repeats again the node which was assigned this address previously would have left the MANET. Hence, we avoid duplication and reclaim address automatically.

7.1 Addressing Scheme

In our approach, the network part of the IP address can vary depending on the length of the primes we have chosen. For example, consider Table 7.2, in it we chose two doubly safe primes of 13 bits each and get their product N of 25 bits. As we are working in the mod there will be no host part of the IP address generated by our program which will exceed 25 bits for this particular scenario. Therefore, we support Classless Interdomain Routing(CIDR) [25]. With so called CIDRized network addresses, the network part of an IP address can be any number of bits long, rather than being constrained to 8, 16 or 24 bits.

Our program is designed to support CIDR but we also have the capability to easily support four classes of the original Internet addressing architecture. The classes of classful addressing we support are class A, class B, class C and class D respectively. As in [10, 15] class A IP address has a "0" in the most significant bit position. Each class A network can configure 16,777,216 (2^{24}) unique hosts. A class B IP address has a "10" in the two most significant bit positions and it can configure up to 65,536 (2^{16}) unique hosts. A class C has a "110" in the three most significant bit positions and is capable of assigning 256 (2^{8}) unique addresses in the network.

All three of these unicast addressing schemes work with our program because our addressing function can provide unique host address values and we can fix the network portion of the IP address. Each value of the host address can be appended to the fixed value of the network address to form an IPv4 network layer IP address. It is obvious that same procedure could be repeated to form an IPv6 network layer address.

7.2 Optimized Algorithm

As compared to the original algorithm, the optimized algorithm has tremendously reduced the search space in the input size. What we mean is that the optimized version in the outer for loop iterates exactly once for each Quadratic Residue, which is lot less than N. For example in experiment 1, it iterated for 1,911,499 times instead of 7,651,669 and gets all the possible QRs. Also, instead of computing the Jacobi-symbol (x, N) we compute it for (x, q) because q is almost half the length of N and this would work faster. In addition to this, the first line which does an intersection operation makes sure that we don't have to test the Jacobi symbol to see if a QR is already present. It will simply return false and execution will go back to the outer loop.

Clearly this saves us computation time of more than three fourths as compared to our original algorithm. The idea behind this is that if p is a prime, for example take p=11, then there are $p-1$ numbers relatively prime to p. We divide the $p-1$ numbers into two parts; both parts contain $p-1$ numbers, but they are the exact opposite of each other in mod N.

part 1 = 1,2,3,4,5
part 2 = 6,7,8,9,10

Note that part 2 in mod N can be written as -5,-4,-3,-2,-1

Now all the Quadratic Residues can be obtained by successfully squaring the numbers in part 1 and reducing them modulo N. It gives Quadratic Residue = 1, 4, 9, 5, and 3.

There are exactly 5 Quadratic residues in mod p which are equivalent to $\frac{p-1}{2}$. Therefore, by squaring the first half of the relatively prime numbers we get all the QRs. Then when we take a product of two prime's p and q, the situation is bit different but we can guarantee the result with at least 99.9 % accuracy that all the QRs will be obtained by using the above algorithm in mod N. In the case where we are missing something, it will only be an extremely small cycle which is negligible in our case.

QRCycles(n::integer,p::integer,q::integer)

T:={ }; x;
$\phi(n) = (p-1) \times (q-1)$
Number of QR $= 1/4 \times \phi(n)$

for i *from* 1 *to* *Number_of_QR* *do*
if ($T \bigcap i = \emptyset$) then C := { }; $x :=$ i;
if (*jacobi*(x , p) == 1 *and* *jacobi*(x , q) == 1) *then*
while not member(x, C) do
$C := C \bigcup x$;
$x := x^2 \ mod \ N$;
od;
Period := *nops* (C);
$T := T \bigcup C$;
print(*Period*);
fi; fi;
od;

end:

Figure 7.2: Optimized Algorithm

7.3 Performance of Algorithm

As the results in Figure 7.3, show that we get all the possible number of cycles and the quadratic residues present in mod N by using our optimized algorithm. For this purpose we have repeated the experiment done for Tables 7.1 and 7.2. Moreover, we have results from two other types of settings. Firstly, where a network is formed without using safe primes, with $p = 2939$ and $q = 3739$. Secondly, with the combination of a safe prime and an odd prime having values $p = 1907$ and $q = 1511$.

The available IP address space increases monotonically with N. We have shown this with a bar chart in Figure 7.4. We have shown the values for up to millions of addresses. Each vertical bar represents the available address space at some specific value of N.

In Figure 7.5, we see that the size of the available IP address space grows as the value of N increases. We ran several experiments with different configurations of safe primes and ordinary primes all of the form of 3 mod 4. In all cases the available IP address space increases monotonically with N. The curve represented by QR shows the available permanent IP addresses and the second curve QNR shows the temporary address space that is available to nodes upon arrival in the MANET. Clearly, temporary addresses are twice the size of the available IP addresses. During the address allocation process a node is provided a temporary address as part of the process of acquiring a permanent address. Note that both QR and QNR values grow as N increases.

We ran 90 experiments with different configurations of p and q, and compared the network size collisions in the MANET. Each curve represents the IP address of a network. Each value of the curve indicates the duplication percentage when a MANET joins another MANET. We see that initially for all networks the collision rate is low. The maximum value of the collision rate we observed is 12.5 percent and the lowest value is less than 0.5 percent. This proves that our solution is highly scalable and when two networks come within close proximity of each other they can easily detect present and possible future collisions

Cycle Length	Number of Cycles	Number of QR
1	1	1
29	38	1,102
1,732	1	1,732
50,228	38	1,908664

(a) Safe primes network

Cycle Length	Number of Cycles	Number of QR
1	1	1
1,199	2	2,398
2,458	1	2,458
2,947,142	2	5894284

(b) Doubly safe primes network

Cycle Length	Number of Cycles	Number of QR
1	1	1
2	1	2
3	2	6
6	2	12
11	8	88
12	21	252
22	8	176
28	12	336
33	16	528
66	16	1056
84	360	30,240
132	168	22,176
308	96	29,568
924	2880	2,661,120

(c) Simple primes network

Cycle Length	Number of Cycles	Number of QR
1	1	1
4	1	4
15	10	150
60	10	600
68	70	4760
1020	700	714,000

