Auto-configuration in Multi-hop Mobile ad hoc Networks

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

Priyanka Sinha

Certificate of Approval:

Jitendra K. Tugnait James B. Davis Professor Electrical and Computer Engineering	Prathima Agrawal, Chair Samuel Ginn Distinguished Professor Electrical and Computer Engineering
Chris A. Rodger Scharnagel Professor Mathematics and Statistics	George T. Flowers Interim Dean Graduate School

Auto-configuration in Multi-hop Mobile ad hoc Networks

Priyanka Sinha

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
May 10, 2007

Auto-configuration in Multi-hop Mobile ad hoc Networks

Priyanka Sinha

_____

Signature of Author

_____

Date of Graduation

Vita

Priyanka Sinha obtained a Bachelor of Technology degree from Indian Institute of Technology Guwahati, in Computer Science and Engineering. She was awarded the Institute Merit Award in 2000-2002. At Auburn, she was a Vodafone fellow in 2005-2006. She has been a Graduate Teaching Assistant, a Graduate Research Assistant and a Graduate Fellow during her masters studies. Her research interests are in the broad area of computer systems, networking, security and wireless.

Thesis Abstract

Auto-configuration in Multi-hop Mobile ad hoc Networks

Priyanka Sinha

Master of Science, May 10, 2007
(Bachelor of Technology, Indian Institute of Technology Guwahati, 2004)

105 Typed Pages

Directed by Prathima Agrawal

As devices with wireless access technology, greater memory and longer battery life, proliferate, there is a need to be able to allow more than one hop communication amongst them. Also, such a network, with wireless enabled devices, is easier to deploy in case of a disaster. But, these devices that participate to form a multi-hop Mobile Ad Hoc Network (MANET), need a unique identifier such as an Internet Protocol (IP) address to be able to participate in networking. To deploy and access services in this network they also need a service management infrastructure. In the wired and wireless local area network such an identifier is provided either by static assignment or via auto-configuration protocols such as Dynamic Host Configuration Protocol (DHCP). To support service discovery, the Domain Name System (DNS) translates from the name of the service to the IP address associated with it. Distributed Hash Table (DHT) based protocols such as Chord have been used for service discovery and content management. This thesis exposes the problem of configuring networks that experience partitions and mergers due to disasters. Thus, physical damages to networks may affect service architectures. This thesis illustrates with

specific examples, how adaptation of existing solutions does not solve the problem. It recommends the use of a protocol that allows concurrent joins and leaves.

Acknowledgments

I would like to dedicate this thesis to my parents Swapna Sinha and Madhusudan Sinha. I am very grateful for the advice and support of my advisor, Prof Prathima Agrawal, for her feedback and for keeping me focused in my research. I also thank her for her guidance, encouragement and financial support throughout my stay at Auburn. I am thankful to the Vodafone Foundation for granting me a fellowship, and to Toshiba America Research for supporting this work. This work is partially supported by US National Foundation CNS-0417565. Prof Vishwani has been instrumental in my coming to Auburn, for which I am grateful. I would like to extend my gratitude to Dr Anthony McAuley and Dr Raquel Sempere Morera of Telcordia Technologies for exposing me to this problem in large mobile ad hoc networks. I would like to thank Dr Subir Das also of Telcordia Technologies for helping me understand the Dynamic Rapid Configuration Protocol (DRCP) and Dynamic Configuration and Distribution Protocol (DCDP) protocol. I would also like to thank fellow graduate students at Auburn, Anand, Dong, Kalyan, Pratap, Ravi, Santosh and Sowmia.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliography conforms to those of the transactions of the Institute of Electrical and Electronics Engineers

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`. Figures and plots were created using GIMP and MATLAB

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Communication in an Internet uses a layered approach as shown in Figure 1.1. Each node consists of several layers of standard communication protocols commonly called the network stack. The physical layer consists of protocols that allow bits to be transmitted and received using the communication medium. The data link layer along with Medium Access Control (MAC) mechanisms consist of protocols for point to point communication over the physical layer. The network layer consists of protocols that allow multipoint routing amongst nodes using the data link layer. The transport layer consists of protocols that allow point to point connections over multiple links using the network layer. The application layer consists of services such as DNS. In a device following such an approach, any upper layer cannot communicate with another device if its lower layers cannot.

1.1 Description of a mobile ad hoc network

MANET is formed out of a cluster of nodes equipped with wireless communications and networking capability that are deployed without a predefined topology [1]. Each node can communicate with any other node within its' interfaces' radio range. To communicate with nodes beyond its radio range, it forwards packets to nodes within its radio range that have a path to the destination node. Hence, each node in the network potentially works as a router. Apart from physical layer connectivity, medium access control and multi-hop routing protocols, nodes also participate in services that are accessed by others.

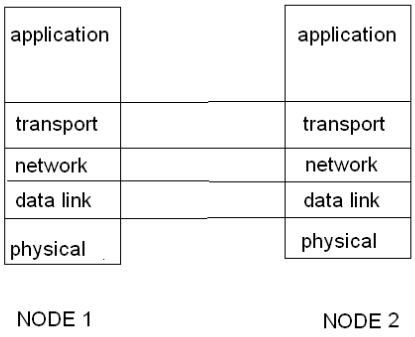

Figure 1.1: Layered model of communication

1.1.1 Modeled as a graph

In wireless Local Area Network (LAN)s, according to the standard, IEEE 802.11 [2] interfaces associated with the same Basic Service Set Identifier (BSSID) are part of the same network. In a MANET, the network can be represented as a graph G = (V,E), where physical interfaces are represented as vertices in V and existing links are represented as edges E [1]. According to the transmission range of each interface, the topology emerges. The graph is assumed to be connected and simple. This graphical representation of a MANET holds for a time instant. In this instant, the topology can be represented as a graph where the edges depend on a distance relation, i.e., if the distance between the vertices v_1 and v_2 is less than the range of each interface there exists a possible edge. An example graph is shown in Figure 1.2. Furthermore, due to the use of directional antennas, these links may also be asymmetric and modeled as directed edges. To note, for interfaces in the network that connect to a wired network, the edge between the wired interfaces need not be representative of physical distance between them.

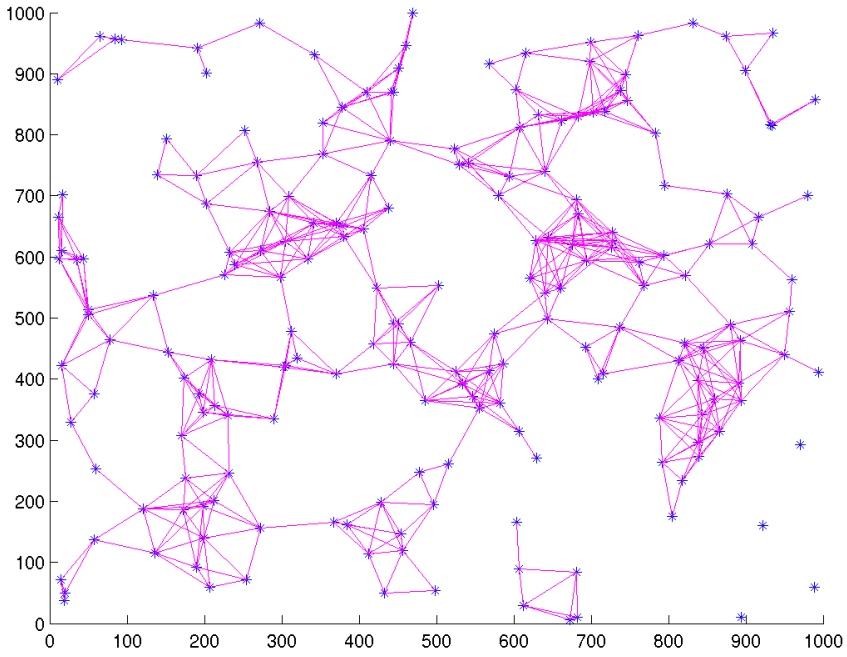

Figure 1.2: MANET

Devices participating in a MANET need not consist of interfaces of the same type [3]. If the devices participating in the MANET have more than one type of wireless access technology, the possible edges form an overlay graph where the vertices are partitioned with respect to their access technology. Vertices in V_1 have interfaces with range D_1, in V_2 with D_2, in V_3 with D_3, where V_1,V_2,$V_3 \subseteq$ V, $V_1 \bigcup V_2 \bigcup V_3$=V. The edge set is the union of all edges from each access technology type with the assumption that there are some nodes that have more than one interface with different access technologies that bridge between them.

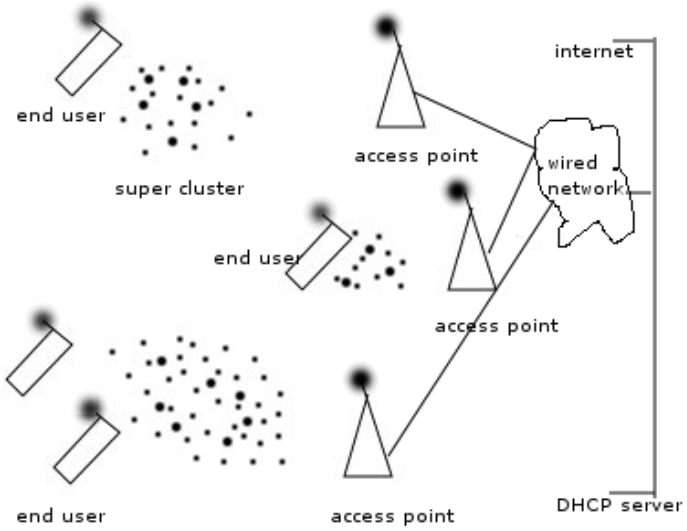

Figure 1.3: Wired Wireless Scenario

1.2 Gateways to wired networks

A MANET may also have nodes that connect it to the wired Internet [4].In the wired Internet, parts of the network are predetermined topologies, deployed and maintained by network administrators.

The above Figure 1.3 depicts a typical office environment deployment of wired and wireless LAN, where the configuration DHCP [5] server is part of the wired network.

1.3 Description of domain name system

1.3.1 How names are essential in the Internet

Most useful services such as web services, distributed databases, email, instant messaging, and gaming are not hosted or served from a fixed topological location(s). Change in entities' location results in its network address and routing path to change. In the Internet,

instead of discovering the topological location of a service every time it is required, an infrastructure is maintained for scalable access to translations of a service name to topological location, making service management an important framework of a network. It allows the logical name of an entity such as a service or a user to remain constant irrespective of its location in the network. The aim of such a system is if nodes are topologically reachable, as in the transport layer can establish a data connection, then it should be logically reachable, i.e., all services on nodes within this connected network, should be accessible to all others, using the service infrastructure. Any such scheme needs to adapt to failure of nodes and changes in underlying network conditions.

In such a service, functions should be able to lookup the current translation, update to change the translation, assign responsibility of the translation, allow new name servers to join, delete failed name servers, and maintain association amongst name servers, (also known as the naming scheme). Considering a few different ways in which the servers may be logically related to one another, such as Peer to Peer (p2p), multicast, replicated database, tree, each of these solutions may be categorized based on whether they make use of a hierarchical structure in the naming scheme/topological placement/search (routing) scheme. One structure is where all the clients and servers form a multicast topology, where each server consists of translations that are registered with it. This result in worst case *O(N)* lookups, i.e., time to search for a translation, where N is the number of servers. It may utilize topological hierarchy but not naming or search hierarchy. Another topology is a DHT. Here each server has a unique identifier and is responsible for entries that map to it. Query may take *O(logN)* * Diameter of the network. A DHT thus usually utilizes hierarchy for search only. Another topology is a centralized scheme where all the translations are

kept with one server. Querying in this setting is of *O(1)* * Diameter of the network since the server is well-known. Updating, i.e., reflecting changes is also of *O(1)*. DNS in its own form utilizes hierarchy in naming as well as search.

Example of such a service that is essential to a large number of applications in the World Wide Web (WWW) is the DNS [6], [7], [8].It is used to lookup the translation of a name to an IP address. DNS is a distributed database that maintains the current name to IP address translations of nodes in the Internet. In the Internet every machine that can host services has a hostname. This hostname's translation to IP address needs to be updated in its authoritative DNS server.

1.3.2 How the DNS works

DNS servers are arranged hierarchically as a multi-way tree with the root of the tree responsible for all records. An authoritative name server for a domain is responsible for all translations whose names end in this domain name. In order to manage unique names in a scalable manner, the names as well as the responsibility of resolving them are arranged in a hierarchical tree structure. A tree structure, for a given number of nodes maintains the minimum number of links. For n nodes, a tree has only *(n-1)* links. The root of the tree is responsible for resolving any query. Any leaf node can obtain the resolution of any node by traversing up the tree. In case it cannot reach some intermediate node, it can always start with the root node. This makes the root server a hotspot, if it collapses, the tree is invalidated.

This is not a problem in the mostly wired Internet since the root server is physically replicated on the Internet backbone and every machine is manually configured with the root

server's IP addresses. So, any temporary failure of an intermediate node would only make its sub tree unreachable. The rest of the tree would still be reachable from the root. Thus, in the wired Internet, there has not been a strong requirement for auto-configuration of the root servers themselves. Currently authoritative DNS servers themselves are statically configured. Update is done statically at each authoritative server. Each name server needs to maintain one link, i.e., to its root. The naming scheme and lookup mechanism are hierarchical. There is no automated assignment of responsibility of translation within the scheme. In order to be resilient to intermittent failures, it relies on replication of the database physically. DNS currently does not support any manner in which their name-IP address translations can be setup ad hoc. The root nodes, which are a constant, are responsible for all translations in the Internet. Searching, that is a very frequent operation, is highly efficient in this structure.

1.4 About peer to peer service protocols

Newer examples of scalable service management are peer to peer networks [9]. Both unstructured ones such as Gnutella, KaZaa, etc and structured models such as CAN, CHORD, Kademlia, Bamboo, SkipNets. The most notable application where this has been used is in file sharing. It is an effective manner to index current location of files or file parts and then access. It differs significantly from the DNS structure in that responsibility of the current translation is assigned dynamically depending upon the current membership in the structure. Further, joining and leaving from such structures need no manual configuration. The cost of such flexibility is in the maintenance of redundant linkages, and regular maintenance

messages. In structured peer to peer networks [9], current location of content is at a designated member of the ring that is searchable from any other member. For unstructured peer to peer networks [9], frequent network wide broadcasts may occur that may cause severe traffic congestion. Location of content is disseminated as the request for it increases. A two level hierarchy as in KaZaa allows unstructured peer to peer networks to reduce the amount of messages. Both structured as well as unstructured peer to peer networks have been successfully used in various applications. Most applications such as publish-subscribe systems, content distribution systems assume the underlying physical network to be connected.

Here, the DHT is a data structure used in structured peer to peer protocols. As against the use of flooding in unstructured peer to peer protocols, DHT's attempt to balance the load and require asymptotically less number of messages for search and maintenance. As described in [10], it is an important data structure for an architecture that manages services. The basic structure consists of a distributed circular linked list. For consistency, only the successor pointers need to be maintained correctly. But, for performance, extra linkages, i.e., fingers are maintained. These are maintained using a passive stabilization routine and are useful to reduce the search time to *O(log(n))*. Current DNS relies on hierarchy in the naming and search scheme, maintaining very few links. Multicast based solutions would rely on hierarchy in the topological sense. DHT or p2p solutions rely on a hierarchical search scheme. Similarly other variations of solutions may be classified.

1.4.1 Description of DHT based DNS

Chord [10] is a structured, DHT based peer to peer protocol. In this scheme of DNS servers, the names would hash into the server identifiers, allowing the name-resolution

responsibility of a node to change dynamically. Each node is analogous to an array slot in a hash table. Its index in the hash table is generated using a consistent hashing technique such as Secure Hash Algorithm (SHA)-1 [11]. DHTs are typically designed to scale to a large numbers of nodes and to handle continual node arrivals and failures. The invariant in Chord is as long as the successor pointer of each node is correct, the ring is stable. The join operation proceeds by requesting a current member of the ring to search for its correct successor in the ring. The passive stabilization routine updates pointers to repair incorrect pointers to predecessor nodes. The join, stabilize and finger update algorithm is described in [10]. In Chord, as long as each node points to its correct successor, all the nodes will eventually form a stable structure where all its members are reachable. The successor is then the node whose identifier is circularly greater than its own and is part of the logical network. Each node in addition to its correct successor also maintains links to *O(log(n))* other nodes in a predetermined manner that allows *O(log(n))* searching. Thus, in Chord, a node n's neighbor is successor(n) and it also forms linkages with successor($n+2^1$), successor($n+2^2$), ..., successor($n+2^k$), where k = *log(N)*,N= maximum number of nodes possible in the network. A node that attempts to join an existing Chord network need just find its successor and configure its logical link to it. As part of a passive stabilization routine, this node, sends a stabilize message to its successor, on receiving which, the successor sends a notify message to its current predecessor informing of its new successor, after which the node changes its predecessor to the new one. Periodically, a node that is part of the logical network, sends a stabilize message to its successor, the successor adjusts its predecessor pointer and sends back a notify message to its earlier predecessor if there is a change. This allows the predecessor to update its successor link as well.

Node	Entry at bit 4	Entry at bit 3	Entry at bit 2	Successor
0000	0001	0010	0100	1000
0001	0010	0011	0101	1001
0010	0011	0100	0110	1010
0011	0100	0101	0111	1011
0100	0101	0110	1000	1100
0101	0110	0111	1001	1101
0110	0111	1000	1010	1110
0111	1000	1001	1011	1111
1000	1001	1010	1100	0000
1001	1010	1011	1101	0001
1010	1011	1100	1110	0010
1011	1100	1101	1111	0011
1100	1101	1110	0000	0100
1101	1110	1111	0001	0101
1110	1111	0000	0010	0110
1111	0000	0001	0011	0111

Table 1.1: Routing Table

Apart from Chord, there are other DHT based protocols, such as Pastry [12] and Kademlia [13] that maintain their links differently. In Kademlia, each node n forms links with $(nXOR2^i)$ nodes where 0 < i < k and k=number of bits in the maximum size of the ring. Although this allows for choice of link based upon topology or other criteria, it can become less balanced than Chord when all the nodes are participating. The invariant in Kademlia is to have at least one member in each sub tree if such a member is present.

In order to compare a DHT based DNS as opposed to current DNS or Logical Name System (LNS), we attempt to find a mapping between them as structures. Here name servers would be the nodes of the structure. To demonstrate the different mappings of Chord as DNS, consider a network where k =4 and all the nodes exist. In this case, the routing table for each of the nodes is as in Table 1.1.

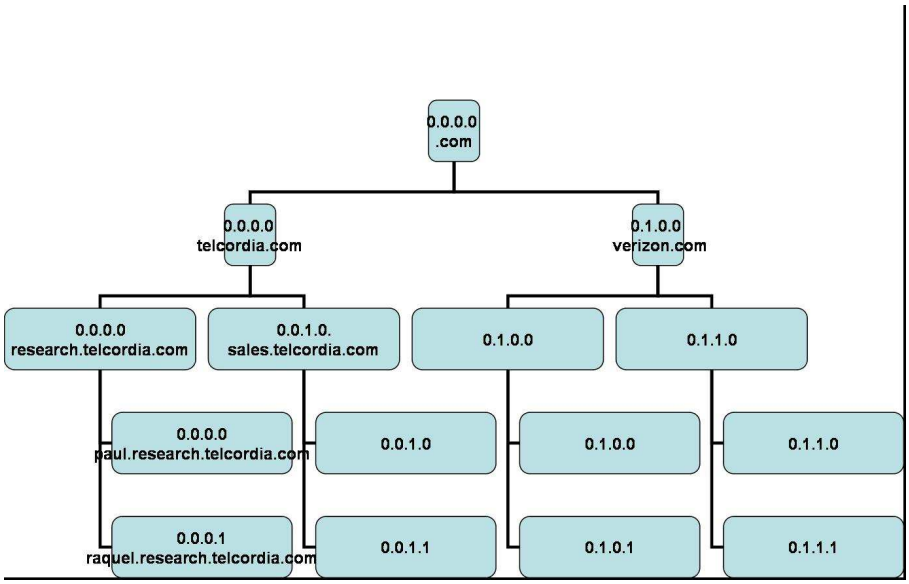

Figure 1.4: mapping Chord to DNS

In the above case, if multiple name servers share the same identifier, i.e., responsibility of maintaining the translations of different authoritative server lie with the same Chord node, an example map of part of the Chord structure is shown to map to a section of the DNS in Figure 1.4. The advantage of using a DHT based scheme over DNS is that in a DHT, the responsibility of the translation is automatically assigned depending on the current members of the logical structure. There is always a node that is currently responsible for a given translation. For an example, we assume that a number of servers can map to the same identifier. Such as *., .com, telcordia.com* all map to 0.0.0.0.

