2009

Hierarchical routing in MANETs using simple clustering

Adam Carnine
University of Nevada Las Vegas

HIERARCHICAL ROUTING IN MANETS

USING SIMPLE CLUSTERING

by

Adam Carnine

Bachelor of Science, Computer Science
University of Nevada, Las Vegas
1999

Master of Business Administration
University of Phoenix, Las Vegas
2008

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science
School of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
December 2009

UNLV
UNIVERSITY OF NEVADA LAS VEGAS

THE GRADUATE COLLEGE

We recommend that the thesis prepared under our supervision by

Adam Carnine

entitled

Hierarchical Routing in MANETS Using Simple Clustering

be accepted in partial fulfillment of the requirements for the degree of

Master of Science
Computer Science

Ajoy K. Datta, Committee Chair

John T. Minor, Committee Member

Yoohwan Kim, Committee Member

Emma Regentova, Graduate Faculty Representative

Ronald Smith, Ph. D., Vice President for Research and Graduate Studies
and Dean of the Graduate College

December 2009

ABSTRACT

Hierarchical Routing in MANETS Using Simple Clustering

by

Adam Carnine

Dr. Ajoy K. Datta, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

This thesis presents both a review of current MANET routing protocols and a new MANET routing algorithm. The routing protocols reviewed include representative samples from the three primary forms of routing found in MANETS: proactive routing, reactive routing and hybrid routing. Secure algorithms are given special treatment in the review. In addition several protocol enhancements are discussed.

The proposed routing protocol is designed to support networks of a medium size, containing over 200 nodes but less than 3,000 nodes. The design is intentionally simple to allow ease of implementation in comparison with other MANET protocols that provide similar functionality.

Keywords: MANET, MANET routing, proactive routing, reactive routing, hybrid routing, clustering, mobile ad hoc network.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

Many people are responsible for getting me to the position of writing this thesis. I must acknowledge Dr. Ajoy K. Datta for his assistance in getting me back on track to complete this thesis and my degree program. Dr. Datta was also kind enough to serve as my examination committee chair.

My thanks also go out to the other members of my examination committee. Dr. John T. Minor, Dr. Yoohwan Kim and Dr. Emma Regentova. Without their time and feedback this document would never have been completed.

A thank you goes out to Ed Jorgensen for his help with this thesis, primarily the feedback on the early basis research. A thank you to Kurtis Kopf for his assistance with proof reading.

I must also thank my loving family for all of their support while I was working on this thesis, and while I was in class, and just for being around to remind me of why sacrificing time is important in the strive for greater things.

Without all of these people this thesis would not have become a reality.

CHAPTER 1

INTRODUCTION

This thesis presents a new algorithm for routing in a mobile ad hoc network (MANET). This algorithm provides the ability to grow the MANET from two hundred nodes up to three thousand nodes though the use of clustering. The algorithm does not provide routing, but rather allows for the use of different routing protocols on top of the clustering protocol. The organization of this thesis is as follows: Chapter 1 provides a brief introduction and background information, Chapter 2 gives an in depth look at the current state of MANET routing protocols, Chapter 3 explains the problem and shows what clustering provides, Chapter 4 gives a detailed description of the protocol and Chapter 5 presents conclusions and future work.

1.1 Background

A MANET is a cooperative network that is formed via wireless connections between several "MANET nodes." Any computing device with a wireless connection is potentially a MANET node, whether that device is a full fledged desktop system, a laptop system, a mobile phone, a mobile internet device, an embedded sensor, or other device with a wireless connection and processing capability. The network requires no fixed infrastructure and is self configuring, thus requiring little administrative effort.

One challenge in the development of MANET applications is ensuring that the underlying MANET routing protocol both functions correctly and provides adequate routing performance, at least for the given MANET scenario. Traditionally proving the adequacy of a routing protocol is accomplished by using a simulation environment. The most prevalent simulation environments today are Network Simulator 2 [1], Global

Mobile Information Systems Simulation Library [2], OPNET Modeler [3], or an ad hoc simulator developed and specifically designed to test a given MANET protocol or scenario.

Alternatively, a researcher may use an exhaustive proof that covers each case of a routing algorithm. While this method does not provide concrete results that the algorithm will work in practice, the method does show that the algorithm will work in theory.

1.2 Outline

Chapter 2 gives a comprehensive background on routing in a MANET. This is broken up into multiple sections. These sections are section 2.1 on proactive routing, section 2.2 on reactive routing, section 2.3 on hybrid routing, section 2.4 on routing protocol enhancements and section 2.5 on security in MANET routing.

Chapter 3 introduces the problem that is addressed by this thesis, namely the inability of standard MANET routing protocols to scale to large numbers of nodes. Section 3.1 presents the high level overview of a proposed solution to this problem.

Chapter 4 presents the simple clustering algorithm. Section 4.1 gives the assumptions made by the protocol. Section 4.2 shows the bootstrapping of the protocol. Section 4.3 gives the initial formation of clusters, while section 4.4 gives the maintenance procedures for a cluster.

Section 4.5 gives a detailed example of cluster formation in a sample network showing the cluster merging in the absence of movement.

Section 4.6 gives the detailed protocol description which includes in section 4.6.1 the roles that apply to each node in the network, in section 4.6.2 the data that is maintained at each node, and finally, section 4.6.3 the detailed description of the packet handling by

each node in the MANET. The handling is based on the type of packet and the roles of the node.

Chapter 5 presents the conclusions of this thesis and proposed future work.

CHAPTER 2

MANET ROUTING BACKGROUND

Routing packets in a MANET is one of the central problems of MANET design. If the routing of packets fails that is the equivalent of failure of the MANET, even if the nodes in the MANET continue to function. MANET routing is based upon a variety of algorithms and currently MANET routing uses three basic approaches to route packets.

One approach is a proactive protocol where each MANET node maintains a local copy of a full routing table for the MANET. Another approach is to use a reactive protocol where each route is built on demand and only maintained while data is actively traveling across the route.

A third approach is a hybrid protocol that combines both proactive and reactive behavior. This combination generally involves partitioning the network into small areas. The behavior of the routing is based on the location of the source node and the destination node. The routing of a packet inside of one of the networks areas is done via a proactive routing protocol. When the packet must cross between areas of the network a reactive routing protocol is used.

Regardless of the routing protocol that is chosen for a MANET, improvements are available. These improvements further enhance the performance of routing in the MANET. Usually these improvements involve “link break” scenarios, but as will be shown, some improvements target other areas of the MANET. A link break occurs when two nodes that were previously in communications are no longer able to communicate for any reason.

Security is another aspect in routing. Some security is added after-the-fact to a routing protocol whereas other routing protocols are designed from the ground up with security in mind. Because of the multiple types of secure protocols: proactive, reactive, or added onto existing algorithms; these protocols will be discussed separately.

2.1 Proactive Routing Protocols

Proactive routing protocols work by distributing routing information amongst the nodes of the network actively through periodic updates. This allows any given source node to have an immediate route to any destination node.

2.1.1 Destination Sequenced Distance Vector (DSDV)

This algorithm was developed by Perkins and Bhaghat in 1994 to provide a simple Layer-2 protocol for routing in a MANET. The purpose of DSDV is to have all of the nodes in the MANET maintain a next hop table for each destination in the MANET. The entries in this table are coordinated by Media Access Control (MAC) addresses instead of using the Layer-3 network addresses. This requires that the routing tables contain both the network address and the MAC address for each node.

The DSDV protocol is based on a combination of the distance vector and the distributed Bellman-Ford algorithm [9]. The revelation demonstrated in [4] was to add sequence numbers to each of the routes stored in the routing table. The sequence numbers allow a MANET node to determine the “freshness” of a route and, therefore, the reliability of that route. The freshness of a route is how recently a packet was successfully relayed along a route.

The DSDV protocol provides two ways of maintaining routes, the first is a full dump of the routing table from a neighboring node, second is an update from a neighboring

node. Further, if the current node has not received a broadcast from a neighboring node within a protocol-specified time, the current node assumes a link break has occurred with the neighboring node.

An incremental update will be performed at regular intervals based upon the current number of changes in the routing tables of a given node. When a MANET node determines that the size of the changes in the nodes routing table surpass a specified amount, typically the amount of information that is broadcast during a network update (perhaps as little as a single packet), then a full dump will be scheduled.

The full dump update is a complete copy of the routing table of a node. This type of update is an "expensive operation." An expensive operation means that the number of packets that must be broadcast to complete the update is large in comparison to either an incremental update or standard traffic on the network. Since the full dump update is expensive, due to the potential size of the routing tables and is, therefore, not broadcast often. This broadcast is done based on the size of the incremental updates and the last time that a full dump update was done.

2.1.2 Octopus

The Octopus protocol falls into a category of proactive routing protocols that requires location services. The goal of the Octopus protocol is to provide fault tolerance when the network has a number of "unstable" nodes. A node is considered unstable if that node connects and disconnects from the network at random intervals [5]. This connecting and disconnecting is caused by problems at the node, either internal to the node or from the environment around the node.

Octopus requires that nodes know information about location, and is based on dividing the network area into a grid containing horizontal and vertical strips. A node will always know the current location as longitude, latitude, horizontal strip and vertical strip. Further a node determines if a neighbor has disconnected if no reply is received to the Octopus location update "HELLO" packets in two successive intervals.

A HELLO packet is a specialized packet that is transmitted by a node to find out information about neighboring nodes. HELLO packets generally have a Time To Live (TTL) of one hop.

The Octopus protocol consists of three subprotocols, location update, location discovery and a forwarding protocol. The location update disseminates information about the location of nodes throughout the network and is initiated by the "border" nodes. A border node is a node that is located at the extreme north, south, east or west of the defined network area such that there is no node further in the border nodes direction. This means for a northern border node in a given vertical strip, no node will be further north in that strip. The location discovery protocol attempts to locate a node by broadcasting the location query request both north and south in the source nodes strip. If no reply is received within a timeout the source node will broadcast another query in the east and west directions. If no reply is again received the source node will assume the destination node is no longer online. The final subprotocol, the forwarding protocol, is invoked once the destination node location has been discovered. This protocol uses a greedy algorithm to forward packets to the next node that is geographically closest to the destination. In the case that a node is a local geographic maximum the node will forward the packet to an alternative target located within the destination nodes strip.

2.1.3 Wireless Routing Protocol (WRP)

Developed in 1996, WRP was one of the first algorithms to break with the traditional development of MANET routing protocols; namely using existing wired network protocols as a base and then extending those protocols onto wireless networks. WRP is not classified as a true MANET routing protocol due to some assumptions made in the definition of the network, such as "input and output queues with unlimited capacity" [6]. The goal of WRP is for nodes to exchange routing information as a means of both keeping an up to date view of the network and of determining the current local topology of the network. If a node does not receive an update for a current neighbor, *n*, within the "router dead time" then that node will remove *n* from the routing table. The fact that *n* was removed will then be included in the next router update message generated by the node.