(d) Safe prime and simple prime combination network

Figure 7.3: Optimized Experiment Result

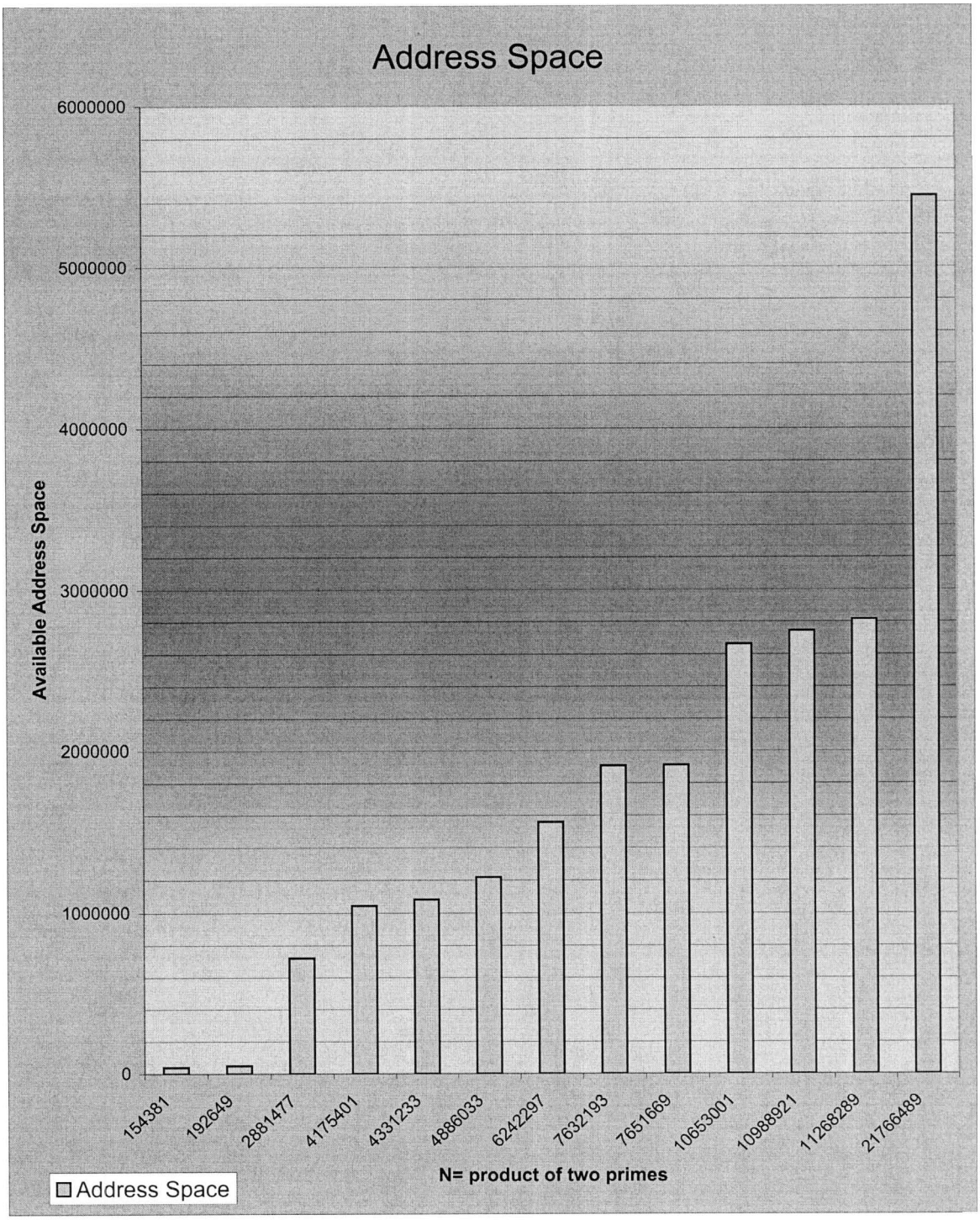


Figure 7.4: Available IP Address Space

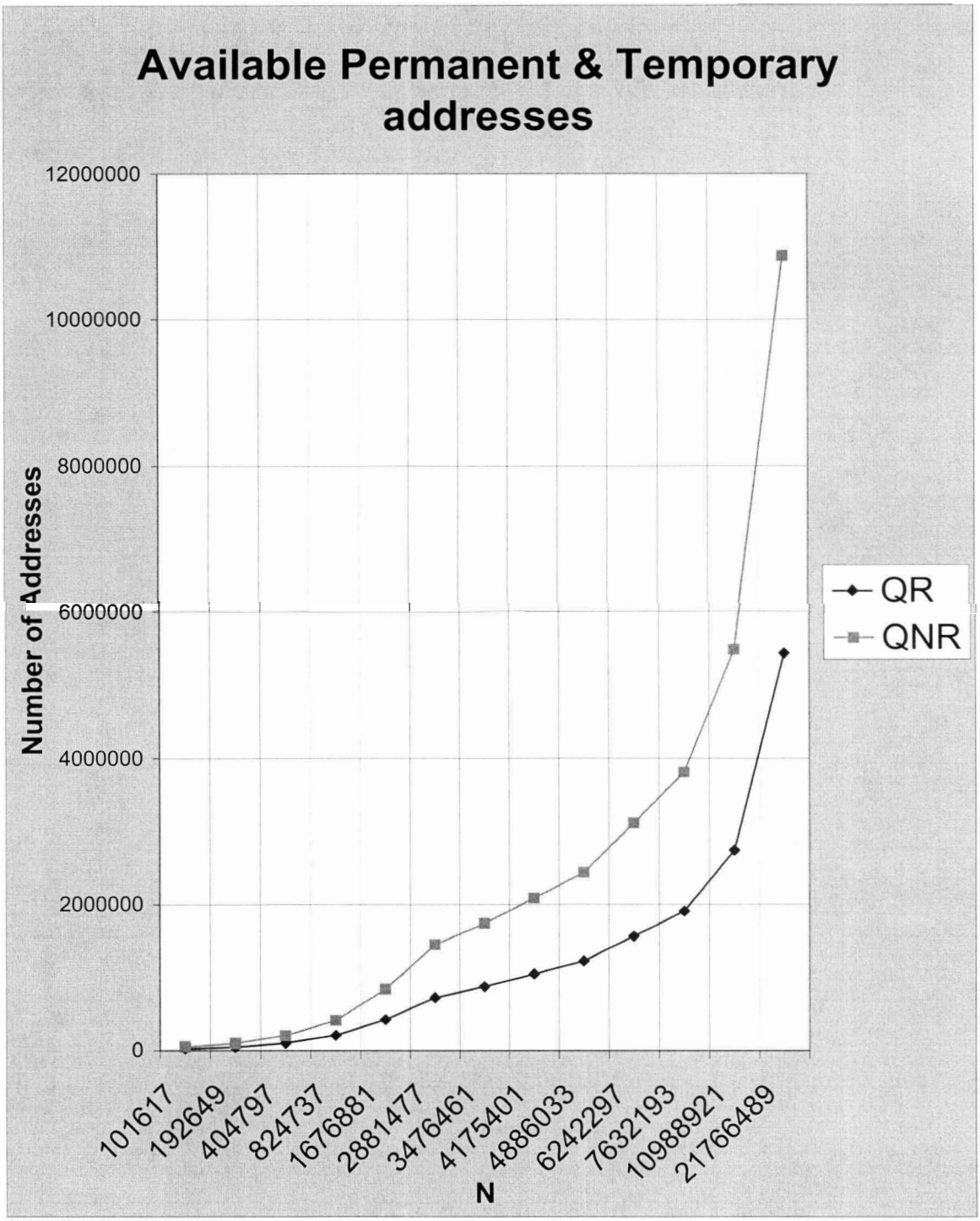


Figure 7.5: Address Space

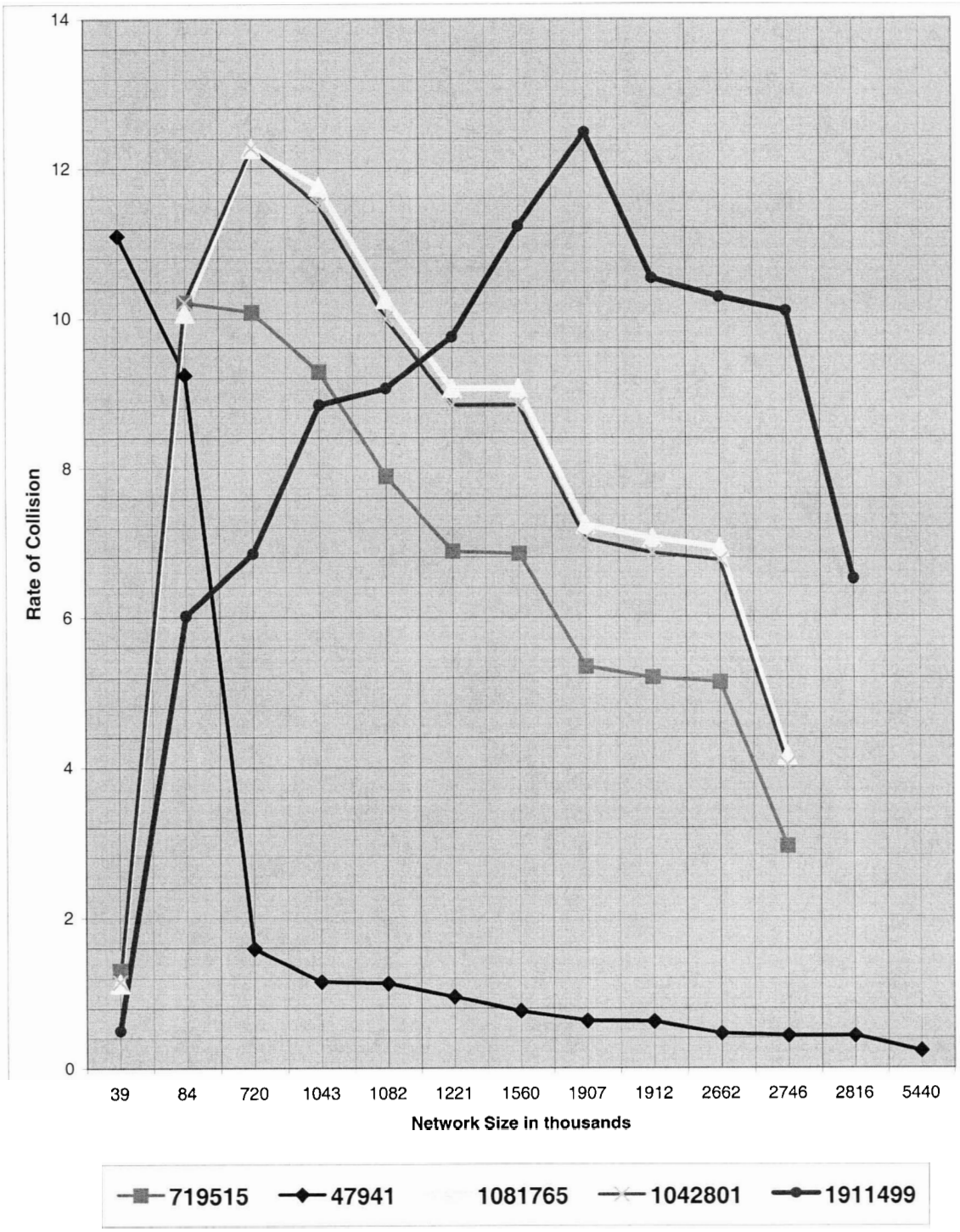


Figure 7.6: Collision Rate

in the MANET. Figure 7.6 shows a network size of nearly 5.5 million addresses. With the exception of the curve representing 47,941 nodes, all MANET collision rates start at a lower value and increase monotonically. After reaching the maximum collision point, the collision percentage decreases monotonically as the network size increases.

The following figures show network set up with only safe primes, ordinary primes and a combination of safe prime and an ordinary prime. We show the value of each network merging with a network of varying sizes. Note that the maximum collision percentage is created when a network merges with a network of almost the same size. At that point, the number of duplicate addresses is greater than any other point in the graph. Either side of the maximum point, the collision rate will be lower than this. Again, as the merged network size grows, the collision rate will decrease monotonically. First the collision rate increases monotonically because the MANET is merging with a smaller MANET. When a MANET merges with a MANET nearly equal to its size, the collision percentage reaches its maximum value. As the MANET merges with a MANET greater than its size the collision percentage decreases monotonically.

We configure three different types of networks: one with using safe primes, one without using safe primes, and one with a combination of a safe prime and an odd prime. What we see is that safe primes always give us a large address block, hence guaranteeing that an address will not be repeated again for an extremely long interval. By contrast simple primes give us an address block of a smaller size when the possible network size is small, but, we can easily see that as network size increases, the address block size increases monotonically.

