A DHT-based Routing Solution for Hierarchical MANETs

by

Echegini, Ngozi Silas

Submitted to the Department of Systems and Computer Engineering in partial fulfillment for the requirements of Master of Applied Science (MASc.)

in

Carleton University, Ottawa - Ontario

Abstract

This thesis presents an effective routing solution for the backbone of hierarchical Mobile Ad hoc Networks (MANETs). Our solution leverages the storage and retrieval mechanisms of a Distributed Hash Table (DHT) to make routing information available in a decentralized fashion, while supporting different forms of node and network mobility scenarios effectively. We do so by splitting a flat network into clusters, each having a gateway who participates in a DHT overlay. These gateways interconnect the clusters in a backbone network. Two routing approaches for the backbone are explored: flooding, which we use as a base approach, and our solution, which is DHT-based. We compare the performance of our solution against the flooding approach via experimentation in a simulator. Our results show that our DHT-based solution, even in the presence of mobility, achieved above 90% success rates and maintained very low and constant round trip times, which was not the case with the flooding approach. The advantage of our proposed approach increases as the number of clusters increases, demonstrating the superior scalability of our proposed approach.

Acknowledgements

First and foremost, I ascribe all gratitude to God for life, grace, guidance and for seeing me through to the end of my studies here at Carleton University.

Special appreciation goes to my supervisor - Professor Thomas Kunz and co-supervisor - Professor Babak Esfandiari, whose patient encouragement, mentoring and insightful criticism brought me to develop in-dept research skills and enabled me see the beauty of being original.

On a personal note, I remember my loving parents of blessed memories, Late Chief & Lolo S. N. Echegini, who laid the foundation I built upon. I love you both and your strides will ever remain green. I also extend warm regards to my siblings Obioma, Uzochukwu, Chinwe, Nnamdi, Adaku and the Nwafors' who played a key role in my acclimatization. Thank you for being there.

I acknowledge the entire body of RCCG (The Overcomer's Place) Pst/Pst Mrs. Femi Olawale and a host of other friends - Dr. Alan Davoust, Prof. François Gagnon, Alexandre Cormier, Walid Abdel Gelil, Frank Ockenfeld, Rania Darweesh Saleh. Thank you for your support and feedback !

Finally, the research work presented in this thesis was sponsored by the US Army RDECOM-Americas. The importance of this funding is gratefully acknowledged.

Contents

List of Tables

List of Figures

Glossary

INET . . . . Internet Protocol stack library for the OMNeT++ simulation environment *Omnet++*.

Omnet++ . . . . Simulation library and framework, mainly for network simulations.

OverSim . . . . Peer-to-peer and overlay network simulation framework for *Omnet++*.

Acronyms

AODV	Ad hoc On-Demand Distance Vector
CEDAR	Core Extraction Distributed Ad Hoc Routing
CGSR	Cluster head Gateway Switch Routing
DCA	Distributed Clustering Algorithm
DHT	Distributed Hash Table
DMAC	Distributed and Mobility-Adaptive Clustering algorithm
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
HNA	Host and Network Association
HOLSR	Hierarchical OLSR
ICMP	Internet Control Message Protocol
IP	Internet Protocol

LCA	Linked Cluster Architecture
LCC	Least Cluster Change
MAC	Medium Access Control
MANET	Mobile Ad hoc Network
MANETs	Mobile Ad hoc Networks
MCDS	Minimum Connected Domination Set
MPR	Multi-Point Relay
OLSR	Optimized Link State Routing
P2P	Peer-to-Peer
QoS	Quality of Service
RFC	Request For Comments
RPGM	Reference Point Group Mobility
RREQ	Route REQuest
RWP	Random Way-Point
SHA-1	Secure Hash Algorithm 1
STAR	Source Tree Adaptive Routing
TC	Topology Control

TTL	Time To Live
WRP	Wireless Routing Protocol
ZHLS	Zone-Based Hierarchical Link State

Chapter 1

Introduction

Mobile Ad hoc Networks (MANETs) are increasingly gaining popularity and finding applications in a range of areas, including emergency response networks, intelligent transportation systems, outdoor enterprises, small businesses etc. [2, 3, 4]. One important characteristic is that they are self-organizing and self-configuring wireless multi-hop networks which do not rely on any existing infrastructure to exist; as nodes are by themselves, servers and clients [5, 6]. Each node must act as a router to forward traffic unrelated to its own use.

Equipping each device to continuously maintain the information required to properly route traffic is the primary challenge in building a MANET. Each device runs a routing protocol which controls how nodes decide which way to route packets between computing devices in a mobile ad-hoc network. Efficient routing mechanisms in MANETs are a subject of long and deep research and numerous algorithms have been researched in the past, which have become the foundation for MANET routing. Initially, two popular approaches were the pro-active or table-driven routing and the reactive or on-demand routing. The scalability of both approaches is limited owing to their inherent characteristics. These protocols have scalability difficulties when the networks have many nodes and/or span a large geographical area.

Hierarchical architectures are utilized to advance the network scalability in these systems [5, 7]. Large networks of flat topologies are divided into clusters, each having hosts and one or multiple gateways who connect the clusters through a backbone network. Hierarchical architectures have gained broad attention and this approach is also adopted in this thesis.

1.1 Motivation and Research Problem

The number of users in MANET applications may vary from just a handful to hundreds of thousands of people and more [8]. As MANETs and mobile devices become increasingly popular and the ensuing networks grow larger, more research effort focuses on devising protocols for route establishment and maintenance in these networks. In a flat network of several interconnected mobile devices, and spanning a large geographical area, for instance, the network will typically incur increasing overheads for route maintenance and establishment and other network functions. In Optimized Link State Routing (OLSR) for example [9], the Multi-Point Relay (MPR) algorithm was designed to solve the problem of flooding control messages efficiently in the network. Yet when the network size begins to grow, the number of MPRs in the network, which are also nodes, increases. The control message traffic handled by MPRs grows approximately with the square of the number of nodes in the network. Yet MPRs can only handle so much network traffic efficiently since their capabilities are no different from other nodes in the network. Scalability in such networks becomes an issue as bandwidth will be heavily impacted, with a large portion of messages for route maintenance and establishment being lost due to congestion [10, 11].

The scalability limitation is true for almost any MANET routing protocol proposed for flat networks. Many routing protocols for ad-hoc networks are either proactive (table-driven) or reactive (on-demand) [12, 13]. Proactive routing protocols like OLSR or Destination-Sequenced Distance Vector (DSDV) originate from the traditional distance vector and link state protocols. They continuously maintain routes to all destinations in a network, whereas reactive (on-demand) protocols like Ad hoc On-Demand Distance Vector (AODV) or Dynamic Source Routing (DSR) will only seek out routes to a destination when necessary. Both routing protocol approaches scale poorly [12, 13]. This is true because of the inherent characteristics of these protocols [14]: on the one hand, the on-demand routing protocols are limited by their route discovery techniques because of the extensive use of flooding. Hop-by-hop flooding usually has a huge negative impact on network performance and often leads to large delays in route discovery [15, 3]. On the other hand, proactive routing protocols have these routes readily available, but it comes at a cost of constant route discovery throughout the lifetime of the network. It is evident therefore that both protocols have scalability issues, which get even worse in the case that nodes are mobile and links become generally unpredictable [15, 3].

A hierarchical routing architecture, when carefully planned, shows its advantage of simplifying routing tables considerably and lowering the amount of routing information exchanged [16, 4], thus increasing search efficiency and increasing scalability. This is best exemplified by the global Internet, which employs a hierarchical architecture and routing structure. The Internet is divided into routing domains. A routing domain typically contains a collection of co-located networks connected by routers (who are nodes) and linked in a common routing domain called the backbone [4].

In this thesis, hierarchical routing is adopted to tackle the problem of incurring increasing overheads for route maintenance and establishment and other network functions in large MANETs. The following concerns are also taken into consideration in this thesis:

- In the context of this thesis, nodes do not necessarily belong to a single network throughout their lifetime. As nodes are mobile, they may change their cluster membership, clusters may merge or split. So, a more general hierarchical routing architecture that supports various mobility scenarios is desirous.

- In order for hosts within a cluster to route packets destined for hosts in external clusters or domains, there is the need for a protocol or scheme which will be the standard for such applications. OLSR supports a HNA message scheme which is primarily for external access, standardized in Request For Comments (RFC) 3626 [17]. This scheme is relevant for the design this thesis implements.

- Since routing will now be intra-cluster wise and globally (inter-cluster), the gateway nodes will require a different mechanism for packet delivery between clusters, and in an efficient fashion as well. The gateways are interconnected through a backbone. A simple, robust but costly routing solution is to flood all messages through this backbone. The backbone network is of concern as it carries the bulk of the user data and control traffic. Thus, there is the possibility of it becoming a performance bottleneck. To avoid incurring increasing and costly overhead, a more efficient routing scheme will be required.

1.2 Context

In the context of this thesis, our hierarchical Mobile Ad hoc Network (MANET) architecture finds application in such environments like a military environment. In a military context for example, there exist platoons which move in groups and each platoon typically consists of soldiers and probably, a dedicated vehicle (armoured tank/truck). These platoons are MANETs which are then considered as clusters in our study. In such a setting, the number of or size of a cluster is typically not known a-priori. Clusters are typically given and may depend on the number of soldiers or the number of clusters desired in such applications. Furthermore, the members of the different platoons will need to communicate with each other. Communication in such environments is usually coordinated in a more efficient manner than just a flat MANET architecture. Typically, a hierarchical approach is adopted to localize control traffic within the various clusters and improve routing scalability in the backbone. Each platoon has a dedicated gateway (this can be the armored vehicle), and, through the gateways, the various platoons are interconnected in a backbone network. The gateways are more powerful devices equipped with capabilities which will enable them effectively support communication between members of their local cluster and the different clusters.

As such, we make the following assumptions as a consequence of these considerations:

- The number of clusters or the number of nodes per cluster is not a design parameter or constraint. The clusters are given as it would in a military context. Furthermore, we assume even distribution of the nodes per cluster.

- Gateway selection and cluster formation procedures are not trivial in general. In this thesis, there are no special algorithms for gateway selection or cluster formation as we assume the cluster structure is determined by the application domain. Gateways are selected well in advance from nodes which have the capacity to become gateways, while other cluster members might associate with different gateways as they switch cluster membership.
- Energy constraints are likely, given that we are dealing with mobile devices. However, we assume that this is not a problem for our gateways given the context of this thesis work. For example, vehicles have considerable energy resources that we can draw on to perform networking tasks.
- Typically, scalability as it applies to a MANET, corresponds to the ability of the MANET to handle an increasing number of nodes. In this thesis, we adopt a hierarchical network structure and keep the number of nodes constant while varying the number of clusters formed. The more clusters are formed, the more gateways will participate in routing messages in the backbone. Thus, scalability as it applies to our work, corresponds to the capability of the backbone network to handle a growing amount of clusters.

1.3 Contributions

We designed a suitable hierarchical routing solution for large-scale MANETs and evaluated it through simulations in *Omnet++* [18]. The proposed solution proposes to use a DHT as implemented in structured Peer-to-Peer (P2P) overlays to manage mobility and to efficiently support inter-cluster routing. *Omnet++* already provides us with implementations of flat MANET routing protocols such as OLSR, and various P2P protocols employing a DHT such as Chord. We extended these as follows:

- OLSR's HNA optional functionality for external access was implemented as per RFC 3626 [17]. This feature allows gateways to inject external route information to their local OLSR MANET, announcing their reachability to other networks.
- We also designed gateway nodes which will support different routing protocols or instances of the same routing protocol on two separate interfaces.
- A flooding protocol was implemented to investigate the effect of flooding when deployed in a backbone network and was used as a base case for our analysis.
- Finally, a DHT application was implemented to leverage the storage and retrieval mechanism of a DHT to achieve a more efficient routing operation in the backbone.

The extensive simulation results show that our proposed improved inter-cluster routing scheme reduces traffic (and therefore congestion) in the backbone, improves application/routing performance, and successfully manages node mobility.

1.3.1 Publications

The work that has been done for this thesis has also lead to two peer-reviewed publications:

- [19] Ngozi Silas, Thomas Kunz, and Babak Esfandiari. Evaluating chord over a hierarchical manet. In *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2017 8th IEEE Annual*, pages 608–617. IEEE, 2017.

A second publication has been submitted and accepted to a journal :

- Thomas Kunz, Babak Esfandiari, Ngozi Silas and Frank Ockenfeld (2018). P2P Overlay Performance in Large-Scale MANETs. In *International Journal of Communications, Network and System Sciences*

.

1.4 Chapter Overview

The rest of this thesis document is structured as follows: Chapter 2 presents the state of the art on routing in MANET environments. Chapter 3 introduces the reader to the author's work. It explains the design of the routing mechanism and provides other related information. Chapter 4 discusses the *Omnet++* [18] simulation environment used in this work and some of its features. The implementation details and the various modules developed for this work are also presented. Chapter 5 discusses the simulation setups and the results that were obtained. We evaluate the effectiveness of our design from the results we obtain. In Chapter 6 we draw some conclusions about this work and also discuss possible future work in this area.

Chapter 2

Background and State of the Art

This chapter provides an overview of the State of the Art in the field of Mobile Ad-hoc Networking. The main aspects and problems in MANETs are described, including a detailed look at existing routing algorithms.

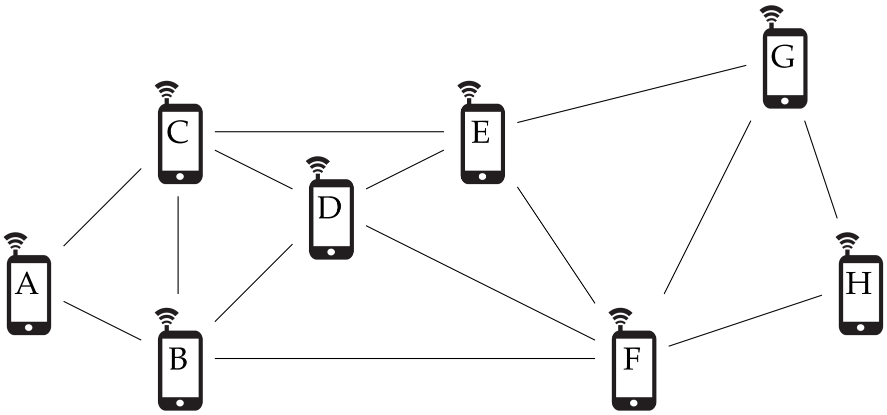


Figure 2.1: Example of a MANET

RFC 2501 [20] specifies MANETs and Figure 2.1 is an example of such. MANETs can operate in isolation or interact with external domains or the Internet. The option to connect to other networks is an option offered on some level since this is very likely to be a requirement in many situations. Nodes that participate in a MANET are generally equipped with wireless capabilities, consisting of a transmitter and a receiver. Each MANET node must perform routing functions to forward a packet to a final destination. Some other MANET features analyzed in RFC 2501 and related to this work include:

- **Limited bandwidth**: The high bit-error rate, noise, fading and other forms of interference in wireless links bring about transmission limitations. A route from a source to a destination in a MANET can consist of multiple physical hops, and will likely accumulate more noise. In addition, many data packets may be lost as a result of collisions when the network topology changes.
- **Dynamic topologies**: Due to the mobility of the MANET nodes, the network topology changes randomly, frequently and rapidly at any time. In addition, links between nodes can become bi-directional and unidirectional links, which results in an unstable network topology.

2.1 Challenges of Multi-hop Forwarding

MANETS are multi-hop wireless networks. In such networks, intermediate nodes along the source to destination route receive and forward packets through wireless links [21, 14]. Wireless multi-hop networks have several advantages. Compared to networks with unique wireless links, wireless multi-hop networks can extend network coverage and improve connectivity. In addition, they allow higher data speeds, resulting in higher performance and more efficient use of wireless media [3, 14]. Multi-hop wireless networks allow operators to avoid extended cable deployment and can be implemented in a profitable way. In the case of multi-hop dense networks, whether covering large geographical areas or having a large number of nodes, several routes could be available to increase the strength of the network [15, 22].

However, as there are advantages to multi-hop wireless networks, there are challenges associated with them as well. Many ad hoc protocols either use network wide broadcasts for route discovery and maintenance or a variation of flooding. Topology information is usually disseminated by flooding as well. As routing follows a per-hop fashion, link availability is a strong requirement for these operations especially when nodes become mobile and links become unpredictable. In some cases, intermediate nodes might become performance bottle necks depending on the extent of traffic they handle. In particular, standard ad hoc routing protocols attempt to minimize the number of relay nodes in the route, OLSR for example, which has the inherent challenge that a node serving as a relay node for transmissions from multiple neighboring nodes can become a performance bottleneck when the network becomes dense [15, 3].

2.2 Routing in Flat MANETs

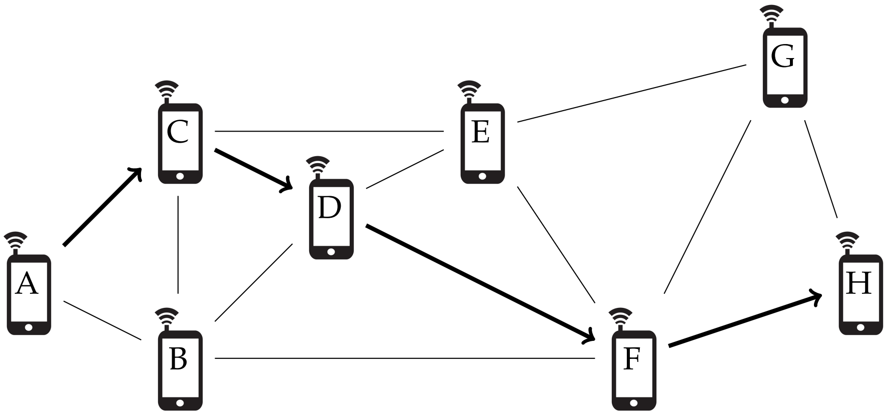


Figure 2.2: Example of Routing in a MANET

In a MANET one of the central problems is routing. 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. For example, in Figure 2.2, hosts A and H are the source and destination hosts respectively, while hosts C, D, and F are intermediate hosts. At each of these intermediate hosts, the next hop is selected based on the entries in their routing tables. If any of these links suddenly become unavailable (because a host moved away, powers down, etc.) and there is no alternative link to do forwarding, then the packet to be forwarded will be dropped. In a flat MANET, no node or address hierarchy exists and all nodes perform equivalent routing roles within the network, operating according to the same routing protocol. The category of protocols traditionally designed for these kinds of environments is further subdivided into reactive and proactive routing protocols. However, more recently, additional categories like hybrid approaches (using some elements of reactive and proactive systems) and hierarchical techniques, have been introduced. The following subsections introduces these routing approaches.

2.2.1 Flooding

Network-wide broadcast is an essential feature for ad hoc networks. The simplest broadcast service is flooding. Flooding is a simple routing technique in computer networks where a source node sends packets through every outgoing link to other nodes in the network. It is a way to distribute routing information updates quickly to every node in a large network. Its advantages are its simplicity and reachability. However, for a single broadcast, flooding generates abundant retransmissions which results in bandwidth and battery power waste. Also, nodes who are close to each other are likely to retransmit at the same time. As a result, flooding quickly leads to message collisions and channel contention, a phenomenon known as the broadcast storm problem [23].