The paper by Murthy & Garcia-Luna-Aceves gives a correctness proof for the algorithm, a complexity analysis, and simulation results that compare WRP to the best wireless routing algorithms that existed in 1996. The conclusion of the paper was that WRP was better suited to routing in a wireless network when compared with the Distributed Bellman-Ford Algorithm (DBF) [7,8,9], Open Shortest Path First (OSPF) [10], Border Gateway Protocol (BGP) [11], and Diffusing Update Algorithm (DUAL) [12]. DUAL is a component of the Enhanced Interior Gateway Routing Protocol (EIGRP) [13].

2.2 Reactive Routing Protocols

2.2.1 Dynamic Source Routing (DSR)

DSR is a reactive protocol that was proposed by Johnson and Maltz in 1996 [14]. The DSR protocol design attempts to remove some of the control overhead in the network. To this end DSR does not have HELLO packets that are seen in other protocols such as the Ad-hoc On-Demand Distance Vector protocol. DSR consists of two sub-protocols, Route Discovery and Route Maintenance.

Route Discovery is the method whereby a source node, *s*, obtains a route to a destination node, *d*. During this phase the source node is known as the "initiator" and the destination node is known as the "target." The initiator will broadcast a route request that has a unique identifier that is determined by the initiator. As this route request propagates towards the target a route is built inside of the route request. This is done because before rebroadcasting a route request the identifier of the intermediate node that is rebroadcasting the packet will be added to the route in the packet.

Once the target receives a route request from the initiator the target will unicast a route reply back to the initiator. When the initiator receives this route reply the route is setup and can be used for transmitting data.

Route maintenance is performed when a link break occurs on an active route. Once a link break has been detected a Route Error (RERR) packet is sent back to the initiator. When the initiator receivers a RERR packet the initiator can either use another route or initiate a new route discovery process.

2.2.2 Ad-hoc On-Demand Distance Vector (AODV)

The specification of AODV is available in Request for Comment (RFC) 3561 [15]. AODV was developed by taking several features from DSR and DSDV and combining these features into a new protocol. The AODV protocol works similarly to DSR in that a route is only built when a route is required and works similarly to DSDV in that route requests contain sequence numbers. AODV also uses HELLO packets to get information about node neighborhoods.

This HELLO packet is used to determine the local neighborhood; these packets are also used to detect link breaks in the neighborhood. The link break is detected if the current node does not receive a HELLO packet within a configurable amount of time from a previously known neighbor. In this case, that neighbor will be removed from the neighborhood.

When a route is required, the local node will broadcast a Route Request (RREQ) packet. The RREQ packet will be forwarded by all neighbors and will eventually reach every node in the network, assuming the network is "connected." A network is considered connected if a route exists between each pair of nodes in the network. Once the destination node receives a RREQ packet, a Route Reply (RREP) packet will be generated and unicast back along the path that the successful RREQ packet had taken.

This reverse path is available because the RREQ packet is modified at each hop to include the previous node. This means that when the RREQ packet reaches the destination a full path back to the source is included in the RREQ packet.

AODV maintains the routing table based on expiration times. The larger the network the longer the expiration times must be. When a route is cached by a node the route will

have an expiration time associated. If that expiration time is reached and no further packets have been relayed along that route, then that route will be deleted from the routing table.

2.2.3 AODVjr, AODV Simplified

AODVjr is a simplified version of AODV that removes many items from the AODV specification. The goal was to take AODV and make the algorithm easier to implement. The following items are removed from AODV: Sequence Numbers, Gratuitous Route Reply (RREP), Hop Count, HELLO packets, Route Error packets (RERR), and Precursor Lists. Further modifications to the AODV protocol are required to produce the AODVjr protocol. Only the destination node is allowed to send a RREP packet. Maintenance is modified to only update a cached route upon the receipt of a packet using that route. The source detects a "route break" when the source fails to receive a packet from the destination after a given timeout [16].

2.2.4 Power-Aware On-Demand Routing Protocol (PAOD)

The goal in PAOD is to maximize the system lifetime of the MANET, in other words how long before the first node in the MANET suffers a failure due to power loss [17]. PAOD makes three assumptions: a node knows the amount of energy remaining, the energy cost of sending a packet, and the source node knows the number of packets that will be transmitted along the requested route. Based on this information a node determines if the node should participate in a route. The basic operation of PAOD is similar to DSR. A node chooses not to participate in a route if the node determines that participation in that route will deplete the node of energy and thus cause a failure.

2.2.5 Greedy On-Demand Routing Scheme Using Location Information (GOLI)

GOLI is a location aided protocol that operates by getting the identifier and location information of the neighbors of the source node only when that information is required to build a route. Further GOLI makes an assumption, similar to Greedy Perimeter Stateless Routing (GPSR) [18] and Location Aided Routing (LAR) [23], that the source node will know, in advance, the identifier and approximate location of the destination node. Similar to Octopus, GOLI uses greedy forwarding to advance the route discovery process after determining the location information of the "1-Hop neighborhood." The *k*-Hop neighborhood is defined as all of the nodes that reside within *k* hops of the given node.

GOLI avoids a typical problem with greedy algorithms in wireless networks, namely that the next node that will be chosen to forward packets is at the edge of the radio range of the current node. This is a problem because when nodes pass out of radio range a link break occurs and additional overhead is incurred in maintaining the route. To solve this problem GOLI defines a threshold that is within the radio range of the current node and if any node is between the threshold and the maximum radio range that node will not be considered for forwarding [19].

2.3 Hybrid Routing Protocols

2.3.1 Zone Routing Protocol (ZRP)

ZRP is the first hybrid MANET routing protocol and was proposed by Zygmunt Haas in 1997 [22]. The novel idea presented by Haas involved using both proactive and reactive routing, in the same protocol.

The main goal in ZRP is to adjust the sizes of the zones relative to the characteristics of the network. For instance, the size of the proactive area of the network is adjusted in

proportion with the speed of the nodes. Thus at very high speed the proactive zone should have a radius of one hop, which is the equivalent of having a purely reactive protocol. As the speed of the nodes decreases, the number of hops for a zone is increased, approaching infinity as the speed of the nodes goes to zero. When the number of hops is infinity the network is the equivalent of a fixed network with purely proactive routing.

In ZRP each node is a member of many local zones, since each node maintains the localized information. If the number of hops of the zone is *k*, then the node will be in the zones of all nodes within *k* hops.

2.3.2 Way Point Routing (WPR)

WPR involves clustering a network into segments. The source and destination nodes will run a high level inter-segment routing protocol, whereas the nodes in a given segment will run a low level intra-segment protocol. The paper that presented WPR did so by using DSR as the inter-segment routing protocol and AODV as the intra-segment protocol [20].

WPR differentiates from other hierarchical routing schemes by only maintaining the hierarchy for active routes, unlike alternatives Cluster-Head Gateway Switch Routing (CGSR) [21] or ZRP. The clustering inside of WPR is done by determining the segment length. At a segment length of 1 hop the protocol will behave exactly as the intra-segment routing protocol or if the segment length is infinity then WRP will behave as the inter-segment routing protocol. The WRP protocol allows the source (start) node to determine the segment length during the route request.

Because of the segmented nature of the route, if a link break occurs only the segment that contains the link break need be rebuilt, versus the typical action taken by many

MANET protocols of rebuilding the entire route. This allows the protocol to have some graceful error recovery and achieve higher goodput.

2.4 Routing Protocol Enhancements

Some research has focused on improving the behavior of existing protocols. This type of research does not yield new algorithms, but rather strategies that are used to improve existing protocols. The enhancements presented in this section are not mutually exclusive and may be implemented side by side to enhance a single existing protocol.

2.4.1 Encounter Age Caching

A source node generates a "directional" route request by caching encounter ages of encounters with other nodes in the network. The concept is that the source node does not look for a route to the destination, but rather looks for a node that encountered the destination more recently than did the source node.

Fresher Encounter Search (FRESH) is an example of an algorithm that uses encounter age caching. The FRESH encounter based search algorithm can be implemented on top of any algorithm that does a network wide search for a node [24]. The FRESH approach involves finding "anchors" on the route, where each anchor is a node that has more recently encountered the destination node. The algorithm will perform the search by using "concentric ring searches" until the next anchor is found. A concentric ring search involves sending out search packets with an increasing TTL normally starting at two hops and increasing until the target node is found.

The FRESH algorithm is designed for large scale networks and will not be suitable in smaller networks since the cost of the concentric ring searches will be larger than the cost

of a single global search of the network. Additionally a side effect of the anchor finding is that the route to the anchor will be setup in the process.

2.4.2 Non-Optimal Route Suppression

During route discovery, intermediate nodes "overhear" the route replies of neighboring nodes. In many cases, an intermediate node determines that a given route request packet will result in a non-optimal route and thus acts to suppress the route to the source. This eliminates overhead in the network by removing some of the control packets that are created during the normal route discovery process. This technique has been applied to DSR in [25] and was originally proposed in [26]. A node overhears a route reply message before receiving the route request message for a given route. By noting that a reply has already been generated the intermediate node suppresses the initial route request and saves some control overhead in the network.

2.4.3 Bloom Filter Service Discovery

This enhancement provides a way for the MANET protocol to piggyback information about available network services into route discovery packets. A Bloom Filter uses a known hash function for each available service and combines, using bitwise OR, the results of each hashed service value into a single-bit array. This array is included with route discovery requests thus spreading the information about services available on the network.

By combining the information about available services into the route discovery process, some of the overhead of discovering network services, such as domain name service (DNS) servers or internet access nodes, is eliminated. This does not completely remove the need for a MANET node to attempt to discover a service directly, but this

discovery request is avoided if the node learns of the service by participating in a route discovery for another node where the newly requested route has existing information about a network service. For more information about this enhancement see [27].

2.4.4 Abstraction of Bidirectional Routes

Bidirectional Routing Abstraction (BRA) [28] is a method for allowing the simulation of bidirectional links in a MANET where some of the links are unidirectional. The reverse link is established over a short loop back involving at least *r* nodes where *r* is specified in the algorithm setup. BRA is not a completely transparent layer that is added, but rather specifies the reverse links with a weight, since a reverse link might consist of up to *r* nodes, whereas the upper level algorithm expects such a link to have the same weight as the obverse link (IE a link from node A to node B is expected to have the same weight as a link from node B to node A). Also specified in [28] is a derivative algorithm, Dynamic-BRA, where the *r* constant is no longer fixed but is dynamic based on the properties determined by each node in the MANET. Because of the existence of unidirectional links, upwards of 30% in any given MANET [28] , this enhancement should be considered for any MANET.