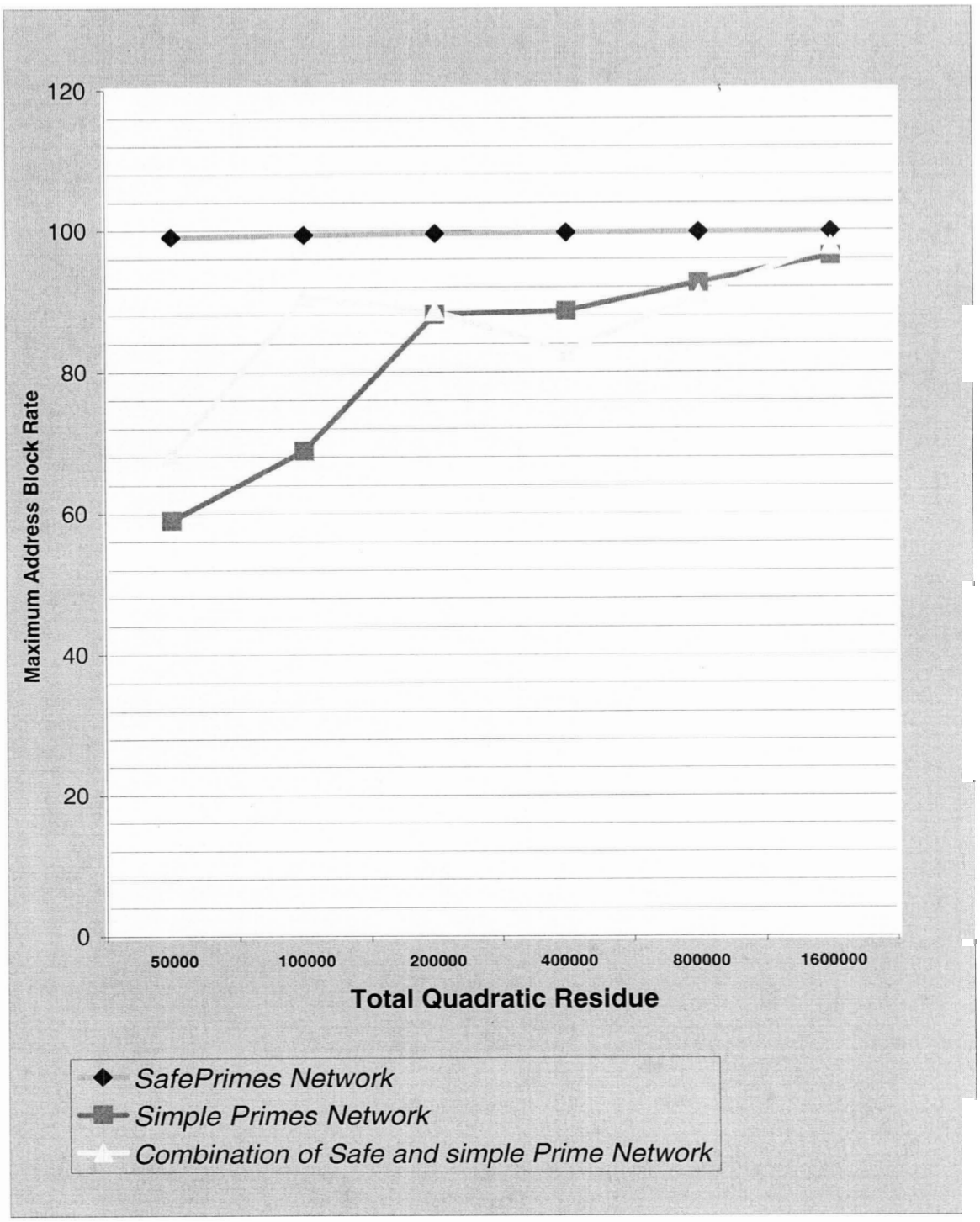
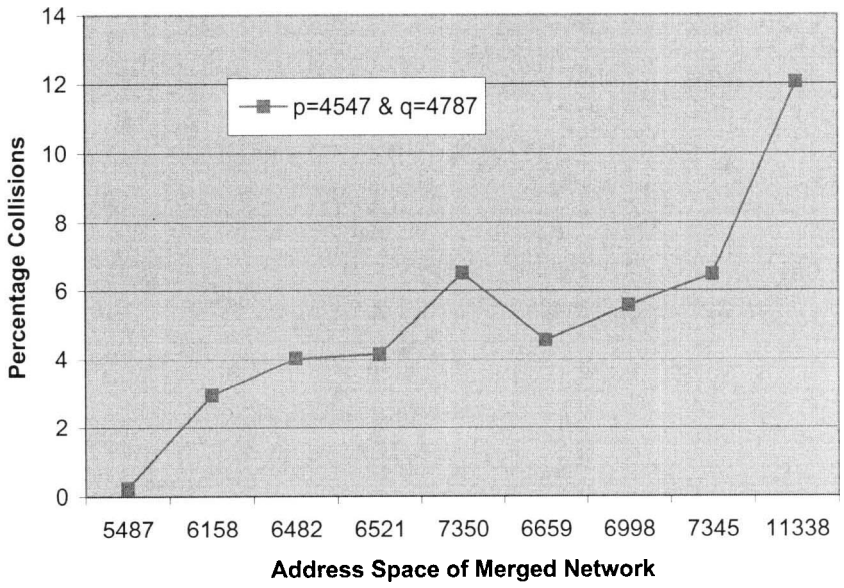


Figure 7.10: Address Block Size



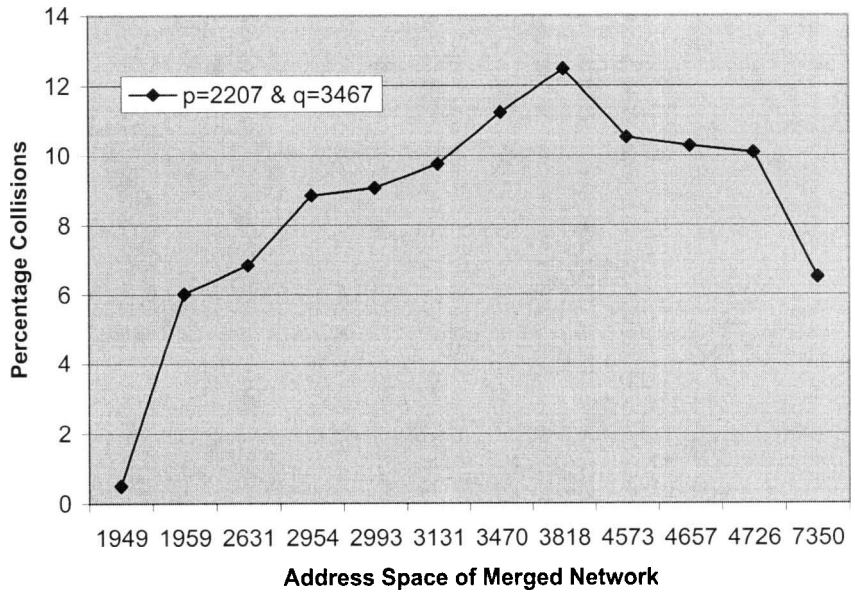
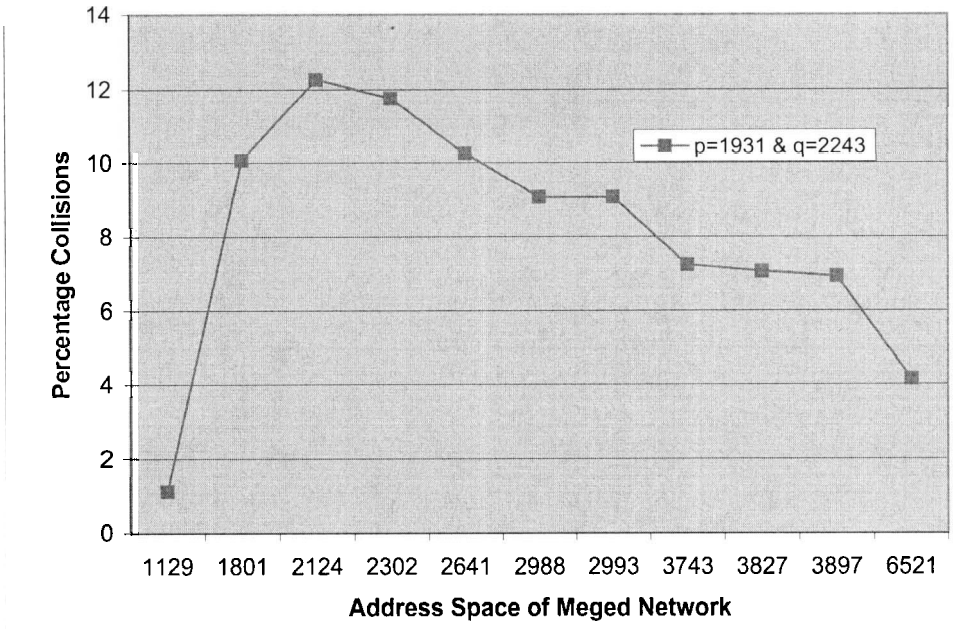