As opposed to Chord, the Pastry routing table where the size of the network is maximum 16, i.e., 4 bits per node identifier, is more flexible. The routing table for a full network would have choices for the entries as in Table 1.2

As against Chord, where each logical node has the same responsibility as all others, in the above, some nodes can be assigned more responsibility than others, even if all of them

Node	Entry at bit 4	Entry at bit 3	Entry at bit 2	Successor
0000	1XXX	01XX	001X	0001
0001	1XXX	01XX	001X	0000
0010	1XXX	01XX	000X	0011
0011	1XXX	01XX	000X	0010
0100	1XXX	00XX	011X	0101
0101	1XXX	00XX	011X	0100
0110	1XXX	00XX	010X	0111
0111	1XXX	00XX	010X	0110
1000	0XXX	11XX	101X	1001
1001	0XXX	11XX	101X	1000
1010	0XXX	11XX	100X	1011
1011	0XXX	11XX	100X	1010
1100	0XXX	10XX	111X	1101
1101	0XXX	10XX	111X	1100
1110	0XXX	10XX	110X	1111
1111	0XXX	10XX	110X	1110

Table 1.2: Options for Routing Table
X here denotes a 0 or 1.

are present in the ring at the same time. For example, an instance of the table can be as in Table 1.3

From the sample routing table, looking at the number of incoming links at each node, we observe 0000 has 15 nodes linking to it, that is, node 0000 occurs in the routing table of 15 other nodes. Node 1111 on the other hand has 1 incoming link. Table 1.4 shows the number of incoming links for each node.

As a complete structure, this is similar to a DNS tree with either 0000 or 1000 as the root node.

1.5 Address resolution protocol

In wired networks, Address Resolution Protocol (ARP) [14] is used to aid in routing as well as IP address assignment. An Ethernet LAN segment can be viewed as a bus

Node	Entry at bit 4	Entry at bit 3	Entry at bit 2	Successor
0000	1000	0100	0010	0001
0001	1000	0100	0010	0000
0010	1000	0100	0000	0011
0011	1000	0100	0000	0010
0100	1000	0000	0110	0101
0101	1000	0000	0110	0100
0110	1000	0000	0100	0111
0111	1000	0000	0100	0110
1000	0000	1100	1010	1001
1001	0000	1100	1010	1000
1010	0000	1100	1000	1011
1011	0000	1100	1000	1010
1100	0000	1000	1110	1101
1101	0000	1000	1110	1100
1110	0000	1000	1100	1111
1111	0000	1000	1100	1110

Table 1.3: Sample Routing Table

Node	Number of Incoming links
0001	1
0010	3
0011	1
0100	7
0101	1
0110	3
0111	1
1000	15
1001	1
1010	3
1011	1
1100	7
1101	1
1110	3
1111	1

Table 1.4: Number of incoming links

where a host can broadcast to other hosts on it and every such host can rebroadcast to all those hosts. For the purpose of address resolution, the ARP request packet has destination MAC layer address FF:FF:FF:FF:FF:FF (broadcast address), its own MAC address as the host address, host IP address and destination IP address. This broadcast reaches all hosts connected by protocol stack layer 2 and below elements (such as bridges and switches as long as they allow broadcasts to propagate). If there is any host with the destination IP address, it replies.

1.6 Dynamic host configuration protocol

These resolutions are cached in order to reduce the number of broadcast messages. ARP allows auto-configuration. The ARP request for a node requesting to be assigned its IP address, broadcasts with its host MAC address, destination MAC address set to the broadcast address, host IP address set to 0.0.0.0 (, i.e., this host IP address), destination host address set to 255.255.255.255 (which is the broadcast IP address, unless its ARP cache has adverts from DHCP servers). This request is also known as a DHCP request. It is sent as a UDP packet. This ARP request propagates until it reaches a protocol stack layer 3 element such as a router. In this stretch either it encounters a DHCP relay agent or a DHCP server [5]. A DHCP relay agent unicasts the request to a DHCP server which replies back with an IP address assignment and configuration information. A ARP reply with requestors MAC address and IP address as destination and server's MAC address and IP address is rebroadcast and this updates the requestor's ARP cache and assigns it its IP address. Duplicate DHCP servers for the same network segment can exist as long as either their

address pools are non-overlapping or they ping the IP address before assigning to discover any conflicts and avoid duplicates , i.e., perform Duplicate Address Detection (DAD).

Chapter 2

Need for auto-configured DNS and auto-configuration

2.1 Network partitions and mergers in MANET

We note in this section how the underlying network is easily partitioned in a MANET leading to the case of the authoritative servers being unreachable and physically connected nodes being logically unreachable:

In the graph G=(V,E), representing a MANET, a link is sensitive to change in channel conditions, changes in the surrounding environment such as short term fading effects, high error rate or excessive collision as well as mobility of the devices. Thus, not all possible neighbors by distance may be reachable at a particular point in time. Also, apart from the change in possible neighbor relations of a node due to mobility, the above reasons result in the graph representing the network to change with time.

In MANET, a link failure may occur due to mobility. An interface that has moved out of the radio range of another node dissolves the link between them. The resulting link failures between interfaces that have no other redundant paths, lead to network partition. An interface in a MANET is not only responsible for connectivity between its direct neighbors but also for providing paths for other nodes in the network.

Apart from the environmental conditions that affect a MANET network nodes that are currently associated with one network may decide to leave and join another geographically collocated network and thus cause a change in the topology as well. Geographically close nodes need not participate in the same network. For example, in a city wide MANET created out of users that voluntarily participate in it, as nodes move, change the network

they are associated with or switch off, temporary holes in the network may emerge, i.e., parts of the network may get isolated from each other due to such dissolution of links. Further, they may merge back together again as links are formed between interfaces in different partitions. This phenomenon may occur frequently. Hence, not all devices that exist with wireless access technologies may participate in a given network. There may also be devices with interfaces that participate in different networks thus bridging them. At different points of time, even if the nodes do not move, on changing the network it is associated with, changes the graph.

As we note from the above changes, some of the links between nodes may be lost from time t1 to time t2, where t2 > t1. If these deleted edges in E cause the formation of disjoint components, the MANET network has partitioned. This results in nodes belonging to the same network but in different components not being able to communicate with each other. This new graph G' = (V, E) has at least two disjoint components. Similarly, if edges are added to the graph G' to form G", i.e., it goes from G' to G", where G" either does not have more than one component or has re-organized into different components, a network merger has occurred between the disjoint components. For example, take a linearly arranged nodes a, b, c, d, e, f and g, i.e., a-b-c-d-e-f-g. If the link between c and d, and e and f breaks, the resulting graph has three disjoint parts, a-b-c, d-e and f-g. If further, there is a new link formed between c and f, the resulting graph re-organizes to two disjoint components a-b-c-f-g and d-e.

A break in links does not necessarily mean a loss of connectivity. Due to the number of extra links, there may be more than one path between a pair of nodes, which may be utilized when a link breaks on a path being used between two nodes. To note, we make the

assumption that detecting a link break as against congestion or collision is assisted by the MAC layer. We do not assume that a path break can be easily detected.

2.2 Network partitions in other networks

Although infrequent, network wide disruption of infrastructure in wired networks arise in the event of a disaster due to either natural calamities such as tsunamis, volcanoes, hurricanes, earthquakes or man-made such as terrorist attacks, wars. They result in physical faults in the connecting media leading to link failure. This may lead to a partitioning of networks if there are no other paths connecting the two entities. An example is the failing of a fiber-optic cable connecting two routers in different cities for example like in [15], [16]. It is difficult to bring up the communication infrastructure quickly, manually. In a wireless network, it may obliterate a number of interfaces, thereby removing links associated with it that could lead to potential partitions. Damage that is local to the point of disaster, affects far ends of the service infrastructure because of its hierarchical arrangement.

In an infrastructure based wireless network, such as 802.11b or cellular networks, the link failures occur due to mobility of nodes from one infrastructure point to another. Since, these nodes act as hosts and do not serve other nodes, only these hosts are affected by mobility. Thus, when they form a new link with another infrastructure point, they require re-configuration.

Changing the static configuration in clients of the root DNS servers also results in partitioning of the service architecture [17].

2.3 Effect of underlying network partition and merger on DNS

In a given network, the topological links are from one interface in a network to another, while logical links are the topological configuration maintained by logical entities according to their logical relations. Loss of topological links due to failures in the underlying physical connections may lead to loss of some logical links as well. These topological partitions may or may not be recovered for a long time afterwards. They may behave as distinct entities, able to support networking services within their partition. Changes such as addition and deletion of interfaces or links may occur in each of them, while they are partitioned. After a period of time, when these stabilized partitions merge, services from one partition may not be accessible from the other. This could create a partition in the service management framework although the underlying physical network would have healed.

Currently DNS servers are statically configured. In order to be resilient to intermittent failures, it relies on replication of the translations it is responsible for, physically. This solution does not provide robustness by the DNS architecture itself towards network partitions. For example, a large MANET with two different domains breaks down into two partitions each consisting of nodes from each of the domains. In this case, although the MANET can reconfigure itself to setup lower layer communication between the nodes, DNS currently does not support any manner in which their name to IP address translations can be setup in an ad hoc manner. Figure 2.4 depicts the partition in services even when the underlying network partitions have regained communication within.

In the event of a disaster, either natural or man-made, infrastructures are disrupted. Damage that is usually local to the point of disaster is propagated to the far ends of the service infrastructure. Hierarchy and caching improve performance in a stable infrastructure

but do not exist in a disaster scenario. This makes ad hoc auto-configuration solutions relevant.

2.3.1 Root of the DNS as a hotspot

In DNS, the name to IP address resolution responsibility is hierarchically distributed. The root server, i.e., *"."*, is the only one responsible for all the translations. Its IP addresses are constant. All clients are configured with the 13 root server's IP address. If all of the 13 root servers' IP address change, the root server becomes logically unreachable. This creates a single point of failure. DNS roots exist in practice because 13 root servers by design are guaranteed to be a constant for the internet. The following example describes how a network partition and merger would lead to the above assumption to invalidate.

The Figure 2.1 represents a section of the DNS as it exists today. Each of the nodes' *iisc.ernet.in, iitg.ernet.in, cbse.nic.in, eng.auburn.edu* is a host. Each host is associated with an authoritative DNS server which is responsible for all the translations of its domain zone that includes hosts, child DNS name servers. *ernet.in* is the authoritative server for *iisc.ernet.in* and *iitg.ernet.in. nic.in* is the authoritative server for *cbse.nic.in. auburn.edu* is the authoritative server for *eng.auburn.edu. in* is the top level domain server for *ernet.in* and *nic.in. edu* is the top level domain server for *auburn.edu. "."* Is the root server for *edu* and *in.*

In the section of DNS, assume that a disaster causes the underlying network to partition as in Figure 2.1, at time t=0, causing *.edu.* to be inaccessible from all hosts named **.edu.*. Thus, none of the translations for **.edu.* can be accessed by those in the other domain such as **.in*, even if they can reach the root server. Within the *.edu* domain, hosts with

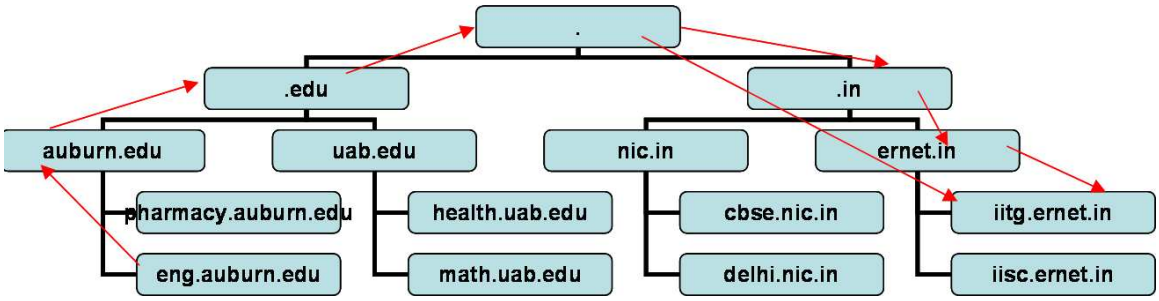

Figure 2.1: Section of the DNS before the partition at time t=0-

the authoritative server *auburn.edu* shall now be able to resolve queries for those within *auburn.edu* but not of other **.edu*, since the server containing the correct translation for authoritative servers for those zones is no longer reachable. The correct name to IP address translations are listed in Table 2.1, using nslookup [18].

The above example describes a situation where the partitioning of the underlying network is not on organizational boundaries that are arranged hierarchically in DNS, resulting in hosts not being able to obtain translations from one broken domain to another. The existing logical structure is not connected although there is physical connectivity. This would inevitably lead to service discontinuities due to the partitions formed. It is likely that the IP address for the corresponding names would have changed and earlier transactions or sessions would break and not resume.

In order to repair the logical structure to resume name resolution within the partition, a new node within the partition would need to become the *.edu* server. Without the original translation table of *.edu.*, the list of **.edu.* names is unknown and so are their translations. It would be difficult for any one **.edu* to take over its responsibilities. This is because all the **.edu* do not know of each other's existence. So, for any one of them to take over the function of the root, does not guarantee that all the **.edu* nodes will be able to modify their

Name	IP Addresses
.	198.41.0.4, 128.63.2.53, ..
.edu	192.41.162.32,..
.in	204.152.184.64, 193.0.0.193,..
uab.edu	138.26.1.2, 207.230.75.50,
health.uab.edu	138.26.153.226
math.uab.edu	138.26.76.18, 207.230.75.50,..
nic.in	164.100.3.1
cbse.nic.in	164.100.52.226,164.100.52.227
delhi.nic.in	164.100.52.74
auburn.edu	131.204.2.251
ernet.in	202.41.97.61
iitg.ernet.in	202.41.110.33
iisc.ernet.in	144.16.64.3
eng.auburn.edu	131.204.110.158
pharmacy.auburn.edu	131.204.250.29

Table 2.1: Name To IP Translations at t=0-

links to the new root and consequently communicate with **.in* nodes. Multiple partitions within **.edu*, aggravate the problem further.

Automatic repair for this logical structure may be possible if there were more logical links maintained. For example if *auburn.edu* had a logical link to *.in*, all the translations for **.in* could be reached from the nodes **.auburn.edu* even without the existence of *"."*.Figure 2.2 depicts the logical links after the merger and a fictitious change in name to IP translations in Table 2.2.

Thus, if an intermediate authoritative server becomes unreachable, there is no way for its child servers to access services from the other parts of the logical tree. Healing a partition depends on whether there are redundant links to different parts of the logical structure that can be accessed in order to access information of parts of this tree. In DNS, since such links are hierarchical in nature, it is simple to say that the loss of the root is an essential loss of all translation. Alternatively, these links can be discovered, but a broadcast discovery at

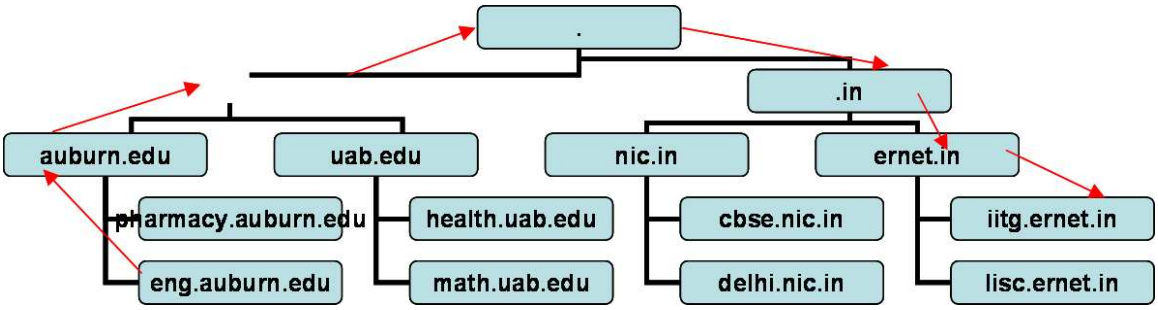

Figure 2.2: Section of DNS after network merger at t=0+

every network partition is an expensive solution in terms of the number of messages and time taken to repair.

In IPv4 [19] networks, there are not enough IP addresses anyway that can be assigned to every interface for its entire lifetime. Hence, translations of name to IP addresses for services on them will change with time. Since the root node is an integral non-changing parameter in the internet, its translation is fixed always and is a constant static configuration in all hosts. In partitioned networks though, name-services can fail especially if there are nodes from two different domains communicating. To note, that one DNS name may translate to more than one IP address that exist in different organizational and geographical locations.

2.3.2 Partitions and mergers in MANET

Figure 2.3 depicts two ad hoc network partitions merging. When they merge together, apart from by the MAC and routing protocols, in such a changing network, nodes may host services that are accessed by other nodes. Services from the combination of nodes should be available to each node. To continue accessibility of services, configuration is made difficult as a central authority within the network cannot always be trusted to be reachable. If

Name	IP Addresses
.	168.3.4.5, 124.3.0.123..
.edu	192.41.162.32,..
.in	178.2.3.19
uab.edu	138.26.1.2, 207.230.75.50,
health.uab.edu	138.26.153.226
math.uab.edu	138.26.76.18, 207.230.75.50,
nic.in	164.100.3.1
cbse.nic.in	164.100.52.226,164.100.52.227
delhi.nic.in	164.100.52.74
auburn.edu	131.204.2.249
ernet.in	202.41.96.11
iitg.ernet.in	202.42.24.12
iisc.ernet.in	144.16.64.3
eng.auburn.edu	131.204.110.158
pharmacy.auburn.edu	131.204.250.29

Table 2.2: Name to IP translations at t=0+

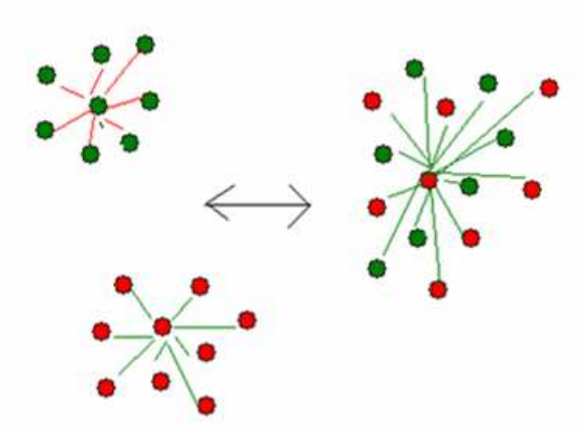

Figure 2.3: Ad hoc Partition and Merge

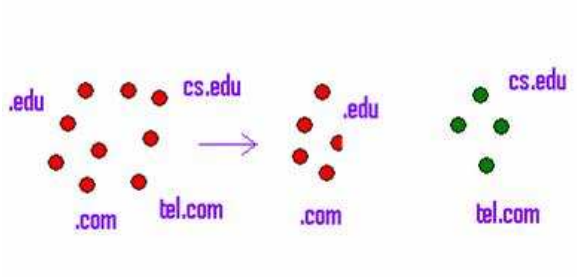

Figure 2.4: Authoritative servers unreachable in MANET

each partition ran their local DNS with local root, in the merged case as neither of them would have translations for the merged nodes. In Figure 2.4, if *.edu* was responsible for all translations ending in it and similarly *.com*, in the partitioned case, services by *cs.edu* would not be accessible to *tel.com* and vice versa, even though they are physically reachable.

In Figure 2.5 above, consider that the name servers form a Chord [10] ring. Consider the entire network as a large group of wireless nodes. The nodes marked 0-9 are the Chord name servers as shown in Figure 2.6. Each of the LNS name server's consists of a topological database of translations of the nodes it serves. The logical association amongst the LNS servers is a ring formed by the hash of its node name. The LNS server that is logically responsible server for a name is the successor(hash(name)). The collision-resistant hash function is considered to be the same for all nodes. As an example, let hash (*eng.auburn.edu*) = 2035, hash (*auburn.edu*) = 900. The reachable name server whose identifier is closest to 900 and greater than or equal to it contains the name to IP mapping of *auburn.edu*.