The broadcast problem has been extensively studied for multi-hop networks. Optimal solutions to compute a Minimum Connected Domination Set (MCDS) [24] were obtained for the case when each node knows the topology of the entire network (centralized broadcast). In particular, several solutions like [25] have been presented in which the broadcast time and complexity is investigated in detail. These solutions are deterministic and guarantee a bounded delay on message delivery, but the requirement that each node must know the entire network topology is a strong condition, impractical to maintain in mobile ad hoc environments. Several broadcast protocols that do not require knowledge of the entire network topology have been proposed. In a counter-based scheme [23], a node does not retransmit if it overhears the same message from its neighbors for more than a prefixed number of times and in a distance-based scheme [23], a node discards its retransmission if it overhears a neighbor within a distance threshold retransmitting the same message. Other protocols, such as the Source Based Algorithm [26], Dominant Pruning [27], Multipoint Relaying [28], Ad Hoc Broadcast Protocol [29], or Lightweight and Efficient Network-Wide Broadcast Protocol [30] utilize 2-hop neighbor knowledge to reduce the number of transmissions. A good classification and comparison of most of the proposed protocols is presented in [31]. Two-hop neighbor knowledge is achievable via periodic hello messages; each hello messages contains the node's identifier (IP address) and the list of known neighbors. After a node receives a hello messages from all its neighbors, it has two-hop topology information centered at itself. A common drawback of the above Neighbor Knowledge methods, which use local information to determine whether to rebroadcast, is their performance in mobile environments. Outdated 2-hop neighbor knowledge corrupts the determination of next-hop rebroadcasting nodes [31], leading to poor broadcast performance.

There also exist more efficient approaches to brute force flooding such as directed flooding [32, 33] and opportunistic flooding [34]. The directed flooding protocol [32, 33] extends the basic flooding approach but optimizes the way packets are diffused in a network. There exists a Generator node, Intermediate nodes and a Sink node. In the basic principle, every node has a geographical location which is known by every other node. The generator node uses a virtual aperture to forward a packet and includes its geographical location. Intermediate nodes will only forward packets based on specific rules: if they are within the generator node's aperture and have enough energy to rebroadcast the packet. This way, only selected nodes in the network get to forward a packet. Opportunistic Flooding on the other hand, has a design tailored for low-duty-cycle networks (mainly sensor networks where nodes stay dormant to conserve energy) with unreliable wireless links and predetermined working schedules. With an energy-optimal tree structure, probabilistic forwarding decisions are made at each sender based on the delay distribution of next-hop receivers. Only opportunistically early packets are forwarded via links outside the tree to reduce the flooding delay and redundancy in transmission. In general, these protocols focus more on energy management as they are more applicable in wireless sensor networks. In our research, we do not focus on energy management related issues given the context of our work.

Overall, considering a hierarchical network paradigm, flooding approaches may be more suited for networks with fewer clusters, which will not typically experience a lot of redundant broadcasts in the backbone network. As the number of clusters increases, the number of gateways will also increase and this will lead to the broadcast storm problem [23] which worsens the performance. Using a more efficient broadcast protocol may improve the backbone performance in the presence of only a few clusters, but will soon run into the inherent capacity limits of the backbone.

2.2.2 Reactive Protocols

The on-demand routing protocols suggested for MANETs, such as the DSR protocol [35, 12, 13], AODV routing protocol [35, 12, 13], etc., basically make use of broadcast-based methods for route discovery. They differ in their routing packet formats, data structures maintained by each node, various optimizations applied in route discovery and in their approach for maintaining routes. In a broadcast-based method, when an originator node wants to send data packets to a target node, and it does not have a valid route to this target node, it broadcasts a route request packet to its neighbors. These neighbors forward the route request packet to their neighbors and this process continues until either the target node or an intermediate node with a valid route to the target node is located. Each node receiving a particular route request packet broadcasts it only once to its neighbors, and it discards the subsequent receptions of the same route request packet, to minimize routing overhead. Duplicate receptions are detected using sequence numbers associated with route request packets. This method of route discovery floods the entire network with the route request packets thus this method of route discovery is also called Flooding Method and does not scale in large MANETS.

The shortcoming of this method is that it floods the entire network with the route requests even when the target node is just a few hops away from the originator node. An improvement over the Flooding Method, as suggested in [21] to reduce the waste of bandwidth of a MANET is the Expanding Ring Method. In the Expanding Ring Method, the originator node initially uses a Time To Live (TTL) field equal to some (small) constant in the route request packet and initiates the route discovery as before. This limits the initial propagation of the route requests messages to a few hops. When no valid route is received by the originator node within a certain timeout interval, it increments the TTL field of the route request packet and re-initiates the route discovery process. This process is repeated till a valid route is received by the originator node or up to a maximum number of retries. Expanding Ring Method, although better than the Flooding Method in terms of overall bandwidth utilization of a MANET, is still not a very efficient method, as it still wastes a lot of bandwidth due to redundant link traversals. In fact, Expanding Ring Method is sometimes more expensive than Flooding Method, e.g. when the originator node and target node lie at opposite extremes of the network. This waste of bandwidth in case of Flooding and Expanding Ring Method is because each participating node normally receives route request packets from all its neighbors and, except for the first one, all subsequent receptions are redundant. Thus although these methods of route discovery are simple in operation, this simplicity comes at the cost of wasting valuable bandwidth.

2.2.3 Proactive Protocols

In proactive routing [35, 22, 12, 3] each node builds and maintains routing information to all other nodes in the network. The information is stored in tables in the nodes and maintained by exchanging information with other nodes. In proactive routing [35], a node can consult its own routing table to get a route from itself towards any final destination. This is possible due to the fact that each routing agent maintains a full routing table to all network nodes. OLSR [36] is a well-known example of a proactive routing protocol. Other examples include DSDV [37], Source Tree Adaptive Routing (STAR) [38, 39], and Wireless Routing Protocol (WRP) [39]. Such approaches create a substantial amount of background maintenance traffic to keep all the routing tables up-to-date in the presence of topology changes. Furthermore, participating mobile nodes are required to maintain their routing table entries even if they are not being used. Also, the routing tables in the nodes grow as the network size increases. Besides these scalability issues, however, proactive approaches have many desirable properties such as low latency route access and Quality of Service (QoS) path support and monitoring. Proactive routing is most suitable for applications which require a low message latency and which have a high message throughput. However, in large scale MANETs, where there are a lot of nodes and the ensuing control traffic continues to grow, scalability becomes an issue [40, 19].

2.2.4 Hybrid Protocols

Proactive routing uses excess bandwidth to maintain routing information, while reactive routing involves long route request delays. Reactive routing also inefficiently floods the entire network for route determination. The key idea behind hybrid routing protocols is that they attempt to combine the best features of both reactive and proactive protocols. Thus, they are both proactive and reactive in nature. It was proposed to reduce the control overhead of proactive routing protocols and also decrease the latency caused by route discovery in reactive routing protocols [15, 22, 12]. This is mostly achieved by pro-actively maintaining routes to near-by nodes and determining routes to far away nodes using a route discovery strategy. Most hybrid protocols proposed to date are zone-based, which means that the network is partitioned or seen as a number of zones by each node [41, 42]. Others group nodes into trees or clusters [15, 22, 12]. By attempting to minimize the number of broadcasting nodes, hybrid routing protocols have the potential to provide higher scalability than pure reactive or proactive protocols. In [41] they do this by defining a structure (or some sort of a backbone), which allows the nodes to work together in order to organize how routing is to be performed. By working together, the best or the most suitable nodes can be used to perform route discovery. For example, in Zone-Based Hierarchical Link State (ZHLS) [41, 42] only the nodes which lead to the gateway nodes broadcast the inter-zone route discovery packets. Collaboration between nodes can also help in maintaining routing information much longer. For example, in SLURP [43], the nodes within each region (or zone) work together to maintain location information about the nodes which are assigned to that region (i.e. their home region). This may potentially eliminate the need for flooding, since the nodes know exactly where to look for a destination every time. Another novelty of hybrid routing protocols is that they attempt to eliminate single point of failures and creating bottleneck nodes in the network. This is achieved by allowing any number of nodes to perform routing or data forwarding if the preferred path becomes unavailable [15, 12].

A shortcoming that hybrid routing protocols face is that a lot of storage/caching of information and processing requirements are involved as compared to reactive protocols. This, in turn, results in more memory and power consumption [12].

2.3 Hierarchical Routing

A large variety of approaches for ad hoc clustering have been proposed, with different approaches typically focusing on different performance metrics. Dynamic routing is one of the important issues in MANETs. However, it has been shown that a flat structure exclusively based on proactive or reactive routing schemes cannot perform well in a large dynamic MANET [16, 4]. In other words, a flat MANET structure faces scalability problems with increased network size, especially when node mobility is also introduced. This is due to their inherent characteristics. The communication overhead of link-based proactive routing protocols increases with the square of the number of mobile nodes in a MANET. For a reactive routing scheme, the Route REQuest (RREQ) (Route REQuest) flooding over the whole network and the considerable route setup delay become intolerable in the presence of both a large number of nodes and mobility [16].

One alternative to flat MANETs is clustering or hierarchical routing. The motivation for exploring hierarchical routing is that it increases scalability, routing efficiency and reduces routing table entries considerably. Rather than assuming that node movement is independent, hierarchical ad-hoc routing protocols group nodes into clusters of nodes that follow the same movement pattern. These protocols are based on the idea that members of a group tend to move together and therefore a node will most likely remain within the same cluster. This allows a node to move freely within its cluster and only inform other cluster members, abstracting node movement within a cluster so that members of other clusters only need to know how to communicate with one of its members. These groups may have some sort of cluster leader, popularly known as gateways or cluster heads. Depending on the algorithm and the clustering technique, there might be gateways providing connectivity with other clusters, and cluster heads who coordinate routing within their clusters and with other clusters. Alternatively a single node plays both roles, as joint cluster head/gateway, providing connectivity with other clusters through a core/backbone network. Clusters can then be organized into a hierarchy.

In [44, 45, 40, 46] the authors present a review of current hierarchical routing protocols and clustering approaches. The authors first provided fundamental concepts about clustering. Then they classify the proposed clustering schemes into six categories based on their main objectives, which are load balancing clustering, Dominating-Set-based (DS-based) clustering, low maintenance clustering, mobility-aware clustering, combined metrics-based clustering, and energy efficient clustering. They also grouped the clustering cost terms into five categories: the required explicit control message exchange, the ripple effect of re-clustering, the stationary assumption, constant computation round, and communication complexity.

One of the earliest clustering protocols is the Linked Cluster Architecture (LCA) [47], developed for packet radio networks. The LCA protocol organizes the nodes into clusters according to the proximity of the nodes. Each cluster has a cluster head and all nodes in a cluster are in the direct transmission range of the cluster head. The choice of the cluster head is based on node identifiers, where the node with the largest identifier in a given area becomes the cluster header. The gateways in the overlapping region between clusters are used to connect clusters. LCA specifies that there should only be one designated gateway to interconnect clusters at a given time. A pair of nodes within transmission range of each other can also be used to connect clusters if there are no nodes in the overlap region.

The authors of [48] have described the Cluster head Gateway Switch Routing

(CGSR) protocol. In this protocol, packets are routed alternately between the cluster leaders and the gateways. The authors define several extensions that can be added to CGSR, such as priority token scheduling and gateway code programming, to control access to the channel. In addition, they define a Least Cluster Change (LCC) algorithm, designed to reduce the number of changes in the cluster leader, since such changes can generate significant overhead.

The authors of [49, 50], take a different approach to clustering and present two clustering algorithms. The first of these is the Distributed Clustering Algorithm (DCA) intended for "quasi-static" networks in which nodes are slow moving, if moving at all. The other algorithm is called the Distributed and Mobility-Adaptive Clustering algorithm (DMAC), designed for higher mobility. Both algorithms assign different weights to nodes with the assumption that each node is aware of its respective weight. The weights are in turn used to determine the cluster leaders. In the DMAC protocol, if two cluster leaders come into contact, the one with the smaller weight must revoke its leader status.

Another approach is that taken by the Core Extraction Distributed Ad Hoc Routing (CEDAR) algorithm [51], which builds a set of nodes (i.e., a core) to perform route computation instead of creating a cluster topology. Using the local state information, a minimum dominating set of the network is approximated to form the core. CEDAR establishes QoS routes that satisfy bandwidth requirements using the directionality of the core path. Link state and bandwidth availability is exchanged to maintain important information for computing QoS routes.

Kleinrock was an early pioneer of hierarchical routing schemes for static networks. In [52], Kleinrock and Kamoun investigated a hierarchical routing scheme with the goal of reducing routing table size. The authors of [53] also adopt a similar approach. The authors determined that the length of the routing table is a strict function of the clustering structure. Clustering generally has the unwanted side-effect of an increase in path length, and so the goal was to find an optimal clustering scheme that optimizes path length. It was determined that the number of entries in a node's forwarding table is minimized when the number of *level-i* clusters in each level *-(i+1)* cluster is e, and the number of levels in the clustering hierarchy equals $\ln N$. In this case, the forwarding table contains $e \ln N$ entries.

The Landmark Routing technique [54] is a distinct approach to building a hierarchy as it is based on landmarks, as opposed to transmission ranges. A landmark is a router whose location is known by its neighboring routers up to some radius. All routers within that radius know how to reach the landmark. A hierarchy of such landmarks is built by increasing the radius of some of the routers. Nodes have hierarchical addresses based on the landmarks with which they are associated. A source node routes to a destination by sending the packet to the lowest level landmark with which both nodes are associated. As the packet approaches the destination, the granularity of routing knowledge about that destination improves, and so the packet can be accurately routed to the destination.

Advantages of hierarchical routing, alongside scalability, include the ability to reduce routing table sizes, to shield nodes within a cluster from mobility in other clusters and to use different routing protocols, with possibly different update frequencies, in different clusters. Disadvantages include the difficulty in maintaining the structure of clusters in the face of high mobility (which has a particularly adverse effect if cluster heads change groups), the possible bottleneck presented by gateway nodes (these nodes also suffer greater resource usage) and the use of suboptimal paths. Examples of hierarchical routing protocols can be found in [40].

In our research, scalability is a strong requirement and so, we try to limit routing knowledge and avoid flooding in the backbone. Thus, we apply a hierarchical approach. In the context we are working with, we do not use any special algorithm for cluster formation nor do we worry about the way the clusters are formed. Clusters are given, and they basically flow from the application/use of the MANET. For example, different platoons of soldiers (in a military context) or different first responder crews joining in as a group etc.. Furthermore, most hierarchical protocols still employ flooding, albeit it within a limited scope. Our proposed solution completely avoids the use of flooding by applying the idea of a structured P2P approach, a Distributed Hash Table (DHT) which we introduce in Chapter 3. OLSR is then used for routing at the network layer and does not worry about exchanging any extra link-state information in order to support mobility.

	Reactive [12, 13, 35]	**Proactive** [3, 12, 22, 35]	**Hybrid** [12, 15, 22]	**Hierarchical** [44, 49, 52, 54]	**DHT-based**
Scalability to large MANETs	×	×	×	✓	✓
Routing Overhead	high	high	medium	medium	low
Mobility Management approach	flooding	flooding	partly-flooding	partly-flooding	DHT-unicast advertisement
Route discovery approach	flooding	flooding	partly-flooding	partly-flooding	DHT-unicast look up
Cluster formation	-	-	algorithm (not trivial)	algorithm (not trivial)	clusters are given
Cluster head selection	-	-	algorithm (not trivial)	algorithm (not trivial)	gateways are given

Table 2.1: Comparing the different routing approaches in the context of Large-scale MANETs

2.4 Summary

Flooding is a very simple but costly solution for the backbone of a hierarchical MANET. Considering a network split into multiple clusters with levels of hierarchy, and in real time applications, for a single broadcast, flooding will generate abundant retransmissions which results in bandwidth waste. As a result, flooding quickly leads to message collisions and channel contention, a phenomenon known as the broadcast storm problem as described above. We implemented a blind flooding algorithm, which we used as a base case for our comparison. Here, the gateways to the various clusters broadcast packets once in the backbone network as is the case with every other flooding algorithm. The results show that broadcasting does result in many collisions and poor overall network performance.

Table 2.1 presents a summary of the routing approaches discussed above. The trade-offs between proactive and reactive routing strategies are quite complex. Which approach is better depends on many factors, such as the size of the network, the mobility, the data traffic and so on. Proactive routing protocols try to maintain routes to all possible destinations. Regardless of whether or not they are needed, routing information is constantly propagated and maintained. In contrast, reactive routing protocols initiate route discovery on the demand of data traffic. Routes are needed only to those desired destinations. This routing approach can reduce routing overhead when a network is relatively static and the active traffic is light. However, the source node has to wait until a route to the destination can be discovered, increasing the response time. The hop-by-hop flooding used for route discovery usually has a huge negative impact on network performance. The hybrid routing approach can adjust its routing strategies according to a network's characteristics and thus provides an attractive method for routing in MANETs. However, a network's characteristics, such as the mobility pattern and the traffic pattern, can be expected to be dynamic. The related information is very difficult to obtain and maintain. This complexity makes dynamically adjusting routing strategies hard to implement and maintain. It is evident therefore that these protocols either have scalability issues, which get worse in the case that nodes are mobile and links become generally unpredictable or have complex implementations in the case of the hybrid protocols.

We adopt a two-tiered hierarchical clustering approach where we split a flat MANET into a number of clusters, each having a gateway which connect the clusters in a backbone network. This way, the different clusters are able to run an instance of the same routing protocol, or different routing protocols and hence, reduce routing overhead, increase efficiency and scalability, and reduce routing table entries overall as it is irrelevant for nodes to know the entire network topology.

Whereas other approaches deployed either a pure flooding based protocol or a variation (AODV protocol for example) in the backbone, we used OLSR protocol with a DHT. The default implementation of AODV routing protocol broadcasts routing packets in the whole network when events such as route creation/discovery or route breakages occur. In fact, the higher the number of nodes is, the higher the number of control packets is. Moreover, the number of control packets increases rapidly when the network topology changes under high mobility and different mobility scenarios. As a result, the unavailability of links will result in packet losses and high latency. OLSR on the other hand when used alone, will experience difficulties because of the level of link state information it will have to exchange in order to support mobility. These disadvantages can, to some extend, be reduced by limiting the flooding scope using a clustered or hierarchical routing approach. But mobility management (tracking the location/membership of nodes within a cluster) still requires broadcast-based solutions.