Figure 7.7: Merged Network Collision One



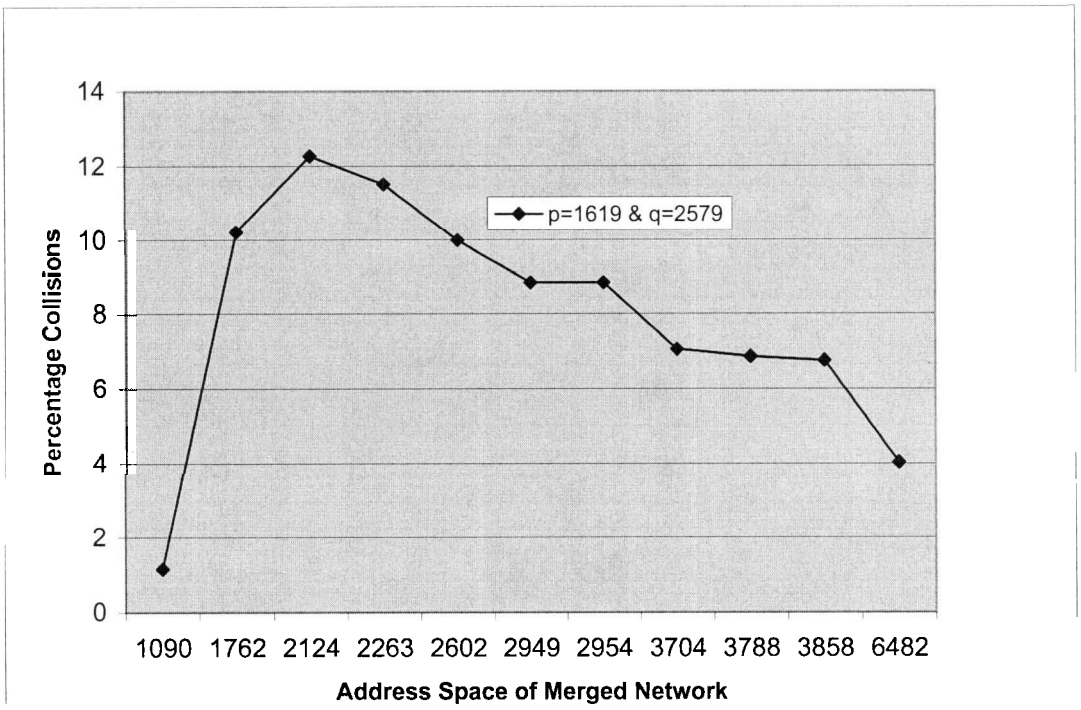


Figure 7.8: Merged Network Collision Two

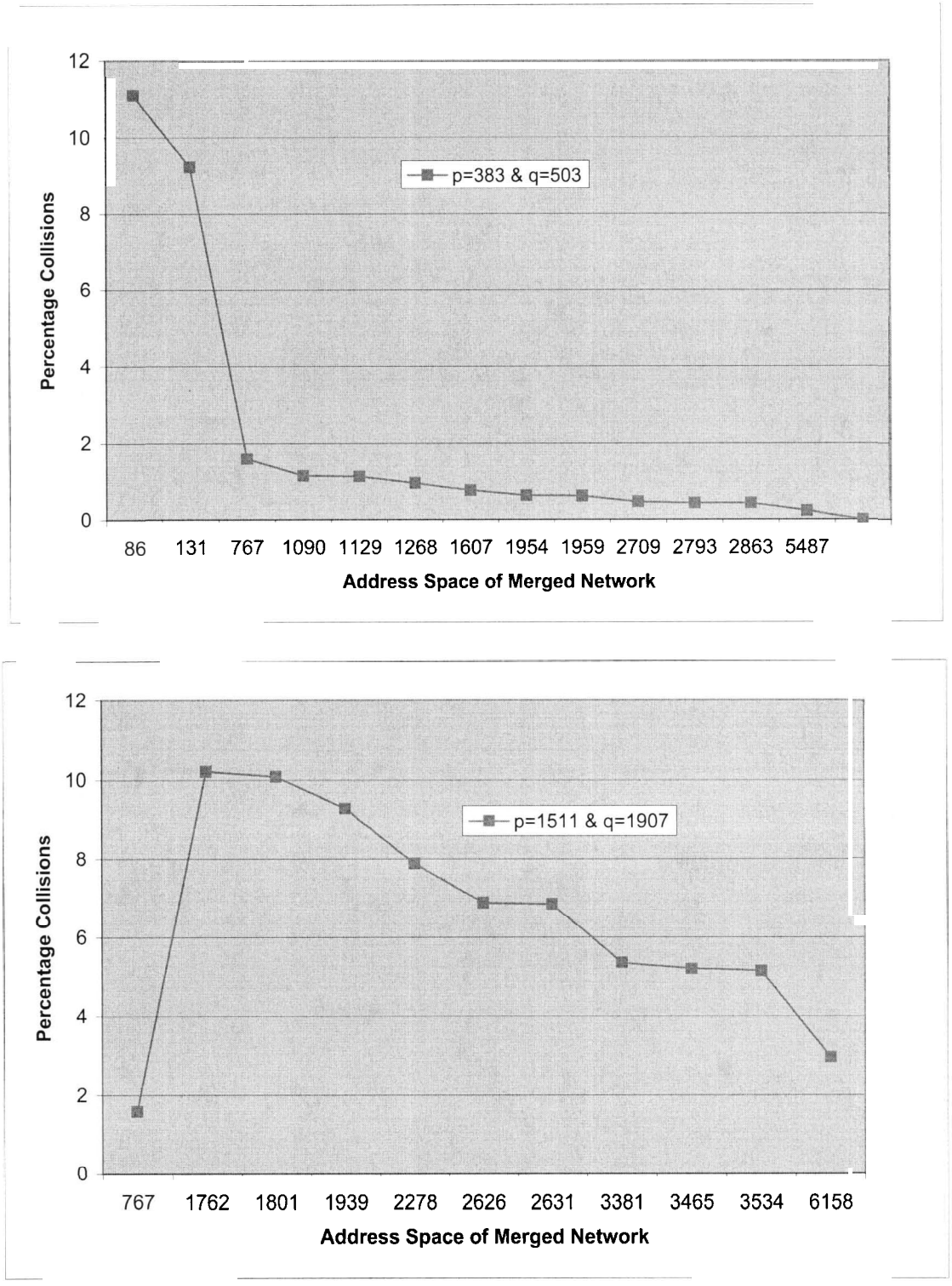


Figure 7.9: Merged Network Collision Three

Chapter 8

Hybrid MANET

Suppose node x with IP address 250.0.0.10 comes within radio frequency range of a router R and it wants connectivity with the global internet. It requests a connection to the internet site at the global IP address $g.g.g.g$ and port number 80. It sends a request to the router R. The connecting router R which is part of the wired network, connects to the MANET with one of its interface IP addresses, 10.10.0.1.