Consider an example case where Figure 2.5 depicts the topological links between nodes. Let hash($m.n.edu$)=9, hash($a.b.com$)=0, hash($p.b.com$)=1, hash($o.n.edu$)=2, etc as depicted in Figure 2.6. Logical links are formed over the underlying network, between reachable nodes that form the network, where each node maintains *O(log(n))* links. For

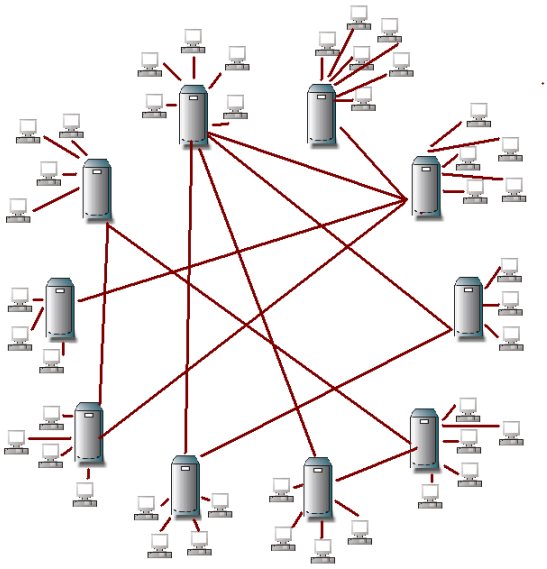

Figure 2.5: Physical links between nodes and LNS servers

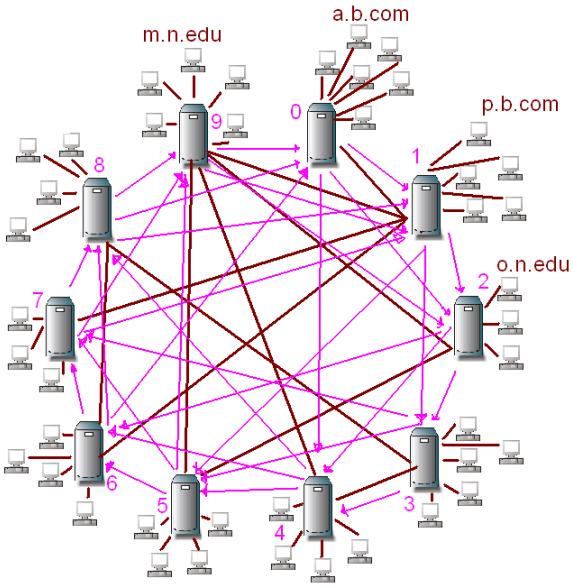

Figure 2.6: Chord logical structure amongst LNS servers

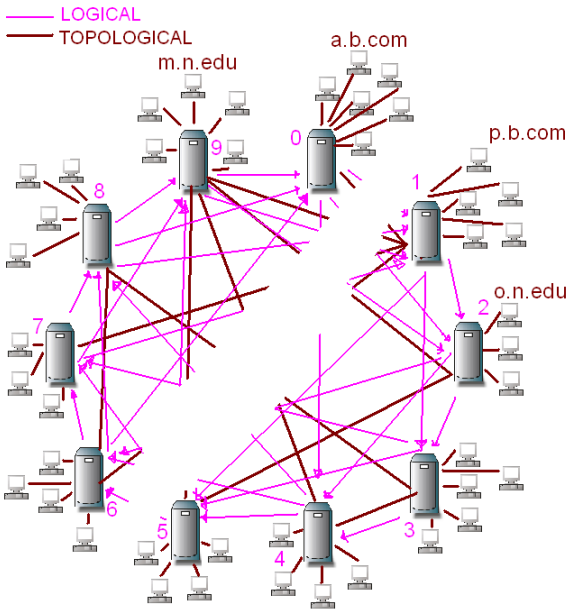

Figure 2.7: Break in underlying network

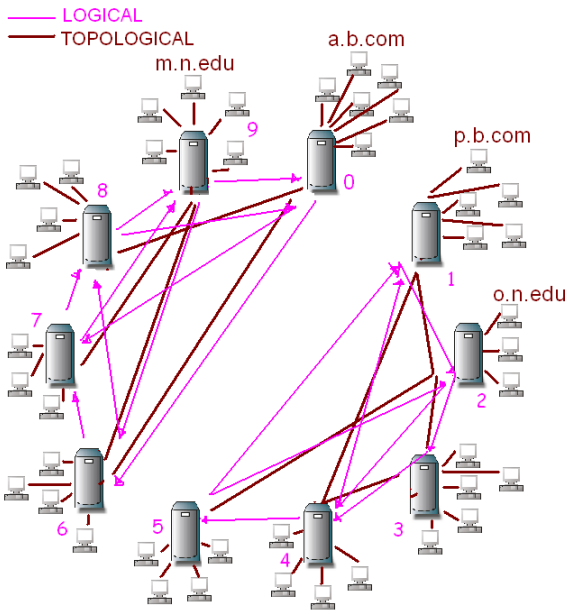

Figure 2.8: Stabilized logical links in partitions

Node	Successor	2nd Successor	4th Successor	8th Successor
0	1	2	4	8
1	2	3	5	9
2	3	4	6	0
3	4	5	7	0
4	5	6	8	0
5	6	7	9	0
6	7	8	0	0
7	8	9	0	0
8	9	0	0	0
9	0	0	0	1

Table 2.3: Before Breaking

example, there is a topological link between 3 and 8, 8 and 6, consequently a path from 3 to 6, i.e., 3 and 6 are reachable from each other through 8. Now, assume that an event causes breaking of the number of topological links, breaking quite a few of the logical ones as well as shown in Figure 2.7. In this event, although a number of the topological links were broken, the logical network is partitioned into two pieces. In these two partitions, nodes whose logical links are lost attempt to re-stabilize the structure and result in discovering each other as shown in Figure 2.8. The following is the description of the breaking and a possible manner of rejoining of links:

Before partitioning, the LNS servers' maintained logical links as shown in Table 2.3 where incoming links at node-from node are 0-9, 9-8, 8-7, 7-6, 6-5, 5-4, 4-3, 3-2, 2-1 and 1-0.

After the partition, the logical links are depicted in Table 2.4 where X is used to denote a null entry. Incoming links at node-from node are 0-9, 9-8, 8-7, 7-6, 5-4, 4-3, 3-2, and 2-1. Logical links would be detected to be broken since they timeout, when pinged or used, although a timeout on a single logical link can mean either that the two nodes are

Node	Successor	2nd Successor	4th Successor	8th Successor
0	X	X	X	8
1	2	3	5	X
2	3	4	X	X
3	4	5	X	X
4	5	X	X	X
5	X	X	X	X
6	7	8	0	0
7	8	9	0	0
8	9	0	0	0
9	0	0	0	X

Table 2.4: Just After Breaking

on different partitions or that there is a lot of congestion on the path connecting the two nodes.

Due to the partition, 0 and 5 have lost their successors resulting in an inconsistent ring. Amongst 0's logical link, is a link to 8, to which it can update as its new successor, which due to the stabilization routine will eventually change to 6. All of 5's logical links are lost. 5 would be able to discover 1 as its successor if all the nodes had all their predecessor links as well. For links that have broken, to discover the right member that returns the logical structure to a consistent state would take time. The existing links that have not broken will not need to be readjusted unless there is a simultaneous change in the membership of the logical structure. The resulting logical links as the partitions stabilize is depicted in Table 2.5 where incoming links at node-from node are 0-9, 9-8, 8-7, 7-6, 6-0, 5-4, 4-3, 3-2, 2-1 and 1-5.

Consider that a merger event occurs after this, where the underlying network has repaired itself. On merging, detecting that a merger has occurred is difficult since there aren't any logical links between the partitions. Assuming that the two partitions do discover

Node	Successor	2nd Successor	4th Successor	8th Successor
0	6	6	6	8
1	2	3	5	1
2	3	4	1	1
3	4	5	1	1
4	5	1	1	1
5	1	1	1	1
6	7	8	0	0
7	8	9	0	0
8	9	0	0	0
9	0	0	0	6

Table 2.5: Broken Stabilized

each other, the logical links would be in the same state as that right after the partition event, i.e., as in Table 2.4, where incoming links at node-from node are 0-9, 9-8, 8-7, 7-6, 6-0, 5-4, 4-3, 3-2, 2-1 and 1-5. As this structure stabilizes itself, it would eventually return to its original stable state as depicted in Table 2.3

2.3.3 Importance of extra links for DNS in MANET

In large ad hoc networks, a single name server may not be sufficient to serve the entire network. For nodes in the largely wired Internet, to access services hosted by nodes in the MANET, IP addresses and DNS need to work in MANET as well. As ad hoc networks partition and merge, name servers within each partition need some manner in which they can update the translation of reachable names. This is to ensure that physically reachable nodes are able to access each other's services in the condition shown in Figure 2.4. Currently, there is no automatic detection of network layer failures and delegation of responsibility of translations to other nodes.

Attempting to get rid of the hierarchy in responsibility of who has the name gets rid of a single point of failure, the root name server. Redundancy in the associations amongst

name servers (i.e., links between them) than physical redundancy leads to robustness. It is self evident that extra linkages are required to make the structure more robust than a tree. Extra links help rebuild the structure. How these linkages are arranged would affect how easy it is to recover from a partition. Backwards compatibility of a solution to current DNS is also desired. To add these extra linkages in a manner that they can be maintained with least cost, in the event of network partitions, explored is the possibility of mapping Chord [10] to DNS. The first step would be to convert names in current DNS to numbers. Hashing functions can do the job to convert names to numbers and then numbers can be compared. As against names, these numbers have fixed length. Collision resistant hash functions with sufficiently large key space allow that each logical name be mapped to a unique number. The ease in creating and maintaining links dynamically in DHT's is because you can compare two numbers and find out which way you need to search. Two arbitrary names cannot be compared in this manner. This predefines a position for the responsibility of resolution that need not force hierarchical naming, and allow transfer of responsibility based upon how the links are maintained. The logical links that a DHT maintains is independent of hierarchy. It maintains *O(log(n))* links on an average, where n is the number of nodes in the network. The logical links that a DNS maintains depends on hierarchy. A DNS authoritative server maintains links to all its child authoritative servers. Most DNS servers are deployed in an organizational manner, i.e., an organization usually has a DNS server(s) of its own that is registered to an external parent server(s). In the mobile ad hoc network case, such kind of servers would not be useful, since a partition in mobile ad hoc networks has no correlation to organizational boundaries. Thus the hierarchical structure does not help.

In terms of membership, DHT has dynamic membership, while DNS has manual membership. A DNS node, i.e., an authoritative server can be deleted and added. To add a node in a DHT, any existing node in the DHT can be contacted and sent a join request to. For a node to delete in a DHT, it can search its predecessor and update its predecessor's successor. Thus, before a scheduled delete, it can also transfer the data that it was responsible for, to its successor. To add an authoritative DNS server requires manual registration with its parent DNS server. To delete an authoritative DNS server requires that the entry for it in the parent server be deleted. This causes the authoritative DNS server and all its children to be unreachable from the rest of the network, as well as, all the translations for them to be lost. In order to service the translations in the deleted DNS server, entries would require to be added to the parent server. DNS clients can send queries to any DNS server.

One of the salient points of using a DHT over the conventional DNS is that, ordering numbers is easier than ordering names. Thus, the concept of distance and direction are difficult using names in DNS. Comparing *eng.auburn.edu* with *auburn.edu* and *us.gov* tells us that *auburn.edu* is closer to it than *us.gov*. It does not say whether for *google.com* translation sending the search request to *auburn.edu* or *us.gov* would be better. If instead of names they were replaced by numbers where the range of numbers is fixed, we can measure the distance between 0 and 19 as well as say that 4 is closer to 0 than it is to 19 and thus to search for 4, we can send the query to 0 than 19.

From the above, we can see that a MANET with a large number of nodes would be a dynamically changing network that needs to be self-organizing and autonomous in order to

scalably maintain its configuration information. Apart from basic connectivity and communication as offered by the MAC and routing protocols, nodes will also be participating in services that are accessed by others. To assure accessibility of these services, tree-structured configuration is made difficult as a central authority within the network cannot always be trusted to be reachable. Services should be available to all nodes that are part of the network and are reachable. In order to withstand disasters, auto-configuration capability is required. This necessitates the existence of a higher number of linkages with the associated databases than the tree structure in DNS. The redundancy here is not in terms of the replication in databases, rather it is the redundancy due to linkages maintained in the logical network of DNS servers.

The research challenges in making a DNS for MANET are thus, scalability in the order of the number of messages that are floating around needed to sustain the proper functioning of the structure, robustness that is to be able to structure consistent even in the case of arbitrary failures, discovery that is to be able to discover services on all nodes that is reachable. A correct solution is one, where the name of every physically reachable node is found. An efficient solution is one which has the least cost and time to maintain as is robust to partitions.

2.4 Auto-configuration

In the event of disruption of communication infrastructure, the location that maintains the mapping between logical and topological entities might not be reachable any longer, while the topology has changed. Thus, auto-configuration is the natural first step to discover

the same. To support the service management framework, the location or path to the participating management servers needs to be configured in these nodes.

2.4.1 Address resolution protocol (ARP) in MANET

The address resolution protocol was introduced to be able to translate between IP addresses and device address on the Ethernet bus. It allows the address used for networking to not depend on the manufacturers address, making adding and removing network interfaces and dynamic routing possible. ARP can be used successfully in situations where the membership of a network interface to other interfaces is constant, i.e., on the link that the network interface participates, the other interfaces are also participating on that link. In MANET, since every node is potentially a router, it participates in more than one link and thus in the multi-hop case the neighbors of neighbors need not be the same. This makes it difficult to use ARP for resolution. It leads to nodes two hops apart to think they are on the same link, while on subsequent ARPing they cannot resolve each other as they are more than one hop away [20].

In multi-hop wireless networks, each interface in the MANET is part of the same network. Each Interface (IF) is a potential router. This may lead to the situation where IF0–r–IF1–r–IF2,where r=maximum radio range for the interface. In this scenario, a one hop broadcast from IF1 reaches IF0 and IF2, but a one hop broadcast from IF0 does not reach IF2. This makes it difficult to define a bus like segment as in Ethernet. If the DHCP server resides at IF1, its broadcast is heard by both IF0 and IF2, but are they considered being on the same link? DHCP server on IF0 can be easily seen to have IF1 on its broadcast reach and naturally on its link. Now is IF2 part of a different link from IF1 even though it

can hear IF1's broadcasts and not IF0's or is it on the same link? One solution to resolve this situation is to have DRCP relay agents running on each node that discover the route to the DHCP server and cache it.

Chapter 3

Previous Work

3.1 Other naming solutions

There has been work towards making DNS more robust. Collaborative Domain Name System (CoDNS) [21] is a scheme where in to increase robustness of a name server, its database is physically replicated as a peer-to-peer network. The effect of churn, that is, the cycle of nodes joining and leaving the DHT, on performance of DHT has been studied in [22], but the effect of churn is significantly different from network partitions as are traditional solutions used for failure resilience and load balancing such as physical server redundancy. In Dynamic Domain Name System (DDNS) [23] the DNS UPDATE message can be used to update translations to an authoritative DNS server. Session Initiation Protocol (SIP) [24] is an elaborate protocol that allows application layer routing based on logical names, with the help of service registration, location servers, etc. These schemes address robustness in the case of node failures and light mobility.

A solution that does concern with change in IP addresses is the Unmanaged Internet Protocol (UIP) [25] and [26]. It proposes the use of Kademlia like routing substrate over IP that works around network partitions and mobility issues. Recent work in UIP has proposed a non-hierarchical DNS like scheme for handling mergers and partitions as described in [26]. Hence, there has been no extensive study into the possibility of incorporating DNS for dynamic ad hoc network topology that results due to disasters or mobility, while it is an essential service for applications.

Skipnet [27], [28] is a proposed scheme using skip graphs [27] to effectively address the problem of DNS unavailability on disconnects at organizational boundaries. It supports simultaneous use of multiple DHTs guaranteeing that message routes traverse only intermediate nodes sharing the same name prefix as do source and destination nodes. It elaborates failure recovery algorithms describing how probabilistic skiplists are used to route by name and numeric Identifier (ID). In each organization segment, a well-known root node is present that forms part of the alternate means of discovery. The assumption here is that partitions occur at organizational boundaries. On organizational disconnect, Skipnets make the assumption that well-known nodes will exist in each disconnected segment that aid in discovery of partitions. In a MANET, there need not be a correlation between a node name and its topological location making it difficult to take advantage of name ID routing.

LNS as proposed in [29], [30], [31], [32], [33] is a naming scheme for large ad hoc networks. A group of LNS servers serve in a MANET. Each LNS server has a topological, logical and home database. The topological database consists of all translations of nodes registered with it. The logical database consists of translations of LNS servers responsible for a set of logical groups. The home database contains translations of nodes' home server. In Figure 3.1 LNS servers form a multicast group.

Every isolated network contains at least one server responsible for containing the name-IP address translation. It broadcasts periodic beacons for nodes to join and register with it. For nodes that are registered within a LNS server, irrespective of their "domain", can access translations of all nodes reachable and registered in the LNS server in its topological database. It is not backwards compatible with DNS.

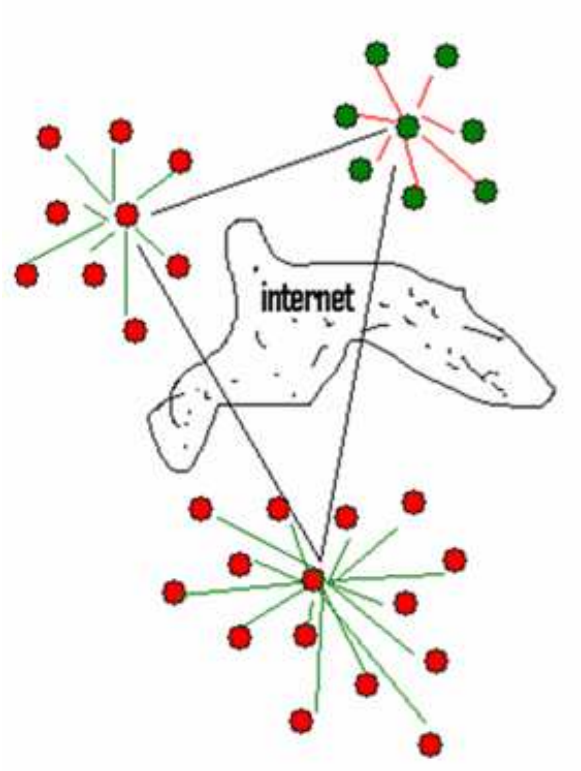

Figure 3.1: LNS servers logically associated as a multicast group in a MANET

The technical report by Xiaozhou Li [34] discusses concurrent management of a ring. It details that a bidirectional ring is required for a ring to maintain consistency in the event of concurrent joins and fails. It mentions an active join and leave algorithm as opposed to Chords passive stabilization method as described in [34]. His more detailed work, compiled as RANCH [35], allows for a locality aware, efficient, concurrent joins and leaves, structured p2p protocol. In the case, where the bi-ring need not be maintained in an ordered manner, performance might be better.

3.2 Auto-configuration in MANET

Different techniques have been proposed for auto-configuration of IP addresses in MANET. One approach is assignment of globally unique conflict free addresses to each network interface, and the other is assignment of addresses with DAD thereafter [4]. For conflict free assignment, in order to reclaim already assigned addresses that are no longer in use, the lease timer allows, for addresses that have not been reallocated to be reclaimed. Schemes that use conflict free mechanisms are Prophet [36] and DRCP [37], [38], [39]. DRCP assumes nodes have formed into IP links. In the single-hop wireless LAN scenario, this is indeed the case, where the DRCP server may reside on the Access Point and any broadcast by the interfaces associated with the Access Point will reach it and vice versa. This allows the Access Point to manage the IP address assignments and DNS server configuration.

3.3 Comparing possible solutions

In order to compare performance, with solutions that reach full logical reachability of physically reachable nodes, the percentage of the network reachable and the time to end in a fully connected network, are relevant. On the other hand, if solutions asymptotically aim parts of the network to be logically connected, and may not result in a fully connected network, then staleness, i.e., the number of incorrect translations due to mobility and failures is a useful measure. In order to measure staleness, the lookup time that nodes take for correct and incorrect translations can be noted. The rate at which updates in the translation tables occur, number of servers looked up on an average to resolve a query and the traffic overhead due to maintenance of the structure.

Chapter 4

J-SIM simulation study of DNS for MANET

Having identified the problem of existing name resolution service architectures not being able to recover from network partitions and mergers, the following protocol is simulated and studied.