2.4.5 Route Caching

The goal behind route caching is for a MANET node to maintain a route in a cache until the route is invalidated. This invalidation may be due to the reception of a link break from an upstream node, other times the route being specifically invalidated by the source or the destination, or the route may be expired from the cache explicitly through the use of a timer.

Beraldi & Baldoni developed a caching scheme for ZRP in [29] that does not rely on the traditional timer method for route expiration from a cache. Instead, the cache is proactively maintained within a zone such that when a node that is a member of the zone receives a link break on node *n*, then that node, after validating the link break on node *n*, will broadcast a message to the zone to delete any route that contains node *n*. This caching scheme works well in any MANET where nodes are subdivided into either zones or clusters, but will cause broadcast storm problems in any undivided MANET.

2.4.6 Chase Packets

The goal of chase packets is to minimize the route discovery overhead by partitioning the network into two regions: the immediate neighborhood of a node and the "beyond neighborhood" [30] of a node. The route request packets will travel at full speed in the neighborhood and will have a slight propagation delay in the beyond neighborhood. The source will send a second "chase" packet to follow the route request immediately after receiving a route reply. This chase packet will catch the route request in the beyond neighborhood and will terminate the broadcast thus saving on route discovery overhead. This algorithm relies on defining the neighborhood such that the partitioning of the network allows for chase packets to catch a stale route request packet; for example, the node should not define the neighborhood to be the diameter of the network.

2.4.7 Route Compaction

A route compaction algorithm is given in [31]. The goal of route compaction is to remove intermediate nodes in a route when the source node transmits a packet that bypasses one or more hops in the route. This bypass is discovered via broadcast. Route compaction is a typically employed in a subset of MANETs where the nodes use

directional antennas for wireless connectivity, since a MANET node that uses an omni-directional antenna will not create such paths. The primary method for broadcast in a MANET with directional antennas is to use a "sweeping broadcast" whereby the antenna is swept through a circle sending out a broadcast once per sector. The sectors are determined by the angle that is reached during a single directional broadcast. The compaction is performed by the source or by any node that is part of the final route. One important thing to note is that route compaction will not find a shorter route, route compaction eliminates hops on the current route.

2.4.8 Swarm Intelligence

Swarm intelligence mimics the foraging behavior exhibited in lower life forms, such as insects, as a routing model to be emulated by the network in order to more efficiently route packets. Many papers have proposed algorithms that are inspired by the behavior of ants [33, 41, 42], pheromone (chemical) trails [32, 34, 36], and swarm intelligence [35, 37, 38].

An ant will leave a trail of pheromones [39] while looking for food. This trail is used to guide future ants along a path that is more likely to lead to food. Further, the chemical in the trail degrades over time so that if no ant goes down a given trail, then eventually that trail is no longer considered to be better than any other trail [40].

A MANET employing swarm intelligence adopts a concept similar to using pheromones by keeping track of which network links are receiving the most traffic. Thus a broadcast may be limited by introducing artificial delays based on the pheromone count for a given outgoing link. Additionally, an algorithm exists that uses the amount of traffic going across a link to estimate network congestion and provide alternate paths [41].

Any network that implements this type of routing enhancement will inherently generate multiple paths between a given source and destination. Additional enhancements based on this property turn naïve ant based routing into a protocol that generates disjoint paths [42]. Two paths are considered disjoint if the only common nodes are the source node and the destination node.

2.4.9 Localized Error Recovery

Localized error recovery is a method to improve routing by attempting to repair a link break. This repair is done by the node that detected the link break before that node sends a routing error message back to the source. When a link break is detected along a route to the source, the detecting node was attempting to forward a packet along the route. This node will attempt to find an alternate route by doing a quick localized search for a route to the destination [43].

Once a route has been repaired in this manner the node should send a special packet to the source to indicate that a repair was done on the route. The source determines how many route repairs will be tolerated before a fresh route must be generated. This is important because each repair will make the current route less optimal. Contrast this strategy with AODV with Backup Routing (AODV-BR) [44] where, in addition to the localized repair, the node will send a RERR packet to the source which will force the regeneration of the route after a single repair.

Many other forms of localized error recovery exist including using information from the 2-hop neighborhood such as Neighborhood Aware Source Routing (NSR) [45], multiple route caching, and multiple route generation. Nodes may also operate in promiscuous mode. This allows the nodes to overhear portions of a route and thus giving

additional network information that may be used to generate alternate routes in the case of a link break.

2.4.10 Global Positioning System (GPS) Enabled Nodes

Location Aided Routing (LAR) is any routing protocol that involves the use of node position. Node position is determined by using GPS information, or in the absence of GPS, using localized information such as relative previous location.

The goal of LAR is reduction in the overhead of determining an initial route. This is accomplished by the definition of an "expected zone" for the destination node, *d*. The expected zone is defined as the area where the *d* is located based on the previous location of *d,* the amount of time that has elapsed and the average velocity of *d* [23].

The source node, *s,* will broadcast the route request to a "request zone." The request zone is a rectangular area that contains the expected zone. The location of *s* determines the size of the request zone. If *s* is within the expected zone then the request zone is the square that has the expected zone inscribed within. If *s* is outside of the expected zone then the request zone is the smallest rectangular area that contains both the source and the expected zone.

By minimizing the area where the route request is broadcast the overhead of the broadcast is reduced. For example, if a node, *n,* receives a broadcast packet and *n* is not within the request zone indicated in that packet, then *n* will drop the packet rather than propagating packet.

Though LAR is not a stand-alone protocol, the use of location in general increases the effectiveness of the broadcasts of existing protocols [46].

Other protocols overlay an artificial grid on top of the expected area of the MANET [47]. The grid is then used when making route discovery decisions based on the location of the source node. Alternatively, the grid may be used to create clusters based on geographic boundaries, such as a single grid cell denoting a cluster [48], or perhaps a group of cells may be used.

2.5 Secure Protocols

None of the above mentioned protocols specify security measures. Security may be added in the form of enhancements to existing protocols. Other times security is the basis for the protocol and drives the implementation. Because of the uniqueness and importance of algorithms that deal with security in a MANET, these routing protocols are presented here in a separate section.

Security in ad hoc routing is based on three principles: Availability, Confidentiality and Integrity. Availability deals with network services that should always be available, and the trust required by a node that the service is not malicious. Confidentiality is a principle that means the data sent from the source will only be interpreted by the destination. Finally, Integrity means that data is received at the destination in the same format as the data was sent by the source; the data does not change in transmission [49].

2.5.1 Trust-Aware Routing Protocol (TARP)

TARP is a security protocol based upon DSR that deals primarily with the availability of network resources. The crux of TARP is the use of several metrics in determining the suitability of a route including software configuration, hardware configuration, battery power, credit history, exposure and organizational hierarchy [49].

The DSR modification is the addition of four bits to the route request packet that includes two bits for minimum battery power required and two bits for software encryption capability. TARP does not specify which encryption mode will be mapped to the two bits, only that the modes available must include RSA, DES/3DES, BLOWFISH, IDEA, SEAL RC2/RC4/RC5/RC6 [49].

2.5.2 Reliable Ad-hoc On-demand Distance Vector Routing Protocol (RAODV)

The primary difference between RAODV and AODV is that RAODV attaches a trust metric to each node in the MANET. This protocol is an enhancement to the AODV protocol that adds a second phase to route discovery and guarantees the reliability of a route. This protocol assumes that node impersonation is impossible and will be equivalent to AODV in the absence of any malicious nodes [50].

2.5.3 Secure Efficient Ad Hoc On Demand Routing Protocol (SEAR)

SEAR is another security extension of the AODV protocol. The main goal of SEAR is to secure the route discovery packets and route error packets by the use of "hash chains." A hash chain involves applying the same hash function some multiple of times, the chain length, to a value [51]. Two hash chains are used in SEAR; one for securing sequence numbers and hop counts, and another to secure the route error messages. By securing these two components any node is able to determine if a received packet is authentic or if the packet was modified by an attacker [52].

2.5.4 Ariadne

This protocol is designed to secure the network level above the MAC level of the wireless network. Ariadne uses one of several different key authentication schemes when setting up the network. These schemes include pairwise shared secret keys, TESLA [53],

and digital signatures. Ariadne further uses per-hop hashing to ensure the nodes in a route are maintained. The route maintenance in Ariadne is based on DSR [54].

CHAPTER 3

THE PROBLEM OF SIZE

As the number of nodes in a MANET grows, the ability to route inside of that MANET is decreased. For proactive protocols, this is because each node attempts to maintain routing information for every node in the MANET. This is difficult due to the memory requirements and due to the control requirements. The number of control packets will increase quadratically based on the number of nodes in the MANET.

Reactive protocols also encounter a problem as the size of the MANET is increased. Since reactive protocols do not maintain the entire network state at the node level, the individual nodes generally do not have problems, for instance, due to memory constraints. The problem with reactive protocols is that the entire route path for any route is entirely contained within the routing packet. As the lengths of these paths increases the packet headers grow. This reduces the ability of the protocol to deliver data and ultimately makes the protocol fail once the path length exceeds a critical number of nodes.

Hybrid protocols suffer in both ways due to the combination of routing protocols. A hybrid protocol operates with more nodes than either a pure proactive or pure reactive routing protocol, but still does not scale well. ZRP, for example, has heavy overlap in zones and the control information is duplicated for each zone.

As the proliferation of devices continues, the potential size of a MANET will continue to increase and a new solution must be obtained.

3.1 Clustering

The solution proposed in this paper is to develop a clustering algorithm that will be independent of the underlying MANET routing protocol. The protocol will use an approach similar to ZRP, where the routing inside of a cluster uses a different protocol from the routing between clusters. Ideally, the clustering protocol may be implemented without requiring modifications to any routing protocol that wished to use clustering, and this algorithm accomplishes that.

The clustering will be done by adding fields to the packet header to indicate different types of packets, such as a cluster control packet or a packet relating to the underlying routing protocol. Further nodes will be modified so that they know the current cluster head, the backup cluster head and maintain several routing tables.

CHAPTER 4

CLUSTER CREATION AND MAINTENANCE PROTOCOL

The cluster creation and maintenance protocol is the heart of this paper and represents a different way of looking at clustering inside of a MANET. The protocol is divided into several pieces including the bootstrapping of the protocol, how a node joins a cluster, how a node determines that there is a disconnect from the "cluster masters", the procedure a node follows once a disconnect has been detected, and finally, how routing is accomplished, both intra-cluster and inter-cluster. The cluster masters are the cluster head node and the backup cluster head node.

Once all of these items have been described, the detailed protocol information will be given. This information will include packet header structure and information about how a node will handle each of the different packet types that will be received during cluster operations.

4.1 Assumptions for Cluster Nodes

This protocol makes some assumptions for the cluster nodes. The first assumption is that each node has a unique identifier. This identifier is generated from some internal information such as a hash of the nodes primary processor identifier and the MAC address from the primary interface of the node.