The MANET connects through a gateway node to the router. This gateway node will send an InetDiscover message and will receive a reply from the router R. Now the gateway node will be aware of its connection with the external wired network or internet. The gateway node will do the selective two hop broadcast in the MANET, with its IP address piggy-backed in the periodic routing *hello message*. This avoids generating extra message or communication overhead to inform the MANET that it has connections available to the internet.

The router will perform Network Address Translation (NAT) and will put the source

IP address on the packet as the router interface outgoing IP address (10.10.0.1). Now the router will receive all the responses from the internet at this address. Here is how the protocol works:

The MANET sends an external network (exterNet) Discover message. The router will know that a MANET is connecting to the wired network. The router will reply with an ACK message.

If the NID or network prefix is the same as that of the router R it will generate an error NACK message. Otherwise it sends an ACK. Since x may move or change its current position, x will advertise the router interface within the MANET. Therefore any node can connect to the internet through this address as long as one of the nodes in the MANET is within the router's range.

If the MANET moves away from the router transmission range, the Hybrid Network gets disconnected. If more than one router is within the transmission range, a node in the MANET will connect to the router depending on its distance (number of hops) from the router or traffic or response time from the router. This will distribute the load evenly. Chances are if a MANET is within transmission range of one router, there will be other routers or wired networks close by.

8.1 MANET Disconnections with Router

8.1.1 Gateway Node Movement

In a MANET it is most likely that a gateway node will move and as a result may go out of the transmission range of the router. But this does not mean that the MANET has lost

connection with the router. Some other node y may have come within the transmission range of the router and as a result the connectivity is still alive. Therefore, we have to keep track of all these gateway nodes and route packets to the proper gateway nodes on the way to the router. Otherwise, we have to advertise the router address to enable a node in the MANET requiring internet connectivity to route the message towards the router even when the gateway node changes. Note that at any point in time there can be two or more gateway nodes connecting to the same router.

8.1.2 Reconnects with Same Router

If the time a MANET is out of reach from the router is very small, a node may not know if the connection was lost. All ongoing connections will remain alive and when the MANET gets back within transmission range, communication will take place as before.

8.1.3 MANET-Router Time Out

If the disconnection of a Hybrid network occurs for a significant time such that the nodes in the MANET are unable to reach the router and vice versa, then the nodes may keep their connections alive for a certain timeout period/interval k. If within this time frame the MANET gets connected with R, communication takes place as usual.

If the time out period k expires, the MANET might not connect to the same router. Therefore, it would double the timeout value to $2k$. If it again times out without getting within transmission range of a router, it will yet again double the time out period to $4k$. If the MANET is highly mobile it will get within a router range soon. If the mobility is slow it might want to increase the timeout as much as 10k before closing all the ongoing connections. Highly mobile MANETs will often have connections with different routers, but

each connection will be relatively short-lived.

Multiple Router Connections

If two or more routers are within the radio frequency range of the router, a MANET can get internet connectivity by connecting with different routers. A node connects to the router closest to it, depending on number of hops from the router, traffic, congestion level, response time, available bandwidth, signal strength and other power issues. Note that there may be multiple gateway nodes connecting to the same router or there might be only a single gateway node connecting to the router. New connections will be set-up depending on the closest available router from the requesting node.

8.1.4 Connecting to Different Router

Consider what happens if the MANET connects to a different router. If this router is the only router connected to the MANET now, then all routes to the previous router have gone obsolete.

1)If previous router R is still connected, all ongoing communications from the MANET to R will take place as previously.

2)If connection with R is lost and the MANET becomes connected to a new router S, then all nodes which were connected to R will connect to S.

3)If a node x in the MANET did not break a connection with R, it will send a message to S to deliver its packet to R, which will route it to its final destination. R will

send the packets it had buffered for x through S to x in the MANET.

If S wants to communicate directly with Z it will send a message *sourceChange* to S containing the address of R. S will know that it has to perform NAT.

Communication between Routers

This is how the intercommunication between two routers takes place.

S will inform destination Z about the change of address. Node x will send the packet to S and as a result S will inform Z about the address change. Communication between S and Z will take place through NAT, such that TCP will be oblivious to address changes at the IP layer. The network layer will direct packets to S address instead of R.

TCP Connections

TCP acknowledgements and sequence numbers will ensure that the packets will be delivered in sequence. Z will resend all the packets that were not received by x and vice versa.

R will put the packets in a buffer and will turn on the timer if no response is received from x. R will timeout and discard/empty the buffer.

In TCP R will not send an ACK of a packet to Z until it receives ACK from the final destination x. Therefore, Z will timeout and resend the same packet to R again.

Note that not sending an ACK from R to Z ensures that if x connects to another router it will receive all the packets that are delivered from Z to R but not delivered successfully from R to x. We know that TCP maintains connection state in the end systems. This connection state includes receive and send buffers. TCP will continue to send a data segment until its receipt has been acknowledged by the destination, regardless of how long reliable delivery takes.

UDP Connections

UDP does not have sequence numbers to guarantee in order delivery nor does it maintain connection state.

For loss-tolerant multimedia applications, a small amount of packet loss is tolerable. But, if the packet loss is significant during the time x gets out of the reach of R and connects to S then there will be a long period of vulnerability. Note that S can send a message to R by performing NAT. S sends an outgoing packet with S's address but inside it will have the address of x.

R will receive packets from S and it will again perform NAT and send packets to Z.

Z will continue to direct packets to R as its destination address.

R forwards packet to S which in turn forward them to x

Thus communication will take place between two end hosts running UDP connections.

Chapter 9

Efficient Address Calculation

Here we will describe a quick and efficient way of finding long disjoint cycles in N, where N is a product of two prime's p and q. Each of the long cycles will be used as a disjoint address block. The address block interval must be long, because when an address repeats, the node which was assigned this address previously would have left the MANET. Therefore, we want to know the length of long cycles without doing much computation. The initiator node when doing future address computation for the MANET must not select a short address block. Here we will explain how to efficiently compute long address blocks.

If a is a pseudo square the problem is that there is no known way to decide if $a \in QR(n)$ in polynomial time if the factorization of N is not known. Note that $a \in QR(n)$ if and only if $a \in QR(p)$ and $a \in QR(q)$.