4.1 Beaconing Chord based LNS

A node participating in this protocol is in either UNJOINED, JOINED, DELETE or REVERSE state. A node in the UNJOINED state is not participating in the logical network, although it can cache topological beacons to update its *contact_id*. When a node is interested in participating, it sends out a search query to a node already participating in the network, such as its *contact_id*. In the UNJOINED state, a node does not respond to any search or join requests that it may erroneously receive. When a node's first search request, which is also its join request, is replied to with the address of the current successor, the node changes state from UNJOINED to JOINED. In the JOINED state it participates in the logical structure. When a node in the JOINED state loses its link to its successor, the structure becomes inconsistent. The node then uses its reverse links, i.e., its predecessor pointers, to find the point in the ring that is its new successor. Thus, when a ping to its successor times out, the node enters the REVERSE state and sends a search query to its predecessor, listening to replies until TIMEOUT. Out of the replies received from different nodes, the reply from the node that is farthest from it is updated as the new successor, making the recovery robust. It is possible that a node does not receive any

replies whatsoever. In that case, the node assumes that it is isolated, and makes its own ring. In the DELETE state, a node is attempting to leave the logical network voluntarily. In this state too it does not reply to any search queries.

```
Node enters network {
  State = UNJOINED
Listens for a topological beacon
If (no beacon)
  Then (becomes MASTER), state = JOINED
    If (MASTER)
      Then (send topological beacon every T1 seconds)

Else
  Beacon rcvd
  JOIN (MASTER)
}

JOIN (MASTER)
  Send JOIN request to MASTER's IP address

RECIEVE (JOIN) from NEW_ID
  SEARCH (NEW_ID)

SEARCH (ID)
```

```
  If (ID in my Range then reply)
    Else send Search (ID) to successor's IP address

On network merge {
  Hear topological beacon from ID,
  If (MASTER) (if Own_ID>ID become SLAVE and JOIN (ID))
}
```

For the scenarios discussed in this thesis, the initial state of a network is assumed to be a stable state where the node with the lowest identifier being the MASTER is topologically beaconing. This beacon makes it easier for other LNS servers to discover partitions to merge into. In the event of a network partition, a change in the node's *contact_id* reflects loss of its association to a ring. Nodes that cannot hear the beacon assume that they no longer belong to the existing partition. When a node's *contact_id* times out, i.e., it cannot hear the topological beacon anymore, it sets the *contact_id* to itself thereby declaring itself to be a MASTER node. Thereafter, the discovery process for the lowest identifier node ensues. Each MASTER node beacons topologically, unless it hears a beacon from a lower identifier stop beaconing and change status to SLAVES. Eventually, the lowest identifier node would remain as the MASTER since in any given topology there is always an entity which has the lowest identifier. Meanwhile, the stabilization algorithm attempts to repair the links. In the event of a merger of existing partitions, the ring consisting of a lower identifier MASTER remains the same, while all others join this ring. The following algorithm is how distance between two identifiers is determined.

```
x1 = num1 - src_id
```

```
x2 = num2 - src_id


if ( x1 >0 && x2>0  )
  then if x1<x2, x2 is further


if (x1 >0 && x2 <0 )
  then x1 is closer


if ( x1< 0 && x2<0 )
  then if abs(x1)<abs(x2), x2 is closer
```

Taking the example above in Figure 2.6, initially in the stable state, 0 beacons topologically as it has the lowest identifier in the network. After a partition as shown in Figure 2.7 occurs, each of the nodes 0,6,7,8 and 9 can hear a beacon from 0, while 1,2,3,4 and 5 cannot. In the partition with 1,2,3,4 and 5, each of them not being able to listen to any topological beacon, change their status to MASTER and start beaconing. That is, at some point, 1, 2,3,4,5 each start beaconing, unless they hear a beacon themselves. This way, eventually, only 1 would be beaconing. Do note that any logical links that has not been lost to a partition is still maintained by each node.

Initially, each of nodes 1,2,3,4 and 5 as shown in Figure 2.7 do not hear any beacons from 0. Hence, when they time out each of them start beaconing. 1 ignores all beacons. 2 ignores 3, 4 and 5, but it changes its *contact_id* to 1, simultaneously it issues a search request for itself to 1. If the search request is delivered back to 2, it assures that there is a logical path from 1 to 2. Similarly with 3, 4 and 5 there is no change in the *contact_id* for

node numbers below their own and for those where the *contact_id* changes. Hence, no links are changed due to the *contact_id* change in each one of them as shown in Figure 2.8

In the stable state, the stabilization routine consists of pinging your successors. Since there are no insertions and deletions at this time, notifies require no change in links. After the partition event, 0 cannot reach its successor, and 6 receives no stabilization messages from its predecessor. In the other partition, 1 does not receive any stabilization messages from its predecessor and 5 cannot reach its successor. In such a scenario, Chord would designate the next lowest link it has as the successor, and the stabilize operation would fix the successor-predecessor pointers of each node. In the worst case scenario, where none of the fingers are reachable or haven't formed, the above method does not work. In this case, one approach would be for 0, whose successor timer has expired to utilize the circular nature of the ring and do a reverse-search for the farthest point to close the ring. The other approach would be for 6, which does not obtain a ping from its predecessor, to issue a search request for its predecessor, informing it, as its successor. The third approach is for an external agent (automated/user-generated) that knows about 6 and its current IP-address to inform 0 of it. Since merging would not be easy with just the stabilization protocol, an anchor node (i.e. the node with the lowest identifier amongst the LNS servers) beacons regularly topologically, making it easier for other LNS servers to discover if there is another partition to merge into.

Further, in the event that the underlying network has reconnected, the logical partitions attempt to merge. In the above case, 0 hears 1's beacon and since 0<1, it is ignored. 1 on the other hand, hears 0's beacon and changes its contact to 0. It then issues a search for 1 to 0, which fails to locate 1 and returns 6 to 1 instead. At this point, 1 informs 0 that it is

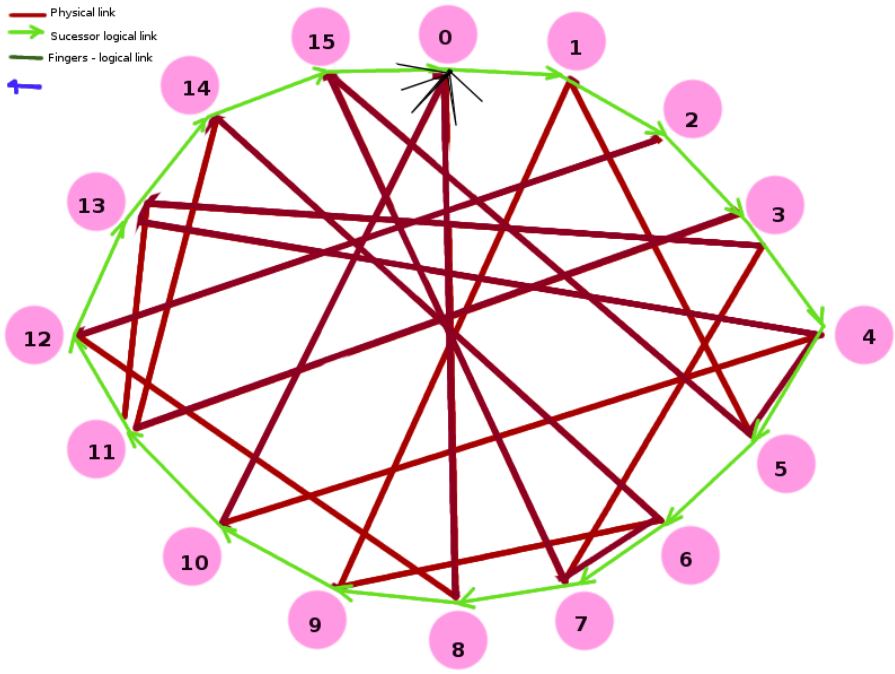

Figure 4.1: Physical and Logical Links at Initial State

its successor and notifies 5 that its successor is 6. 0 subsequently updates its successor to 1 and 6 on stabilization updates its predecessor to 5. This causes the entire ring to merge. SLAVE nodes change their *contact_id* to the lowest identifier beacon heard.

There is an issue here as to what to do with current links when you hear a topological broadcast from a node that is not your *contact_id*, i.e., there is more than one master in your reach. One possibility is if the *contact_id* changes, to break all the existing links and join the more recent leader ring. The other option is to keep the links as is, unless the stabilization algorithm invalidates it. Existing links may be kept because identifiers are globally unique and for the application that we are considering, globally there is assumed to be just one such universal structure. Hence, links between nodes are not those that cannot be fixed with stabilization/search. In essence, since the existing links during and after a

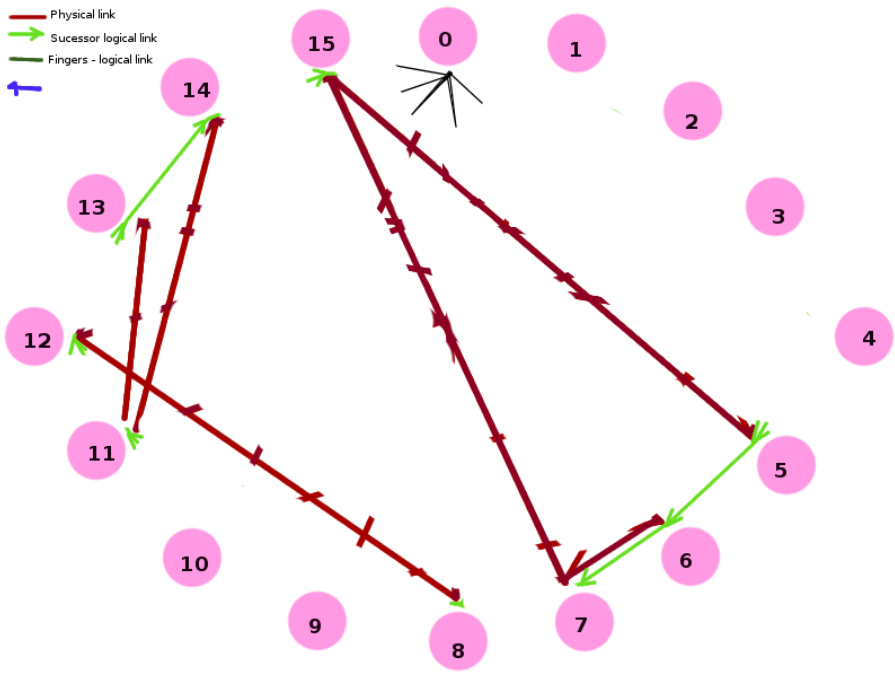

Figure 4.2: Physical and Logical Links after network partition

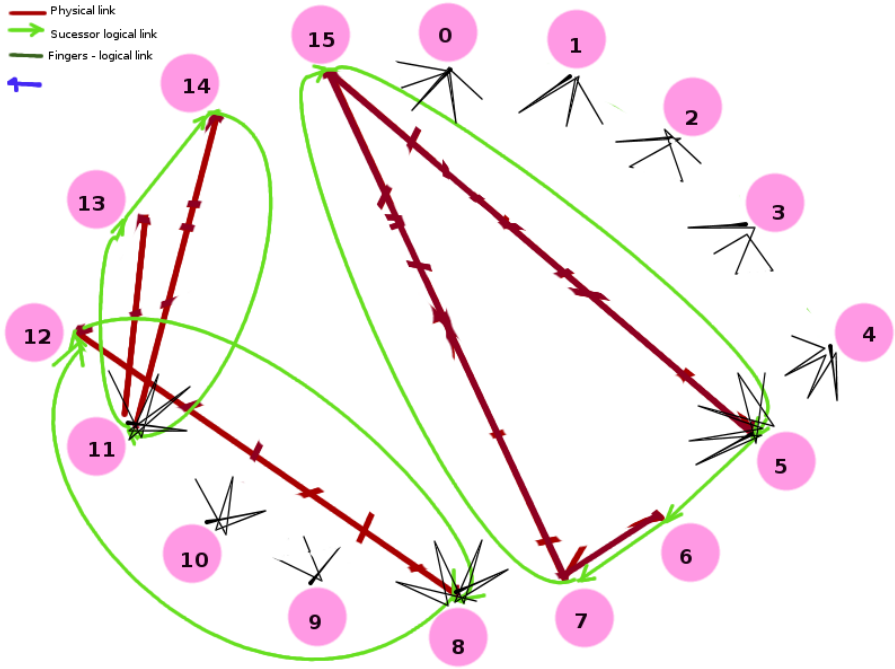

Figure 4.3: Physical and Logical Links after network merger

failure are not incorrect, there is no need to delete them voluntarily. As in the example considered, the logical overlay need not partition into contiguous segments. The Figure 4.1 is the same as in the earlier example. After the break, the topology as shown in Figure 4.2 emerges, leading to non-contiguous segments. Even then, the linkages that existed before the partition, that have not broken, can be used still. The network on stabilizing reforms as shown in Figure 4.3.

This protocol was modeled using J-Sim [40] and its network models, where only the basic Chord ring was modeled. All initial joins in the simulation scenarios were sequential. A successful lookup is when a search query is answered. The links are wired links that are simulated to be broken at various points in time. Network layer routing tables in each node are pre-configured before the start of the simulation, and so there is no route discovery. The Figures 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 show the behavior of the ring under various conditions.

4.2 Failure modes

4.2.1 Examples of pathological cases

In an earlier example, two partitions merge into a consistent ring using the algorithm described. In the case of more than two partitions, assume that initially the ring consisted of 16 members, 0-15 as shown in Figure 4.10. Further, in the event of a partition, it breaks into three parts 8-15, 0-3 and 4-7 as shown in Figure 4.11. These three partitions eventually stabilize themselves independently as shown in Figure 4.12. When this partitioned network reconnects, different structures arise depending on the order in which messages are exchanged. If the network reconnects such that two partitions merge at a time, that is first

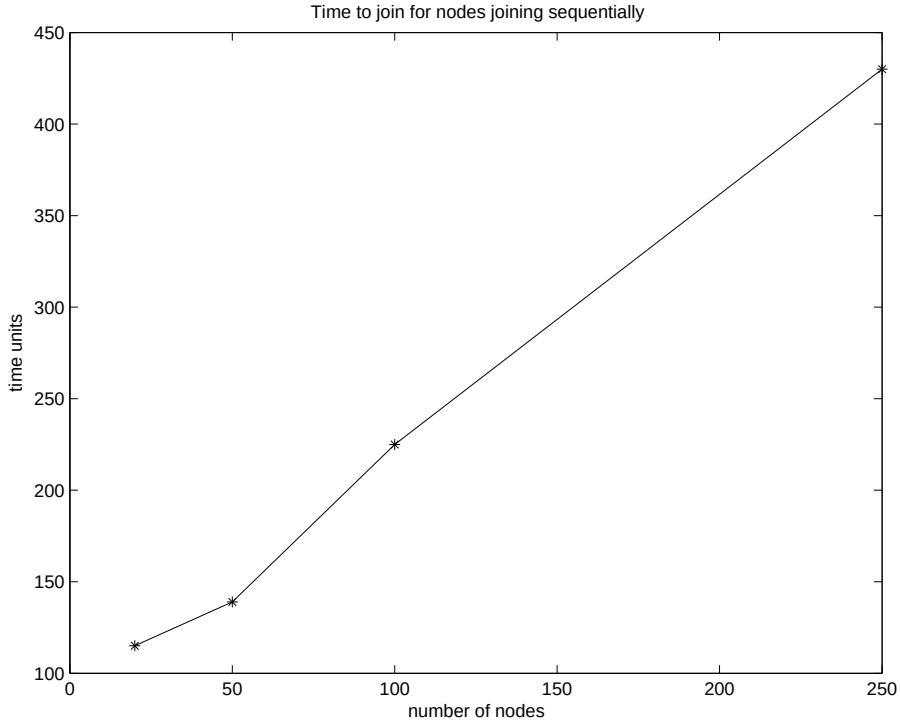

Figure 4.4: Time units required for sequential join of nodes into a Chord ring

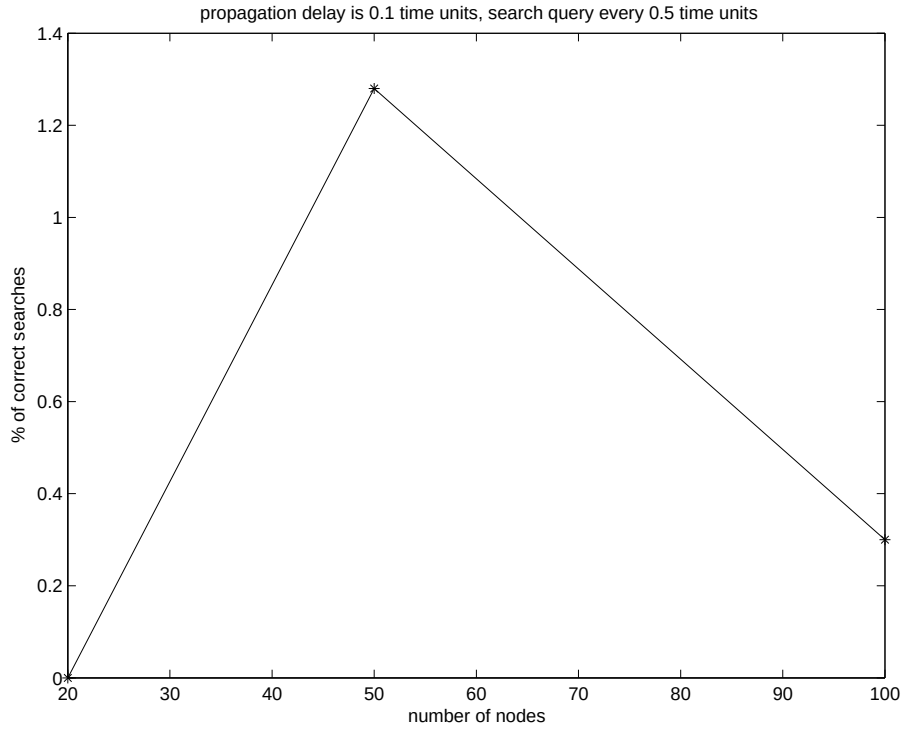

Figure 4.5: Percentage of successful search queries during a network partition where propagation delay of links is 0.1 time units and a search query to all other nodes is sent every 0.5 time units

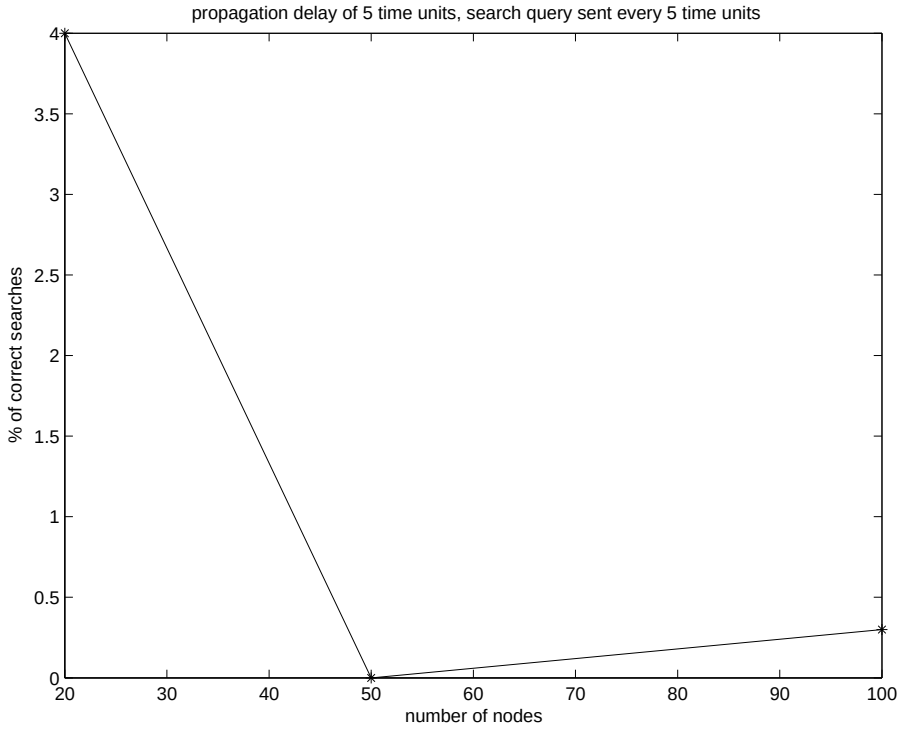

Figure 4.6: Percentage of successful search queries during a network partition where propagation delay of links is 5 time units and a search query to all other nodes is sent every 5 time units

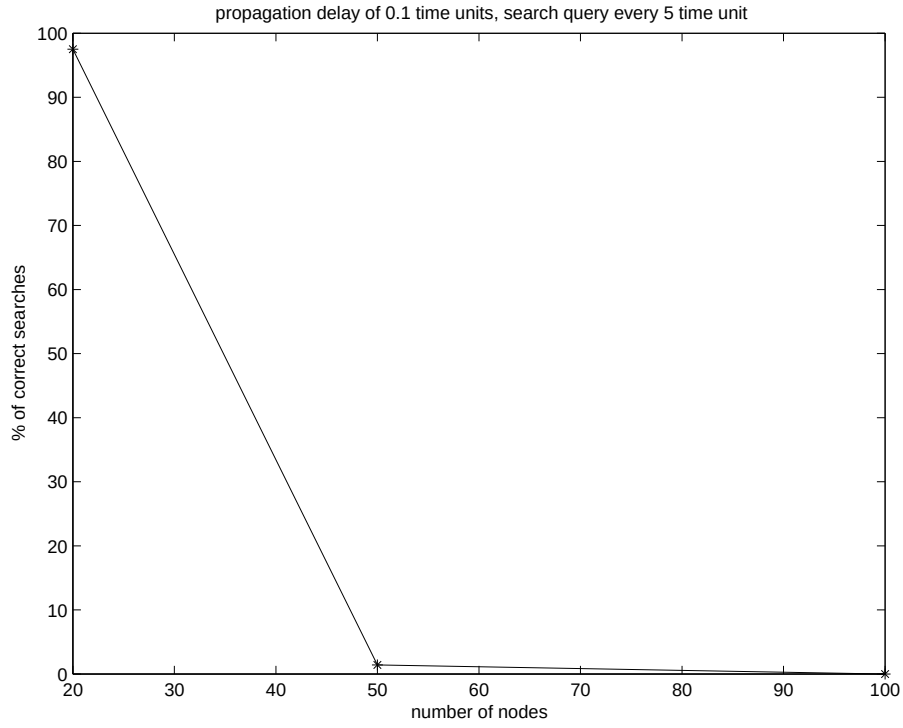

Figure 4.7: Percentage of successful search queries during a network partition where propagation delay of links is 0.1 time units and a search query to all other nodes is sent every 5 time units