The protocol also assumes a maximum number of nodes that are in a cluster to be a fixed number of nodes. The protocol has a default soft limit of 50 nodes to a cluster and a hard limit of 75 nodes. The reason for this range is the instability of the network. As nodes are moving the nodes leave the cluster, further nodes have limited resources and at times simply power down. Though the protocol will attempt to keep an accurate count of

the number of nodes in the cluster maintaining an absolute number is not practical. Thus this fuzzy definition of a maximum that will be used when determining if two clusters should merge.

The protocol assumes a cluster head fitness function on each node. The implementation of this function is left out of this paper because many different metrics are considered for fitness and the deployment of the MANET requires different weightings for each input. For example, the protocol may consider things such as total free memory, current node speed, node processing power, time until power down et cetera. The weights on these metrics may be different depending on the type of MANET nodes. For example, a sensor network has minimum node speed but might be more dependent on the time until a node powered down.

The protocol assumes that bidirectional communication exists between each pair of nodes where communication will take place. This means that for each pair of nodes, {a, b}, that if there exists a communication link $a \rightarrow b$ then there also must exist a communication link $b \rightarrow a$. The protocol does not restrict implementation to a strictly direct communication link, but such a link is preferred.

The protocol assumes that packets will be processed in a synchronized fashion at each node. In other words the nodes will have a single receive queue and will process each packet individually from that queue. No parallelization of packet processing will be performed on any node. The network communications are not assumed to be synchronous.

4.2 Bootstrapping the Protocol

When a node first comes to life, that is to say when a node is booted, the node is not a member of any cluster. The node will create a new cluster and be the head of that cluster. Figure 4.1 gives an example network with 13 nodes labeled A through M. This Figure will be used throughout the protocol description to give insight into what is happening in the MANET at each step of the protocol.

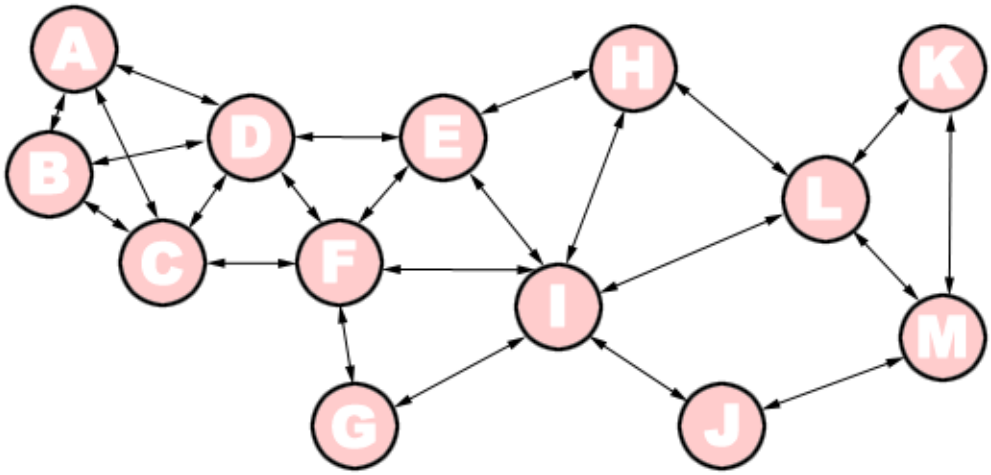

Figure 4.1: Example Network

The nodes are given as a circle with a letter to identify each node. The links are shown as arrows with arrowheads on each end indicating bidirectional communication. Cluster heads are shown with a light fill.

4.3 Cluster Formation

At the inception of the protocol no clusters have been formed and each node is a cluster head in a cluster with a total node count of one. The nodes will each broadcast an

initial Cluster Hello packet (CH). This packet is the basis for determining both the nodes in the cluster and the links between the clusters. Upon receiving a CH packet the node will generate a Cluster Hello Reply packet (CHR) based on whether or not the node is a member of the cluster. If the node is a member of the cluster, then the node will rebroadcast the CH packet and will wait a specified amount of time before formulating a CHR packet. If the node is not a member of the cluster, then the node will not send a CHR packet.

Since the clusters all contain a single node, each node will receive a CH packet from different clusters. Each node will then realize that no other nodes in the neighborhood are a part of the cluster. This is because no neighbor of the node is in the nodes cluster.

The node will send out a Cluster Merge Request packet (CMR). The CMR packet is sent to a cluster gateway and is always forwarded up to the cluster head in the receiving cluster. The receiving cluster head then must make the decision of whether or not to merge with the requesting cluster. If the decision to merge is reached, then the receiving cluster head will send a Cluster Merge Preapproval packet (CMP) back to the original cluster.

Upon receiving a CMP packet the requesting cluster head must now decide to merge or not to merge. If the requesting cluster head decides to merge, then a Cluster Merge Approved packet (CMA) is sent. At this point if requesting cluster head will either be the new cluster head of the merged cluster or will become the new backup cluster head for the merged cluster. If the requesting cluster head will remain the cluster head, then a Cluster Head Backup packet (CHB) will be sent out to the cluster, otherwise the requesting cluster head will become the backup cluster head and will send out a Cluster

Head Takeover packet (CHT). The CHB instructs all current members of the cluster to reset the backup cluster head to be the backup cluster head indicated in the CHB packet. The CHT instructs all nodes to set the backup cluster head to be the current cluster head and to set the cluster head as the cluster head node that originated the CHT packet.

4.4 Cluster Maintenance

Cluster maintenance is performed by the periodic broadcasting of the CH packets and the reception of the CHR packets. The CH packets prove to the cluster nodes that the cluster head is still reachable, provide the latest snapshot from the cluster head of all nodes that are currently in the cluster, and gives the identifier of the backup cluster head. Upon receipt of a CH packet the current node will update the intra-cluster routing table by either reconciling with the node list in the CH packet, or completely rebuilding the table based on the node list in the CH packet. The current node will now generate a CHR packet that contains the identifier of the current node as a cluster member, and a list of all cluster gateway nodes from the current nodes neighborhood table. This will allow the cluster head to have a routing table that contains information on how to reach each neighboring cluster. The current node will now rebroadcast the CH packet to all nodes in the neighborhood.

Upon receipt of a duplicate CH packet the current node will simply drop the packet. The current node tells if a packet is a duplicate because of a sequence number contained in the CH packet.

If the current node does not receive a CH packet from the cluster head, and further did not receive a CHT packet from the backup cluster head, then the current node will assume that due to network changes the current node has become isolated from the

cluster. In this case the current node will reset, as though initially bootstrapping the protocol, and will become the cluster head and backup cluster head of a cluster that contains one node, the current node.

4.5 Detailed Cluster Example

This section will provide a detailed example of the formation of clusters within a small MANET. This example will run through the initial creation of clusters. The network is considered to be stable with respect to clusters when at any time no two clusters may merge. Due to the movement of nodes the network will not remain cluster stable indefinitely, however, for this example node movement will be ignored.

4.5.1 Initial Network Configuration

Initially the network contains only single node clusters. This state is achieved when the nodes in the network boot up for the first time. Figure 4.2 gives the initial state of the network.

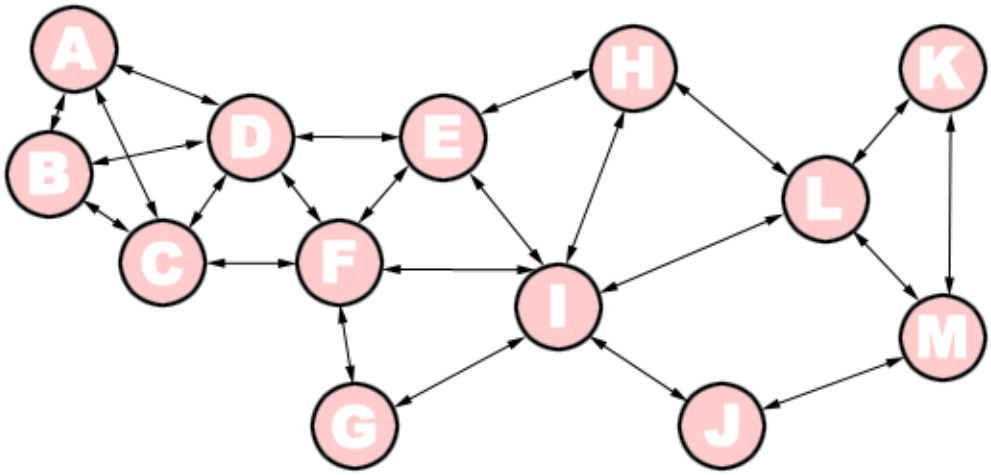

Figure 4.2: Initial Network State

Initially, all nodes are considered cluster heads of a cluster with a size of one. Additionally, all links in the network are cluster gateway links. No merge requests will be made by any node immediately following the initial bootstrapping of the node. Each node will first send out a CH packet which will, in turn, define the nodes local neighborhood. After the initial CH packet, and before sending a second CH packet, the nodes will send a CMR packet.

The CMR packet will not be sent if the cluster size is greater than or equal to the cluster requested size. The cluster requested size is set in the nodes configuration by the operator of the node, or the requested size is set to a default value of 50. The cluster requested size for this example is four nodes. The maximum cluster size, another configuration parameter that is either set by the node operator or by a software default, will be set to seven for this example. In general, the cluster maximum size should be set to a total number of nodes that are easily supported by the intra-cluster routing protocol.

4.5.2 First Merge Requests

The first CMR packets will be sent out by each node before a second CH packet is sent out. If a node has previously accepted a merge request, in other words has sent out a CMP packet, then that node will not generate a CMR packet. Any node that sent out a CMP packet will wait for a CMA packet for two CH intervals. If no CMA packet is received, then the node will send out a new CMR packet. The node will subsequently ignore the CMA from any previously sent CMR.

In the example nodes A, C, E, G, I, K, and M send out CMR packets. Nodes B, D, F, H, J, and L receive these packets and generate CMP packets. Node A receives a CMR packet from node B and sends a CMP packet to node B. Similarly node C receives a

CMR from node D and sends back a CMP. The same is true for the node pairs of E and F, I and H, K and L, and lastly M and J. Node G does not receive a CMP packet and remains a cluster with a single node.

Figure 4.3 gives the network status after the first CMR packets have been acknowledged with CMP packets, and after the final CMA packets have been sent.