In our solution, the DHT handles the mobility aspect and inter-domain routing information maintenance while OLSR supports network layer routing, both of which are unicast. Our solution leverages the storage and retrieval mechanism of a DHT to make routing information available and support different forms of mobility like whole clusters merging and splitting and nodes switching their cluster memberships multiple times. The gateways typically advertise reachability to all the nodes in their cluster by storing $key - value$ pairs in the DHT as such: (node ID, gateway IP), thus, making routing information available in a decentralized fashion. Using a DHT in the backbone has the following advantages:

- The DHT protocol, Chord, maintains the $key - value$ pairs stored in the DHT. As whole clusters merge and split, no special handling of network/clusters merging and splitting is required.

- Routing in the DHT is unicast and will be based on individual node identifiers generated by hashing IP addresses into keys. At the network layer of the backbone, routing will also be unicast as we use a unicast routing protocol, OLSR.

- When a node switches cluster/network membership, the new gateway will simply advertise its reachability to this node in the DHT, overriding the previous entry. Here, we mimic Mobile IP [55, 56] from the (fixed) Internet, but replace the server-based infrastructure (the Home Agent who knows where a node is) with a more dynamic P2P solution that fits the requirements of a MANET. In addition, finding the Home Agent requires that IP addresses have a hierarchical structure (the Home Agent is in a node's Home Network), whereas the DHT does not care about the hierarchical IP address structure.

Chapter 3

Design of our DHT-based Routing Approach

Initially, the backbone network topology of data networks were relatively simple. The operations were centralized, so the star topology was the most logical and, in some cases, it was the only topology admitted by the technology [57]. With this set-up, the network became vulnerable as the center of the star became a single point of failure. With the transition to multiple client-server and P2P relationships, the choice of the topology of the core network is not so clear.

The original purpose of the backbone network is to form an umbrella or parent network which manages high capacity links, infrastructure and high traffic volumes while providing connectivity to individual networks connected to it. The connectivity may cover a local area within a building, vicinity or may have a global outreach that spans vast geographical areas. Such large networks can have multiple tiers of hierarchy between the backbone and the networks connected to it. Given its position at the top of the network hierarchy, two requirements of the backbone topology are reliability and scalability.

In a MANET, nodes are often mobile and there can exist mobile subnetworks/-MANETs which are interconnected through a backbone network/MANET [12, 22]. In flat MANETs, routing is challenging in the presence of mobility because links become unpredictable as a result of the dynamic nature of these networks. This becomes even more challenging in cases where there are mobile sub-MANETs which are interconnected through a backbone network/MANET [7]. In such setups, the routing scheme deployed in the backbone network must be able to maximize the backbone bandwidth, enhancing throughput and reducing end-to-end delays with respect to schemes without a backbone. It must do so without compromising (in fact, possibly enhancing) scalability and fault tolerance. In designing a dynamic system like ours which aims at supporting a variety of applications, maintaining connectivity is a strong requirement. In this section we present our design which is a hierarchical routing solution that provides an efficient routing scheme and supports various mobility scenarios effectively.

3.1 Distributed Hash Tables

In our design, we are ultimately interested in a structure that will support a variety of large-scale applications over MANETs. Such networks are formed by clusters of nodes, representing a community of interest. Each cluster is a MANET which has a cluster head that serves as a gateway to provide connectivity to external networks. Distributed Hash Tables (DHTs) are a form of a distributed database that can store and retrieve information associated with a key, in a network of peers/nodes that can join and leave the network at any time. In this thesis, we leverage this distributed data sharing functionality to provide extra information with which each gateway can carry out routing through the backbone efficiently. By leveraging this functionality, different forms of mobility, including single hosts switching cluster membership and whole clusters merging/splitting, can be managed. Section 3.2 describes our design in more detail. The DHT strategy is used as a lookup strategy in structured P2P systems to specify the location of objects in peers, and it provides all the needed services to look up objects in a decentralized P2P system [58]. The DHT retains mapped information about nodes and peers in the form of key/value pairs (k, v) so that data can be easily located in the overlay network. In DHTs, each peer maintains a storage space to keep a hash table. Many structured P2P systems like Chord [59], CAN [11], Pastry [60] and Tapestry [61] are based on a DHT. The indexing of data facilitates its discovery and search by any peer.

For the DHT protocol, we use Chord, which is one of the well known distributed overlay lookup protocols based on a DHT, that helps locate data/resources located in different peers in an overlay network [62, 59]. Chord assigns an m-bit identifier to nodes and keys. The terms *node* and *key* denote both the node and key themselves as well as their identifier, without ambiguity. The value of m should be large enough to prevent collision by assigning the same identifier to other peers. $SecureHashAlgorithm1(SHA-1)$ [63], a consistent hash function, is the base hash function used by Chord to generate an identifier with m bits. Both the peer's IP address and the resource name is hashed to generate a key to determine the peer's identifier and the key identifier. Keys are assigned to peers using consistent hashing. This entails that peers are organized around an identifier circle modulo 2^m. Key k is then assigned to the peer whose identifier is equal to that of k, if such a peer exists, or to the first peer whose identifier follows k's on the circle.

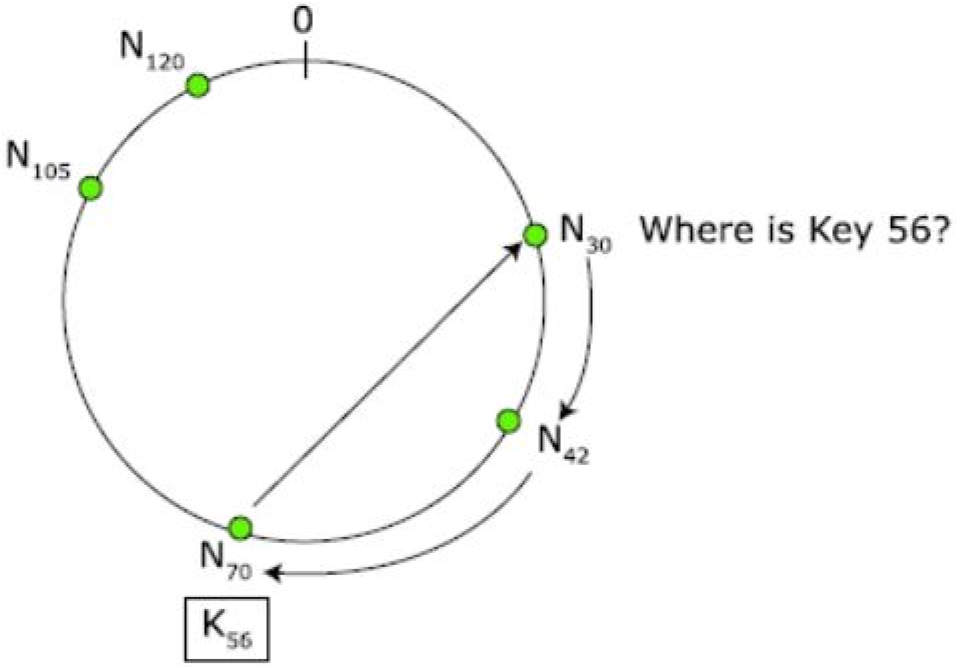


Figure 3.1: Chord Ring Showing a Lookup Process, [1]

Two high level interfaces, PUT and GET, are used to either retrieve or store information in the DHT [62, 59, 58]. The PUT message is used to store/index a resource in the DHT and it has the signature $PUT(k, v)$, where k is a *key* obtained by executing $SHA - 1$ on a peer's IP Address and v corresponds to the data/resource to be stored. The GET is a query, used to retrieve a resource from or lookup a resource in the DHT. It has the signature $GET(k)$, where k is a key obtained in a similar way as with the PUT signature. An example use of the DHT for storage might proceed as follows: assume that the key-space is the set of 160-bit strings. To index a resource v with name *research* in the DHT, $SHA - 1$ is executed on *research* to generate a 160-bit key k, and a message $put(k, v)$ is sent to any peer participating in the DHT. The message is forwarded through intermediate peers in the overlay network until it arrives at the single peer responsible for k as specified by the key-space partitioning. That peer then stores the key and the resource. In order to execute efficient lookups, each peer keeps a table of m entries named finger table. Fingers are other peers that are tracked by each peer. In the finger table of a peer n, the identifier of the first peer (at i^{th} entry) that comes after n is determined by $(n + 2^i)$. The lookup operation is simple. When a $GET(k)$ query is sent to a peer, the peer first needs to inspect its own local storage to ensure that it carries the desired data item. If it holds the desired data item, it simply sends the result to the requester. Otherwise, it redirects the query to its nearest peer according to its finger table [62, 59]. Subsequently, the nearest peer also redirects the $GET(k)$ query to the peer closest to it, and so on, until the query reaches the peer that carries the result. The result follows the reverse path to reach the requester peer (the peer that sent the query). With a system of N peers, the complexity of the search algorithm is $O(log\ N)$. Figure 3.1 represents a peer $N30$ that needs to lookup the key $k56$ stored at the peer $N70$. As explained above, if there existed a peer $N56$, then it would be responsible for $k56$. However, in this case, $N70$ is the first node whose identifier follows $k56$'s on the Chord ring and so, it becomes responsible for $k56$. The query, therefore, proceeds as such: $N30$ sends the $GET(k56)$ query to peer $N42$ which does not carry the key and finally $N42$ forwards the query to peer $N70$ which does carry the key. $N70$ will reply with whatever the stored resource is.

3.2 Hierarchical Structure

In Chapter 2 we mentioned that clustering is a technique that partitions a network into different groups or clusters, creating a logical hierarchy in the network. By partitioning a network into different clusters, communication overheads for maintaining

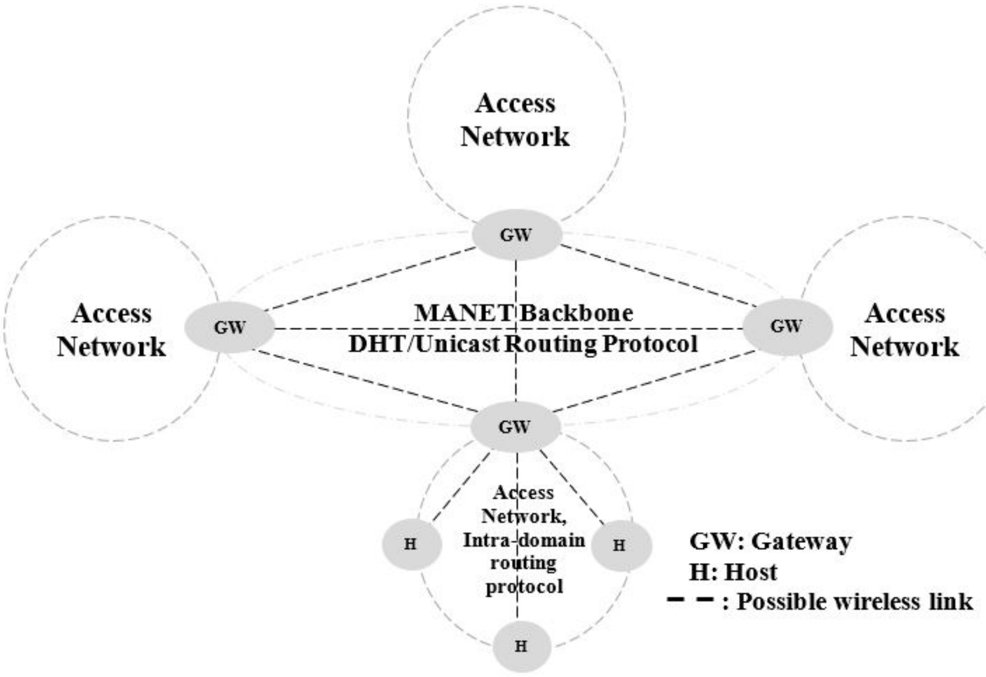

Access Network
Access Network
Access Network
GW
GW
GW
GW
MANET Backbone
DHT/Unicast Routing Protocol
Access Network, Intra-domain routing protocol
H
H
H
GW: Gateway
H: Host
– – : Possible wireless link

Figure 3.2: Two-Tiered Network Architecture

up-to-date routing information can be significantly reduced [16, 64, 4, 19].

Figure 3.2 is a representation of the hierarchical architecture we deploy. It is a two-tier hierarchical architecture where we divide a flat MANET of N nodes comprising h hosts and g gateways into c clusters. Each cluster is therefore a MANET on its own and has a gateway node. In Figure 3.2, the clusters are the circular areas having dotted lined borders and they form level one of the two-tier hierarchy. These clusters are formed by :

- Allowing each cluster to communicate on a separate communication channel such that no matter how close they might be with each other, they do not hear each other.

- Physically separate the clusters from each other by assigning them unique cluster spaces in the network area, and allow them to communicate locally over the same communication channel.

The gateway nodes are connected in a backbone MANET and connect their local clusters to the rest of the network. With this architectural setup, each cluster is basically a separate and independent routing domain, running its own intra-domain routing protocol. Each cluster may be configured to run a separate routing protocol (OLSR, AODV, OSPF, etc.) or each cluster may be configured to run an instance of the same protocol. This is further explained in Section 3.3.

Similarly, the backbone, which is on its own a MANET, requires a routing protocol to coordinate routing functions among the different clusters/MANETs represented by the gateway nodes. In the Internet, BGP is the typical inter-domain routing protocol which interconnects these networks [65, 66]. BGP works well with managing autonomous systems which are fixed networks governed by a single entity. It was designed to cope with the scale and operational challenges of the Internet. Compared to the Internet, MANETs are considerably smaller yet have highly dynamic environments. In MANETs, nodes are generally mobile and when an ad-hoc network supports sub-networks, these sub-networks may be mobile as well. As a result, a range of challenges like single networks splitting, multiple networks merging, or nodes moving between networks may ensue. When this becomes the case, either new inter-domain routing protocols are required, or the dominant BGP needs to be modified [67, 68]. BGP has been rather effective in the wired world [65, 66]. Modifying it to suit MANET environments results in poor routing performance [67]. Instead of modifying BGP, an alternative approach is to borrow the core design principles of BGP (enable opaque interoperation, effectively coordinate inter-domain routing), and take a clean slate approach to enable inter-domain routing in MANETs [65]. The routing aspect of our design is explained in

Section 3.3.

3.3 Routing Approaches Explored

Routing in MANET environments is challenging because of the dynamic nature of the network topology. In hierarchical MANETs specifically, routing becomes even more complex because, coupled with the presence of mobility, there is more than one level of hierarchy and destinations are distributed in multiple routing domains which a source might want to reach. Thus, the routing approach deployed for inter-domain routing must be capable of efficiently delivering packets between the various routing domains.

In the Internet, the principle of BGP is to enable opaque interoperation, where each domain has the administrative control over its intra-domain routing protocol and inter-domain routing policy, which is not known (or opaque) to the other domains [65, 66]. Similarly, the routing approach deployed in a hierarchical MANET must be able to coordinate opaque interoperation between the various routing domains.

3.3.1 Approaches to Inter-domain Routing in MANETS

Flooding: Flooding is a simple but inefficient and costly routing approach for the backbone of a MANET[19]. A source node in the backbone network will broadcast a message in the backbone once, and all other nodes, except the destination node, will rebroadcast these messages once. Figure 3.3 represents a flooding network. Here, GW_A is the originator of the packet and GW_H is the destination. All other gateways in the network rebroadcast the packet once [19], and our results in Section

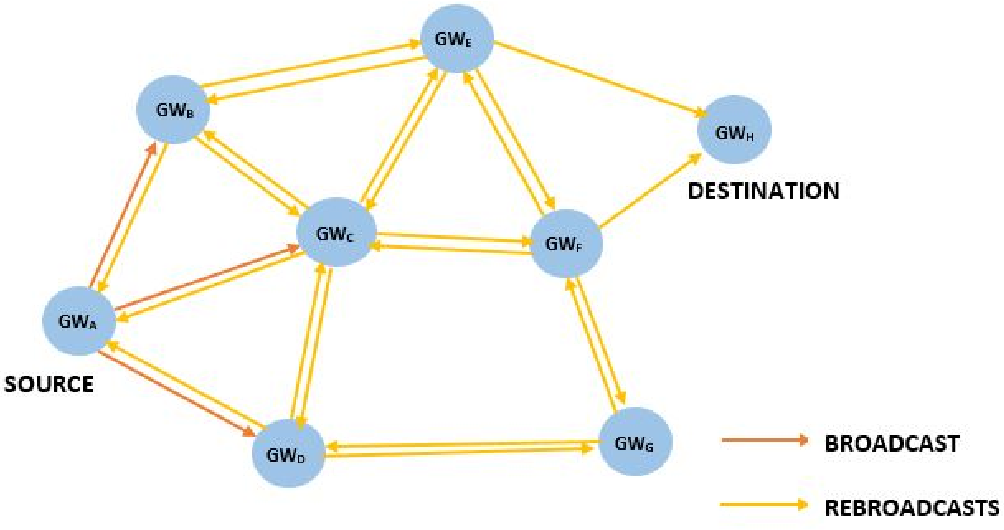


Figure 3.3: An Example of Flooding

5.1.4 show that the outcome of such an approach is unfavorable performance-wise and merely provides a best effort service [69]. In this work, we implemented a flooding protocol which we use as a base case to measure the performance of our solution.

Reactive Routing (AODV): Reactive protocols, as described in Section 2 and [37, 35, 22, 12, 20, 13], broadcast routing packets in the whole network when events such as route creation/discovery or route breakages occur. The routing packets for route discovery are in two folds: route request packets (RREQ) and route reply packets (RREP). Figure 3.4 represents a simple route discovery process with AODV routing protocol. Here, GW_A initiates the route discovery process and GW_H is the gateway to the sought destination. In such networks, the higher the number of nodes is, the higher the number of control packets is. Moreover, the number of control packets increases rapidly when the network topology changes under high mobility and different mobility scenarios. As a result, the unavailability of links will result in packet losses and high latency. When reactive protocols create a route,

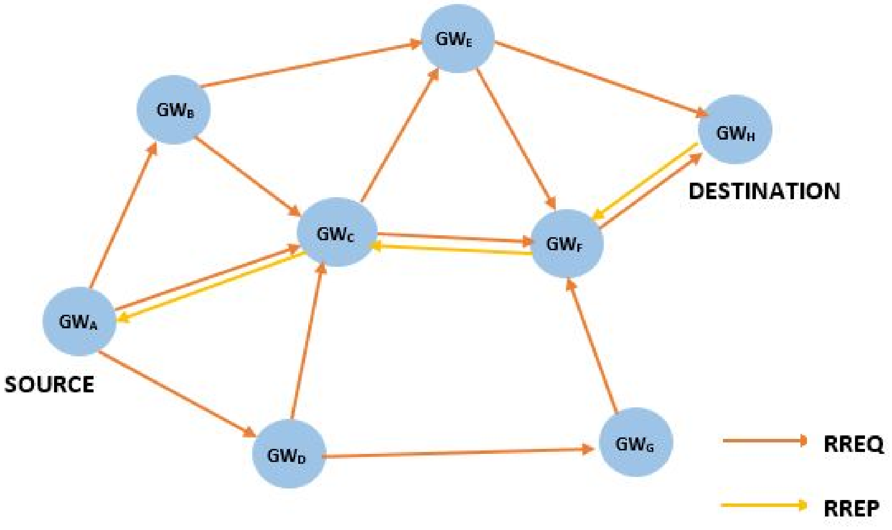


Figure 3.4: An Example of Routing with AODV Protocol

they broadcast many redundant control packets. Indeed, some control packets may be sent in the opposite direction of the destination, thus they will not be used to reach the destination. Bandwidth consumption due to these packets may be high and will negatively impact the overall protocol performance.