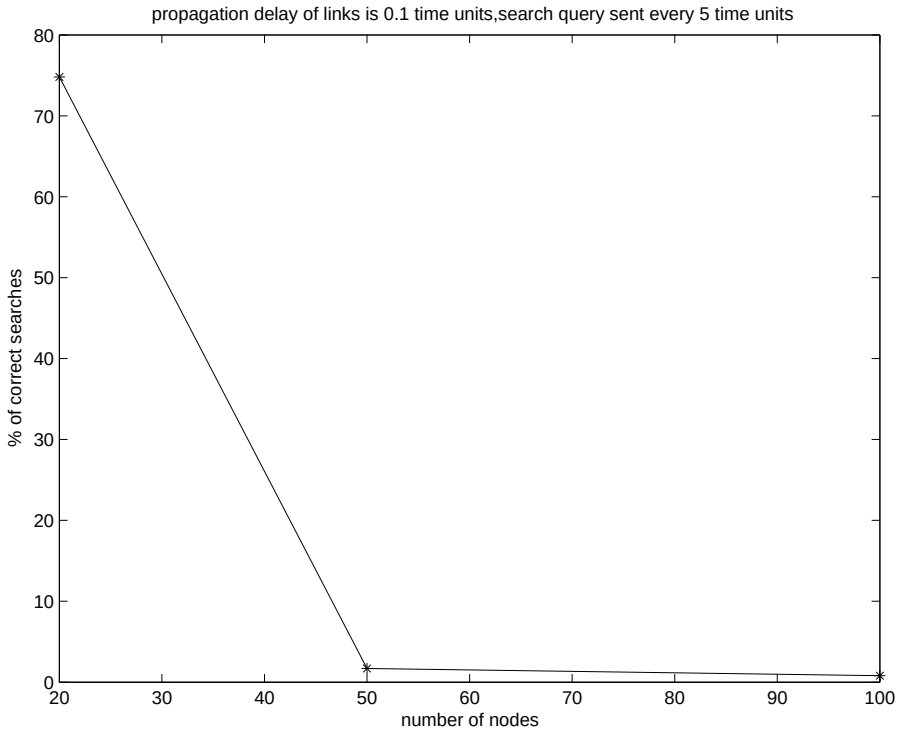

Figure 4.8: Percentage of successful search queries in consistent state where propagation delay of links is 0.1 time units and a search query to all other nodes is sent every 5 time units

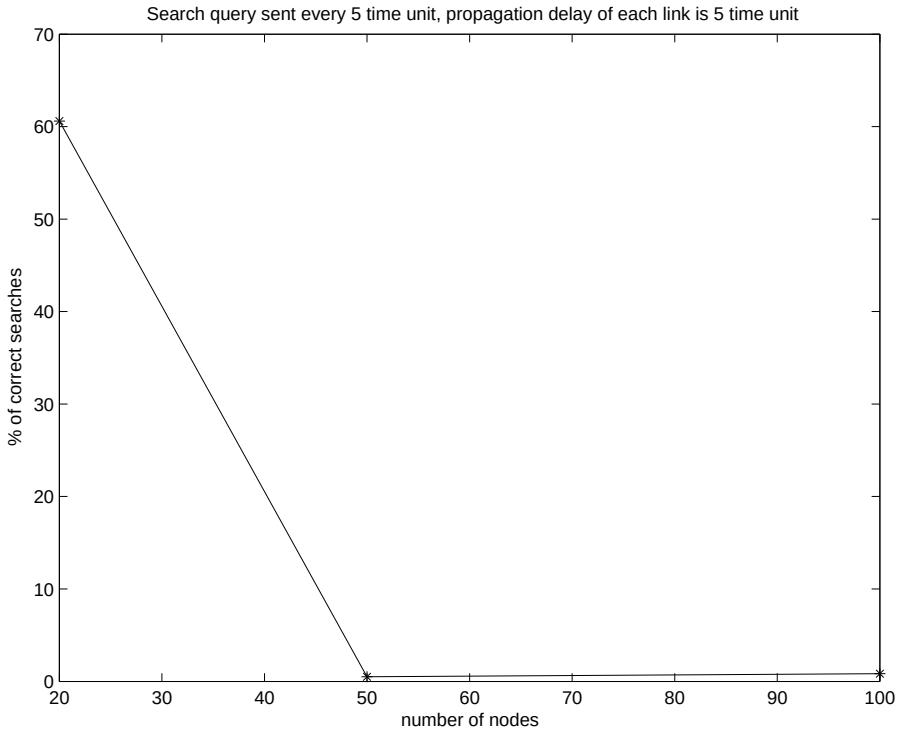

Figure 4.9: Percentage of successful search queries in consistent state where propagation delay of links is 5 time units and a search query to all other nodes is sent every 5 time units

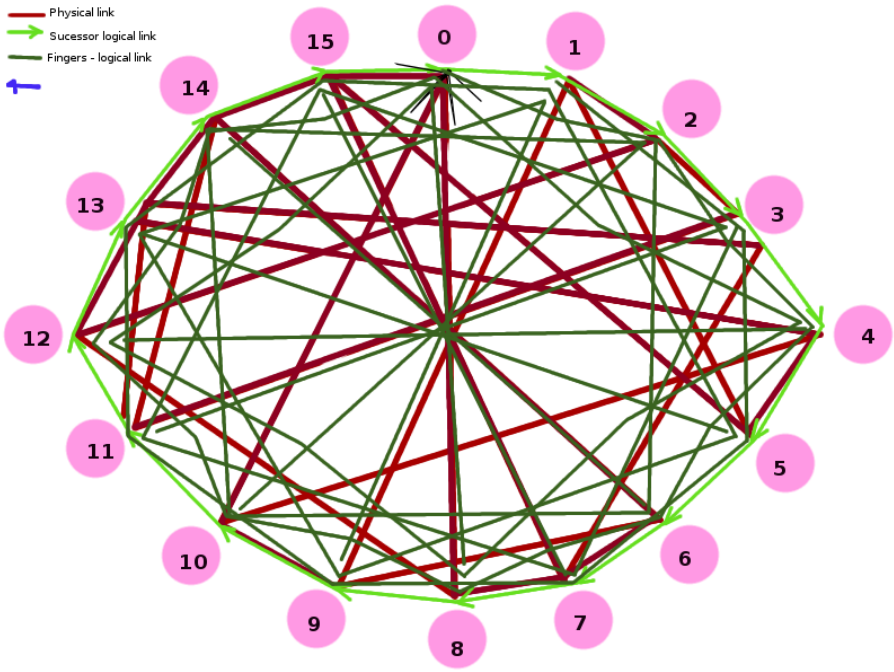

Figure 4.10: LNS servers logically associated as a Chord ring

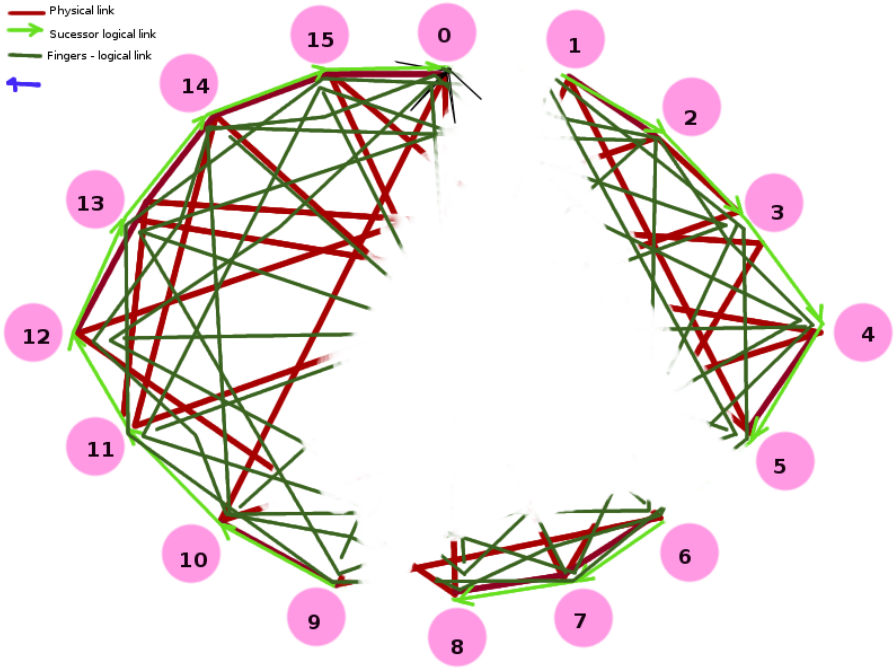

Figure 4.11: Disruption in the logical connections due to network partition

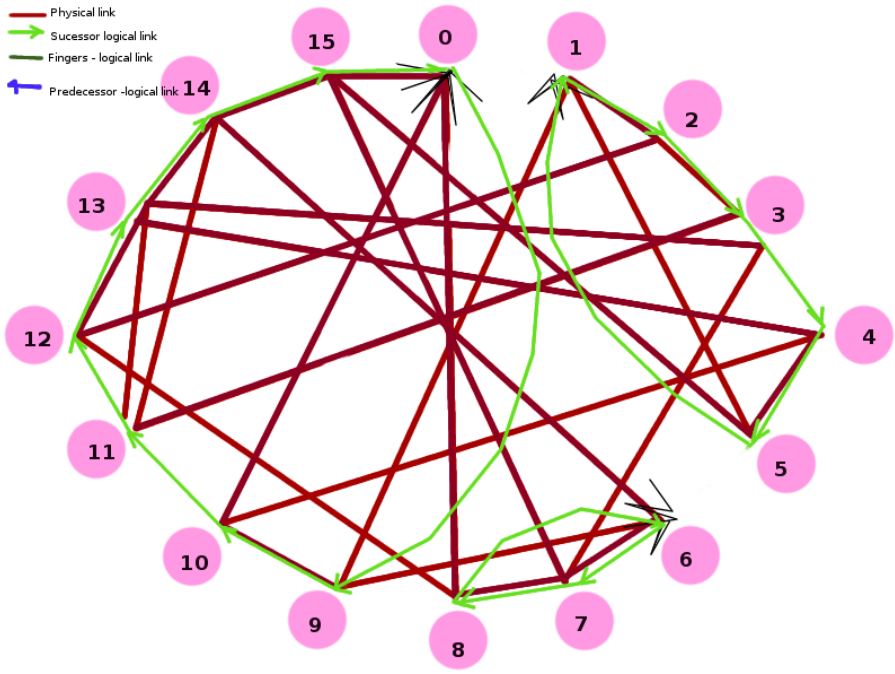

Figure 4.12: Stabilized logically partitioned rings after a network merger

two merge and then the merged ring merges with the last partition, it results in stabilizing to the original stable state. But, if several topological links are formed, such that all three networks merge concurrently, irrecoverable structures may emerge. In the scenario described, 6 may hear both 0 and 1's beacon and change its *contact_id* accordingly. 1 would change its *contact_id* only on hearing 0's beacon. One possible scenario is the ring $6-8$ attempts to join $1-5$ while $1-5$ joins $9-0$. Here, as part of the stabilization routine, 6 requests 1 for its predecessor, i.e., 5, while 1 requests 0 for its predecessor, i.e., 9. 6 notify 5 that it is its successor, while 1 notifies 0 that it is its successor. 6 notify 8 that 1 is its successor, while 1 notifies 5 that 9 is its successor. 6 then changes its predecessor to 5, while 1 change its predecessor to 0. Both 6 and 1 stop beaconing after that, making it difficult to detect such a condition. Both of these configurations are depicted in Figure 4.13 and

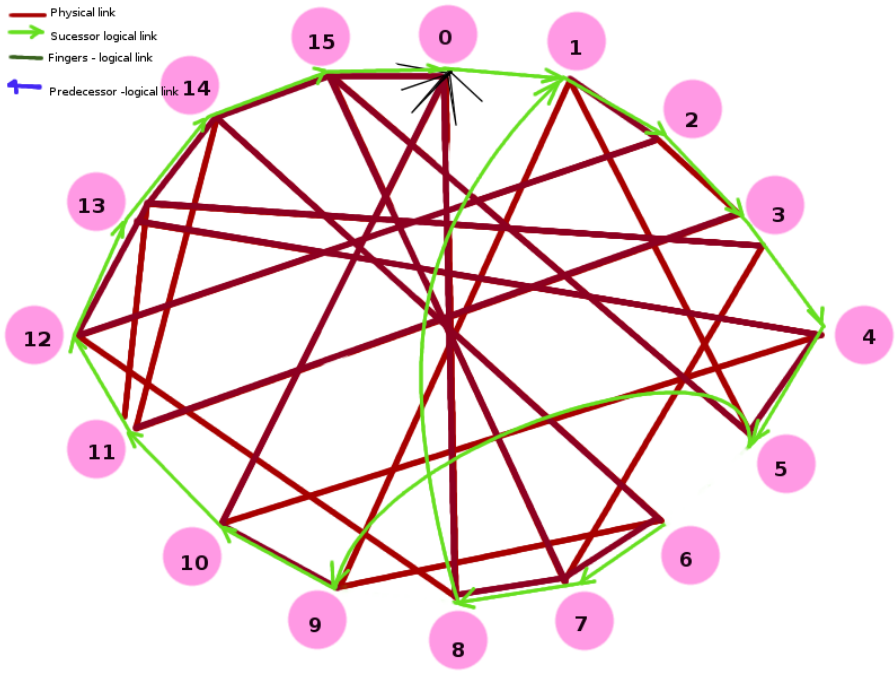

Figure 4.13: Irrecoverable scenario when more than one partition attempts to merge

Figure 4.14. The stabilization routine running in the background would not be able to recover from this to a consistent single ring. If $6-8$ and $1-5$ try joining $0-9$ at the same time, the race conditions for the successor-predecessor pointers at 0 and 9 would lead to irrecoverable rings as well.

For robust recovery from partitions and mergers, if we consider that each node maintains both successor and predecessor links as shown in Figure 4.15, that partitions as shown in Figure 4.16, and the partitions stabilize as shown in Figure 4.17, when they merge, it may lead to irrecoverable cases. In the case where $6-8$ joins $1-5$ while $1-5$ joins $0-9$, 6 requests 1 for the current location of its predecessor, that being 5, and successor, that being 2, while 1 requests 0 for the current location of its predecessor, that being 15, and successor, that being 9. 6 notifies 8 that its new successor is 1 and 1 that its new predecessor is 8;

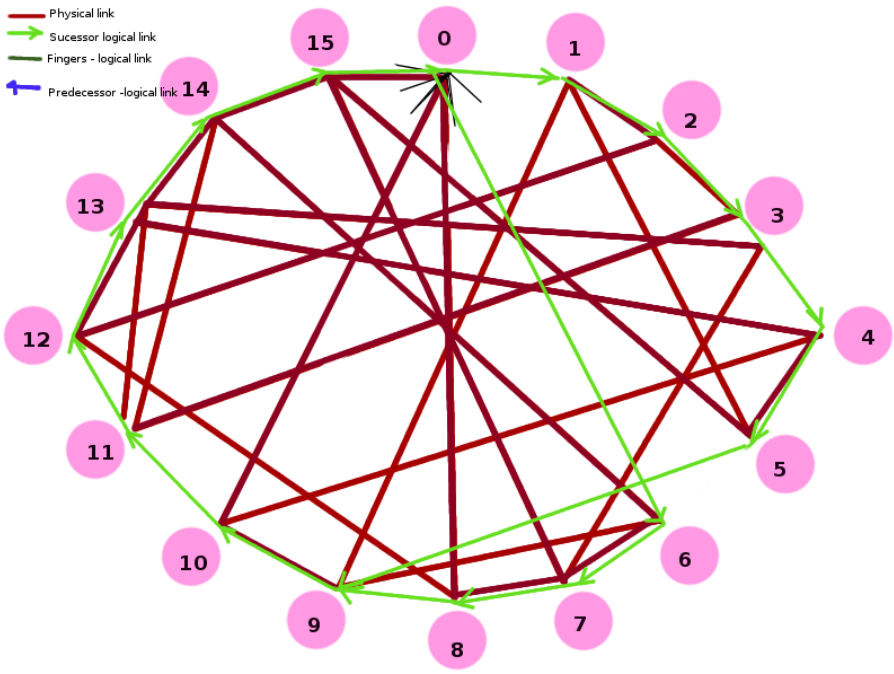

Figure 4.14: Other Irrecoverable scenario when more than one partition attempts to merge

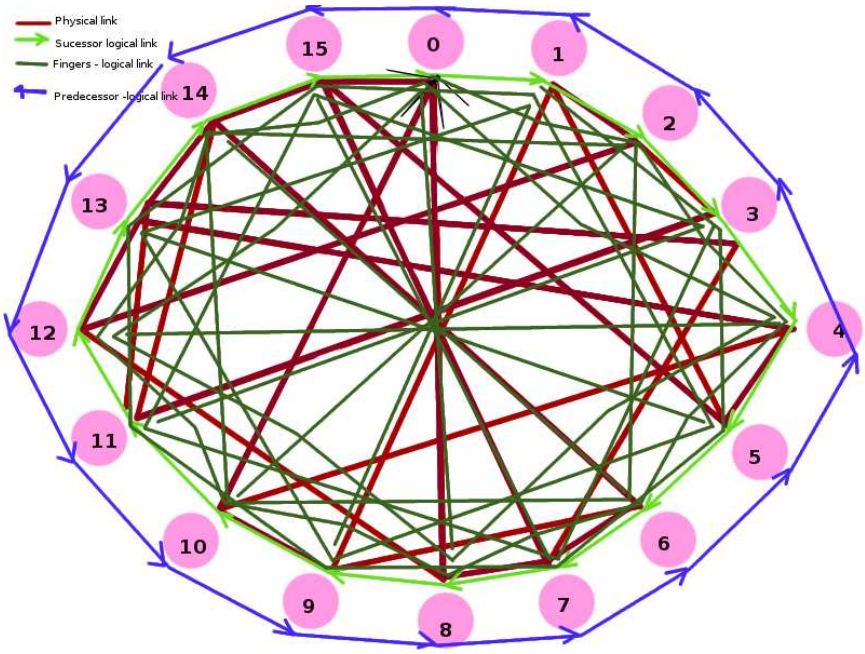

Figure 4.15: 16 LNS servers logically associated with both successor and predecessor pointers maintained