There are two types of cycles as discussed before in the address generation section. They are;

$$s_1, s_2, \bullet \bullet \bullet\bullet, s_{i-1}, s_i, s_{i+1}\, \circ\circ\circ\circ\circ\circ, s_z, s_i, \cdots$$

We see that s_i belongs to two different sequences. Firstly s_{i-1} squares to s_i and secondly, s_z squares to s_i. We are interested in the scenario where the sequence repeats again exactly in the same order as before. This can occur when primes are chosen to be congruent to 3 mod 4. If p and q are both primes congruent to 3 mod 4 we get the address range of our address block.

Proof: Computing the Addressing Range of each Seed

Let two primes p and q be of the form $p = 2p^{'} + 1$ and $q = 2q^{'} + 1$. Let the number of addresses be computed by following the sequence as specified in address generation section

Let

$$p^{'} = \frac{p-1}{2} \tag{9.1}$$

$$q^{'} = \frac{q-1}{2} \tag{9.2}$$

Let

$$Y = \frac{p^{'}+1}{2} \tag{9.3}$$

and

$$Z = \frac{q^{'}+1}{2} \tag{9.4}$$

We note that

$$2^{Range} \equiv 1 \ mod \ p^{'}$$

This means that

$$p^{'} \mid 2^{Range} - 1$$

or

$$\frac{2^{Range} - 1}{p^{'}} \equiv \ a$$

We can write this as

$$P \mid 2^{Range} - 1 - p^{'}$$

Dividing both sides by 2 gives

$$p^{'} | (2^{\ Range\ -\ 1} - Y) \tag{9.5}$$

A similar procedure for q results in

$$q^{'} | (2^{\ Range\ -\ 1} - Z) \tag{9.6}$$

From (9.5) We can rewrite the equation as

$$2^{\ Range\ -\ 1} \ \equiv \ Y \ \ mod \ p^{'} \tag{9.7}$$

From, here we can compute by taking $log_2\ mod\ p'$ such that

Range-1 = a mod p

Range = a + 1 mod p

Where a+1 is the cycle range.

Now by similar process, we can compute q from equation (3)

Range-1 = b mod q

Range = b+ 1 mod q

Similarly, b is also the cycle range.

Note that these are the lengths of short cycles which are present modulo p and modulo q. What we are interested in is the range of long cycles, so that a big block of address space can be assigned in order to ensure that an address does not repeat before the node which was assigned that address has left the MANET.

From above, we can easily compute length of the long cycles as the lowest common multiple (LCM) of a and b.

Address Block Length = LCM (a, b)

This method not only works for safe primes but also for other primes.

9.1 Practical Application

Jacobi Symbols can be computed in polynomial time using the Euclidean algorithm and hence are a much more efficient test for quadratic residues then using the Euler Criterion. The Euler Criterion uses the square and multiply algorithm.

We can choose doubly safe primes such that $p = 2p^{'} + 1$ where $p^{'} = 2p^{''} + 1$. This gives us prime p of the form:

$$p = 4p^{''} + 3$$

It can be easily seen that such doubly safe primes gives all primes p which are congruent to 3 mod 4.

Chapter 10

Security in MANET

If a MANET decides beforehand two primes P and Q and computes their product N, a range of addresses in QR mod N can be used or any specific cycle can be used, such that valid values of Quadratic Residue mod N are known and they have the ability to configure themselves with valid addresses. An outside party even if know N cannot find which number can be a QR of N unless they factor N. If a node needs to be assigned the address, the sender will encrypt the address with a public key $\boldsymbol{b}$.

The requester upon receiving the address will decrypt it with public key $\boldsymbol{a}$. Such a system is computationally secure, because the requester cannot decrypt the message to get a valid address unless it knows the secret key a. One can only compute the decryption exponent a if the factorization of N is known. If p and q are large primes this system cannot be broken. An arbitrary node cannot factorize N. This kind of systems is particularly useful in army or intelligence operations, disaster relief or emergency situations, conference rooms and school facilities.

If a node decides to assign address x it first encrypts it using public key cryptography

$$y = x^b \mod N \tag{10.1}$$

Requester upon receiving y, performs the inverse operation

$$x = y^a \mod N \tag{10.2}$$

This way if there is an ongoing intelligence operation it would not allow a malicious node to be assigned an address, such that it becomes the part of the MANET. Therefore, this scheme assigns addresses to a restricted set of nodes and an arbitrary node cannot join in until it is authorized to do so. Also they can assign addresses from one cycle out of the various cycles present in N.

10.1 Co-existence of Schemes

What if someone randomly chooses an address and tries to break the system? How can we enforce our system? There are two cases. In the first case, if we are trying to be flexible we can allow someone else using some other scheme of assigning addresses to keep on assigning addresses as long as they don't conflict with our addressing scheme. But if we allow such activities then they might select addresses which are QR. Two scenarios can occur where these addresses might have been already assigned, in which case a conflict detection algorithm will work to detect the duplicate address assignment and any such assigning node will need to compute a conflict-free address. Note that even if a current conflict is not detected there might be a potential conflict. What we mean is if an address is a QR mod

N which has not been assigned yet, then, this potential address has been taken away from us. If this will happen not only one, but a block of potential addresses which lie on the QR cycles can be assigned and taken away from us. Therefore, this cannot be allowed in our system.

Suppose a node joins our MANET and it is using an address with different NID. We might consider this a network merger. If we know the node is not using QR addresses, we will assign a new QR address to this node and make it drop its current address. Hereafter, this node will assign addresses based on our scheme. What if a node tries to assign addresses which are not QR? As no address checking is done when an address is assigned, an arbitrary address assignment might go undetected.

If the Network ID part of this address is different we will treat this similar to the previous case. If the NID of the host is the ID of the network but the Host ID is not a QR then with our current scheme nodes will communicate with this node without any problems, since this address is not in any conflict. This is the case where someone is trying to break our system. We can test the Host ID part by checking if the Host ID is a QR in p and in q. If not, then we declare this address invalid and send an address cleanup message which indicates that no more communication will take place with any such address. From there on, if any node x request a service or connection with any node y, y will make sure that x's address is a QR in N and is a valid address. This way we keep track of any such illegal address assignments.

Chapter 11

Conclusion

Our approach assigns unique addresses to new nodes with low latency without relying on either periodic or reliable flooding which consumes a considerable amount of bandwidth. It distinguishes between concurrent address requests and replies, and handles the mobility scenarios at the time of address assignment. We prevent misrouting between nodes. Also, we do not incur extra communication overhead or bandwidth for reclaiming the address space once all the addresses have been assigned. After reaching the maximum address pool size our addressing function makes sure that addresses repeat automatically to ensure conflict-free future address allocation. Each subsequent occurrence of an address block is an exact replica of the one before. This ensures the address space is disjoint and no two nodes share the same address. The interval before an address block enters a cycle must be long enough to ensure uniqueness. Therefore, there will be no two nodes having the same IP address in a MANET at any given time.