DHT-Based Unicast Routing: Just like an intra-domain routing protocol, using a unicast protocol like OLSR alone in the backbone will carry out routing functions among the members of the backbone network which include supporting the exchange of messages and maintaining routing tables. This is not sufficient, since both routing domains are independent of each other and will, therefore, need some extra information to route messages between both domains. For example, when a packet destined for an external domain shows up at a gateway, the gateway has information about its local cluster and the backbone network in its routing table, but requires information about the location of the destination in order to select the right gateway to forward the message to.

Each gateway GW_X can participate in a DHT based P2P overlay and leverage its storage and retrieval functionality to advertise reachability to a host X_X in their local cluster in the DHT by storing a key-value pair as such: (X_X, GW_X). Each participating gateway can then retrieve information about a destination host X_X by issuing a query to the DHT: *GET(X_X)* which returns GW_X. Using a DHT within the backbone has a number of advantages:

1. In the event that clusters merge/split, hosts switch cluster membership, the DHT protocol will maintain the data (key-value pairs stored in the DHT).

2. Routing will be based on individual host IDs, and in the event that clusters merge or split, no special handling of networking merging and splitting will be required.

3. Host mobility (both intra and inter-cluster/MANET) can be supported efficiently.

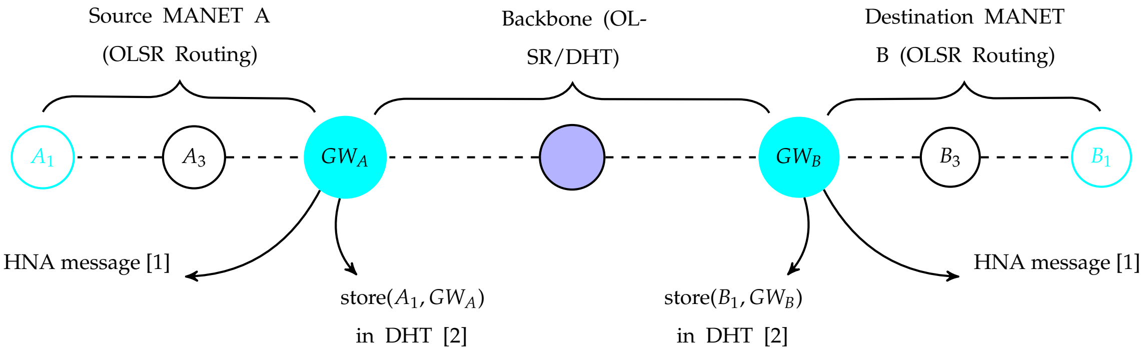

Source MANET A (OLSR Routing)
Backbone (OLSR/DHT)
Destination MANET B (OLSR Routing)
A_1
A_3
GW_A
GW_B
B_3
B_1
HNA message [1]
store(A_1, GW_A) in DHT [2]
store(B_1, GW_B) in DHT [2]
HNA message [1]

Figure 3.5: Initial Routing Setup

As earlier mentioned, our approach is based on the use of a DHT together with a MANET routing protocol to support routing in the network backbone, while individual clusters may employ any MANET routing protocol of their choice, such as OLSR, ADOV, etc. In both our cluster and backbone MANETs, we use OLSR [36]. Routing of messages between nodes in any of these clusters is efficiently handled by this intra-domain routing protocol. Routing between hosts in different clusters is a more complex venture. For example, a host A_1 in cluster A needs to route a packet to a host B_1 in cluster B, as shown in Figure 3.5. The source host A_1 has the address of the destination host B_1 with which it creates the IP packet it wishes to route. The challenge is that A_1 does not know how to forward the packet to B_1 because it discovers from its routing table that B_1 is not a member of its cluster. For such operations, where a host might need to reach a host in a different cluster, the OLSR protocol supports an optional auxiliary functionality, ”Host and Network Association (HNA) Message”, whereby a gateway announces its reachability of other clusters to its local cluster members. The gateways in each cluster (here GW_A for cluster A and GW_B for cluster B) will periodically broadcast HNA messages to all hosts in its cluster to advertise reachability of hosts outside the cluster through them (1). The content of an HNA message is a list of Network/Network-Mask Addresses of all external network destinations (clusters/MANETs in our case) the gateway can reach.

Following the default implementation of HNA messages [17], the gateways will have to advertise a Network/Mask address pair for each network it is associated with to its cluster members who will have to create an entry in their routing table per Network/Mask address pair received in the HNA message. This will become inefficient in the case where networks come and go, and gateways will have to frequently update their local cluster of these changes and/or external networks become numerous and gateways will have to advertise a long list of addresses (in the order of 100s or 1000s) which will increase the size of the routing tables. In our design, gateways essentially advertise a global default route *(0.0.0.0/0.0.0.0)* to all members of their cluster. Each host will then create an entry in its routing table as such: Destination-[0.0.0.0], Next hop-[Gateway IP], Netmask-[0.0.0.0]. With this entry, messages destined for external networks will be forwarded to the gateway.

HNA routes are added to the routing table based on hop count, and are dynamically updated as the topology changes. That is to say that if there exist multiple external access advertisements, then the smallest hop count is chosen. When a packet destined for an external cluster shows up at a gateway node, the challenge becomes how to forward the packet through the backbone to the destination host. This is where the inter-cluster routing protocol comes to play.

All gateways will learn about all hosts in their cluster through OLSR, which will populate a routing table with entries for each host in the cluster. The cluster heads form a backbone MANET and join a P2P overlay based on a Distributed Hash Table (DHT). Each gateway node *GW* advertises reachability to all hosts *N* in its cluster within the backbone by storing the following key-value pair in the DHT: *(*k, GW) (2). Key *k* is generated by hashing the IP address of the host to be advertised in the DHT. Chord protocol provides the services to manage and maintain the DHT data as whole clusters merge/split or hosts switch cluster membership, distributing all key-value pairs among the peers (here the gateway nodes).

3.3.2 Routing in Static Scenarios

The more general/basic case of routing in our design, and similar with what happens in the Internet, is the routing procedure in the static scenario where nodes belong to a particular cluster for the entire lifetime of the network.

Figure 3.6 shows the actual routing of data packets in the general case. Once

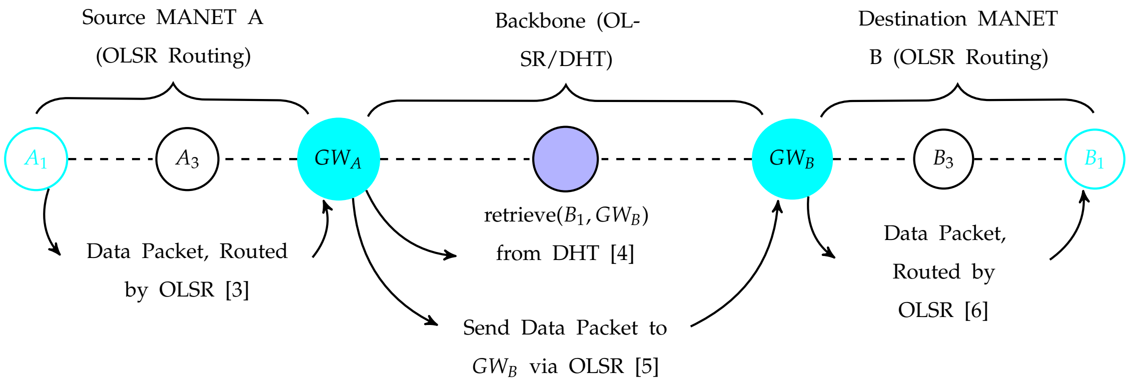

Source MANET A (OLSR Routing)
Backbone (OLSR/DHT)
Destination MANET B (OLSR Routing)
A_1
A_3
GW_A
GW_B
B_3
B_1
Data Packet, Routed by OLSR [3]
retrieve(B_1, GW_B) from DHT [4]
Send Data Packet to GW_B via OLSR [5]
Data Packet, Routed by OLSR [6]

Figure 3.6: Packet Forwarding

A_1 has a data packet to send out, it will consult its routing table which has been populated by the intra-cluster routing protocol, OLSR, and determine that B_1 is not in its own cluster. Using the Host and Network Association (HNA) message route initially created, it will, therefore, route the packet to its gateway GW_A (3). Once the gateway receives such packets, it executes $SHA-1$ on the IP address of B_1 to generate key k, and queries the DHT using this key. This operation retrieves the associated cluster head, GW_B, for B_1's cluster (4). GW_A now has a next-hop address with which it forwards the message to GW_B through the backbone (5). The IPv4 Datagram to be forwarded already contains a source address and a destination address. Thus, GW_A will encapsulate the datagram such that the new datagram will have source address GW_A and destination address GW_B, a process called Tunneling. Once GW_B receives the packet, it understands from the nature of the datagram that it is a tunneled packet. It then decapsulates the datagram and forwards the packet within its cluster based on the local instance of the intra-domain routing protocol, here OLSR again (6). A good advantage to this is that a separate data structure maintains all routing information which a gateway retrieves from the DHT and reuses the same for subsequent routing operations. A potential downside of the proposed architecture is that the source cluster gateway needs to perform a DHT retrieval before it can route the first packet to a new/unknown destination address. But this is not substantially different from other on-demand routing protocols widely used in MANETs, such as AODV or DSR, where routes to new destinations are only discovered when the first data packet is destined to such a host. However, these lookups are done with unicast routing in the P2P overlay, rather than broadcasting/flooding the backbone. And unlike such protocols, any topology changes in the backbone will be hidden from the gateways, reducing the need to rediscover routes.

3.3.3 Routing in the Presence of Mobility

In addition to what happens in the static/general case of routing we described in Section 3.3.2, we also include routing support for various mobility scenarios which are likely to result in MANET environments. Such scenarios, like mobility within the local cluster, whole clusters joining or leaving, clusters merging or splitting, and hosts switching cluster membership or completely disappearing (can no longer be reached) are possible instances of mobility. In this thesis, we study the effect of mobility when hosts are mobile within their local cluster, whole clusters merge/split, and hosts switch cluster membership. Hosts moving within their local cluster is simply a variation of the static scenarios and so, the same basic routing principle holds and works. The challenging mobility scenarios are when clusters merge/split or a host changes cluster membership.

In Chapter 4, we discuss in detail the two approaches we explored in building cluster architectures which are independent of each other: separating clusters physically and allowing them to communicate over the same channel, or assigning different channels to individual clusters. Of the two, the only approach that works for all the mobility scenarios we consider is to physically separate the clusters from each other and allow them to communicate over the same communication channel. This way, as whole clusters move among each other in the network and happen to overlap, they do so both physically and logically, forming a single cluster. When they move away from each other, they become individual/independent clusters again. Thus, clusters merging and splitting is made possible because all nodes communicate on the same channel and since they each run an instance of the same routing protocol, OLSR, they will learn of each other by exchanging OLSR hello messages. Similarly, hosts can switch cluster membership seamlessly and still maintain the same IP address they initially had. The challenging part of these scenarios is to support routing efficiently in the presence of these mobility scenarios. In the basic routing case we described in Section 3.3.2, there was no need to update the DHT with any information because the hosts each gateway advertises belong to the same cluster as the gateway for the entire network lifetime. With mobility, it becomes necessary for the gateways to update the DHT with recent routing information regarding the hosts they can now reach or no longer reach. For example, when two clusters (A and B) merge, the ensuing cluster will have two gateways GW_A and GW_B who will now advertise reachability of all the new hosts they can now reach in the DHT. Similarly, when the clusters split, the gateways will have to delete entries they initially advertised in the DHT, for hosts they can no longer reach. In the case of hosts switching cluster membership, a gateway GW_X to the cluster X which the host n arrived in will learn about the presence of n in its cluster via the intra-domain routing protocol, OLSR, and will have to advertise to the DHT that it can now reach host n. Similarly, the gateway GW_Y to cluster Y

where host *n* came from will have to update the DHT that it can no longer reach host *n*. These updates are a critical requirement to efficiently support routing in the presence of mobility, following the routing design we adopt in the backbone.

When two clusters merge, there is the probability that some hosts are better reached through either of the gateways as a function of the number of hops away. The hosts, when they receive HNA messages from different gateways, will compute the route which has the shortest hop count. On the other hand, since the backbone handles most of the traffic for messages exchanged between different clusters, there is the tendency of it becoming a performance bottleneck with high traffic loads especially when messages have to travel multiple hops through the backbone before getting to the destination gateway. In this work, we are particularly concerned about efficient routing through the backbone to reduce the likelihood of the backbone becoming a performance bottleneck. Thus, we focus more on optimizing routing through the backbone. In the clusters, the intra-domain routing protocol, OLSR here, will always select the shortest path to deliver packets to the destination host although the selected gateway might not have the least number of hops to the destination host. To optimize routing through the backbone, the value stored against every key in the DHT which ought to be the IP address of a destination gateway can be maximized to take more information.

Instead of one IP address, the IP addresses of both gateways can be serialized as an encapsulated list and stored as the value *v* in the *PUT* message. This way, a source gateway that retrieves this information, will use the hop count information in its routing table to select the closest gateway. Overall, selecting the closest gateway through the backbone does not translate to the shortest path to a destination host. This approach serves to minimize the traffic through the backbone.

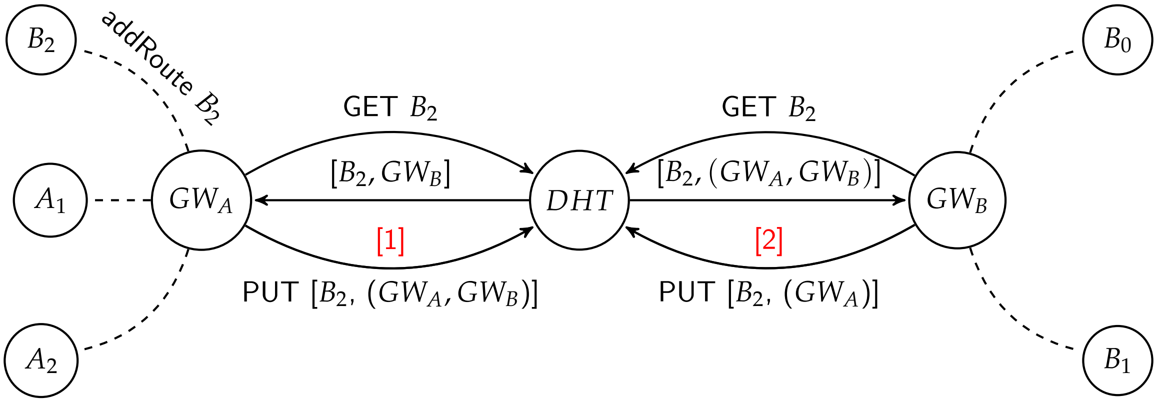


Figure 3.7: DHT Updates [A]

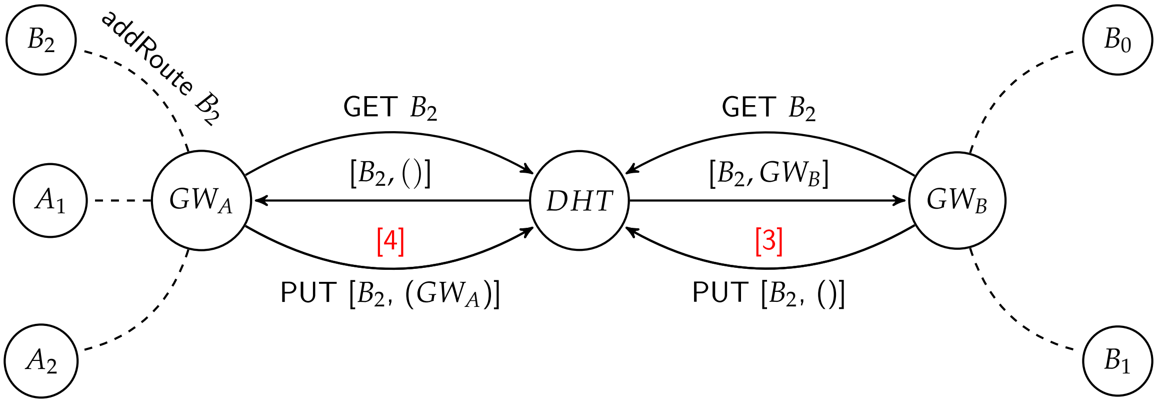


Figure 3.8: DHT Updates [B]

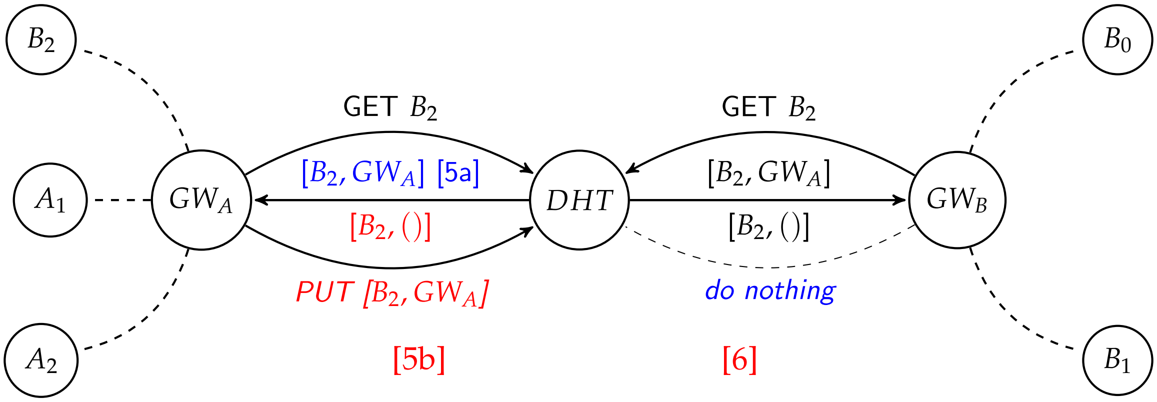


Figure 3.9: Check Race Condition

Figures 3.7, 3.8, and 3.9 show the way the DHT is updated to support the different mobility scenarios we study in this thesis. In each of these figures, hosts A_0, A_1, A_2 and gateway GW_A are members of cluster A while hosts B_0, B_1, B_2 and gateway GW_B are members of cluster B. B_2 is a host previously in cluster B but moved to cluster A. The arrows represent DHT queries and responses while the dotted lines are wireless connections. In order to provide a solution that supports all our mobility scenarios, every gateway queries the DHT before adding an entry to it. One query is distinguished from the other via different code names: ADD_ROUTE, $DELETE\_ROUTE$, $LOCAL\_UPDATES$. According to each of these code names, a gateway GW_X will handle the response to each query differently. The three mobility cases together with the steps involved in updating the DHT in each case are described below.