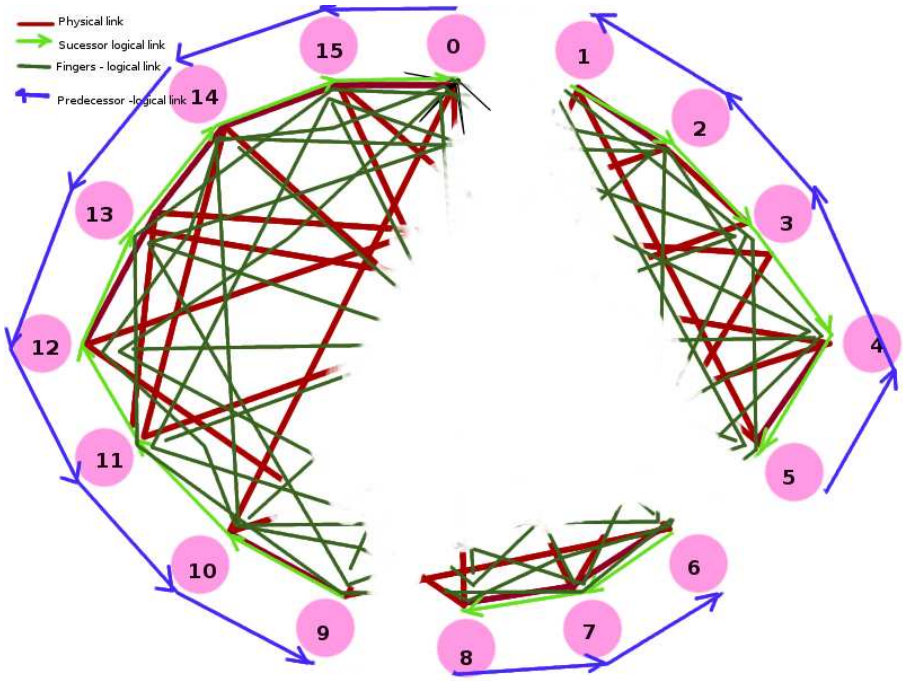

Figure 4.16: Effect on the logical structure due to network partition

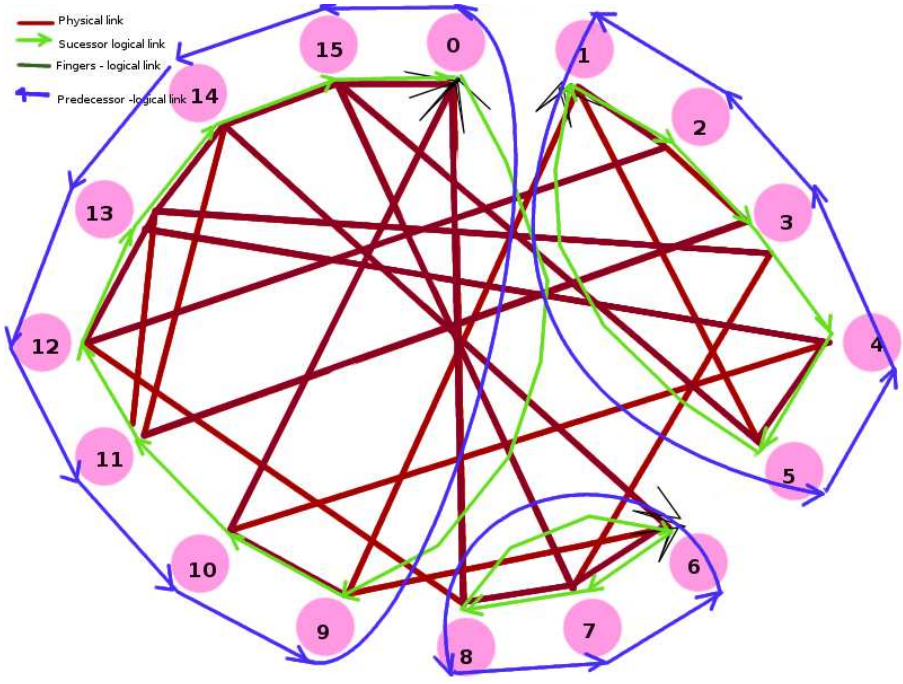

Figure 4.17: State after a network merger

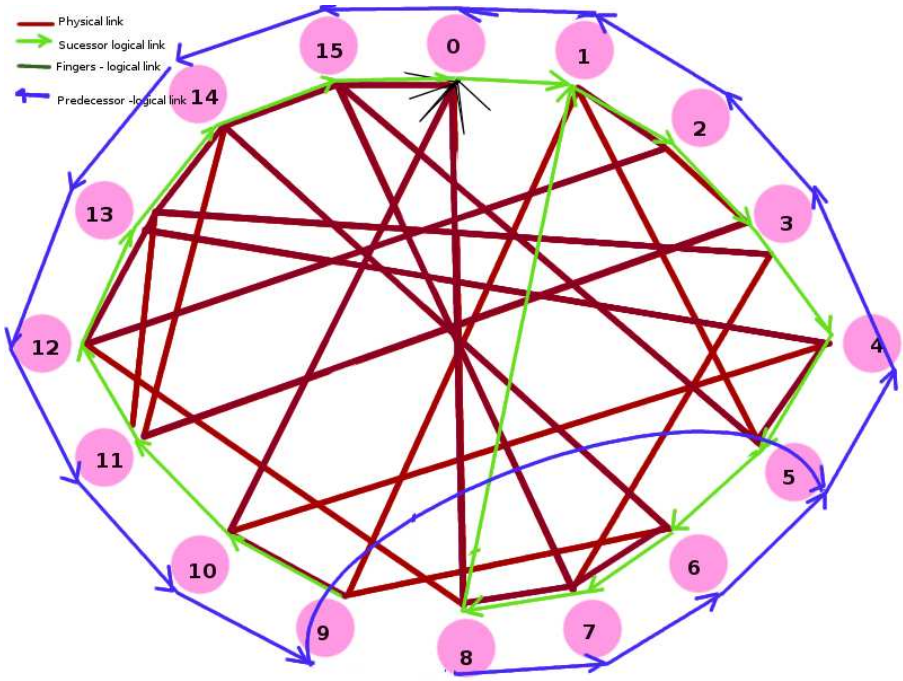

Figure 4.18: One Merger scenario

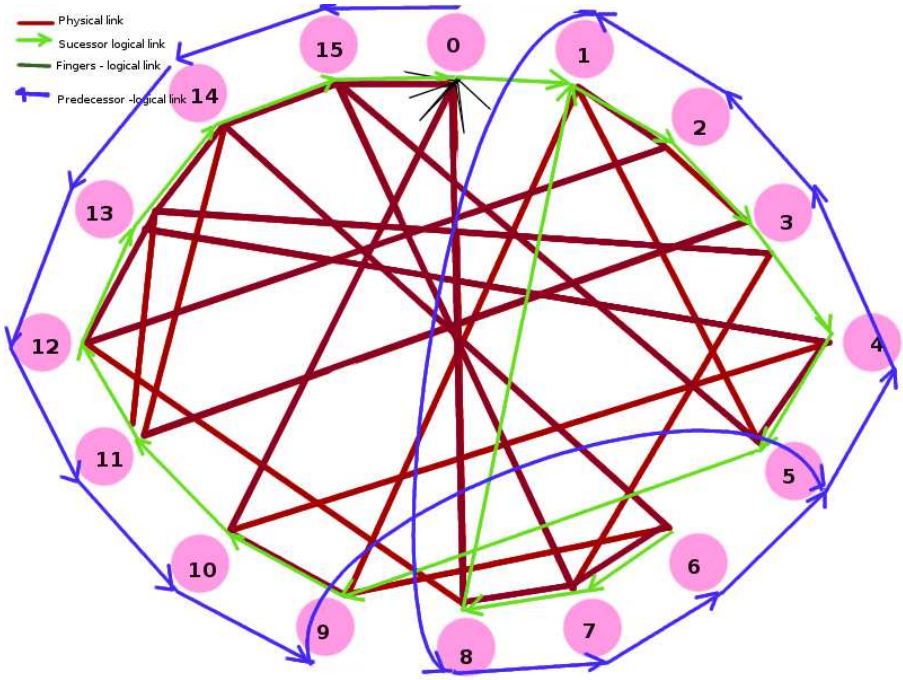

Figure 4.19: Other Merger scenario

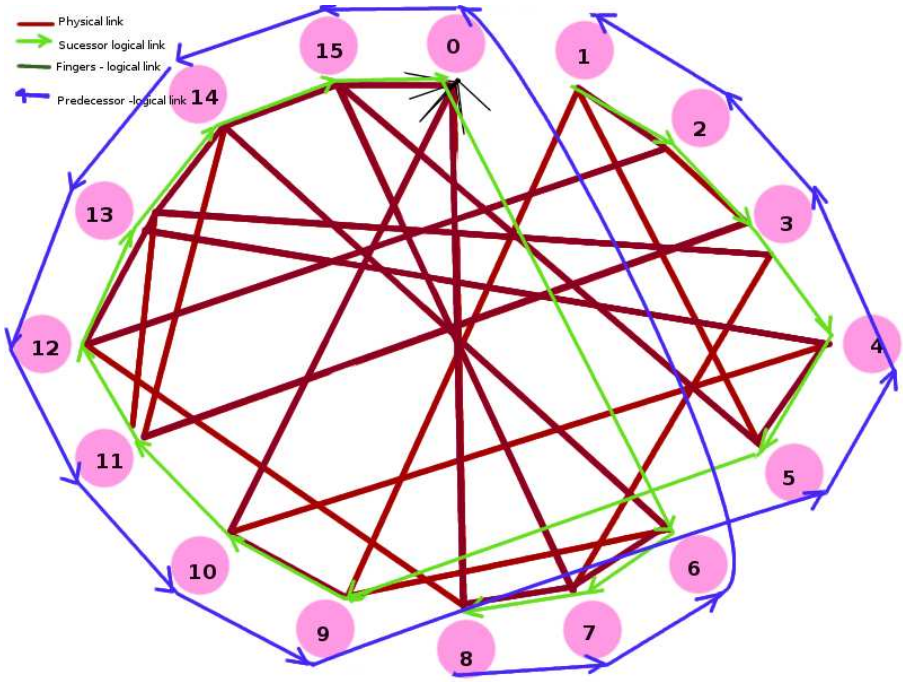

Figure 4.20: Yet another merger scenario

while 1 notifies 9 that its new predecessor is 5 and 5 that its new successor is 9. This result in the following irrecoverable bi-directional ring as shown in Figure 4.18. Similarly, when both 6-8 and 1-5 attempt to join 0-9, it may result in the following scenarios as shown in Figure 4.19 and Figure 4.20.

There are many problems with using a unidirectional ring based DHT with passive stabilization technique as we observe in this section, particularly with respect to behavior while merging. Not described is the case as to what happens, when the partitioned nodes leave the rings immediately on detecting a new partition and joins it. One of the obvious problems is that it would cause concurrent joins which the DHT may not be able to handle. The other problem is that beacons from the non-leader partitions may not subside at the same time, causing nodes to join and leave the wrong partitions and lead to irrecoverable

topologies as well. The passive maintenance protocols do not help in recovering from network partitions. After such partitions attempt to merge, they form linkages that do not eventually lead to a consistent ring. This causes sections of the logically connected overlay, to lose service connectivity to parts of the network without possibility of recovery. The need here is to be able to reach all current services management servers translations. During the partition or the merger, a part of the services may be available, but as the network stabilizes, all translations should be available. The current DNS as it exists does not allow this. Most of the configuration is handled manually, but in a highly distributed manner. This may not scale in the disaster or high mobility or frequent node death scenarios that lead to partitions. Further, a DHT structure allows dynamic assignment of responsibility.

4.3 Use of concurrent management

As opposed to Chords passive stabilization method, in order to maintain consistency while there are concurrent joins and leaves, [34] proves that active joins and leaves, i.e., the successor and predecessor pointers be managed at join and leave time rather than using a periodic background process, and a bidirectional ring, i.e., one where both the predecessor and successor pointers are correctly maintained, is required. This idea is exploited in the above cases allowing partitions to heal with higher probability. When rings start merging, MASTER nodes that hear beacons from a lower identifier node, leave the ring, passing the beacon to the next available server, and join the leader ring. Figure 4.21 states the condition at the start of a merger due to network merger, Figure 4.22 depicts the first few nodes leaving the non-leader rings as does Figure 4.23, while Figure 4.24 shows the point when all the non-leader rings have been dissolved and nodes concurrently attempt to join

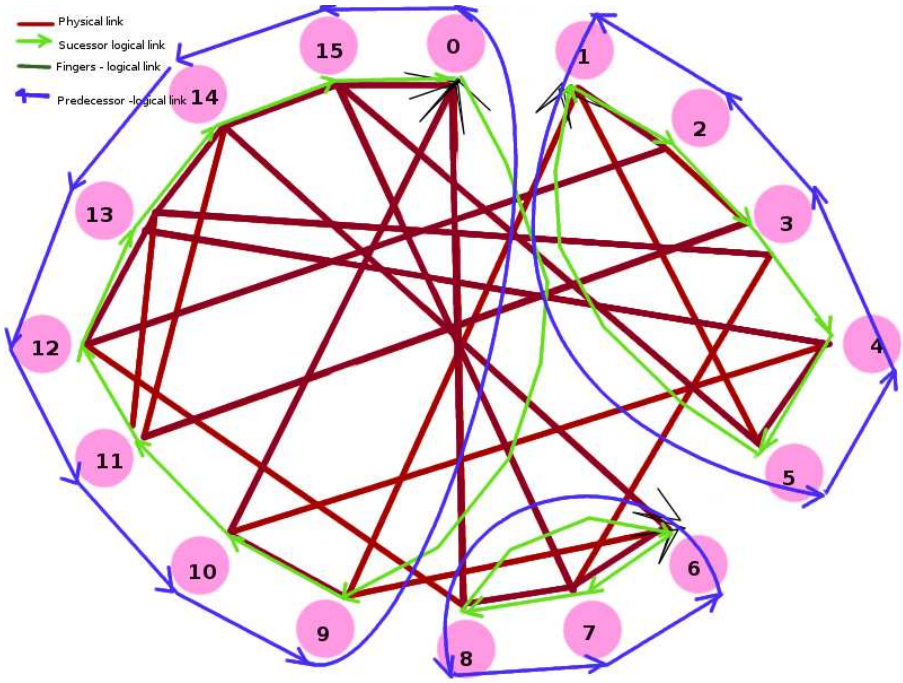

Figure 4.21: Stable rings at start of merger

the leader ring. Even if they do attempt to join a non-leader ring, the non-leader rings would eventually deplete.

Without the detailed examples, it is also suspected that concurrent joins and deletes during repair due to partitioning or merging may make the structure irrecoverable as may merging before the partitions have stabilized on either sides of the network break. Further, if there are nodes that join or leave the partitioned service management infrastructure, they change the logical linkages, such that merging may no longer be simple since the ring may need to be broken into further parts in order for the node to be merged. Such joins may not be avoided since the network may merge after a long time and more servers may be required for sharing the load.

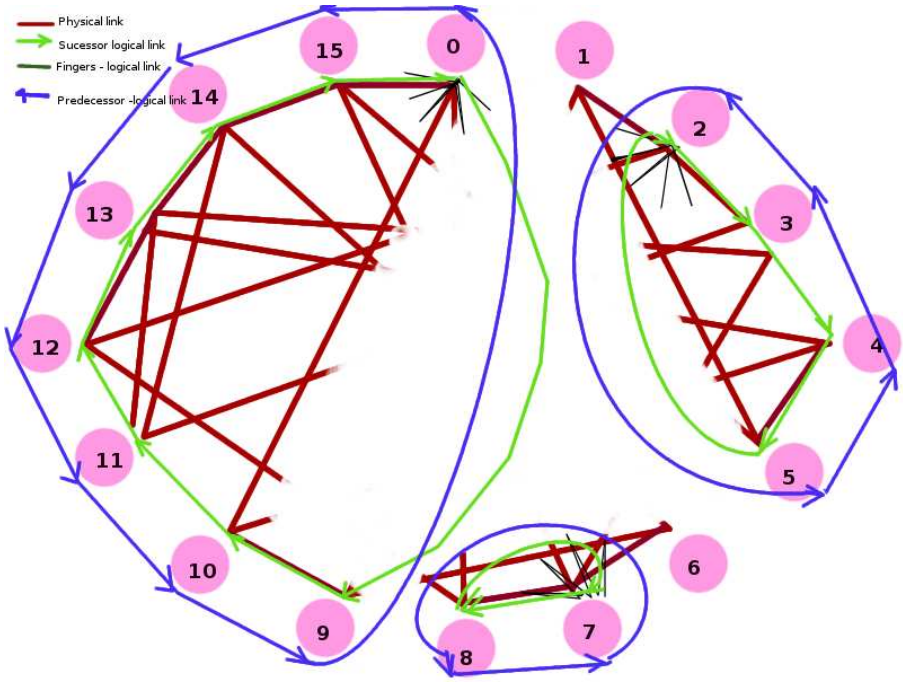

Figure 4.22: Nodes leaving non leader rings

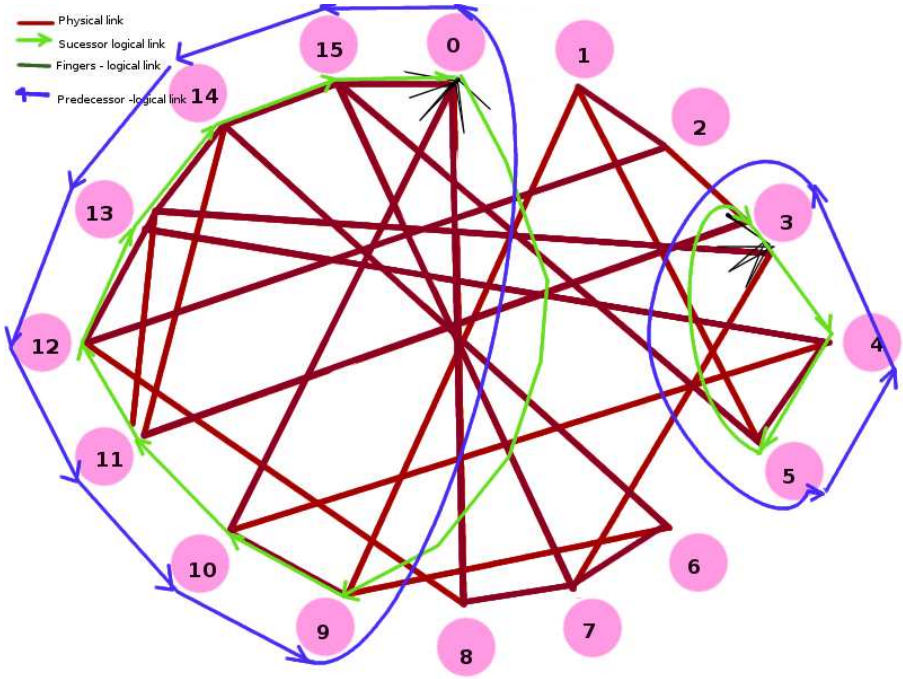

Figure 4.23: More nodes leaving non leader rings

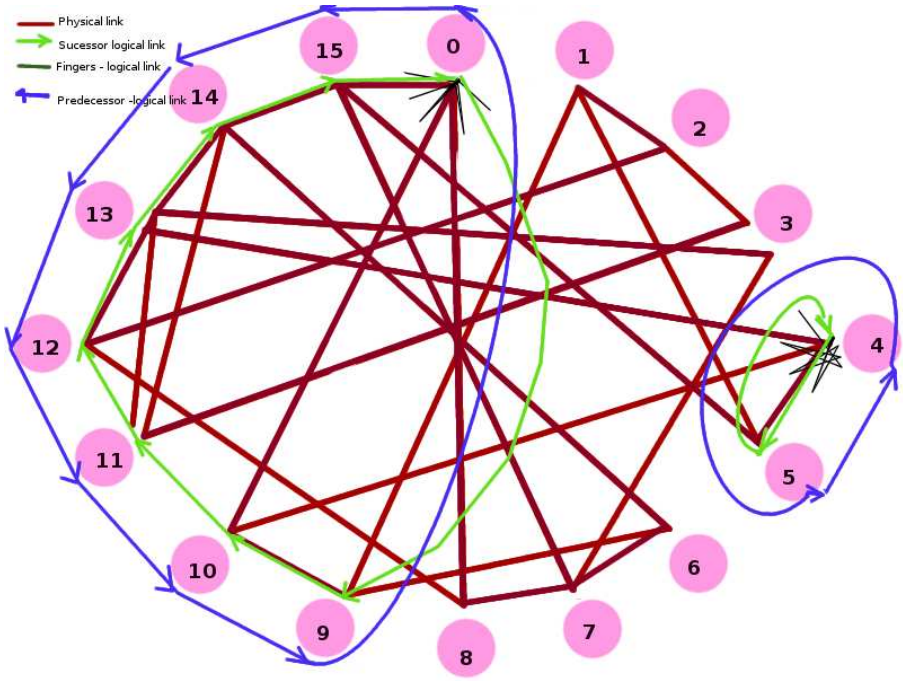

Figure 4.24: Dissolution of non leader rings

Chapter 5

Auto-configuration in MANET

Nodes that connect to the internet, either through wired interfaces or through last hop wireless, are special nodes that need at least one interface to route to the MANET and others that interface to the internet. For nodes participating in a MANET, routing consists of interfaces that use MANET routing protocols such as AdHoc OnDemand Vector (AODV) [41], Directed Source Routing (DSR) [42]. They discover routes that create and maintain entries in a node's routing table. In accordance with the layered approach to protocols, interfaces participating in networking require a unique IP address or identifier. During a node's assignment of an IP address, it is also configured with its local DNS server's IP address to which it can send its lookup queries. In most MANET routing algorithms, each node contains the IP-MAC translations of its neighbors. The following is a manually worked out example of how the above Chord based protocol for MANET may be used.

The scheme would be that a node, when it wakes up, either joins or creates its own MANET. It decides to be a configuration server or client. As a configuration server, it uses the auto-configuration protocol to discover at least one other configuration server to join into the group of configuration servers. As a host client, it uses the auto-configuration protocol to discover one other configuration server, to which it can search for its current "responsible" server and then upgrade its configuration information. It can also issue searches to it for service discovery for all reachable nodes. If it becomes a host, it needs to find a server, if it cannot reach a server it becomes its own server. As new servers discover this server, they join and form a bi-directional Chord like ring. The beaconing is then by the server which

has the lowest identifier. Since, the servers form an ordered logical ring it is easy to assign the beaconing to the lowest identifier node.