Moreover, our protocol provides reliable state synchronization in presence of packet loss, delays and network merging. We manage the address space in a way such that address

leakages and inconsistent states that may result in an exhausted address space or unnecessary address changes are avoided. In our approach, network partitioning and subsequent merging does not induce duplicate addresses. It handles temporary and permanent network partitions. We provide a way to distinguish between two different networks. The two different networks can be two independent MANETs or a MANET and a wired network. Our solution ensures scalable and extremely low overhead solutions for network mergers. In the best case scenario, we do not have to run the DAD mechanism to check for duplicate address entries, and as a result broadcasting and flooding costs are saved. In our approach, no explicit procedure is invoked for handling network mergers. This greatly reduces the overhead cost and uses minimum bandwidth and saves the battery power of mobile nodes.

This approach is scalable and is capable of configuring at least millions of nodes with even distribution. Our address allocation protocol is independent of the underlying routing protocol but can also be integrated with a routing protocol. We provide support for proactive, reactive and hybrid routing protocols in MANETs. For IPv6 addressing formats we can get a unique IP address by embedding a 48-bit MAC address plus a Quadratic Residue or a randomly generated large number. But this is not possible for the 32-bit IPv4 address format.

We also provide Hybrid network connectivity and a procedure to maintain connections and deliver in-order packets to the end host. Connections are maintained and packets are eventually delivered to their destination, even when a MANET moves from one wired network to another wired network. We also provide *ad-hoc* network security for small scale *ad-hoc* networks. In the future we would also like to enhance our security measure, to prevent a malicious node from holding or exhausting the address space.

Bibliography

[1] *Duplicate MAC Addresses on Cisco 3600 Series*, http://www.cisco.com/warp/public/770/7.html, 1997.

[2] A.McAuley A. Misra, S.Das and S.K.Das. Autoconfiguration, registeration, and mobility management for pervasive computing. In *Proc. of IEEE Personal Comm.*, page 24.

[3] C.Perkins ET Al. Ip address autoconfiguration for ad hoc networks. In *Internet Engineering Task Force (IETF), Internet Draft*, http://people.nokia.net/ charliep/txt/aodvid/autoconf.txt, Nov. 2001.

[4] Tayal A.P. and Patnaik L. M. An address assignment for the automatic configuration of mobile ad hoc networks. In *Proc. of ACM Personal and Ubiquitous Computing*, 2004.

[5] E.Belding-Royer C. Perkins and S.Das. Ad hoc on demand distance vector (aodv) routing. In *Internet Engineering Task Force (IETF)*, RFC 3561, July 2003.

[6] C.Perkins. Ip mobility support. In *Internet Engineering Task Force (IETF)*, RFC 2002, Oct. 1996.

[7] D Maltz D. Johnson and Y.Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr). In *IETF Internet draft*, draft-ietf-manet-dsr-10.txt.

[8] R. Droms. Dynamic host configuration protocol. In *Internet Engineering Task Force (IETF)*, RFC 2131, March 1997.

[9] D.Stinson. *Cryptography: Theory and Practice*, pages 173–178. Chapman and Hall/CRC-Press Inc, 2nd ed edition, 2002.

[10] Buck Graham. *TCP/IP Addressing*. Morgan Kaufmann, second edition, 2001.

[11] M. Guines and J. Reibel. An ip address configuration algorithm for zeroconf mobile multihop ad hoc networks. In *Int'l Wksp. Broadband Wireless Ad Hoc Networks and Services*, Sophia Antipolis, France, August 2004.

[12] L.Ni H.Zhou, M.Mutka. Ip address handoff in the manet. In *Proc. of IEEE INFOCOM 2004*, March 2004.

[13] M.Mutka H.Zhou, L.Ni. Prophet address allocation for large scale manets. In *Proc. of IEEE INFOCOM*, San Francisco, CA, April 2003.

[14] M. Faloutsos J. Eriksson and S. Krishnamurthy. Scalable ad hoc routing: The case for dynamic addressing. In *Proc. of IEEE INFOCOM*, March 2004.

[15] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Addison Wesley, second edition, 2003.

[16] A.J. MccAuley and K.Manousakis. Self-configuring networks. In *Proc. of 21st Century Military Comm. Conf. Proc.,*, volume 1, pages 315–319, 2000.

[17] M. Mohsin and R.Prakash. Ip address assignment in mobile ad hoc networks. In *Proc. of IEEE Milcom*, Anaheim, USA, Oct. 2002.

[18] N.Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *Proc. of ACM Mobihoc*, June 2002.

[19] P. Patchipulusu. Dynamic address allocation protocol for mobile ad hoc networks. Master's thesis, Texas A and M University, 2001.

[20] J.L. Peterson and T.A.Norman. Buddy systems. In *Proceedins of Comm. ACM*, June 1977.

[21] B.Aboba S.Cheshire and E.Guttman. Dynamic configuration of ipv4 link-local addresses. In *IETF Internet draft, July 2004, Work in Progress*, http://files.zeroconf.org/draft-ietf-zeroconf-ipv4-linklocal.txt, July 2004.

[22] R.Prakash S.Nesargi. Manetconf: Configuration of hosts in a mobile ad hoc network. In *Proc. of IEEE INFOCOM 2002*, New York, NY, June 2002.

[23] S.Thomson and T. Norten. Ipv6 stateless address autoconfiguration. In *Internet Engineering Task Force (IETF)*, RFC 2462, Dec. 1998.

[24] A.Laouiti-et al. T. Clausen, P.Jaqcuet. Optimized link state routing protocol (olsr). In *Internet Engineering Task Force (IETF)*, RFC 3626, Oct. 2003.

[25] J.Yu V.Fuller, T.Li and K. Varadhan. Classless inter-domain routing (cidr): an address assignment and aggregation strategy. In *Internet Engineering Task Force (IETF)*, RFC 1519, September 1993.

[26] K. Weniger. Passive duplicate address detection in mobile ad hoc networks. In *IEEE WCNC 2003*, New Orleans, LA, March 2003.

[27] K. Weniger. Pacman: Passive autoconfiguration for mobile ad hoc networks. In *IEEE Journal on Selected Areas in Communications (JSAC)Special Issue 'Wireless Ad hoc Networks'*, March 2005.

[28] K. Weniger and M.Zitterbart. Ipv6 autoconfiguration in large scale mobile ad hoc networks. In *Proc. of European Wireless 2002*, 2002.

[29] K. Weniger and M. Zitterbart. Address autoconfiguration in mobile ad hoc networks: Current approaches and future directions. In *IEEE Network Magazine*, volume 18, August 2004.

[30] Y.S.Chen and S.M.Lin. Raa: A ring based address autoconfiguration protocol in mobile ad hoc networks. In *International Conference on Mobile Ad Hoc and Sensor Networks*, December 2005.

[31] E. M. Belding-Royer Y.Sun. A study of dynamic addressing techniques in mobile ad hoc networks. In *Proc. of Wireless Communications and Mobile Computing*, April 2004.

[32] Prince Samar Zygmunt J. Haas, Marc R. Pearlman. The zone routing protocol (zrp) for ad hoc networks. In *IETF Internet draft,*, draft-ietf-manet-zonezrp04.txt.