1. ADD_ROUTE

   One way to guarantee that the DHT is always up to date with the most recent routing advertisements is that every gateway periodically advertises reachability to the hosts they can reach as per their routing table entries. However, this is an inefficient scheme with high redundancy because there is a high possibility that gateways will frequently update information that has not changed. Another downside to it is the redundant control overhead that will be generated for all the redundant updates. In our scheme, we do controlled updates, which is similar to what happens in reactive MANET routing protocols [70, 71].

   Initially, every gateway advertises reachability to all hosts in its cluster in the DHT as explained in Section 3.6. Subsequently, in events like clusters merging or hosts joining, the gateways will learn about these new hosts after their routing tables have been updated by the intra-domain routing protocol, OLSR. The gateway then hashes the individual Internet Protocol (IP) addresses of the new hosts and queries the DHT. This query will either be successful and return a valid value, a NULL value, or completely fail. According to the result of a query for a host B_2, a gateway GW_A will do the following:

(a) A valid value means that an entry for B_2 already exists in the DHT. If the value retrieved corresponds to the glsip address of GW_A, this entry need not be updated as such: Figure 3.9 - [5a]. Otherwise, GW_A will add its glsip address to the existing entry and update the DHT as such: Figure 3.7 - [1].

(b) A NULL value means that no entry exists for B_2 in the DHT. GW_A will then hash the IP address of B_2 to generate key k and initiate a PUT message to advertise its reachability to N in the DHT with the message: $PUT(k, GW_A)$ as such: Figure 3.8 - [4].

(c) When a query fails for whatever reason (neither returns a valid value nor a NULL value), GW_A chooses a random back-off interval between 5-10 seconds and re-issues the query to the DHT.

2. DELETE_ROUTE

When a host B_2 leaves a cluster, or clusters split and move away, the gateway(s) get to know that a host or hosts have left after their routing tables have been recalculated and invalid routes are flushed by the routing protocol, OLSR in this case. The gateways will hash the IP address of a host B_2 to generate key k and query the DHT, which will return the value currently stored against key k as such: Figure 3.7 - [2], Figure 3.8 - [3] or Figure 3.9 - [6]. If the value retrieved is a list of entries for which one of the entries corresponds to GW_A's IP address, it removes its IP address from the list and updates the DHT as such: Figure 3.7 - [2]. Otherwise, for a single entry corresponding to its IP address, it removes the entry, and updates the DHT with a NULL value as such: Figure 3.8 - [3]. As it is difficult to synchronize the DHT updates between GW_A and GW_B for host B_2, there is the possibility of a race condition between both gateways. For example, in Figure 3.7, event [1] happens before event [2] and in Figure 3.8, event [3] happens before [4]. These two scenarios are the best cases under which DHT updates will always succeed. However, in Figure 3.8, if event [4] happens before event [3], an invalid entry will be advertised for B_2 in the DHT and B_2 will be out of reach unless the right entry is updated in the DHT. To handle this, each gateway will choose a random time in 1-5 second intervals and check with the DHT to make sure that its update was successful, as such: Figure 3.9 [5], [6]. This solution does not guarantee that race conditions will be handled (if they happen at all), but it will reduce their possibility/occurrence. As our simulation results show, such race conditions do not occur frequently enough to visibly impact the protocol performance.

3. LOCAL_UPDATES

   Each gateway maintains a separate data structure, *TEMP_ROUTING_TABLE*, where it stores all the routing information for the different destinations

$X_X, ..., Y_Y, ..., Z_Z$ it retrieved from the DHT as such:

$$destinationAddress = X_X \; ; \; gateway = GW_X \; ; \; validityTime$$

$$destinationAddress = Y_Y \; ; \; gateway = GW_Y \; ; \; validityTime$$

$$destinationAddress = Z_Z \; ; \; gateway = GW_Z \; ; \; validityTime$$

In situations like hosts switching cluster membership, the routing information becomes stale. To get the most recent route advertisements, one way is for the gateway GW_X responsible for a hosts X_X, to inform all other gateways that it can no longer reach X_X, and probably provide information about X_X's new location. Another way is for X_X to leave instructions with a home agent in its home cluster/MANET on how it can now be reached which is similar to what happens in Mobile IP [55]. This brings about a whole lot of complexities and location update message exchange. To avoid this complexity and still provide similar services, the gateways query the DHT periodically at the expiry of each entry's validity time, to obtain the most recent route advertisements for each destination $X_X, ..., Y_Y, ..., Z_Z$ stored in *TEMP_ROUTING_TABLE*. After a successful query, the most recent route information is updated for each destination and the validity time is incremented. When a host Y_1 in cluster Y who was initially sending messages to X_1 in cluster X leaves cluster Y, the gateway GW_Y will no longer send pings on the behalf of Y_1. Therefore, the route corresponding to destination X_1 in *TEMP_ROUTING_TABLE* will no longer be updated. However, if there exists multiple sources from cluster Y, sending messages to destination X_1, then the route will be updated and kept fresh in *TEMP_ROUTING_TABLE*. In our study, we selected a 30 second interval for these periodic updates. We choose this interval after experimenting and observing the average time it takes gateways to update the DHT with recent route advertisements. We were also concerned with avoiding too much traffic in the backbone, which may be redundant as hosts might not switch cluster membership at all or too often throughout the network life time.

3.4 Node Design

<table>
<tr><td colspan="2"><b>APPLICATION LAYER</b>
<i>DHT Application</i></td></tr>
<tr><td colspan="2"><b>TRANSPORT LAYER</b>
<i>TCP/UDP</i></td></tr>
<tr><td><b>NETWORK LAYER</b>
$OLSR_1$</td><td><b>NETWORK LAYER</b>
$OLSR_2$</td></tr>
<tr><td><b>LINK LAYER</b>
<i>IEEE 802.11</i></td><td><b>LINK LAYER</b>
<i>IEEE 802.11</i></td></tr>
<tr><td><b>PHYSICAL LAYER</b>
<i>wlan 0</i></td><td><b>PHYSICAL LAYER</b>
<i>wlan 1</i></td></tr>
</table>

Figure 3.10: Gateway Node/Cluster Head Protocol Stack

In order to support inter-domain routing and intra-domain routing, the cluster heads, which serve as gateways to their local clusters, are specially designed to support two interfaces and run separate routing protocols or an instance of the same routing protocol on the separate interfaces they support. At the physical layer, each gateway node communicates with members of its local cluster on *wlan 0*, and with the members of the backbone network over *wlan 1*. Both interfaces are physically separated from each other using channel assignment. Each *wlan* receives/sends packets from/to the next layer in the hierarchy, the link layer, which implements two instances of IEEE 802.11 to coordinate the activities of the different routing domains. Similarly, the network layer supports two routing protocols which operate independently. These routing protocols build their individual routing table data structures separately and populate a single routing table with routing entries which the gateway uses for routing over the different interfaces. At the network layer, both routing protocols are separated from each other using input and output gates. This way, the network layer is able to properly handle information concerning the different routing domains efficiently.

Chapter 4

Implementation of Algorithms

Section 4.1 gives a brief overview of the simulation environment we use in this study, here *Omnet*++. As some of the important features and modules which we need for our study were not available in the existing libraries *Omnet*++ uses, we had to implement them. We wrote code for all our additional functionality in C++ which is the implementation language for *Omnet*++. In Sections 4.2 to 4.5, we present all the algorithms which we implemented towards this study and discuss them.

4.1 The Simulation Environment

Omnet++ [18, 72] is an open source C++-based discrete event simulator for modeling communication networks and other distributed or parallel systems. It is available on common platforms including Linux, Mac OS/X, and Windows, using the GCC tool chain or the Microsoft Visual C++ compiler. *Omnet*++ represents a framework approach. Instead of directly providing simulation components for computer networks or other domains, it provides the basic machinery and tools to write such simulations. Specific application areas are supported by various simulation models and frameworks such as the Mobility Framework or the *INET* Framework. These models are developed independently of *Omnet*++ and follow their own release cycles. *Omnet*++ 4.6 is the version of *Omnet*++ we used.

The INET Framework [18, 73] is an open-source model library for the *Omnet*++ simulation environment. It provides protocols, agents, and other models for researchers working with communication networks, and is especially useful when designing and validating new protocols, or exploring new or exotic scenarios. INET contains models for the Internet stack (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), wired and wireless link layer protocols (Ethernet, PPP, IEEE 802.11, etc), support for mobility, MANET routing protocols (OLSR, AODV, etc), several application models, and many other protocols and components. INET benefits from the infrastructure provided by *Omnet*++. Beyond making use of the services provided by the *Omnet*++ simulation kernel and library (component model, parameterization, result recording, etc.), this also means that models may be developed, assembled, parameterized, run, and their results evaluated from the comfort of the *Omnet*++ Simulation IDE, or from the command line. The version of INET Framework we used is INET 2.0.

OverSim [74] is another open-source overlay and peer-to-peer network simulation framework for the *Omnet*++ simulation environment we use in this study. It contains several models for structured (e.g. Chord, Kademlia, Pastry) and unstructured (e.g. GIA) P2P systems and overlay protocols. The version of Oversim we used is OverSim-20121206.

GW_A's Routing Table			
Destination	Next-hop	Hop-count	Interface
A_0	A_0	1	wlan0
A_1	A_1	1	wlan0
A_2	A_2	1	wlan0
0.0.0.0	255.255.255.255	-	wlan1

Table 4.1: GW_X's Routing Table

Algorithm 1: SIMPLE FLOODING PROTOCOL : *forwarding*

```
Input: IP Packet
begin
    if $\langle IP\ packet\ \in Duplicate\rangle$ then
        drop IP Packet
    else
        $Duplicate \leftarrow \langle Duplicate\ \cup IP\ Packet(creationTime,\ validityTime = creationTime + holdTime)\rangle$
        route IP Packet
    $Duplicate \setminus expired\ IP\ Packets$
```

4.2 Flooding-based Protocol

Flooding or network-wide broadcasting is the process in which one node sends a packet to all other nodes in the network, or to every outgoing interface except on which it arrived on. Many applications as well as various unicast routing protocols such as DSR, AODV, and the likes use broadcasting or a derivation of it. The principal use of flooding in these protocols is for Location Discovery and for establishing routes. A straightforward approach for broadcasting is blind flooding, where each node will rebroadcast any packet it receives on all its interfaces except the one it came from. Blind flooding will generate many redundant transmissions, which may cause what is called the broadcast storm problem [19] and waste wireless resources.

The flooding protocol we implemented works the same way as blind-flooding. Only the gateways do flooding and so, they add a route to their routing table as such: $destAddress = 0.0.0.0, sourceAddress = 0.0.0.0, nextHop = 255.255.255.255, interface = backbone\_interface$. Table 4.1 shows the content of a gateway's routing table after the route has been added. When a gateway GW_X receives a packet for the first time on either its local interface destined for B_Y in cluster Y or on the backbone interface, it keeps a duplicate copy of this packet in a duplicate set before forwarding the packet on the backbone interface according to Algorithm 1. The input to Algorithm 1 is an IP Packet. Every packet is distinguished from the other by their individual creation time which is in the order of microseconds. Thus they, with high probability, are distinguishable. In addition, every entry in the duplicate cache is assigned a validity time equal to $creationTime + holdTime$, after which it will be discarded. This is important because the gateways handle a lot of packets per second and it is redundant storing these packets for the entire network lifetime as we only need them for short periods of time, in order to track duplicate copies of the same packet. Moreover, the cache does not have infinite capacity. Depending on the transmission delay experienced at the gateways, a delayed packet at GW_X which was previously processed by GW_Y might show up again at GW_Y after GW_X finally transmits it. If GW_Y has prematurely discarded information about this packet and therefore fails to recognize it as a duplicate, it will broadcast it in the backbone as a new packet and other gateways will do the same. Thus, the hold time has to be chosen carefully. After tests we performed with a range of values, we chose 10 seconds for the hold time. Subsequently, for every packet GW_X receives, it will check with its duplicate packet cache and will not forward the same packet more than once. The duplicate cache contains all the packets a gateway has processed once on the backbone interface. After forwarding a packet, GW_X will always check with the duplicate set to remove all packets that have expired.

4.3 Host and Network Association (HNA)

RFC 3626 describes the Optimized Link State Routing (OLSR) protocol for mobile ad hoc networks [17, 36]. In this document, the functionality of OLSR is divided into the core and auxiliary functionality respectively. The core specification is always required for the protocol to operate; specifying the behavior of a node participating in the MANET and running OLSR as routing protocol on its OLSR interfaces. On the other hand, in addition to the core functionality, there will be instances where additional functionality is desired. The auxiliary specification thus provides additional functionality which may be applicable in specific scenarios. An example auxiliary functionality is a situation where a node in the MANET has multiple interfaces, some of which participate in other routing domains (Might be non-OLSR interfaces or OLSR interfaces completely isolated from each other). This kind of auxiliary functionality is called external access [17]. In order to provide connectivity from the local OLSR MANET interface(s) to these external interfaces, a node (usually a gateway or cluster head) SHOULD be able to inject external route information to the local OLSR MANET. These external interfaces may be point-to-point connections to other singular hosts or may be connections to separate networks or the Internet. *Omnet++* [18] uses the INET library [73] which already has the core functionality for OLSR implemented. RFC 3626 [17] provides a step by step approach towards implementing other functionalities. This section describes the external access component of OLSR's auxiliary functionality, called HNA (Host and Network Association) messages, and how we implement this functionality in the *Omnet*++ simulator.

As stated in RFC 3626, a MANET can exist unregimented - isolated/independent of other networks. However, in situations where MANETs coexist or may need to contact other non MANET interfaces, the Internet for example, the option to connect to other networks should be offered on some level since this is very likely to be a requirement in many situations [17].

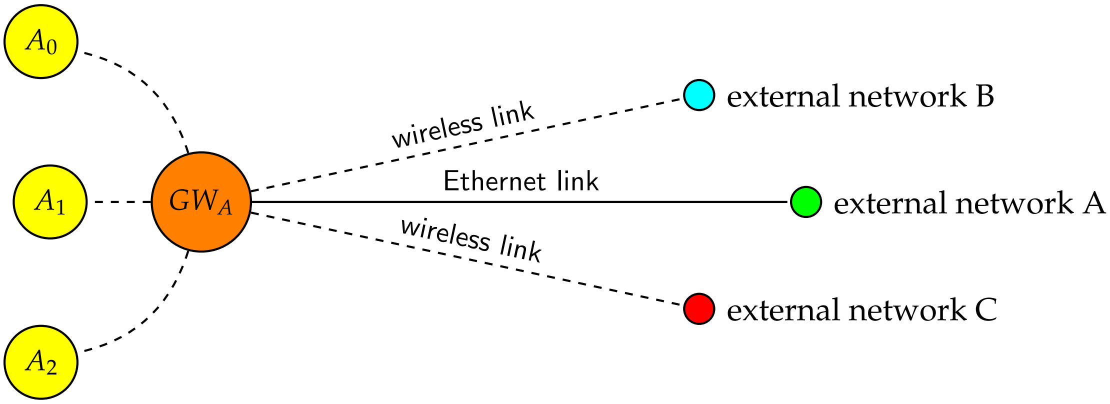


Figure 4.1: A Gateway Node Associated with External Networks

A node associated with such non-MANET interfaces, mostly a gateway or cluster head, in order to provide this capability of injecting external routing information into an OLSR MANET, periodically issues a Host and Network Association (HNA) message, containing sufficient information for the recipients to construct an appropriate routing table entry.

Figure 4.1 illustrates the most basic case of a typical external access scenario. In the figure, the gateway, GW_A, has an Ethernet link and wireless links on which it has external access to A, B and C, and wishes to offer connectivity to hosts $A_0, A_1, and\ A_2$ in the MANET. GW_A will, therefore, announce its reachability to external networks A, B and C to hosts $A_0, A_1, and\ A_2$ by propagating HNA messages in the MANET. All nodes do not need to support the HNA functionality for HNA messages to be flooded throughout the MANET. Using OLSR's default forwarding algorithm, these messages are propagated through the network. However, all nodes MUST support HNA-message processing and route calculation for HNA routing to work. For example, since routing in MANETs follows a per-hop fashion, if a node within the MANET routes external traffic to an intermediate neighbor, who is the next hop address based on HNA information, the intermediate neighbor acts as a relay and must also have set up an HNA route for the traffic to be routed along the path. Therefore, the neighbor, and other intermediate nodes in the path, must support HNA functionality [17] for external traffic to ultimately reach the gateway.

4.3.1 HNA Message Format and Generation

Algorithm 2: HNA Message Generation

```
Input: Timer Expire
begin
    generate HNA message
    $srcAddress = GW$
    $destAddr = 255.255.255.255$
    $interface = cluster$
    $message = (network \rightarrow 0.0.0.0; mask \rightarrow 0.0.0.0)$
```

A HNA message is basically a list of Network address and Netmask pairs corresponding to external domains reachable by a gateway/cluster head of a MANET. This message content is sent as the data part of the OLSR packet with the ”Message Type” field set to HNA_MESSAGE or its integer value ('4'), the Time To Live (TTL) field set to 255 and Validity_time set accordingly to the value of HNA_HOLD_TIME (3 x HNA_INTERVAL), where HNA_INTERVAL = 5 seconds as recommended in

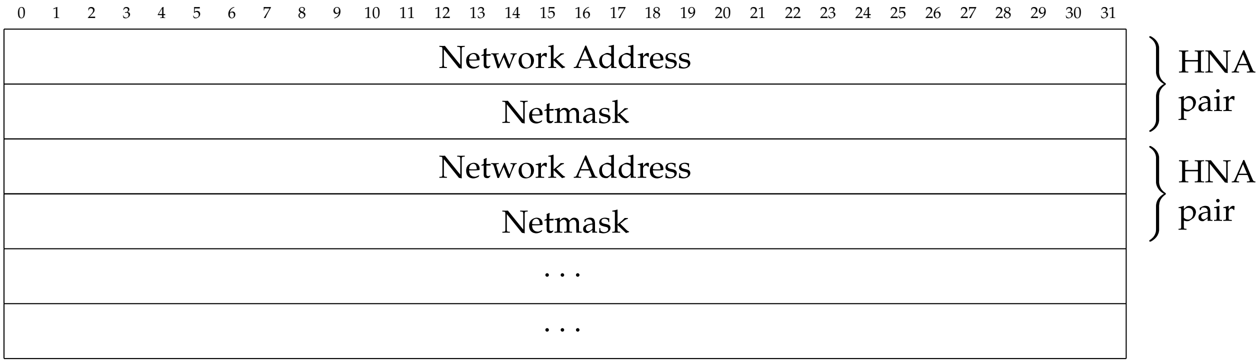


Figure 4.2: HNA Message Format

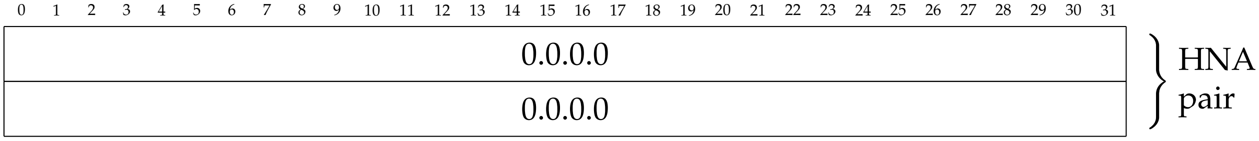


Figure 4.3: Optimized HNA Message

RFC 3626 [17]. However, this is not a fixed value and can be configured to whatever desired value.