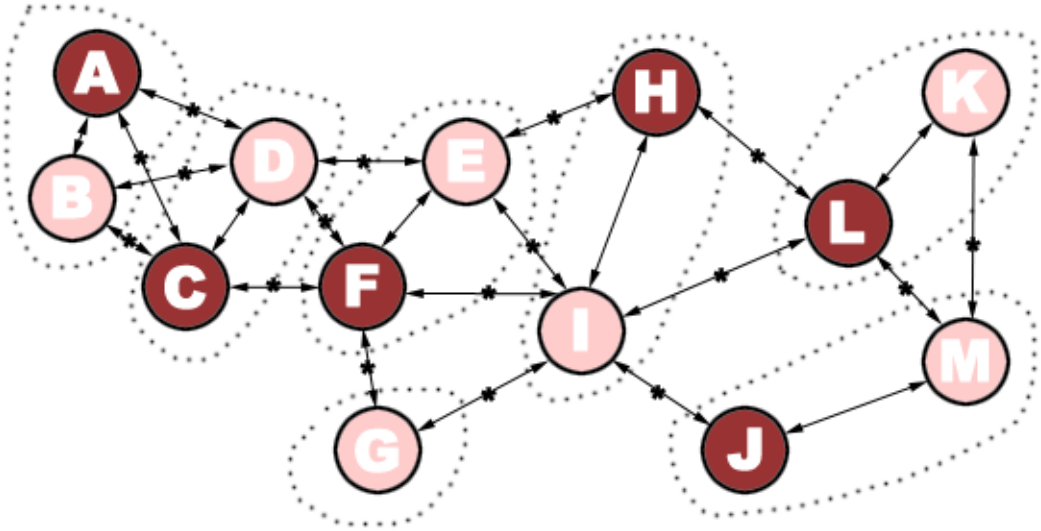

Figure 4.3: Network State after First Merges are Completed

After merging each cluster will contain either one or two nodes. From the example a cluster of nodes A and B exists where node B is the cluster head and node A is the backup cluster head. When two clusters merge one of the cluster heads will be made the backup cluster head of the newly formed cluster. Any existing backup cluster head will be demoted and will simply be a regular cluster node. Cluster gateway links are marked with an asterisk (*).

4.5.3 Subsequent Merge Requests

The merging will continue until the cluster becomes stabilized. In the example, two more merge phases will be needed to achieve cluster stability. In the first round, four nodes will issue CMR packets that will be accepted (generating a CMP packet) and completed (generating a CMA packet). Nodes B, K, G will issue CMR packets that will be accepted by nodes D, M, and I respectfully. Node E will also issue a CMR packet but will not receive a response.

Figure 4.4 gives the network state after the clusters have merged.

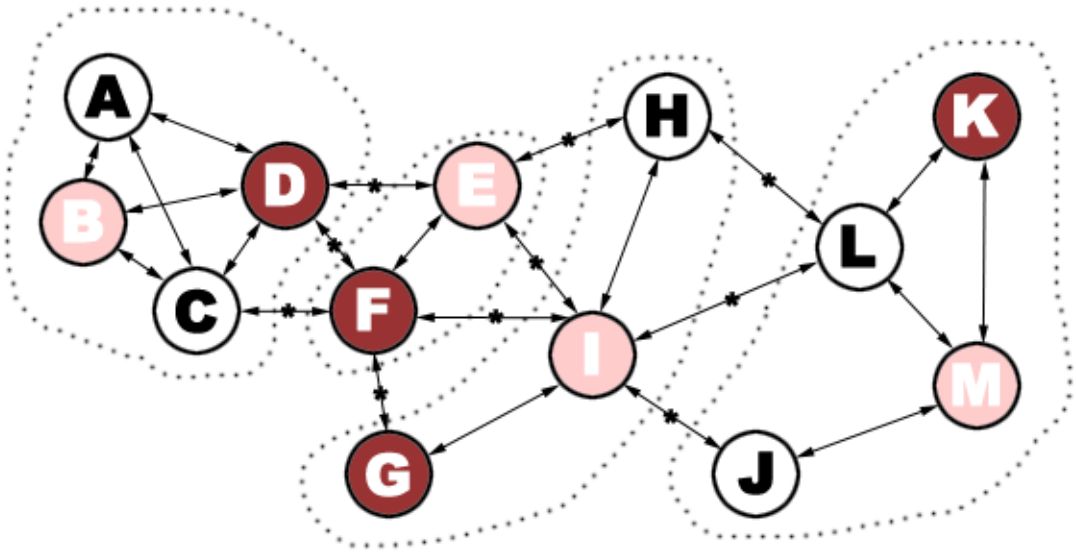

Figure 4.4: Network State after Second Round of Merging

Looking at Figure 4.4, the example now shows four clusters. Two of the clusters now have four nodes. The first cluster with four nodes is the cluster containing nodes B, D, C and A. The second cluster that contains four nodes is the cluster containing nodes M, K, L and J. Because these two clusters have reached the cluster requested size, the

cluster heads M and B will not issue CMR packets. Further, because these clusters have reached the cluster requested size, cluster heads M and B will not send out a CMP packet in response to any received CMR packet.

In the example, one final round of merging is needed to bring the network into cluster stability. In this round, node E will send out a CMR and node I will send back a CMP. Node E will respond with a CMA and then those two clusters will merge producing the final network diagram given in Figure 4.5.

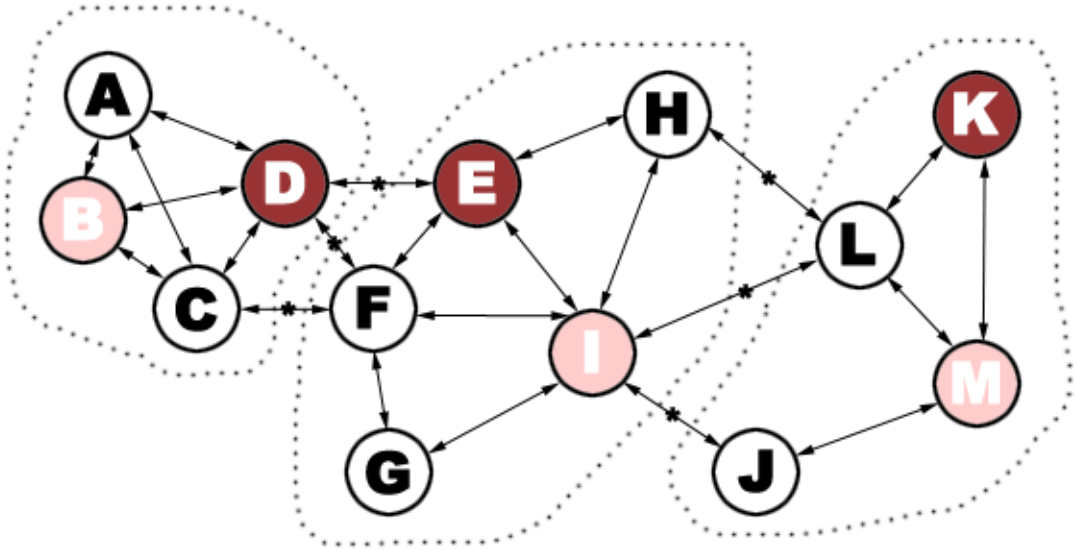

Figure 4.5: Final Cluster State

The final cluster state is achieved when the cluster containing nodes E and F merges with the cluster that contains nodes G, H and I. This final merge shows that the cluster head that initiated a cluster merge may not become the cluster head of the final cluster. This happens based on the results of the cluster head fitness function in nodes I and E, the two cluster heads in the merging clusters.

Also of note is the fact that the final cluster has five nodes. This is more than the requested size of the clusters in the network but less than the maximum size of a cluster. Thus this network configuration is perfectly valid.

4.6 Detailed Protocol Description

This section contains a detailed description of the simple cluster protocol. The protocol is described both by the data structures maintained on the individual nodes of the MANET and by the handling of the various packets defined by the protocol.

4.6.1 Node Roles

Each node in the protocol must have the ability to maintain certain data structures that are appropriate for the roles of that node. Each cluster in the simple cluster protocol will contain nodes that must fill the various roles. The nodes may have one or more roles in the cluster. Assuming that the cluster contains *N* nodes, then Table 1 gives a listing of the various roles that nodes may have in this protocol.

Role	Description	Maximum Number / Cluster of N Nodes
Cluster Head	The cluster head is the master node in the cluster.	1
Cluster Head Backup	The backup cluster head is a mirror of the cluster head and takes over in the event of a cluster head failure.	1
Cluster Gateway Node	A gateway is any node that links between two clusters.	*N*
Cluster Node	A cluster node is any node in the cluster and may also be a cluster head, a backup cluster head, or a gateway node.	*N*

Table 4.1: Node Roles in Simple Cluster Protocol.

Cluster Head. The cluster head node for cluster *K* is responsible for coordinating communication between nodes in cluster *K* and nodes in all other clusters in the MANET. The cluster head will also maintain the list of nodes that are currently in cluster *K*.

The cluster head will maintain two lists of nodes, a list of gateway nodes to reach other clusters, and a list of nodes that are in cluster *K*. The list of gateway nodes will be queried whenever a node inside of the cluster, the source node, needs to route packets to another cluster. The cluster head will be responsible for setting up the route between the source node and the cluster where the destination node resides. The route will be built as a list of cluster hops, where the last cluster hop is the cluster that contains the destination node. The routing between the gateway of the final cluster and the destination node will be handled internally to that cluster.

The list of nodes in the cluster will be broadcast as part of the CH packet. This will allow all nodes in the cluster to know which nodes currently reside in the cluster. Due to the mobile nature of the network and the timing of CH packet broadcasts this list of nodes is merely a best guess estimate based on the results of the previous CH packet broadcast. A more stringent algorithm may be developed to maintain the list of nodes in the cluster, however the goal of this protocol is ease of implementation, and more complexity is intentionally being avoided.

Cluster Head Backup. The cluster head backup is a node that will take over cluster head duties in the event of either an active or passive cluster head failure. Cluster heads fail in two different ways. An active failure results when the cluster head node determines that a failure is imminent, perhaps due to a lack of power. In this active failure case, the cluster head proactively promotes the current backup cluster head to the role of cluster

head. Alternatively, a passive failure occurs when the cluster head fails without opportunity to take action. This may be due to node motion or catastrophic failure of the cluster head node. In this case, the cluster head backup will not have received any traffic from the cluster head for a predetermined number of HELLO periods and will then self promote to cluster head by sending out a CHT packet.

The cluster head backup will maintain the same data structures as the cluster head. These data structures include the nodes that currently reside in the cluster, and the list of gateway nodes to other clusters. The list of gateway nodes to other clusters will be updated periodically from the cluster head via a CS packet.

The cluster head backup provides redundancy so that a failure of the cluster head does not result in the immediate disbanding of the cluster. If the cluster head backup is promoted, then the newly promoted cluster head must choose a new cluster head backup. The choice of a new cluster head backup is done immediately upon the promotion of the new cluster head by a special CHBT packet.