5.1 Extended DRCP to work with Chord LNS

In MANET, consider the case of three wireless interfaces IF0, IF1 and IF2 separated at maximum range from each other linearly (such as IF0-maximum range of the interface-IF1-maximum range of the interface-IF2). Let the DRCP server process be running on IF0. It has a list of IP addresses that it can assign to requesting interfaces. If IF1 sends a DRCP SOLICIT requesting configuration, IF0 will be able to hear it. If IF2 sends a DRCP SOLICIT requesting configuration, IF1 will be able to hear it. In order to limit broadcasts and floods, once IF1 is configured, it can respond to IF2's DRCP SOLICIT by acting as a relay agent for IF0.

The proposed is the modification to the DRCP client to modify it as a DRCP relay agent. The current DRCP client state transition diagram is modified to Figure 5.1. Note that only configured clients participate in routing, non-configured clients do not and neither receive nor forward messages. A non-configured client either receives a configuration or broadcasts a DRCP SOLICIT.

It is assumed that devices in the internet have some non-identical and non-conflicting data that can be used to generate a unique ID. At boot time, an interface decides to be either a server or a client. If it is a server it may broadcast its availability using DRCP_REQUESTs. If it is a client, it broadcasts DRCP SOLICITs link locally for Maximum Hops to a Configuration Server (MAXHOPS)*(Time for one hop), before becoming a server. This results in a two-level hierarchy.

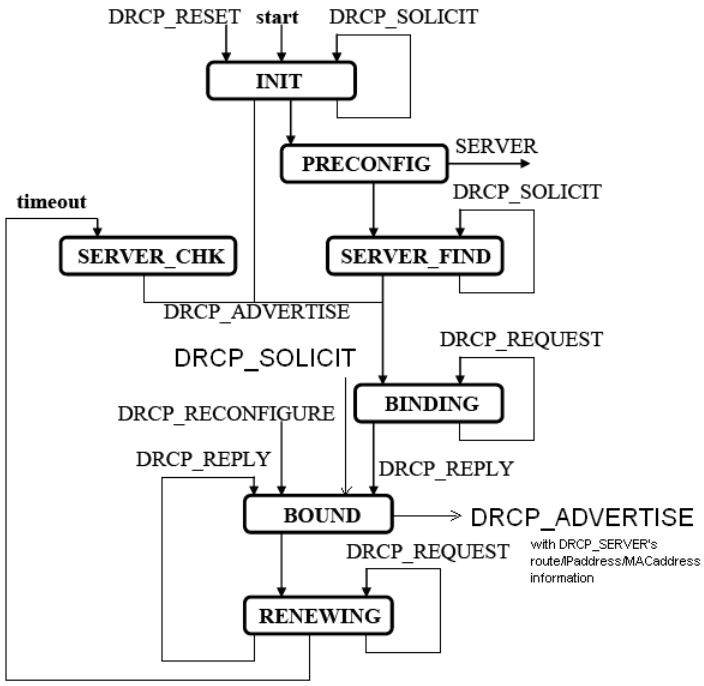

Figure 5.1: Modified DRCP client state machine for multi-hop configuration

5.1.1 Behavior of a server

If an interface elects to be or becomes a server, it creates a unique signature id of a fixed length, for example using SHA-1 [11]. On doing so, it starts one-hop broadcasting, its $\langle ID, timestamp \rangle$. It is then configured.

Once configured, it listens for messages of the form $\langle IDA, timestamp, routenodelist \rangle$. This allows it to build up its routing table to other nodes. Each next hop destination MAC address is obtained from the routing table and neighbor set.

As a server, it also disseminates information about itself. This allows a client interface to test if its original contact is alive and in the case that it is no longer reachable it registers itself with the advertised server.

For communication, originating at a server, the server sends packets of the form Server's MAC address as Src, Next Hop Client as Dst, Server's IP address as Src, Desired Dst IP, message.

At the next hop Client, it changes the header to Client's MAC address as Src, Next hop as Dst, Server's IP address as Src, Desired Dst IP, message. Since, from a server, all the hops are configured, this will ensure delivery of the packet.

When an interface receives an ID, it can use it unless it is changed by some other server. Once assigned, until the interface registers with another server and changes it, it remains the same. For a server, that has no connections to other servers, it assigns addresses sequentially, starting with its own ID. For example if the ID of a server is 192.134.24.56, the address that it assigns to the next client that requests for one is 192.134.24.57.

5.1.2 Behavior of a client

If an interface decides to be a client, then it waits for MAXHOPS*(Time for one hop) time before it transforms into a server. On boot up, it sends out a message with Client's MAC as Src, Broadcast MAC as Dst, A default IP as Src (such as 0.0.0.0), Link local Broadcast IP address as Dst.

This is broadcast to one hop neighbors. Amongst the one hop neighbors if any of them is a DRCP server, they can reply back to notify the client of their existence. This allows the client to select one of the possible servers to associate with. To the selected server it then sends a bind message. The bind message is of the form Client's MAC ID as Src, Server's MAC is as Dst, Server's IP as Dst, A default IP as src, to which the server's reply would be Server's MAC ID as Src, Client's MAC ID as Dst, Server's IP as Src, Client's IP as Dst,

message. This would assign the IP to the client along with configuration information and keep a searchable copy on the server.

If there are no servers within its immediate server set, the neighbor clients that are configured, send their server information as the reply. The client then sends the bind message to one of those clients, which the client routes to the server that it has been configured by. This bind message looks like Client's MAC ID as Src, Neighbor's MAC ID as Dst, Neighbors IP as Dst, A default IP as src. The neighbor client caches this, and sends a new request to the server for a new IP address for the client MAC address. The server in turn does that and sends back the results to the client acting as a relay. This relay client on receiving the translation sends out a bind reply similar to the one above to the client, thus configuring it.

For communication, originating at a client, the configured client sends packets of the form Client's MAC ID as Src, Next Hop Client's MAC ID as Dst, Client's IP as Src, Desired IP as Dst, message. Since, clients that are not configured do not participate in routing, routing to the client can be through paths that have already configured nodes.

For a client that does not obtain a response from any client supporting a route to a server, it re-broadcasts its query every exponential timeout waiting until MAXHOPS *(Time for one hop). A server is expected to exist at least MAXHOPS away.

When a client does not receive a response of a path to the server after MAXHOPS * (Time for one hop), it is considered to have no existing servers to obtain configuration from. Hence, it is forced to take on the role of a server. At this point, the client changes its status to server, and proceeds as a server. In this manner during zero-configuration if the

intermediate clients need time to configure them, some of them will turn into servers and eventually the further hop clients shall be serviced.

5.1.3 Discovery of other servers

The above description details the process of obtaining an IP address and other configuration information at boot time. The following description enlists how to lookup for translations and how the clients and servers behave once configured.

The above scenario is in consideration that no new LNS servers pop up and existing LNS servers do not decide to stop being so, which may not necessarily be true.

Apart from basic networking, servers attempt to discover other servers. This it does, using regular client notifications. As in, an un-configured client, when it receives replies to its broadcast configuration request from more than one client, it compares them and sends out the notification to each one of them of the other. This works during zero-configuration, when servers are trying to assign addresses to a large number of nodes. In this set up there are a large number of nodes that do not have any configuration information. The servers either simultaneously or at different times wake up to service the clients around them. As clients time out, they start becoming servers. This allows that as clients configuration spreads, at borders, clients will receive information of more than one server as available.

With the newly discovered servers, a given server can handoff part of its list of IP-MAC translations to the new server and delete them from its own, thus transferring responsibility of configuration information to it. This occurs frequently in the scenario where a server with ID1 has assigned IP addresses up to ID3, where ID3>ID1, and it further discovers a new server ID2 where ID3>ID2>ID1. In this situation, ID1 hands off all translations from

ID2 to ID3 to server with ID2 and then ID2 reconfigures them if it already has a set of clients assigned up to ID3. If not, it keeps a pointer to the current server, i.e., ID1 that manages the translations up to ID3 and thus it sets its current starting assignment ID to ID3.

Consider the sample scenario where in there is one client configured via ID1 surrounded by clients configured by ID2. When a new client enters the neighborhood, it obtains info from ID1 and ID2 assigned clients, and can thus inform each other of the existence of the other. In a stable scenario, there are also a few situations, one, wherein, there are servers that already exist or new servers are formed for load balancing requirements or a client wishes to modify itself to a server for accounting purposes. In that case, there may not be any new nodes to be configured at boundaries to be able to inform nodes of the presence of respective servers. The manner in which to discover each other to participate in the overlay network can follow the following mechanism where a server with node ID1 sends a MAXHOPS+2 limited message to discover another server. This message is a hop limited broadcast. For each client that receives this message <ID1, Searching Other Nodes, Hop Count (HOPCNT)>, it looks up its own registered server id.

This broadcast need not be frequent. The message may also contain IDs, for which that are already discovered and do not need to be replied for. This list can be compared by the client attempting to send back a reply.

```
if(server_id == ID1 or dont have a server_ID)
  further broadcast the message after decrementing the HOP_CNT;
else
```

```
{
reached a boundary area.
Reply back to the discovering server with route information.
}
```

The following figures depict a hand worked example of how Figure 5.2 is the initial state of the MANET where some nodes have decided to become servers and have a self generated unique ID, Figure 5.3 is where the one hop nodes from the servers can detect its presence and have been configured by their respective servers, Figure 5.4 is where the second hop clients are configured through the help of some first hop clients acting as DRCP relay agents and two servers are able to discover each other as one of the second hop nodes' receives request for configuration from two different servers, Figure 5.5, Figure 5.6, Figure 5.7 depict further hopes being configured and Figure 5.8 is the state where all of them have been configured.

5.1.4 Changes due to mobility

As long as no new nodes enter, no change in configuration needs to take place. If a node moves that is not a server, only the entry in the server's translation table needs to be timed out as shown in Figure 5.9. If a server moves out, then if there are new nodes, that cannot find a server to register with, it becomes the new server.

Discovering other servers, allows a server to distribute its database of IP-MAC address translations to them and deleting those entries. For entries that it deletes, it sends out a server identifier, timestamp, route node list to new server. When it moves out voluntarily, it can hand out the database of IP-MAC translations to the node that is responsible next.