According to this scheme, if a node in a MANET has reachability to network 192.168.10.0/24, this node will announce the following information in a HNA message: ***network_address :: 192.168.10.0, netmask :: 255.255.255.0*** and in the format of Figure 4.2. This node will periodically generate a Host and Network Association (HNA) message, containing pairs of (network address, netmask) corresponding to the connected hosts and/or networks. The periodic value for this transmission is every HNA_INTERVAL (5 seconds) [17]. A host within a MANET which has no associated hosts or networks SHOULD NOT generate HNA messages. Such hosts will only participate in the default forwarding exercise as per the OLSR core functionality [17]. As described in Section 3, we optimize the HNA message size and also reduce the routing table size by limiting the number of announced networks to a singe default route as represented in Figure 4.3. Algorithm 2 shows how we implement HNA generation. It takes two inputs : 'Timer Expire' which triggers a HNA generation and parameters c, H, GW which denote $currentnode, Host, GateWay$. We assume that the current node is the gateway. Once its HNA timer expires, it generates and propagates a HNA message in its local MANET as such: $network\_address$: 0.0.0.0, $netmask$: 0.0.0.0 .

4.3.2 HNA Message Processing

Algorithm 3: HNA ROUTING TABLE COMPUTATION

```
Input: HNA Message $\langle network,\ mask,\ gateway \rangle$
begin
    if $\langle network \in routing\ table \rangle$ then
        if $\langle HNA\ Message \rightarrow gateway\_hopCount \rangle < \langle routing\ table \rightarrow gateway\_hopCount \rangle$ then
            $routing\ table = routing\ table \setminus \langle routing\ table \rightarrow gateway \rangle$
            $routing\ table = routing\ table \cup \langle HNA\ Message \rightarrow gateway \rangle$
    else
        $routing\ table = routing\ table \cup \langle network,\ mask,\ gateway \rangle$
```

We implemented HNA processing functionality as per RFC 3626, which provides a standard step-by-step approach [17]. Each node maintains an association base for all the HNA routes it receives. In scenarios where a MANET might have multiple gateways, there will be multiple HNA sources. A host will, therefore, maintain a copy of all the HNA messages it received from the different sources in its association base. Since each node receives an HNA message every HNA interval, it will check with its association base to update or create a new entry for each HNA message according to RFC 3626 [17]. After a Host has processed an HNA message, it then creates a routing table entry as explained in Section 4.3.3.

4.3.3 HNA Route Calculation

Algorithm 3 shows how we implement HNA route calculation as per RFC 3626 [17]. It takes three inputs: ′OLSR Message′ which specifies the type of message received (*HNA message*), ′routing table′ with which it makes its comparison before computing a route, and parameters c, H, GW which denote $currentnode, Host, GateWay$. We assume that the current node ′c′ is a host. For every entry a host maintains in its association base, it selects the best route as a function of the number of hops it takes to reach the originator of the HNA message, here the gateway. This process is repeated every time a host processes an HNA message. Thus, in a scenario where a host receives HNA messages from multiple sources, it will always select the best route (i.e., to the closest gateway). This is particularly useful in cases where nodes become mobile and routes are not fixed.

4.4 DHT Application

Algorithm 4: NETWORK LAYER: MESSAGE HANDLER(*DHT Requests*)

```
Input: Routing Table Update ⟨routeAdded, getDestination, routeDeleted⟩
begin
    if ⟨message = [ routeAdded || routeDeleted || getDestination ]⟩ then
        DHT Notification
```

In this section, we detail the implementation related to our DHT application. As earlier stated, Oversim [74] is one of *Omnet*++′s [18] libraries which includes several structured and unstructured P2P (peer-to-peer) protocols, including Chord [62, 59], which our DHT uses.

Our DHT carries out two major operations: route advertisement, denoted as

Algorithm 5: DHT APPLICATION : *DHT Notification*

```
Input: notifications $\langle route,\ getDestination,\ routeAdded,\ routeDeleted \rangle$
begin
    if $\langle notification == routeAdded \rangle$ then
        if $\langle \langle route.interface == local \rangle \&\& \langle route \ni localCache \rangle \rangle$ then
            $local\ cache = local\ cache \cup route$
            $DHT\ queries[DHT\_GET(route.destAddress)]$
        else
            if $\langle route \ni gatewayCache \rangle$ then
                $gatewayCache\ cache = gatewayCache\ cache \cup route$
    if $\langle notification == routeDeleted \rangle$ then
        if $\langle \langle route.interface == local \rangle \&\& \langle route \ni localCache \rangle \rangle$ then
            $DHT\ queries[DHT\_GET(route.destAddress)]$
    if $\langle notification == getDestination \rangle$ then
        if $\langle route.interface == local \rangle$ then
            $DHT\ queries[DHT\_GET(route.destAddress)]$
```

addRoute, and route deletion, denoted as *deleteRoute*. Other operations which are updates to the DHT and checks with the DHT are variations of these two operations and are discussed below.

Algorithm 4 implements a message handling service which handles the message exchange between the DHT application and the routing layer. The OLSR protocol in INET [73] implements a notification service 'Routing Table Update' and by subscribing to this service, our message handler gets to know when new routes are added to a gateway's routing table. Other notifications which Algorithm 4 handles are to signal route delete and route request operations to the DHT. When the local instance of the OLSR protocol adds a route for destination host $'X_X'$ to gateway GW_X's routing table, Algorithm 4 notifies the DHT by calling Algorithm 5. The inputs to Algorithm 5 are *getDestination, routeAdded, and routeDeleted* notifications. Algorithm 5 understands from the type of the notification that this is a *routeAdded* operation. A cache, *dhtCache*, is used to track all the entries registered in the DHT. Before registering an entry in the DHT with Algorithm 6, Algorithm 5 checks with the *dhtCache* for an existing entry and creates a new one where none exists. GW_X only advertises hosts in its local cluster in the DHT while other nodes, which are the gateways belonging to the backbone, are maintained in a separate cache, *gateway_Cache*. As routes to all destinations which GW_X can reach are maintained in a single routing table, Algorithm 5 differentiates between the hosts and the gateways with the interfaces from which these routes are added to GW_X's routing table, for example ($wlan0$, $wlan1$). The inputs to Algorithm 6 specify the kind of DHT query (a PUT or a GET) and the gateway's IP Address. It then issues a $GET(k)$ query to the DHT by hashing the IP address of the destination node in the message it receives to generate key k, using $SHA1$ hashing algorithm. Oversim already has $SHA1$ implemented.

Algorithm 6: DHT APPLICATION : *DHT queries*

```
Input: DHT query $\langle GET(IP\_Address); PUT(IP\_Address); GW\_IP\_Address \rangle$
begin
    if $\langle query == DHT\_GET \rangle$ then
        $k \leftarrow SHA1(IP\_Address)$
        $send\_GET(k)$
        $attach\ control\ information$
    else
        $k \leftarrow SHA1(IP\_Address)$
        $list \leftarrow serialize(GW\_IP\_Address)$
        $v \leftarrow list$
        $send\_PUT(k, v)$
        $attach\ control\ information$
```

Algorithm 7 returns the response to all the queries made to the DHT. Its inputs

Algorithm 7: DHT APPLICATION : *DHT query responces*

```
Input: DHT query responses $\langle GET\ response(data); PUT\ response(data); GW\_IP\_Address \rangle$
begin
    if $\langle response == GET\_response \rangle$ then
        if $\langle response == FAILED \rangle$ then
            $choose\ backoff\ interval\ in\ range(1-5)$
            $attach\ control\ information + reschedule\ GET(k)$
        if $\langle response == NULL$ then
            if $\langle GET.kind == ADD\_ROUTE \rangle$ then
                $attach\ control\ information$ + call Algorithm 6
            if $\langle GET.kind == ROUTE\_REQUEST \rangle$ then
                $choose\ backoff\ interval\ in\ range(1-5)$
                $attach\ control\ information + reschedule\ GET(k)$
        if $\langle response == SUCCESS \rangle$ then
            if $\langle GET.kind == DELETE \rangle \&\& \langle value = GW\_IP\_Address \rangle$ then
                $value \leftarrow NULL$
                $attach\ control\ information +$
                  $DHT\ queries[DHT\_PUT(data.destAddress,\ value)]$
            if $\langle GET.kind == ADD\_ROUTE \rangle$ then
                $value \leftarrow GW\_IP\_Address$
                $attach\ control\ information +$
                  $DHT\ queries[DHT\_PUT(data.destAddress,\ value)]$
            if $\langle GET.kind == ROUTE\_REQUEST \rangle$ then
                $listOf(Dest\_GW\_IP) \leftarrow \langle data \rightarrow v \rangle$
                $attach\ control\ information + pass\ down\ routing\ information$
    if $\langle response == PUT\_response \rangle \&\& \langle response == Failed \rangle$ then
        $choose\ backoff\ interval\ in\ range(1-5)$
        $attach\ control\ information + reschedule\ PUT(k)$
```

are the 'DHT query responses' which specify the kind of DHT response and the gateway's IP Address. Sufficient control information is attached with every query which is then used to distinguish between different responses. For an *addRoute*, the possibilities are that $GET(k)$ was successful and returned a valid or a NULL value, or it completely failed. For a failed response, the query is reissued after a back-off interval which is a normal distribution $N(rand(5), 2)$. A NULL response signifies that no entry existed in the DHT against key k. In this case, a value corresponding to the IP Address of GW_X is then stored against key k by issuing a $PUT(k, GW_X)$ message with Algorithm 6. A response with a valid value (for example GW_Y) indicates that a value existed in the DHT against key k. This usually happens when a host that was previously in cluster Y switches to cluster X. In such a case, GW_X will include its IP Address as one of the values stored against key k, and update the DHT by issuing a $PUT(k, [GW_X, GW_Y])$ message with Algorithm 6. The $PUT(k, [GW_X, GW_Y])$ operation can either be successful or completely fail. If the PUT failed, the query, $PUT(k, [GW_X, GW_Y])$, is reissued with Algorithm 6 after the same back-off interval used above.

Algorithm 8: DHT APPLICATION : *Best gateway calculation*

```
Input: Receive routing information
       ⟨$control\ information, listOf(Dest\_GW\_IP),\ GW\_IP\_Address$⟩
Output: best route to be passed down to the network layer for routing
begin
    $buffer \leftarrow listOf(Dest\_GW\_IP)$
    for ⟨$i = 0;\ i = size\_of\_buffer;$⟩ do
        select the best gateway from the list
        create IP datagram [Dest_GW_IP, control information]
    $Stote\ routing\ information[IP\ datagram]$
```

When GW_X has to deliver X_X's packet to destination Y_Y, GW_X does not find a

Algorithm 9: NETWORK LAYER : *Store routing information*

```
Input: receive notification from application Layer $\langle destHost,\ destGateway \rangle$
begin
    for $\langle i = begin;\ i \neq end; \rangle$ do
        if $\langle [destHost,\ destGateway] \in temp\_routing\_table \rangle$ then
            update temp routing table
        else
            $temp\_routing\_table = temp\_routing\_tabl \cup$
              $[destHost,\ destGateway,\ validityTime = 30sec]$
        if $\langle validityTime == expired \rangle$ then
            $DHT\ queries[DHT\_GET(destHost)]$
```

route to Y_Y in its routing table. Through the message handler, Algorithm 4, it sends a notification message, *'getDestination'*, to the DHT which receives notifications from the routing layer through Algorithm 5. Algorithm 5 understands from the message type, *'getDestination'*, that this is a route request from the DHT. Through Algorithm 6, a $GET(k)$ query is issued to the DHT by executing $SHA1$ on Y_Y to generate key k. Algorithm 7 receives the response to this query which could be successful - returning a NULL or a valid value or completely fail. For a failed response, the same procedure as described above is followed. Otherwise, a NULL will indicate that no entry is currently stored against key k in the DHT. This never happened in the course of our study but an explanation for this is that a host, X_X, initially in cluster X, switched to cluster Y and its previous gateway, GW_X, removed the value it originally advertised against key k from the DHT before GW_Y was able to advertise its reachability to host Y. In such a case, the packet destined for X_X is dropped and the DHT is again queried when there is a route request for destination X_X. For a successful response, Algorithm 8 is called which sends a message containing the route retrieved from the DHT to Algorithm 9 at the routing layer. Algorithm 8 takes two inputs: the value retrieved from the DHT along side some control information and the gateway's IP Address. In the case that the value retrieved from the DHT contains multiple gateways to host Y_Y, Algorithm 8 selects the best route based on the number of hops it takes to reach each gateway through the backbone. It obtains the number of hops from the gateway's routing table. Algorithm 9 takes one input: the routing information retrieved from the DHT. It caches every new route it receives and replaces already existing ones with the most resent routing information received as such : *source, destination, gateway*. These routes can then be reused for subsequent routing to the same destination. Each route has a validity time after which the gateway will issue a query through Algorithm 4 to the DHT.

Algorithm 10: NETWORK LAYER : *Routing table diff operation*

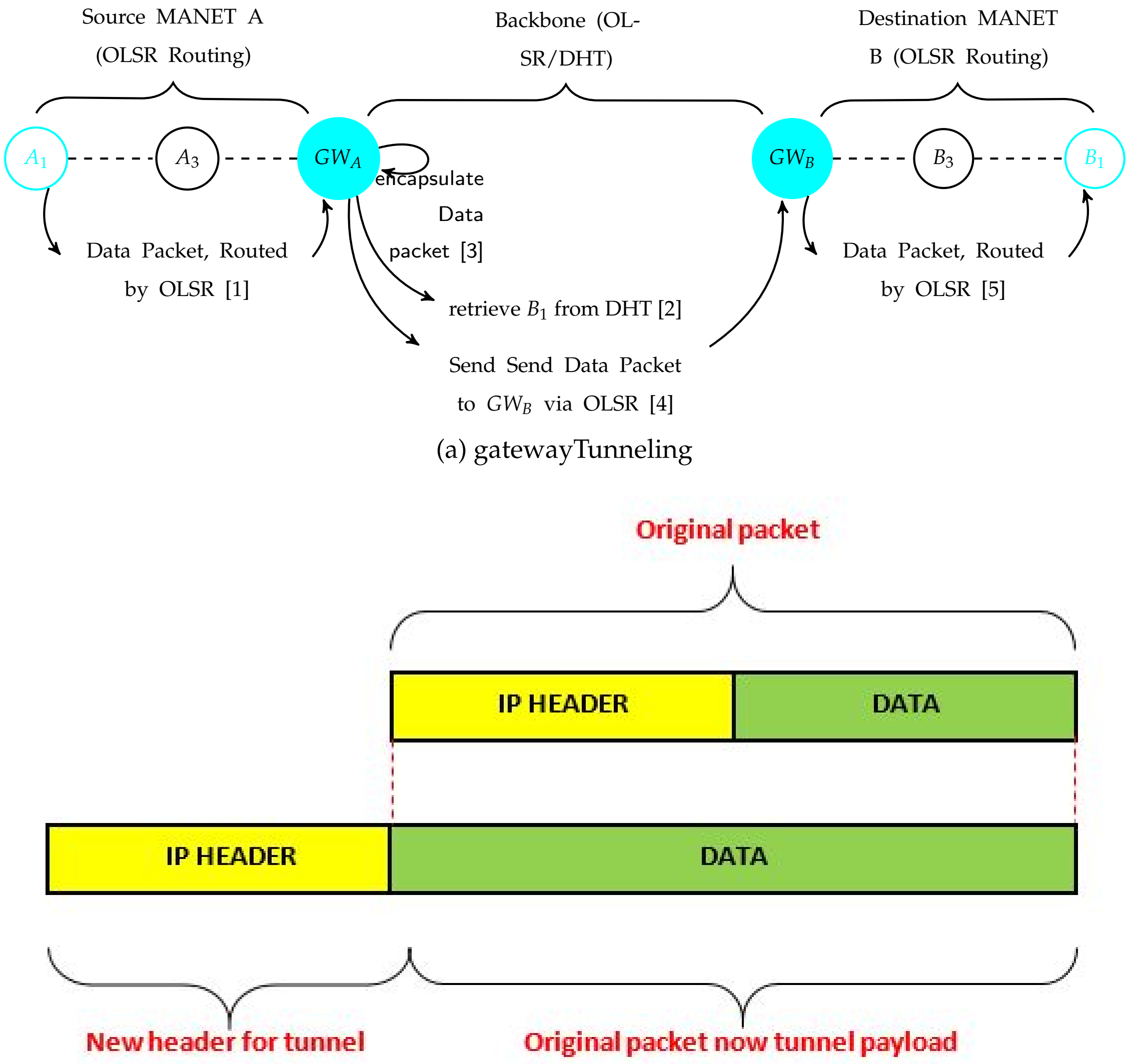
```
Input: Routing Table $\langle Routes \rangle$            ▷ the gateway's current routing table
Input: diff data Structure $\langle entries \rangle$      ▷ the gateway's previous routing table
begin
    for $\langle i = entries.begin;\ i \neq entries.end; \rangle$ do
        for $\langle i = Routes.begin;\ i \neq Routes.end; \rangle$ do
            if $\langle entry[i]\ \ni Routes \rangle$ then
                $GET.kind \leftarrow routeDeleted$
                $DHT\ queries[DHT\_GET(entry[i])]$
                $entries = entries \setminus entry[i]$
    for $\langle i = Routes.begin;\ i \neq Routes.end; \rangle$ do
        for $\langle i = entries.begin;\ i \neq entries.end; \rangle$ do
            if $\langle Routes[i] \ni entries \rangle$ then
                $entries = entries \cup Routes[i]$
```

In the event that host Y_Y moves from cluster Y to Z, while GW_Z advertises its reachability to Y_Y in the DHT by following the *addRoute* operation procedure described above, GW_Y will have to remove the value it stored against Y_Y in the

DHT as it can no longer reach Y_Y. For this reason, we implemented Algorithm 10. Algorithm 10 takes two inputs: the most resent routing table entries and the previous copy stored in the diff data structure. It maintains an older version of a gateway, 'GW_X's' routing table and carries out a diff operation with the newer version. In the event that a host or hosts formerly in GW_X's local cluster are no longer reachable, this will be reflected in GW_X's routing table after the local instance of OLSR updates its routing table. This operation thus yields the hosts who have left the cluster. The process then notifies the DHT via Algorithm 4 with a *routeDelete* message. Algorithm 5 receives this message and through Algorithm 6, a $GET(k)$ query is issued to the DHT for each host that can no longer be reached. Algorithm 7 receives the response to each query and processes it the same way others were processed in the description above.