Cluster Gateway Node. A cluster gateway node has at least one neighboring node that resides in another cluster. The only differentiation between a cluster gateway node and a cluster interior node is that the interior node does not have any neighbors in another cluster. The cluster gateway node will forward traffic from the current cluster to the other cluster to which the gateway node is connected.

The cluster gateway node does not actively provide route lookup responses to nodes and is only treated as a gateway.

Cluster Node. The cluster node role applies to all nodes in the cluster. All cluster nodes know the current cluster head, current backup cluster head and have a list of all nodes in

the cluster. Generally, a node is referred to as a cluster node if that node does not perform any of the other cluster node roles such as being a cluster head, backup cluster head, or a cluster gateway node.

4.6.2 Maintained Data

Every node in the cluster is responsible for maintaining a certain amount of data. This data is what allows the nodes to make decisions about how to perform routing both within the cluster and between other clusters. Depending upon the roles of the node some of the data may not need to be maintained. The data to be maintained by each node role is given in Table 4.2.

Data	Role	Description
Cluster Head	All	The current cluster head of the cluster.
Cluster Head Backup	All	The current backup cluster head of the cluster.
Cluster Neighbors	All	All neighbor nodes that are members of this cluster.
Last Hello Sequence Number	All	The sequence number of the last cluster hello packet from the current cluster.
Cluster Nodes	All	The list of all nodes that are currently in the cluster.
Best Hop to Cluster Head	All	The best node to use to send a packet to the cluster head.
Gateway Neighbors	Cluster Gateway	All neighbor nodes that are members of a different cluster.
Cluster Gateway List	Cluster Head / Backup Cluster Head	The list of all gateways to other clusters.
Neighbor Cluster Size	Cluster Head / Backup Cluster Head	The number of nodes in a neighboring cluster.

Table 4.2: Data Maintained by Cluster Node Role

4.6.3 Packet Handling

This section will detail how a node in the cluster will react to each of the different types of packets that are received by this protocol. Table 4.3 gives a listing of all of the cluster specific packets for the simple cluster protocol.

Packet Type	Packet Abbreviation	Usage
Cluster Hello	CH	Cluster/Neighborhood Maintenance
Cluster Hello Reply	CHR	Cluster/Neighborhood Maintenance
Cluster Merge Request	CMR	Cluster Expansion (Generic Request)
Cluster Merge Preapproval	CMP	Cluster Expansion (Locking Response)
Cluster Merge Approval	CMA	Cluster Expansion (No lock, destructive change to cluster)
Cluster head Takeover	CHT	Promotion of a node to cluster head.
Cluster head Backup Takeover	CHBT	Promotion of a node to backup cluster head.
Cluster Sync	CS	Synchronization between the cluster head and backup cluster head.
Low Level Routing	LLR	A packet that contains data to be routed between two nodes in the network.

Table 4.3: Simple Cluster Protocol Packets

The processing of each of these different types of packets is the basis for the Simple Cluster Protocol. These packets each provide a piece of the functionality required for this protocol. Only the Lower Level Routing packet contains data to be routed between nodes in the MANET.

Cluster Hello Packet. The Cluster Hello packet (CH) is the beacon that maintains the cluster. This packet is sent out periodically by the Cluster Head. This packet contains a snapshot of all nodes in the cluster given by the cluster head. Figure 4.6 gives the pseudo-code for how a cluster node will react to receiving a CH packet.

```
1.   function receive_cluster_hello (packet)
2.   {
3.     if (my_cluster != packet.cluster)
4.     {
5.       add_gateway_link (packet.node, packet.cluster);
6.       drop_packet (packet);
7.       return;
8.     }
9.     if (last_hello_seq < packet.sequence)
10.    {
11.      process_hello_packet (packet);
12.    }
13.    else
14.    {
15.      drop_packet (packet);
16.    }
17.  }
18.
19.  function process_hello_packet (packet)
20.  {
21.    last_hello_seq = packet.sequence;
22.    add_cluster_neighbor_node (packet.node);
23.    create_cluster_node_list (packet.node_list);
24.    rebroadcast_packet (packet);
25.    hello_reply = create_cluster_hello_response (packet);
26.    unicast (packet.node, hello_reply);
27.  }
```

Figure 4.6: Pseudo-Code for Reception of a CH Packet

The first check that is made, on line 3 of Figure 4.6, is to see if the CH packet is from the same cluster. If not this means that the node must add a gateway link based on the node and cluster of the packet. Line 9 shows the check to ensure this node has not received this CH packet before and thus this CH packet is valid. Finally, lines 21 to 26 show the handling of the CH packet, including the neighborhood maintenance on line 22, the cluster node list maintenance on line 23, a rebroadcast of the CH packet on line 24, and finally lines 25 and 26 show how the CHR packet is generated and then unicast back to the originating node.

The CH packet for cluster *K* maintains the gateway links in all clusters connected to cluster *K*. Consider a node, *n*, that is not in cluster *K* and receives a CH packet from a node, *m*, in cluster *K*. Since *n* is not in cluster *K*, *n* will not respond to *m* so the link is maintained unidirectionally only. The other direction of the link, from the *m* to *n* will be maintained when a CH packet is sent from *m* to *n*. Alternatively *m* will infer the gateway link if a routing attempt is made using the gateway link from *n* to *m*.

Cluster Hello Reply Packet. The Cluster Hello Reply packet (CHR) is generated by a node, *n*, when *n* receives a CH packet from a node in the same cluster as *n*. The CHR packet for each node is propagated back to the cluster head for that nodes cluster. Figure 4.7 gives the pseudo-code for how a node will handle receiving a CHR packet.

```
1.   function receive_cluster_hello_reply (packet)
2.   {
3.     if (is_cluster_head (packet.cluster))
4.     {
5.       process_hello_response (packet);
6.       return;
7.     }
8.     if (is_backup_cluster_head (packet.cluster))
9.     {
10.      process_hello_response (packet);
11.    }
12.    unicast (best_hop_to_cluster_head, packet);
13.  }
14.
15.  function process_hello_response (packet)
16.  {
17.    add_cluster_node (packet.node);
18.    update_gateway_list (packet.node, packet.gateway_list);
19.  }
```

Figure 4.7: Pseudo-Code for Reception of a CHR Packet

Only the cluster head and the backup cluster head will process a CHR packet. All other nodes will relay this packet to the cluster head. The backup cluster head will process the packet and then relay the packet to the cluster head.

The processing for a CHR packet involves maintenance of two items: the list of nodes in the cluster and the gateway links from the node that generated the CHR packet. On line 17 the node is added to the list of nodes in the cluster. On line 18 the list of all gateway links from that node is updated since a node may be connected to multiple clusters beyond the cluster to which the node belongs.

Cluster Merge Request Packet. The Cluster Merge Request (CMR) packet is the first step in a three step process by which two clusters merge to become a single cluster. Only the cluster head will process the CMR packet. All other nodes in the cluster will relay a CMR packet to the cluster head. Figure 4.8 gives the pseudo-code for handling a CMR packet.

```
1.   function receive_cluster_merge_request (packet)
2.   {
3.     if (is_cluster_head (packet.target_cluster))
4.     {
5.       process_merge_request (packet);
6.       return;
7.     }
8.    unicast (best_hop_to_cluster_head, packet);
9.  }
10.
11.  function process_merge_request (packet)
12.  {
13.    if (currently_merging)
14.    {
15.      drop_packet (packet);
16.      return;
17.    }
18.    new_cluster_size = nodes_in_cluster + packet.nodes_in_cluster;
19.    if (new_cluster_size > MAXIMUM_CLUSTER_SIZE)
20.    {
21.      drop_packet (packet);
22.      return;
23.    }
24.    currently_merging = true;
25.    merge_preapproval = create_merge_preapproval (packet);
26.    gateway_node = get_gateway_node (packet.cluster);
27.    unicast (gateway_node, merge_preapproval);
28.  }
```

Figure 4.8: Pseudo-Code for Reception of a CMR Packet

In Figure 4.8 on line 3 the node checks to see if the node should process the merge request, if not on line 8 the node forwards the merge request to the cluster head.

The cluster head will process the merge request. If the cluster head has already committed to attempting to merge with another cluster, then this merge request is dropped. Similarly, if the total number of nodes in the requesting cluster plus the total number of nodes in the target cluster is greater than the maximum number of nodes allowed in a cluster, then the packet is dropped. Finally, if the cluster head determines

that a merge is possible, then the cluster head sets the "currently merging" flag on line 24 and then sends a Cluster Merge Preapproval packet in lines 25 through 27.

Cluster Merge Preapproval Packet. The Cluster Merge Preapproval (CMP) packet is the second step in the process of merging two clusters. This packet indicates that the target merge cluster has agreed to merge with the cluster that sent the initial merge request packet. The pseudo-code for how a node will handle a CMP packet is given in Figure 4.9.

```
1.   function receive_cluster_merge_preapprove_request (packet)
2.   {
3.     if (is_cluster_head (packet.target_cluster))
4.     {
5.       process_merge_preapprove_request (packet);
6.       return;
7.     }
8.     unicast (best_hop_to_cluster_head, packet);
9.   }
10.
11.  function process_merge_preapprove_request (packet)
12.  {
13.    if (currently_merging)
14.    {
15.      drop_packet (packet);
16.      return;
17.    }
18.    new_cluster_size = nodes_in_cluster + packet.nodes_in_cluster;
19.    if (new_cluster_size > MAXIMUM_CLUSTER_SIZE)
20.    {
21.      drop_packet (packet);
22.      return;
23.    }
24.    merge_approval = create_merge_approval (packet);
25.    gateway_node = get_gateway_node (packet.cluster);
26.    unicast (gateway_node, merge_approval);
27.    clean_up_gateway_links ();
28.    if (my_fitness > packet.target_fitness)
29.    {
30.      cluster_head_backup = packet.cluster_head;
31.      backup_takeover = create_backup_takeover_packet (backup);
32.      broadcast (backup_takeover);
33.    }
34.    else
35.    {
36.      cluster_head = packet.cluster_head;
37.      backup_takeover = create_backup_takeover_packet (this);
38.      broadcast (cluster_head_backup_takeover);
39.      cluster_head_takeover = create_takeover_packet
(cluster_head);
40.      broadcast (cluster_head_takeover);
41.    }
42.  }
```

Figure 4.9: Pseudo-Code for Reception of a CMP Packet

Similar to the CMR packet, the CMP packet is only processed by the cluster head. When the cluster head receives a CMP packet, several checks are performed before the final approval packet is sent out. These checks include a check to see if this cluster is merging with a different cluster, done on lines 13 through 17. A sanity check on cluster size is performed in lines 18 through 23. If these two checks have passed the merge is approved, a CMA packet is generated and sent to the new cluster.