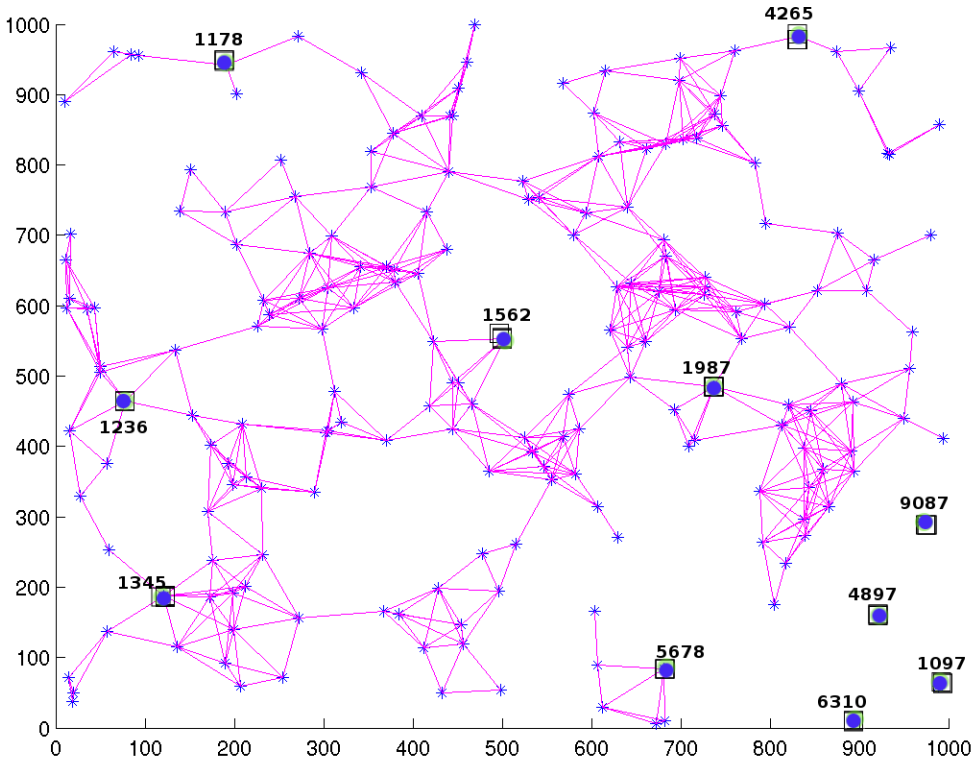

Figure 5.2: Initial state with servers

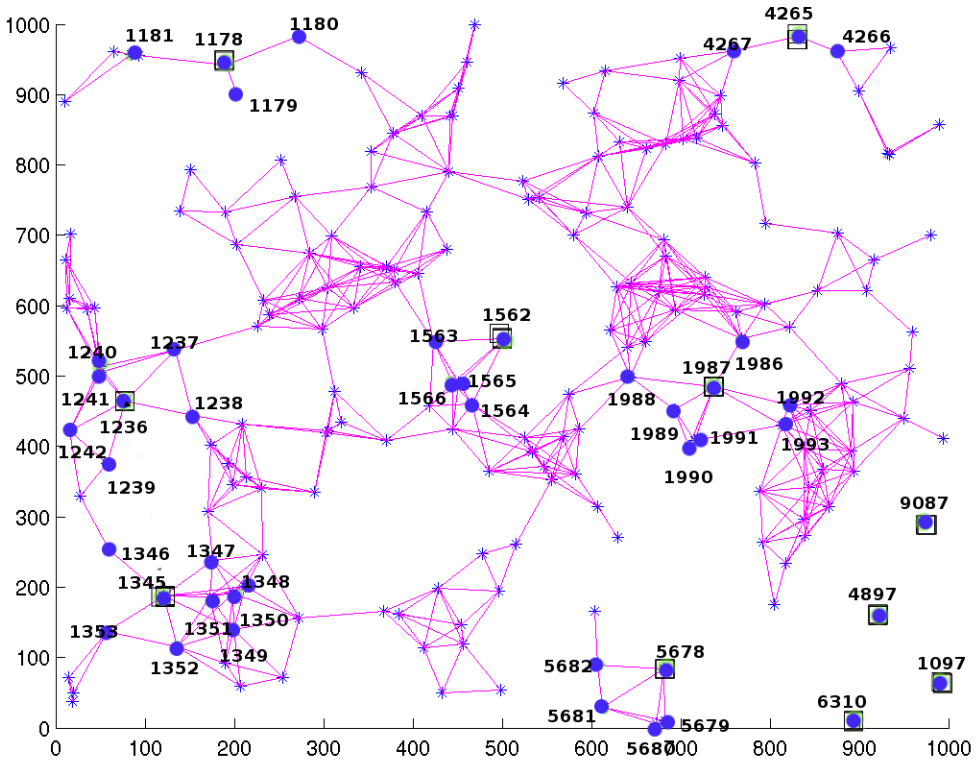

Figure 5.3: First Hop Clients Configured

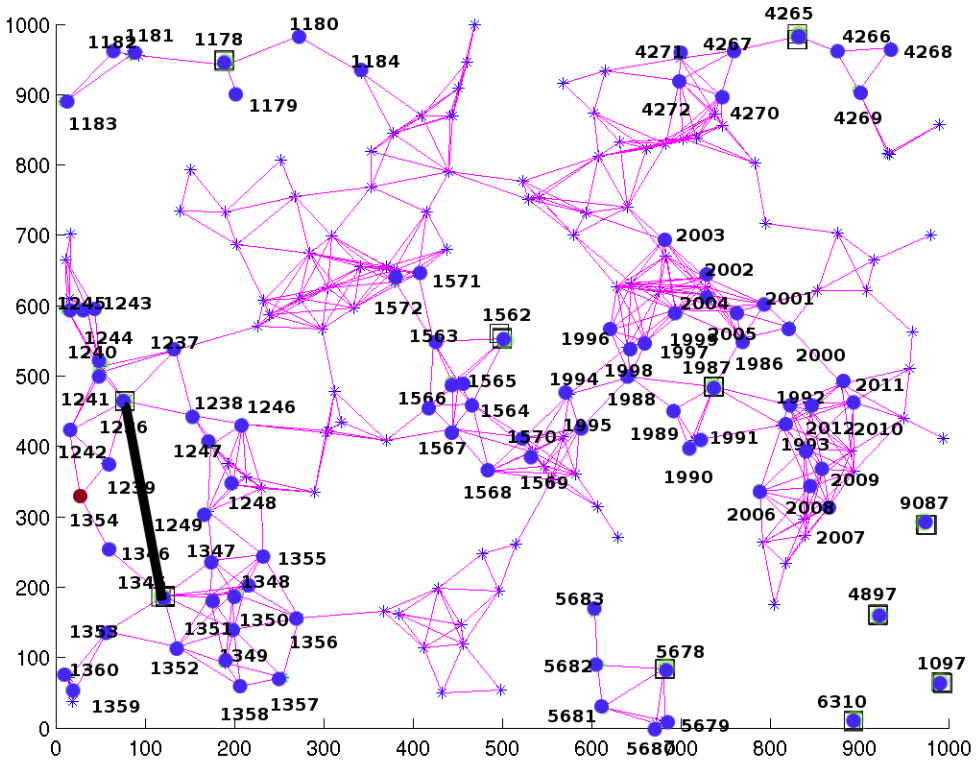

Figure 5.4: Second Hop Clients Configured and First Two Server Discover

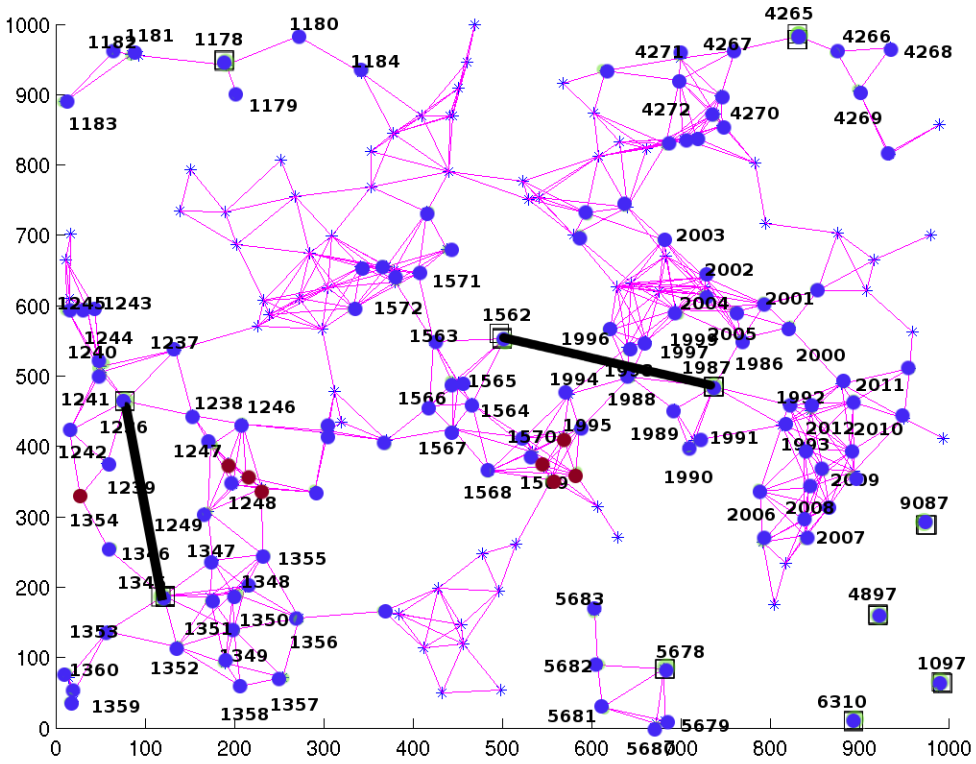

Figure 5.5: Third Hop Clients Configured Boundaries Detected Servers Discovered

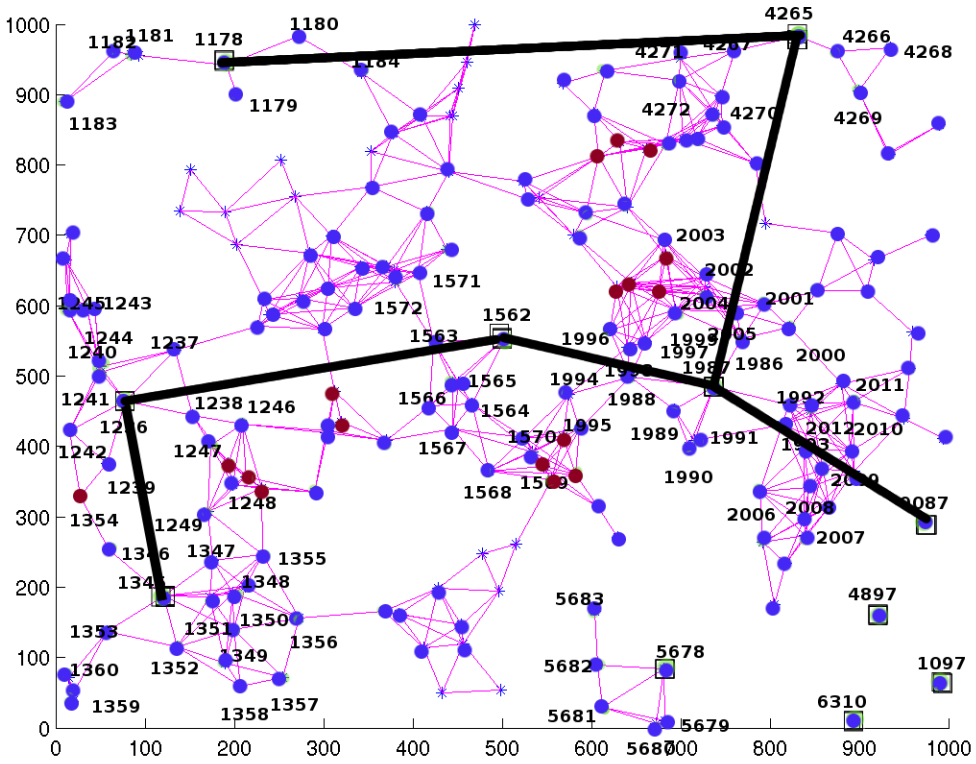

Figure 5.6: Fourth Hop of Clients Configured

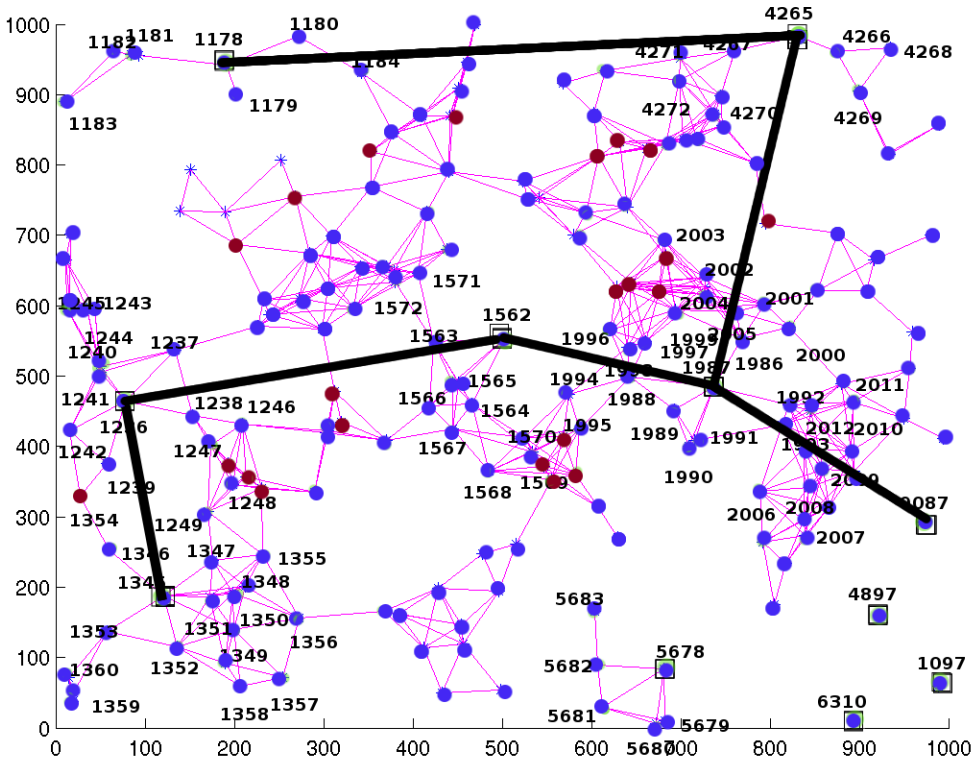

Figure 5.7: Fifth Hop of Clients Configured

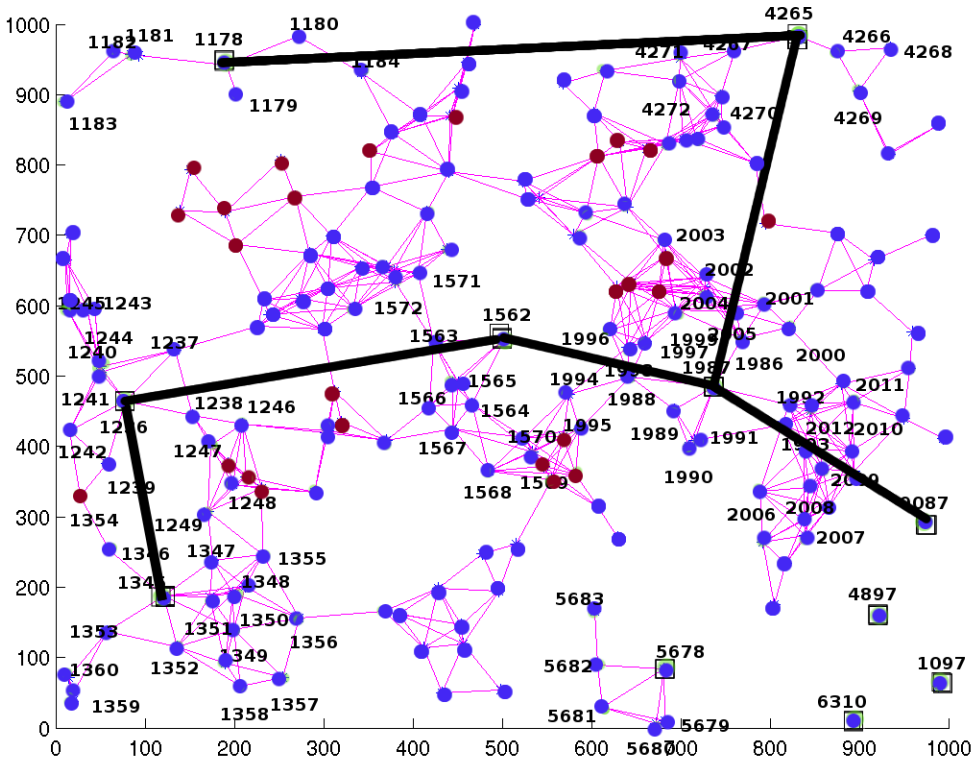

Figure 5.8: All clients configured

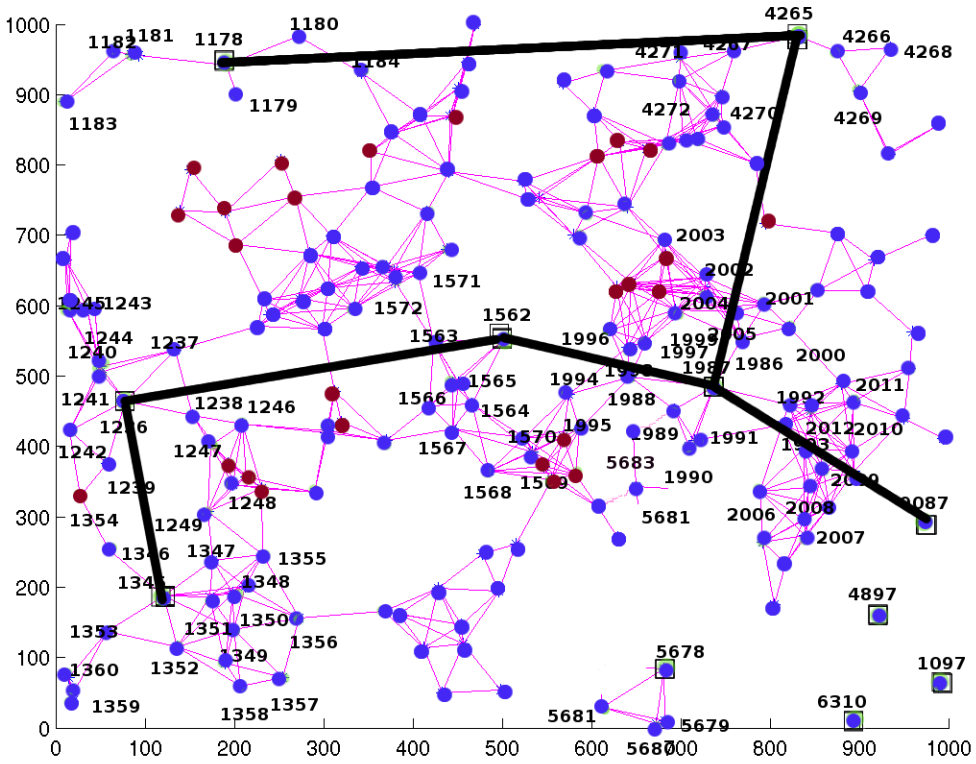

Figure 5.9: Movement of a client

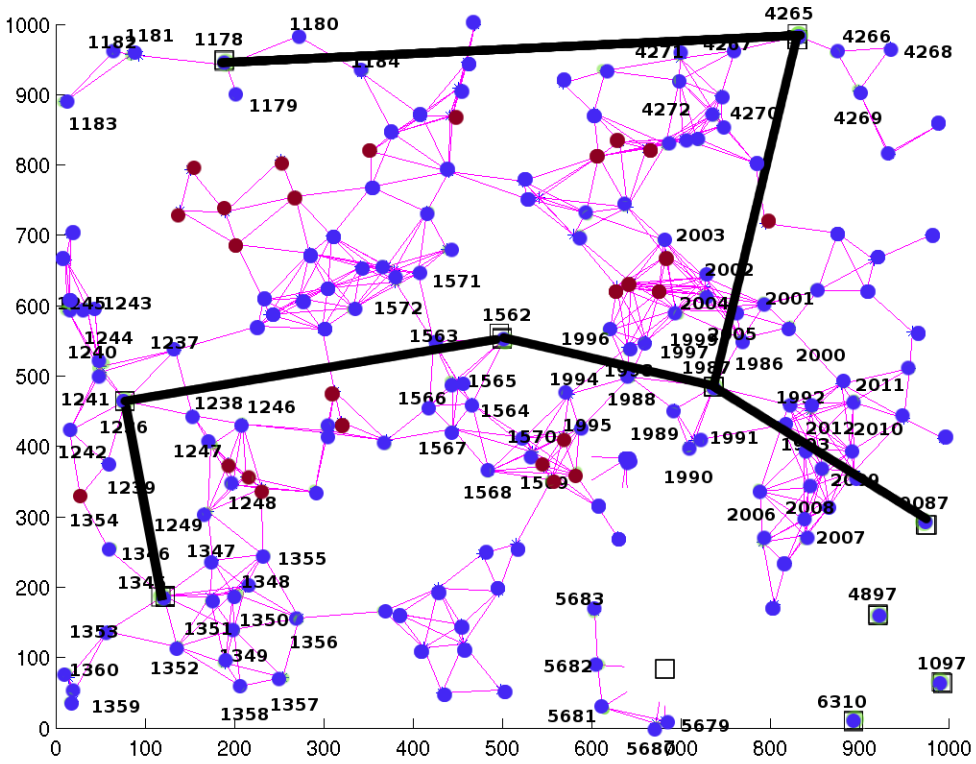

Figure 5.10: Movement of a server

If there are no neighbors at all, then it can force one of its next hop nodes to become a server and hand over its database to it, along with its ID. One example scenario is depicted in Figure 5.10

If a server does not leave voluntarily, then its neighbor set changes. Lets say that Node ID - 2346 has neighbor clients 2563, 2564 and 2367 and it contains IP-MAC translations of all of these as well as for 100's more. If it moves to a different region, its neighbor set may change, and these nodes may have a route to it as well, but they may be registered with some other server. Thus, the route re-discovery may have to be done. If it leaves out of the network, then all translation are all lost. In that case, no node in the network will have the current MAC-IP translations, but the existing assignments would still remain valid.

Chapter 6

Conclusions and Future Work

For a large MANET of wireless devices in a battlefield, it would not be uncommon for network partitions and mergers to occur. For networking amongst devices, being able to uniquely identify or number each interface is necessary and to be able to participate in services, access to a name resolution service is necessary. A large MANET would also participate with the current internet through last hop wireless or wired interfaces. Proposed solutions should be compatible with protocols used in the Internet.

An extensive survey of the literature has been done for available name resolution protocols. From that it has been evident that they do not sufficiently address the problem of the lack of service reachability in the event of network partitions and mergers. Robust service reachability appears as a computationally intensive problem, wherein mobility exacerbates the situation. This thesis exposes the problem in detail.

In order to further study possible solutions, one of the existing name resolution protocols has been adapted to work in a MANET and modeled using the network simulator J-Sim. Through the various simulations of node joins, search for translations using the model, insight has been gathered into the problem that passive stabilization creates. This has led to identifying specific counterexamples wherein it fails to maintain consistency during network partitions and mergers. Further, examples of how an active join and leave technique helps is described.

In order to gain understanding of how such a name resolution service would work in a MANET, IP address and auto-configuration solutions such as DRCP have been looked at.

DRCP is a one hop configuration protocol that assume that interfaces have formed into IP links. This is perhaps a simplistic assumption for a dynamic MANET environment and is more suitable for last hop cellular networks and wireless LAN environments. By allowing DRCP clients to act as relay agents, a manually worked out example of how the service architecture and auto-configuration would work in a MANET has been described.

For this yet unsolved problem, future work includes, modeling of the problem in a graph theoretic manner; a more intensive study of proposed protocols through simulations; analysis of solutions in terms of their computational complexity and heuristics that could address a lot of the issues and finally result in a more robust protocol.

Bibliography

[1] J. P. Macker and M. S. Corson, "Mobile ad hoc networking and the IETF," *SIGMOBILE Mobile Computing Communication Review*, vol. 2, no. 2, pp. 9–12, 1998.

[2] "IEEE Standard 802.11-1997 Information technology- telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements-part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications," 1997.

[3] S. Pandey, S. Dong, P. Agrawal, and K. Sivalingam, "A hybrid approach to optimize node placements in hierarchical heterogeneous networks," in *WCNC 2007: In Proceedings of IEEE Wireless Communications Networking Conference*, IEEE Press, 2007.

[4] F. Templin, S. Russert, I. Chakeres, and S. Yi, "MANET autoconfiguration." IETF Draft, February 2007.

[5] R. Droms, "Dynamic host configuration protocol." RFC 2131, 1997.

[6] P. Mockapetris, "Domain names - concepts and facilities." RFC 1034, 1987.

[7] P. Mockapetris, "Domain names - implementation and specification," 1987.

[8] J. Postel, "Domain name system structure and delegation." RFC 1591, 1994.

[9] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials*, pp. 72–93, 2005.

[10] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the ACM SIGCOMM '01 Conference*, (San Diego, California), August 2001.

[11] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)." RFC 3174, 2001.

[12] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, (London, UK), pp. 329–350, Springer-Verlag, 2001.

[13] P. Maymounkov and D. Mazires, "Kademlia: A peer-to-peer information system based on the XOR metric," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, (London, UK), pp. 53–65, Springer-Verlag, 2002.

[14] D. C. Plummer, "An ethernet address resolution protocol or converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware." RFC 826, 1982.

[15] X. N. Agency, "Earthquakes in taiwan disrupt regional telecommunications," 2006.

[16] B. Wire, "As effects if taiwan earthquake continue to disrupt internet communications, companies find buisness continuity with akamai," 2007.

[17] J. R. Minkel, "Could the internet fragment? [domain name system]," *Spectrum*, vol. 43, no. 6, pp. 20–21, 2006.

[18] A. Cherenson, "nslookup." University of California, Berkeley, 1985.

[19] J. Postel, "Internet protocol darpa internet program protocol specification." RFC 791, 1981.

[20] F. Tobagi and L. Kleinrock, "Packet switching in radio channels: Part ii–the hidden terminal problem in carrier sense multiple-access and the busy-tone solution," *IEEE Transactions on Communications*, vol. 23, no. 12, pp. 1417–1433, 1975.

[21] K. Park, V. S. Pai, L. Peterson, and Z. Wang, "CoDNS: Improving DNS performances and reliability via cooperative lookups," in *OSDI 04: Proceedings of the 6th Symposium on Operating Systens Design and Implementation*, pp. 199–214, USENIX, 2004.

[22] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *USENIX 2004, Procedings of the Annual Technical Conference*, pp. 127–140, USENIX, 2004.

[23] P. Vixie, S. Thomson, and Y. Rekhter, "Dynamic updates in the domain name system (DNS UPDATE)." RFC 2136, 1997.

[24] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol." RFC 3261, 2002.

[25] B. Ford, "Unmanaged internet protocol: Taming the edge network management crisis," in *Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, (Cambridge, Massachusetts), ACM SIGCOMM, November 2003.

[26] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Morris, "Persistent personal names for globally connected mobile devices," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI '06)*, (Seattle, Washington), November 2006.

[27] M. T. Goodrich, M. J. Nelson, and J. Z. Sun, "The rainbow skip graph: a fault-tolerant constant-degree distributed data structure," in *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, (New York, NY, USA), pp. 384–393, ACM Press, 2006.

[28] N. Harvey, M. Jones, M. Theimer, and A. Wolman, "Efficient recovery from organizational disconnects in skipnet," in *In Proceedings of Second International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Springer-Verlag, 2003.

[29] S. Galli, R. Morera, and A. McAuley, "An analytical approach to the performance evaluation of mobility protocols: The handoff delay case," in *VTC 2004-Spring: In proceedings of the 59th Vehicular Technology Conference*, pp. 2389–2393, IEEE, 2004.

[30] S. Galli, R. Morera, and A. McAuley, "An analytical approach to the performance evaluation of mobility protocols: the overall mobility cost case," in *PIMRC 2004: In proceedings of the International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 3019–3024, IEEE, 2004.

[31] A. McAuley and R. Morera, "Adapting DNS to Dynamic Ad Hoc Networks," in *MILCOM 2005: In proceedings of Military Communications Conference 2005*, pp. 1303–1308, IEEE, 2005.

[32] A. McAuley and R. Morera, "Name and address decoupling in support of dynamic networks," in *MILCOM 2002 In proceedings of Military Communications Conference 2002*, pp. 970–975, IEEE Press, 2002.

[33] S. Galli, R. Morera, and A. McAuley, "LNS-SID mobility management in dynamic ad hoc networks," in *VTC 2003-Fall: In proceedings of the 58th Vehicular Technology Conference*, pp. 1994–1998, IEEE, 2003.

[34] X. Li, J. Misra, and G. Plaxton, "Concurrent maintenance of rings," Tech. Rep. TR-04-03, University of Texas at Austin, Austin, Texas, 2004.

[35] X. Li, *Ranch: A Dynamic Network Topology*. PhD thesis, University of Texas at Austin, 2004. TR-04-36.

[36] H. Zhou, L. Ni, and M. Mutka, "Prophet address allocation for large scale MANETs," in *INFOCOM 2003: Proceedings of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 1304–1311, IEEE, 2003.

[37] A. McAuley, K. Young, S. Corson, and K. Manousakis, "Survivable mobile networking," in *ATIRP 2001: Proceedings of Advanced Telecommunications and Information Distribution Research Program*, pp. 4.1–4.36, University of Maryland, 2001.

[38] A. Mcauley and K. Manousakis, "Self-configuring networks," in *MILCOM 2000 21st Century Military Communications Conference Proceedings*, pp. 315–319, IEEE Press, 2000.

[39] A. Mcauley, A. Misra, L. Wong, and K. Manousakis, "Experience-with autoconfiguring a network with IP addresses," in *MILCOM 2001. Military Communications Conference Proceedings*, pp. 272–276, IEEE Press, 2001.

[40] H. Y. Tyan, "J-sim, version 1.3. http://www.j-sim.org," 2004.

[41] C. Perkins and E. Royer, "Ad hoc on demand distance vector routing," in *WMCSA '99:Proceedings of the Second IEEE Workshop of Mobile Computing Systems and Applications*, pp. 90–100, IEEE, 1999.

[42] D. Johnson, D. Maltz, and J. Broch, *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, ch. 5, pp. 139–172. Addison-Wesley, 2001.

Appendix

List of Acronyms

AODV AdHoc OnDemand Vector

ARP Address Resolution Protocol

BSSID Basic Service Set Identifier

CoDNS Collaborative Domain Name System

DAD Duplicate Address Detection

DCDP Dynamic Configuration and Distribution Protocol

DDNS Dynamic Domain Name System

DHCP Dynamic Host Configuration Protocol

DHT Distributed Hash Table

DNS Domain Name System

DRCP Dynamic Rapid Configuration Protocol

DSR Directed Source Routing

HOPCNT Hop Count

ID Identifier

IF Interface

IP Internet Protocol

LAN Local Area Network

LNS Logical Name System

MAC Medium Access Control

MANET Mobile Ad Hoc Network

MAXHOPS Maximum Hops to a Configuration Server

p2p Peer to Peer

SHA Secure Hash Algorithm

SIP Session Initiation Protocol

UIP Unmanaged Internet Protocol

WWW World Wide Web