On the other hand, GW_X will no longer be able to deliver host X_X's messages to host Y_Y with the previous destination gateway address it retrieved from the DHT, here GW_Y. For this reason, each gateway implements an internal timer which is triggered every 30 seconds. When triggered, a *getDestination* notification is sent to the DHT via Algorithm 4. This way, the gateways are able to get the most recent route advertisements from the DHT. Before querying the DHT, GW_X checks for irrelevant routes and removes them. For example, if host X_X in cluster X was previously sending messages to host Y_Y in cluster Y, and moves away to cluster Z, GW_X, before querying the DHT for a route update for Y_Y, will check with its routing table that it can still reach X_X otherwise, it will not issue the query. Subsequently, the DHT will not be queried for destinations that messages are no longer routed to. Algorithm 5 receives this notification and understands that this is a *getDestination* operation. It then sends a $GET(k)$ query to the DHT for the particular route requested through Algorithm 6. Algorithm 7 receives the responses which are treated in the same way as other GET responses discussed above.

4.5 IP Packet Tunneling


Source MANET A (OLSR Routing)
Backbone (OLSR/DHT)
Destination MANET B (OLSR Routing)
A_1
A_3
GW_A
GW_B
B_3
B_1
encapsulate Data packet [3]
Data Packet, Routed by OLSR [1]
retrieve B_1 from DHT [2]
Send Send Data Packet to GW_B via OLSR [4]
Data Packet, Routed by OLSR [5]
Original packet
IP HEADER
DATA
IP HEADER
DATA
New header for tunnel
Original packet now tunnel payload

(a) gatewayTunneling

(b) Tunnelled Packet

Figure 4.4: IP Packet Tunneling

In IP networks, encapsulation is suggested as a means to alter the normal IP routing for datagrams, by delivering them to an intermediate destination that would otherwise not be selected based on the (network part of the) IP Destination Address field in the original IP header [75]. At this intermediate destination node, the IP packet is decapsulated, yielding the original IP datagram, and delivered to the destination as per the original Destination Address field. This process is frequently referred to as "tunneling" the datagram, and the encapsulator and decapsulator are the "endpoints" of the tunnel [75].

In the most general tunneling case we have : $source \Rightarrow encapsulator \Rightarrow decapsulator \Rightarrow destination$; with the source, encapsulator, decapsulator, and destination being host A_1, *nodes* GW_A, GW_B, *and host* B_1 respectively as illustrated in Figure 4.4a. GW_A is considered the "entry point" of the tunnel, and GW_B is considered the "exit point" of the tunnel.

In Figure 4.4, after A_1 routes a packet destined for B_1 to GW_A using the HNA route [1], and GW_A retrieves GW_B's address from the DHT [2], it does not immediately route A_1's packet to B_1. This is because GW_A is not the original destination of this packet and will normally drop the packet since it can not identify the destination based on the (network part of the) IP Destination Address field in the original IP header. GW_A will, therefore, encapsulate the packet [3] by creating a new IP packet with source address : GW_A and destination address : GW_B, while the original packet, including header, assumes the payload of the new packet as illustrated in Figure 4.4b. It then sets the kind of the IP packet to '*tunnelled_packet*'. This way, GW_B, upon receiving this packet [4], will understand that this is a tunneled packet, decapsulate the packet, and route the original IP packet to B_1 using the local instance of OLSR [5].

Chapter 5

Simulations and Results

The performance of the routing protocol deployed over the backbone network is evaluated in this chapter via experiments which were conducted on various network architectures. Data collected from these experiments are also discussed here. The experiments clearly demonstrate how the routing operation is carried out in all the scenarios presented. Each experiment has been designed to demonstrate the different aspects under which the protocol is likely to portray different behaviors.

The routing solution proposed in this thesis leverages the storage and retrieval mechanisms of Chord DHT as described in Section 3 to achieve a more efficient routing in the backbone network. The performance of this approach is then compared with a flooding-based approach which forms a baseline for the performance measure. The study is carried out on two MANET scenarios: one without mobility (stationary scenario) and the other with mobility included. For the stationary scenario, seven experimental setups are used to investigate the performance of the network. The configurations follow each other in an order of increased cluster number and reduced cluster size, keeping the total number of nodes constant. The metrics which are of interest are to capture both the efficiency and effectiveness of the protocols. More specifically, we measured the ping success rate (successful pings sent for which responses were received, as a percentage of total attempted pings), the ping Round Trip Time (RTT), and the network traffic (in packets) generated by the DHT protocol.

5.1 Static Scenarios

We define one static scenario for each of the network architectures considered in this study. This scenario is a simple one where all nodes are uniformly distributed in their different cluster areas, and remain members of the same cluster for the entire network lifetime.

5.1.1 Traffic Generated and Evaluation Metrics

Ping is a widely used tool for collecting metrics to measure the performance of a network [76, 20, 77]. A sender A generates and sends an Internet Control Message Protocol (ICMP) echo request packet, destined for an end system/host B. At the sending end, the sender starts a timer prior to sending the ping packet. B, upon receiving an ICMP echo request packet, simply reverses the ICMP header and sends an ICMP echo response or reply to A. At A, the timer is halted, and the elapsed time is recorded [76]. In the path from A to B, there are a few possibilities. The best case is that the packet traverses the path from source to destination and from the destination back to source successfully. Other cases that are possible include loss of a packet due to a congested network, no path exists from the source to destination or destination back to the source, or some firewall exists and drops ICMP packets in the end-to-end path from the source to the destination [76, 20]. If ping packets are successful, it is an indication that an end-to-end path existed between source and destination, the network protocol can deliver packets, the target host is connected to the network and is in a working condition to respond to the ping packet [76]. Furthermore, successful ping packets reveal a wealth of information which can then be used to measure network protocol performance. A stream of ping packets generated and injected into the network allows the tracking and calculation of the minimum, average, maximum and variance of the elapsed time between echo request sent and echo reply received [76, 20, 78]. A careful interpretation of the response times and their variance can provide an indication of the degree of traffic, or the load being experienced in the network [76]. When the network begins to handle more and more traffic, this will result in increased delay and increased variance, due to the interaction of the intermediate node buffers with the traffic flows along the path elements as load increases. When a relaying node's buffer overflows, it is forced to discard packets. Under such conditions, increased ping loss is observed. In addition to indications of network load, high delay and loss within a sequence of ping packets may be a sign of routing instability (routes are unavailable, etc.). The metrics used, and their definitions are as follows:

1. Ping Success Rate (PSR): The ratio of the ping echo responses received to those generated by the sources. This metric can only be measured after a sender receives a response to an ICMP echo request. Until the sender receives an ICMP response packet, it is hard to know the status of a transmitted ICMP echo request packet. If a response is not received, the fate of the packet is not exactly predictable.

2. Ping Round Trip Time (PRTT): This includes all possible delays caused by buffering during route discovery, queuing at the interface queue, retransmission delays at the Medium Access Control (MAC) layer, and propagation and forwarding times in the path between the query and the response. The ping application is implemented in such a way that it includes per host information about the minimum, maximum, mean, variance, and standard deviation of the round-trip time for the total number of ping responses received [79, 78, 77]. One advantage of measuring latency this way is that it reduces the need for clock synchronization: both timestamps are recorded on the same node, with reference to the same local clock.

3. Routing overhead: The average of the total OLSR traffic (HELLO, Topology Control (TC), Host and Network Association (HNA) messages) sent by nodes in the local clusters, and the average of the OLSR and DHT maintenance traffic sent by the gateway nodes in the backbone.

The first two metrics are the most important for best-effort traffic. The third is an interesting measure because it provides information about what fraction of the network resources is lost to network maintenance traffic and validates that hierarchical clustering in large-scale MANETs reduces routing overheads and increases performance. Considering the DHT maintenance traffic, we do not investigate occurrences like node churn which will generally bring about a lot of overhead depending on the churn frequency. The gateways basically retrieve information from the DHT once, and reuse the same for subsequent routing operations. Thus, the DHT maintenance traffic will largely comprises of ”keep-alive messages” which is not a significant issue. Furthermore, other DHT protocols like OneHopOverlay4Manets use a cross-layered approach, and no separate DHT maintenance is usually required.

In addition to the metrics discussed above, it is important to consider the networking context in which a protocol's performance is measured. The network size (which is a measure of the number of nodes in the network), or the network architecture, is an essential parameter that should be varied. In varying the network size, an approach could be to start out with an initial number of nodes and then increase the number of nodes linearly. In varying the network architecture, an approach could be to start out with a flat network of *N* nodes and build hierarchical architectures of clusters, increasing the number of clusters while keeping the number of nodes constant [80]. With either of these variations, the chosen metrics - PRTT, success rate, protocol efficiency, etc., are then measured for each network architecture or network size. In this thesis, the study will be carried out using the network architecture variation, where the total number of nodes is kept constant and different hierarchical architectures (clusters of varying number and sizes) are created [80].

5.1.2 Simulation Setup

A simulation study was carried out with 40 nodes comprising 30 hosts and 10 gateway nodes in a 4000M x 4000M network area. Initially, all 40 nodes participate in a flat network where all nodes are hosts who participate to support routing in the network and 30 of the 40 nodes exchange ping messages with each other. The 40 hosts are then divided into 2, 4, 5, 6, 8 and 10 clusters having 2, 4, 5, 6, 8 and 10 gateways. The clusters are MANETs and the gateways are part of a backbone MANET through which they inter-connect all the clusters. To make this possible,

Parameter	**Value**
Mobility Model	Stationary
Network area	4000 M * 4000 M
Network architecture	flat and hierarchical
Network structure	1, 2, 4, 5, 6, 8 & 10 clusters
Network protocol (cluster-wise)	OLSR
Network protocol (backbone)	OLSR/Chord, Flooding
Traffic Type	ICMP (Ping) traffic
No of nodes	40 (10 gateways, 30 hosts)
No of Ping sources	30 hosts
Local-Cluster Ping sources	All hosts in the flat network configuration and 30% of the hosts in the 2, 4, 5, 6, 8 & 10 cluster network configurations.
Ping rate	*N(0.3,0.01)*, $\sim$ 100 pings/sec
Number of simulations	10 runs per architecture
Randomness per simulation	Seeded
Node positions per cluster	Random
Simulation time	900 seconds
Time allowed for stabilization	210 seconds
Measurement time	690 seconds
Ping packet size	64, 128, 256 & 512 bytes
Interface queue management type	drop tail
Interface queue capacity	100 packets
Metrics under measure	Ping Round Trip Time, Ping Success Rate, Network traffic (in packets)
MAC protocol	IEEE 802.11g
Transmission range	1000 M, 800 M, 600 M, 500 M, & 300 M in the 1, 2, 4, 5, 6, 8 & 10 cluster configurations
Transmission rate	11 Mbps
Number of radios per node	1 per host, 2 per active gateway

Table 5.1: Simulation Parameters for the Static Scenarios

the gateways have a special design and are equipped with two radios with which they can communicate over two separate interfaces: the cluster they belong to, and the backbone network. The hosts, on the other hand, have a more basic design which allows them to have one radio for communication within the cluster they belong to. The resulting network is a two-tier hierarchical network structure, with the gateways serving to provide connectivity between members of their cluster and external networks. In the 2, 4, 5, 6, and 8 cluster configurations, 8, 6, 5, 4, and 2 gateways respectively, do not participate in the backbone network but exist to support routing in the clusters they belong to. Thus, they behave like hosts but do not send nor receive any ping messages. In the flat network configuration, which can also be called a one cluster configuration, all hosts are uniformly distributed in the network area and they assume this position throughout the entire simulation duration. In the other cluster configurations, each cluster is assigned a unique area in the network area and the member nodes are also uniformly distributed within their cluster space. Similarly, nodes in their individual clusters assume the same initial position for the entire simulation duration. Table 5.1 summarizes all the parameters used in the simulations.

5.1.3 Simulation Tool

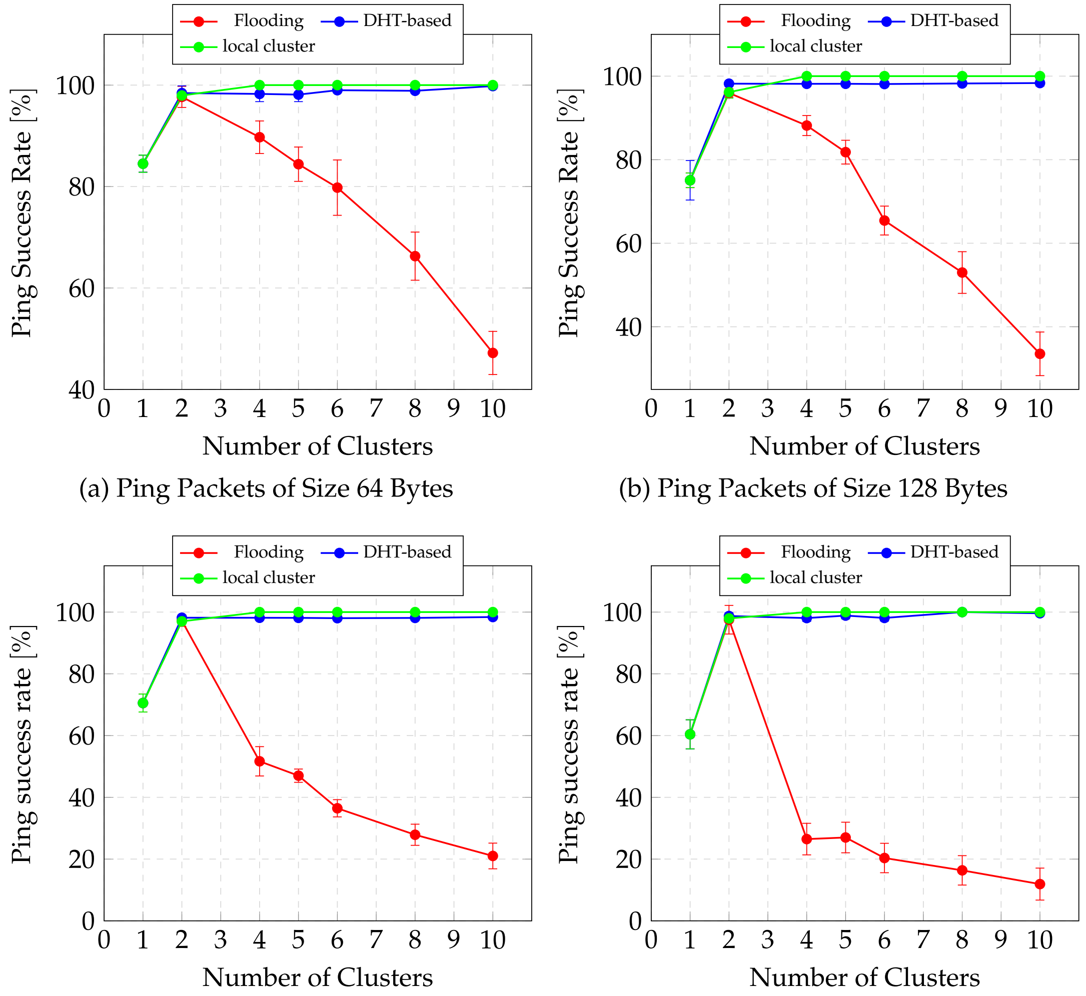
Omnet++ is the simulation environment we chose for our study as presented in Section 4.1. To evaluate the performance of the DHT-based unicast solution in comparison with the flooding solution, simulations were run for 900 simulated seconds. The first 210 seconds was allowed for the routing tables of each node to be populated and for the gateway nodes to build the overlay network and reach stability. We arrived at this figure after observing from several experiments, the approximate time it took for the network to attain stability. Measurements and event recording then set in for the remaining 690 seconds. Each of these network architectural setups were then simulated for ten simulation runs and with different randomly generated seeds. As a result, node locations and the distribution of ping messages vary in each run. The results reported here are the averages over the 10 repetitions.

5.1.4 Simulation Results and Analysis

In this section, the DHT-based unicast solution and the flooding-based solution are compared using the metrics discussed in Section 5.1.1, and for different cluster architectures. In this study, we are particularly interested in the performance of the network protocol deployed in the backbone. Results of the Ping Success Rate and the Ping Round Trip Time of the DHT-based unicast solution and the flooding solution are presented and analyzed for each of the cluster architectures, and the intra-cluster routing protocol performance is presented and analyzed as well. For the intra-cluster routing protocol, we focus more on the impact of the hierarchical clustering architectures on the routing overhead. Graphs for each metric and all architectural setups are then plotted and the margin of error displayed on all graphs represents 95% confidence intervals.