At this point the merge must be completed. Line 27 shows the gateway links are cleaned up, removing any gateway links to the cluster that no longer exist. The first step to finalize the merge is to determine which cluster head will head the new cluster, and which will become a backup cluster head on the new cluster. This is done by checking the fitness function for each of the cluster head nodes. The cluster head with the highest fitness becomes the new cluster head. If both cluster heads have the same fitness, then the cluster head that sent the CMP packet will become the new cluster head, and the cluster head receiving the CMP will become the backup.

If the current cluster head will remain a cluster head, then the only task is to generate a Cluster Head Backup Takeover packet and broadcast that packet to the cluster as is illustrated in lines 30 through 32.

If the current cluster head will become a backup cluster head, then the first step is to assign the new cluster head in the current node shown in line 36. Lines 37 and 38 show that a CHBT packet is generated with the current node becoming the new backup cluster head. Finally in lines 39 and 40 a Cluster Head Takeover packet is generated naming the new cluster head.

Cluster Merge Approval Packet. The CMA packet is the final step in the cluster merge process. When the CMA packet is sent from the target cluster, destructive changes have already been done, and this packet allows the requesting cluster to finalize the merge and perform post merge clean up, as required. Figure 4.10 gives the pseudo-code for the reception of a CMA packet.

```
1.   function receive_cluster_merge_approval (packet)
2.   {
3.     if (is_cluster_head (packet.target_cluster))
4.     {
5.       process_merge_request (packet);
6.       return;
7.     }
8.    unicast (best_hop_to_cluster_head, packet);
9.  }
10.
11.  function process_merge_approval (packet)
12.  {
13.    clean_up_gateway_links ();
14.    if (my_fitness >= packet.target_fitness)
15.    {
16.      cluster_head_backup = packet.cluster_head;
17.      backup_takeover = create_backup_takeover_packet (backup);
18.      broadcast (backup_takeover);
19.    }
20.    else
21.    {
22.      cluster_head = packet.cluster_head;
23.      cluster_head_takeover = create_takeover_packet
(cluster_head);
24.      broadcast (cluster_head_takeover);
25.      sleep (CLUSTER_HELLO_INTERVAL * 2);
26.      backup_takeover = create_backup_takeover_packet (this);
27.      broadcast (cluster_head_backup_takeover);
28.    }
29.    currently_merging = false
30.  }
```

Figure 4.10: Pseudo-Code for Reception of a CMA Packet

Only the cluster head takes action, beyond relaying, based on receiving a CMA packet. The action taken based on the CMA packet is the reverse of the action taken by the cluster head that generated the CMA packet.

Line 13 shows the gateway link cleanup, common to both the CMA and CMR packets. Lines 16 through 18 show the cluster head generating a CHBT packet to promote the other cluster head to a backup cluster head based on the fitness function. Lines 22 through 26 show the current cluster head promoting the new cluster head and then being demoted to a backup cluster head. Finally in line 28 the merging status is cleared.

Cluster Head Takeover Packet. The CHT packet signifies that a new cluster head is taking control. This phenomenon happens under two conditions; the backup cluster head determines the cluster head has failed or two clusters merge, whereby one of the clusters will have a new cluster head. Figure 4.11 gives the pseudo-code for when a node receives a CHT packet.

```
1.   function receive_cluster_head_takeover (packet)
2.   {
3.     if (cluster_head == packet.old_cluster_head)
4.     {
5.       cluster_head = packet.cluster_head;
6.       broadcast (packet);
7.       return;
8.     }
9.     drop_packet (packet);
10. }
```

Figure 4.11: Pseudo-Code for Reception of a CHT Packet

The CHT packet is only processed if the cluster head of the receiving node is the same as the old cluster head listed in the CHT packet. Line 3 shows this check and then in lines 5 and 6 shows that the processing involves changing the current cluster head and rebroadcasting the packet.

Cluster Head Backup Takeover Packet. The Cluster Head Backup Takeover (CHBT) packet is similar to the CHT only instead of applying to the cluster head the packet applies to the cluster head backup. Figure 4.12 gives the pseudo-code for handling the reception of a CHBT packet.

```
1.   function receive_cluster_head_backup_takeover (packet)
2.   {
3.     if ((cluster == packet.cluster) &&
4.         (backup_cluster_head != packet.backup_cluster_head))
4.     {
5.       backup_cluster_head = packet.backup_cluster_head;
6.       broadcast (packet);
7.       return;
8.     }
9.     drop_packet (packet);
10. }
```

Figure 4.12: Pseudo-Code for Reception of a CHBT Packet

Handling of the CHBT packet is similar to how the CHT packet is handled. On lines 3 and 4 the criteria for processing the packet is given as the cluster of the current node must match the cluster of the packet and the backup cluster head of the current node must not match the backup cluster head of the packet. If these two conditions are met, then on line 5 the backup cluster head is set to the new backup cluster head and on line 6 the packet is rebroadcast.

Cluster Sync Packet. The Cluster Sync (CS) packet is used to synchronize the information contained in the cluster head with the backup cluster head. Nodes in the cluster do not maintain a best link to the backup cluster head, due to the fact the backup cluster head does not send out CH packets. The cluster head, however, does have a path to the backup cluster head that was determined by the last CHR packet received from the backup cluster head. Due to this fact the CS packet will contain the full path from the cluster head to the backup cluster head in addition to the data that must be synchronized.

Figure 4.13 gives the pseudo code for the reception of a CS packet.

```
1.   function receive_cluster_sync (packet)
2.   {
3.     if (is_backup_cluster_head (packet.backup_cluster_head))
4.     {
5.       process_sync_packet (packet);
6.       return;
7.     }
8.     next_hop = get_next_hop (packet, node)
9.     if (next_hop != null)
10.    {
11.      unicast (packet, next_hop);
12.      return;
13.    }
14.    drop_packet (packet);
15.  }
16.
17.  function process_sync_packet (packet)
18.  {
19.    update_gateway_links (packet.gateway_links);
20.  }
21.
22.  function get_next_hop (packet, node)
23.  {
24.    if (packet.route.contains (node))
25.    {
26.      return (packet.route.get (node).next_hop);
27.    }
28.    return (null);
29.  }
```

Figure 4.13: Pseudo-Code for Reception of a CS Packet

The CS packet is processed only if the current node is the backup cluster head. This is shown on line 3 and 5. The processing is shown in lines 17 through 20. If this node is not the backup cluster head, then this node pulls the next hop from the CS packets embedded route to the backup cluster head in lines 22 through 29 and sends a unicast of the CS packet to that next hop on line 11.

Low Level Routing Packet. The Low Level Routing (LLR) packet is an encapsulation of a lower level routing protocol. This packet will contain a route to the destination node.

This route will be either an intra-cluster or inter-cluster route. An intra-cluster route is entirely within a single cluster, whereas an inter-cluster route spans multiple clusters. The low level routing protocols used do not need to be modified. Figure 4.14 gives the pseudo-code for a node that receives an LLR packet.

```
1.   function receive_low_level_routing (packet)
2.   {
3.     if (this_node == packet.destination)
4.     {
5.       process_data_packet (packet);
6.       return;
7.     }
8.     next_hop = get_next_hop (packet, node)
9.     if (next_hop != null)
10.    {
11.      unicast (packet, next_hop);
12.      return;
13.    }
14.    error (packet);
15.  }
16.  function process_data_packet (packet)
17.  {
18.    response = generate_response_packet (packet);
19.    unicast (packet, packet.last_hop);
20.  }
21.  function get_next_hop (packet, node)
22.  {
23.    if (packet.is_intracluster && packet.route.contains (node))
24.    {
25.      return (packet.route.get (node).next_hop);
26.    }
27.    if (packet.is_intercluster)
28.    {
29.      next_node = packet.route.get (cluster).next_cluster_node;
30.      next_hop = lookup_route (next_node);
31.      if (next_hop == null)
32.      {
33.        next_hop = find_next_hop (next_node);
34.      }
35.      return (next_hop);
36.    }
37.  }
38.  function lookup_route (next_node)
39.  {
40.    // Find the next_node in the routing table
41.  }
42.  function find_next_hop (next_node)
43.  {
44.    // Use the intercluster routing protocol to find the next hop
45.  }
```

Figure 4.14: Pseudo-Code for Reception of a LLR Packet

When the LLR packet is received the first check that is done is to determine if the current node is the destination node. Figure 4.14 shows this in line 3. Line 8 determines if this node finds the next hop, if so the packet is forwarded in line 11, otherwise an error handler is called in line 14. The error handler is specifically generic as this may be as simple as dropping a packet or may be a more complex thing such as localized error recovery.

Getting the next hop for the LLR packet is more complex than for any of the other packets in this protocol. The first check is to determine if the route is intra-cluster, and if this node is contained in the route, done on line 26. If both of these things are true, then the next hop is retrieved from the packet and returned on line 28.

If the packet is an inter-cluster packet, then the next node is contained in the packets route based on the cluster of the current node. The node will get the next node from the packet on line 32 and then check to see if the next node is in the local routing table on line 33. If the next node is not in the local routing table then the node will attempt to find the next node on line 36.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Many areas of research remain incomplete regarding the presented cluster protocol. This section presents the conclusions of this paper and outlines some possibilities for future work that should be accomplished to help vet and position the proposed protocol for use in both experimental and production environments.

This paper presented a new idea for a novel MANET routing protocol that allows for mid-range scaling of the number of nodes in a MANET. By providing a clustered approach that does not directly specify the underlying routing protocols, more flexibility is given in the deployment of the MANET. The underlying routing protocols can be chosen suit the specific MANET situation.

This protocol was specified with both a detailed example of cluster organization and with pseudo-code to demonstrate proposed implementation.

Additionally, this paper gave a background of different areas in MANET routing, including the three types of MANET routing protocol: proactive, reactive and hybrid. The proactive approach to routing maintains complete network information at each node so that each node, at any given time, deduces the appropriate neighbor to use when forwarding packets. The reactive approach to routing involves building a route from scratch whenever a source wishes to send packets to a destination. The hybrid protocols allow a combination of both proactive and reactive algorithms, which allows both types of algorithms to be used where they are most effective.

Security in MANET routing protocols is given as a separate section to allow the opportunity to discover some of the ways that a MANET is secured against malicious nodes.

The conclusion of this paper is that clustering is not suited to all possible MANET situations, and is detrimental if the size of the MANET is small. If the MANET contains less than 200 nodes the overhead of the clustering protocol will cause the routing in the MANET to be less efficient. This algorithm is postulated to be effective once the number of nodes exceeds 200, depending upon the size of the clusters. This is due to the increased efficiency of determining a route due to the reduction of flooding in the network.

Simple Clustering provides, in the basic implementation, a hybrid routing protocol where the network can be considered divided into two areas, intracluster and intercluster. Routing in each of these areas can be accomplished via different protocols. One future goal would be to extend the clustering implementation from a single level of clustering to provide *N* levels of clustering.

Future work should include simulations that compare this protocol to other clustering protocols and to a pure flooding based protocol. These simulations can also verify the break even point of the algorithm under various MANET scenarios. An avenue to be explored involves the use of location information to help clusters avoid forming when the links that join the clusters are estimated to be short lasting.

Further research has been proposed involving various methods to improve the performance of routing protocols that are independent of the protocol. Some of these methods included localized link repair, bidirectional route abstraction, route compaction

and chase packets. None of these enhancements route packets, however, but rather the enhancements improve the performance of an existing routing protocol. In some cases this performance is through the reduction of control packets.

BIBLIOGRAPHY

[1] *The Network Simulator 2* [computer program]. Version 2.0. Available from Information Sciences Institute: http://www.isi.edu/nsnam/ns/, 2005

[2] *Global Mobile Information Systems Simulation Library* [computer program]. Version 2.0. Retrieved January 3, 2009. Available from UCLA Parallel Computing Laboratory: http://pcl.cs.ucla.edu/projects/glomosim/, 2000

[3] *OPNET Modeler* [computer program]. Version 15.0 PL3. Available from OPNET Technologies, Inc.: http://www.opnet.com/, 2009

[4] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In ACM SIGCOMM Computer Communication Review, 24(4), pages 234-244, 1994.

[5] R. Melamed, I. Keidar and Y. Barel. Octopus: A Fault-Tolerant and Efficient Ad-hoc Routing Protocol. *Wireless Networks*, 14(6), pages 777-793, 2008.

[6] S. Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications,* 1(2), pages 183-197, 1996.

[7] R. Bellman. *On a Routing Problem.* Quarterly of Applied Mathematics, 16(1), pages 87-90, 1958.

[8] L. R. Ford, Jr. and D. R. Fulkerson. *Flows in Networks*, Princeton University Press, 1962.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* Sixteenth printing, pages 532-535, 1996.

[10] Internet Engineering Task Force. Open Shortest Path First IGP (ospf). http://www.ietf.org/dyn/wg/charter/ospf-charter.html, Updated April 4, 2009, Accessed February 7, 2009.

[11] K. Lougheed and Y. Rekhter. Border Gateway Protocol (BGP). http://www.bgp4.as/. Published June 1989. Accessed February 8, 2009.

[12] J. J. Garcia-Luna-Aceves. *Loop-Free Routing Using Diffusing Computations,* IEEE/ACM Transactions on Networking 1(1), 1993.

[13] B. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle. *EIGRP – A Fast Routing Protocol Based on Distance Vectors,* in Proc. Networld/Interop'94, 1994.

[14] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing,* (353), 1996.

[15] C. E. Perkins, E. M. Belding-Royer, E. M. and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. Internet Engineering Task Force. http://www.ietf.org/rfc/rfc3561.txt. Published July 2003. Accessed February 15, 2009.

[16] I. D. Chakeres and L. Klein-Berndt. AODVjr, AODV Simplified. *ACM SIGMOBILE Mobile Computing and Communications Review, 6*(3), pages 100-101, 2002.

[17] K. Wang, G. Chen and Y. Wu. Power-Aware On-Demand Routing Protocol for MANET. *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops,* (7), pages 723-728, 2004

[18] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking,* pages 243-254, 2000.

[19] H. Nakagawa, K. Ishida, T. Ohta and Y. Kakuda. GOLI: Greedy On-Demand Routing Scheme Using Location Information for Mobile Ad Hoc Networks. In *Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems,* page 1, 2006.

[20] R. Bai and M. Singhal. DOA: DSR over AODV Routing for Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing,* 5(10), 2006

[21] C. Chiang, M. Gerla. Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. *IEEE 6th International Conference on Universal Personal Communications,* (2), pages 546-551, 1997.

[22] Z. J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of Institute of Electrical and Electronics Engineers 6th International Conference on Universal Personal Communications,* (2), pages 562–566, 1997.

[23] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks,* 6(4), pages 307-321, 2000.

[24] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Age Matters: Efficient Route Discovery in Mobile Ad Hoc Networks Using Encounter Ages. *International Symposium on Mobile Ad Hoc Networking & Computing,* pages 257-266, 2003

[25] B. Seet, B. Lee and C. Lau. DSR with Non-Optimal Route Suppression for MANETs. Cornell University Archive Site. http://www.arXiv.org. Updated May 29, 2006. Accessed July 2, 2009.

[26] B. Seet, B. Lee and C. Lau. Optimization of Route Discovery for Dynamic Source Routing in Wireless Ad Hoc Networks. *IEEE Electronics Letters,* 39(22), pages 1606-1607, 2003.

[27] F. Outay, V. Verque and R. Bouallegue. Bloom-filter Based Combined Service and Route Discovery for Mobile Ad Hoc Networks. In *Proceedings of the The 2007 International Conference on Intelligent Pervasive Computing,* pages 188-193, 2007

[28] V. Ramasubramanian and D. Mosse. BRA: A Bidirectional Routing Abstraction for Asymmetric Mobile Ad Hoc Networks. *IEEE/ACM Transactions on Networking,* 16(1), pages 116-129, 2008.

[29] R. Beraldi and R. Baldoni. A Caching Scheme for Routing in Mobile Ad Hoc networks and Its Application to ZRP. *IEEE Transactions on Computers,* 52(8), pages 1051-1062, 2003.

[30] M. A. Al-Rodhaan, L. Mackenzie and M. Ould-Khaoua. A New Route Discovery Algorithm for MANETs with Chase Packets. *International Journal of Simulation Systems, Science & Technology Special Issue on: Performance Modeling of Computer Networks, Systems and Services*, 8(3), pages 1-12, 2007.

[31] V. Kolar, P. Rogers and N. B. Abu-Ghazaleh. Route Compaction for Directional Route Discovery in MANETs. *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications,* 3(22-24), pages 101-108, 2005.

[32] P. B. Jeon and G. Kesidis. Pheromone-aided robust multipath and multipriority routing in wireless MANETs. In *Proceedings of the Second ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks,* pages 106-113, 2005.

[33] M. Umlauft and W. Elmenreich. QoS-aware ant routing with colored pheromones in wireless mesh networks. In *Proceedings of the Second International Conference on Autonomic Computing and Communication Systems,* article no. 31, 2008.

[34] S. Tadrus and L. Bai. A QoS Network Routing Algorithm Using Multiple Pheromone Tables. In *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, page 132, 2003.

[35] C. Shen and C. Jaikaeo. Ad hoc multicast routing algorithm with swarm intelligence, *Mobile Networks and Applications,* 10(1-2):47-59, Feb. 2005.

[36] L. F. M. Vieira, U. Lee and M. Gerla. Phero-Trail: a bio-inspired location service for mobile underwater sensor networks. In *Proceedings of the third ACM international workshop on Underwater Networks,* pages 43-50, 2008.

[37] F. D. Rango and M. Tropea. Swarm intelligence based energy saving and load balancing in wireless ad hoc networks. In *Proceedings of the 2009 workshop on Bio-inspired algorithms for distributed systems,* pages 77-84, 2009.

[38] C. X. Mavromoustakis and H. D. Karatza. Swarm-based Active Tunable Routing for Overhead Reduction in Multiservice Networks. In *Proceedings of the 39th annual Symposium on Simulation,* pages 294-303, 2006.

[39] D. Jackson and F. Ratnieks. Communication in ants. *Current Biology*, 16(15), pages R570-R574, 2006.

[40] S. Goss, S. Aron, J. L. Deneubourg & J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften* 76(12) pp. 579-581, 1989

[41] Z. Liu, M. Z. Kwiakowska and C. Constantinou. A Biologically Inspired Congestion Control Routing Algorithm for MANETs. *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops,* pages 226-231, 2005.

[42] Z. Wu, H. Song, S. Jiang and X. Xu. Ant-based Energy Aware Disjoint Multipath Routing Algorithm in MANETs. *Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering,* pages 674-679, 2007.

[43] A. M. Alshanyour and U. Baroudi. Bypass AODV: Improving Performance of Ad Hoc On-Demand Distance Vector (AODV) Routing Protocol in Wireless Ad Hoc Networks. *Proceedings of the 1st international conference on Ambient media and systems,* (17), 2008.

[44] S. J. Lee and M. Gerla. AODV-BR: backup routing in ad hoc networks. *Wireless Communications and Networking Conference,* (3), pages 1311-1316, 2000.

[45] M. Spohn and J. J. Garcia-Luna-Aceves. Neighborhood Aware Source Routing. *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing,* pages 11-21, 2001.

[46] M. Mohseni, S. Vahedi and M. Naderi. A New Position-Based Routing Algorithm for the Reduction of Overhead in Ad-hoc Networks. *Proceedings of the Second International Conference on Systems and Networks Communications,* page 7, 2007.

[47] R. Friedman and G. Korland. Timed Grid Routing (TIGR) Bites off Energy. *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing,* pages 438-448, 2005.

[48] Z. Wu, H. Song, S. Jiang, and X. Xu. A Grid-based Stable Backup Routing Algorithm in MANETs. *Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering,* pages 680-685, 2007.

[49] L. Abusalah, A. Khokhar, G. BenBrahim and W. ElHajj. TARP: Trust-Aware Routing Protocol. *Proceedings of the 2006 international conference on Wireless communications and mobile computing* , pages 135-140, 2006.

[50] S. Khurana, N. Gupta and N. Aneja. Reliable Ad-hoc On-demand Distance Vector Routing Protocol. *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies,* (23-29), page 98, 2006.

[51] L. Lamport. Password Authentication with Insecure Communication. *Communications of the ACM,* 24(11), pages 770-772, 1981.

[52] Q. Li, M. Zhao, J. Walker, Y. Hu, A. Perrig and W. Trappe. SEAR: A Secure Efficient Ad Hoc on Demand Routing Protocol for Wireless Networks. *Proceedings of the 2008 ACM symposium on Information, computer and communications security,* pages 201-204, 2008.

[53] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *RSA CryptoBytes,* 5(2), pages 2-13, 2002.

[54] Y. Hu, A. Perrig and D. B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks,*11(1-2), pages 21-38, 2005.

VITA

Graduate College
University of Nevada, Las Vegas

Adam Carnine

Home Address:
3205 Broadway Avenue
North Las Vegas, Nevada 89030

Degrees:
Bachelor of Science, Computer Science, 1999
University of Nevada, Las Vegas

Master of Business Administration, Business, 2008
University of Phoenix, Las Vegas

Thesis Title: Hierarchical Routing in MANETS Using Simple Clustering

Thesis Examination Committee:
Chairperson, Dr. Ajoy K. Datta, Ph.D.
Committee Member, Dr. John T. Minor, Ph.D.
Committee Member, Dr. Yoohwan Kim, Ph.D.
Graduate Faculty Representative, Dr. Emma Regentova, Ph.D.