5.1.4.1 Ping Success Rate

The study was carried out on 4 different ping packet sizes and in increasing order: 64 bytes, 128 bytes, 256 bytes, and 512 bytes. In Figures 5.1a, 5.1b, 5.1c and 5.1d the red line represents the Ping Success Rate and in Figures 5.3a, 5.3b, 5.3c and 5.3d, the orange line represents the total packets sent at the MAC layer of each gateway, considering the flooding solution. The graphs in the figures mentioned above all have similar trends, as the number of clusters increases for each ping packet size. However, as the ping packet size increases, the ping success rates deteriorate more and more while the total number of MAC layer packets also deteriorates but not as much as the former. The analysis of the two graphical representations show that



(a) Ping Packets of Size 64 Bytes

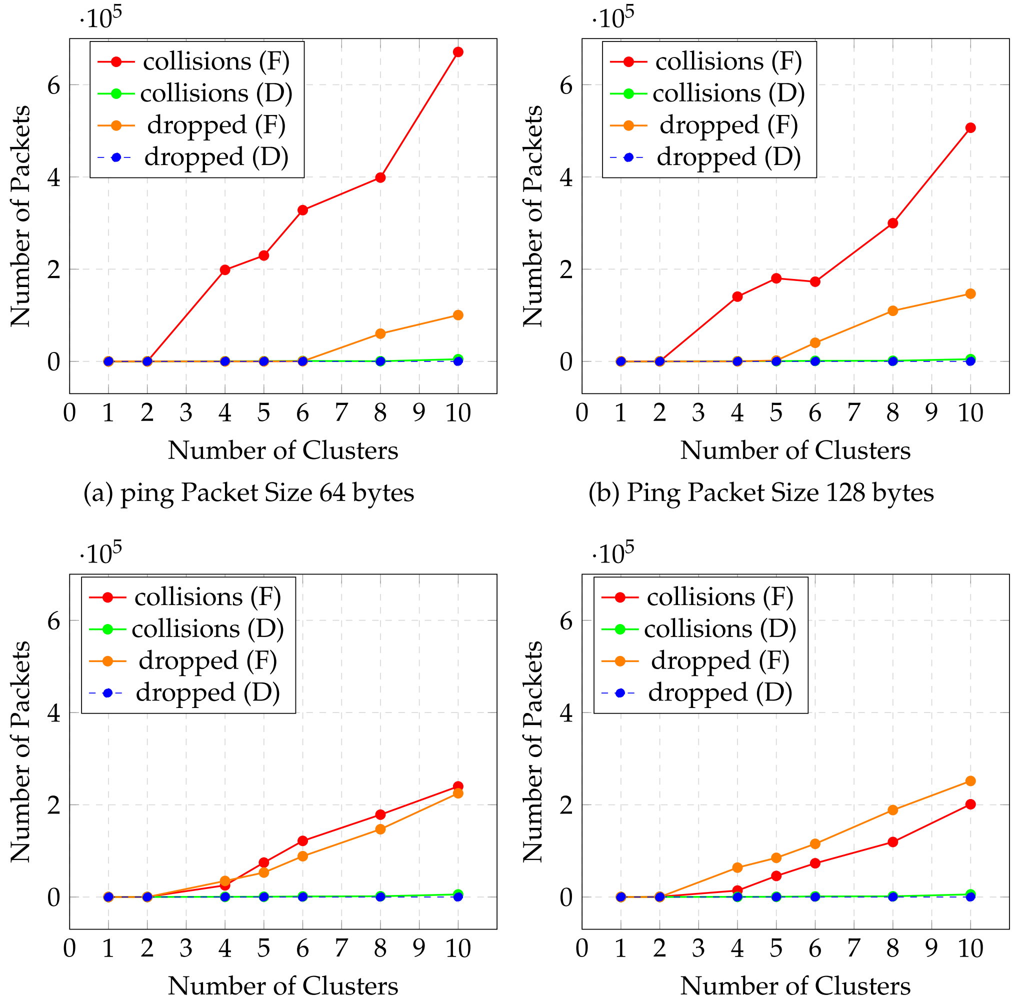
(b) Ping Packets of Size 128 Bytes

(c) Ping Packets of Size 256 Bytes

(d) Ping Packets of Size 512 Bytes

Figure 5.1: Ping Success Rate for the Static Scenarios

the number of successful pings for which responses were received, deteriorates as we increase the number of clusters and deteriorates even more as we increase the ping packet size. In the flat network configuration, all the nodes belong to the same cluster and ping each other while all the gateways behave like hosts to support routing in the network. Thus, there is no backbone traffic in this architecture. As we increase the number of clusters to 2, 4, 5, 6, 8 and 10 respectively, the backbone handles more and more traffic because of the redundant rebroadcasts of each packet



(a) ping Packet Size 64 bytes

(b) Ping Packet Size 128 bytes

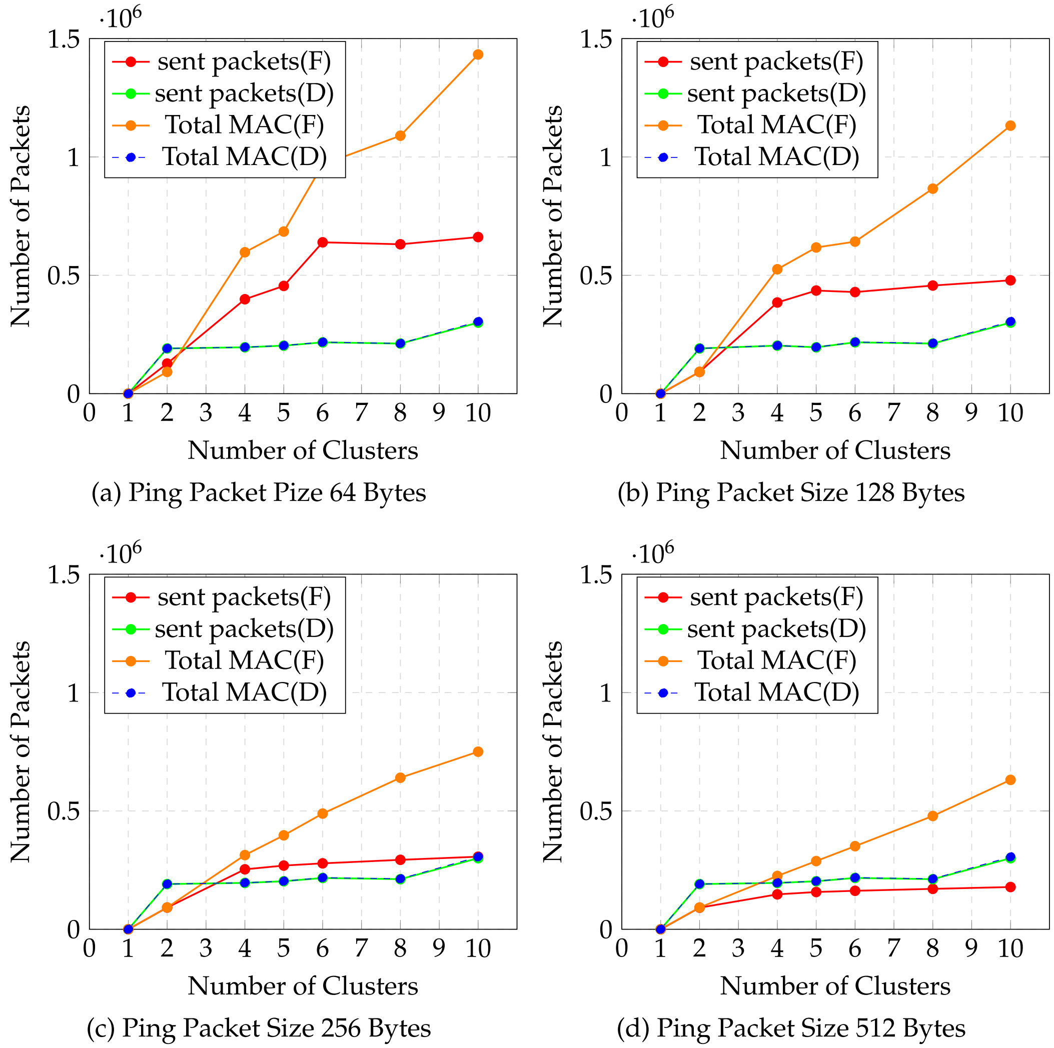
(c) Ping Packet Size 256 bytes

(d) Ping Packet Size 512 bytes

Figure 5.2: Collisions and Dropped Packets at the MAC Layer for the Static Scenarios

(F) = Flooding approach	**(D) = DHT-based approach**

broadcast in the backbone. This happens because, an architecture of *'N'* clusters will have *'N'* gateway nodes, and for each packet broadcast in the backbone, *'N-1'* copies of the same packet will be generated because each gateway will rebroadcast the same packet once. As each node in the network generates ping traffic at a rate which is uniformly distributed, there will be steady broadcasts in the backbone by



(a) Ping Packet Pize 64 Bytes

(b) Ping Packet Size 128 Bytes

(c) Ping Packet Size 256 Bytes

(d) Ping Packet Size 512 Bytes

Figure 5.3: Total and Sent Packets at the MAC Layer for the Static Scenarios

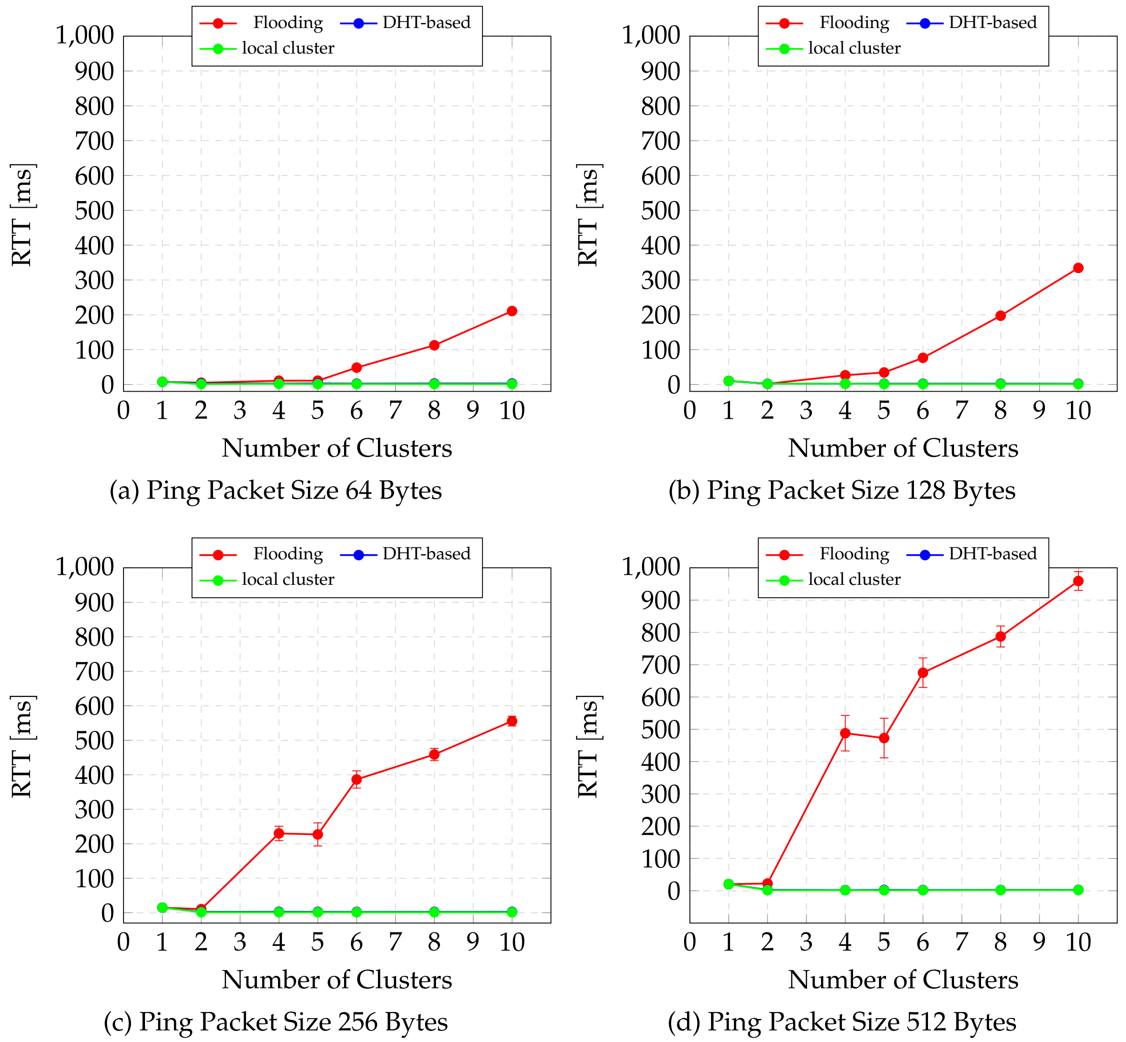
each gateway node and the remaining gateways who rebroadcast these packets. This makes packet collision in the backbone inevitable. Broadcast packets are treated differently than unicast packets in the MAC layer. Under the IEEE MAC 802.11 Distributed Coordination Function (DCF) [81], broadcast transmissions are implemented so that the broadcast packets are sent to the neighbors as soon as the radio channel is sensed to be free (carrier sensing). However, no collision detection is used to guarantee successful receipt of the packets at the destination nodes. Every gateway ought to listen for the channel status for a DCF interframe space (DIFS) interval. If the channel happens to be busy during this interval, the gateway defers its transmission. In flooding, multiple gateways contend for the wireless medium. As such, when they sense the channel busy and defer their access, they will also virtually simultaneously find that the channel is released and then try to seize the channel. DCF specifies a random backoff, which forces a gateway to defer its access to the channel for an extra period [81]. However, with multiple gateways transmitting over the same medium and at the same time for the most part, there is a high probability that multiple gateways will select the same minimal random backoff and attempt transmission at the same time. As a result, collisions occur and every gateway hears this collision. This effect does not occur only once but multiple times in the simulation duration, resulting in frequent collisions and high packet losses. The red line in Figures 5.2a, 5.2b, 5.2c and 5.2d show the rate of collision as the number of clusters increases for each ping packet size and across the different ping packet sizes. Although the line trend is the same for the different ping packet sizes, the number of collisions per cluster architecture decreases as we increase the ping packet size. An explanation for this is that it takes more time to transmit larger ping packets and so packets which arrive during a transmission will be queued. With the rate of packet arrival superseding the rate of packet transmission, the queue will get full faster, and packets who arrive to meet a full queue will be dropped. More and more packets are dropped as the ping packet size increases and as a result, only packets who make it into the queue will have a transmission attempt. A fraction of these packets is again lost to collisions. The major challenge with the dropped packets and the collisions is that these packets are lost entirely and cannot be re-transmitted. This explains the drop in the number of collisions per cluster architecture as the ping packet size increases. Overall, the sum of dropped packets and the collisions at the MAC layer of each gateway provides an indication of the total number of lost packets. The difference between the total lost packets and the total packets at the MAC layer of each gateway node gives the number of packets which were successfully transmitted. Of the successfully transmitted packets, there is no guarantee that all these packets make it to the destination host or that the source host received a response from the destination host. The reason is that there might be other factors in the path from the source to the destination and back to the source - for example, route unavailability at intermediate relaying nodes or at the destination host that could result in packet loss in the path. Overall, as the number of clusters increased for each ping packet size and across the different ping packet sizes, the frequent rebroadcasts generated more and more overhead which utilized most of the backbone channel resources and as a result, fewer ping packet exchanges were successful.

In Figures 5.1a, 5.1b, 5.1c and 5.1d, the blue line represents the ping success rate for the DHT-based unicast solution and the dashed blue line in Figures 5.3a, 5.3b, 5.3c and 5.3d represent the total number of packets sent at the MAC layer of each gateway node as the number of clusters are increased for each ping packet size and across the different ping packet sizes. The graphical representations show that the protocol adopted in the DHT-based unicast solution outperformed the flooding protocol in terms of efficiency. This is true because the way packets were delivered to their destination via the backbone eliminated the ripple effect caused by the redundant rebroadcasting of packets in the backbone. Here, a gateway node 'A' retrieves routing information from the DHT as per the design discussed in

Chapter 3, which includes the IP address of the gateway node 'B' responsible for the cluster where the destination host is located. Using the routing table maintained by the backbone routing protocol OLSR, A then forwards the packet to 'B' who then delivers the packet to the destination host in its cluster using the cluster-wise routing protocol. In this case, when packets arrive at the gateway and must be queued, most of the queue is occupied by data packets which wait for their transmission. Although the gateways handle other packets like the OLSR control traffic and the DHT maintenance traffic, the OLSR control traffic here are the HELLO messages only, which are transmitted periodically every 2 seconds. The least of the DHT maintenance traffic has a periodic frequency of 5 seconds. "HELLO messages" are the only OLSR traffic in the backbone because ours is a fully connected backbone and there are no MPRs to send Topology Control (TC) messages as all gateways are within reach of each other. There are also no HNA messages because the gateways do not announce reachability to the backbone members. As the periodic frequency of these packets is far lower than that of the data packets, most of the queue capacity is occupied by the ping packets. This goes further to explain the low rate of dropped packets shown by the dashed blue line in Figures 5.2a, 5.2b, 5.2c and 5.2d respectively. A few packet drops in the order of tens were experienced as the ping packet sizes increased, for the same reason discussed in the flooding solution, and there were a few collisions as well. Overall, the protocol efficiently delivered over 95% of the packets to their destinations which remained constant for all architectures and ping packet sizes. It is also worth pointing out that with our proposed solution, the ping performance is the same whether pings are within a cluster or across the global network. This is clearly not the case for the flooding-based backbone.

5.1.4.2 Ping Round Trip Time

As mentioned in Section 5.1.1, the ping round trip time is the interval between the ping echo request and the response. Statistics collected from this metric can also reveal the load experienced by the network. The study was carried out on 4 different ping packet sizes as was with the previously discussed results, and for the same architectures.



(a) Ping Packet Size 64 Bytes

(b) Ping Packet Size 128 Bytes

(c) Ping Packet Size 256 Bytes

(d) Ping Packet Size 512 Bytes

Figure 5.4: Ping Round Trip Time for the Static Scenarios

The red line in Figures 5.4a, 5.4b, 5.4c and 5.4d represents the ping round trip time for the flooding solution and the blue line represents the ping round trip time for the DHT-based solution. Comparing both results, both solutions have similar values for the one- and two-cluster architectures but begin to vary significantly beginning with the four-cluster architecture. The one-cluster architecture is the baseline for the remaining architectures and is expected to produce similar results. The two-cluster architecture involves only two gateways, so the flooding solution will behave like a unicast solution since every "broadcast" packet in the backbone will be directed to just one gateway node who happens to be the gateway to the destination host. In this case, there will be no redundant broadcasts, and this is true for all ping packet sizes. However, as we increase the number of clusters to four and above, the round-trip time in the flooding solution begins to increase, and it gets worse as we increase the ping packet size. One factor that has a major impact on this is the flooding of packets in the backbone which increased more and more, as more gateways were added to the backbone network. As explained in Section 5.1.4.1, as more gateways are added to the backbone network, the gateway nodes will have to queue more redundant packets than actual ping packets, especially when the rate of packet arrival exceeds the rate of packet transmission. Since packets are handled in a FIFO fashion, cases where the front of the queue is occupied by redundant packets will be dominant, and it will take a longer time to transmit a ping packet which most of the time will be at the back of the queue. This time equals the sum of the total wait time in the queue and the total transmission time. Since there is a response for each ping packet, the same sequence of events will replay in the response path, which increases the ping round-trip time even more. As the ping packet size increases, the round-trip time increases even more. As the bandwidth available to each gateway is a fixed value, the gateways now take a longer time to transmit larger ping packets, which is an added delay to the already discussed delay. This accounts for the overall increase in the ping round trip times for all the cluster architectures and across all ping packet sizes.

The DHT-based solution, however, does not face the same ordeal. In the same figures, the blue line clearly shows that with the DHT-based solution, the ping packets typically have lower round-trip times and remain somewhat constant across all cluster architectures. For every packet a gateway GW_X is routing for the first time to an external destination, GW_X queries the DHT to obtain the routing information required to route that packet. In this period, the first few packets for which routing information is not available are queued and are forwarded once the gateway successfully retrieves routing information from the DHT. Although there is some delay incurred by this route discovery process, it is in the order of milliseconds, and since the nodes remain at the same position for the entire simulation duration, the routing information retrieved is cached and reused for successive packet forwarding in the same direction. In the backbone, which is of more concern as it can become a bottleneck with heavy network loads, the gateway nodes are within reach of each other, and so packets will typically be delivered in one hop. As discussed in Section 5.1.4.1, the DHT solution introduces new routing overheads, which are the DHT maintenance messages and the OLSR control messages, but these control messages have fixed intervals of 5 seconds and 2 seconds respectively. This is much less frequent than the ping frequency, which has an average interval of 0.3 seconds. The HELLO messages are flooded in a multicast fashion and are never forwarded, while the DHT control messages are unicast packets. The control overhead increases as the number of gateway nodes increases but does not congest the backbone to the same extent as the flooding solution. When the ping packet size increased, the gateways took longer times to transmit packets and packets arriving during a transmission were queued. However, queued packets did not spend as much time in the queue as was the case for flooding, and packets were not queued most of the time. Overall, the ping round trip time in the DHT-based solution remained at about 2.5 ms for all the cluster architectures and across all the ping packet sizes. Again the performance of inter-cluster pings, using the DHT-based solution, is very comparable to the performance of local (intra-cluster) pings, which clearly is not the case for a flooding-based backbone.

5.1.4.3 Routing Overhead

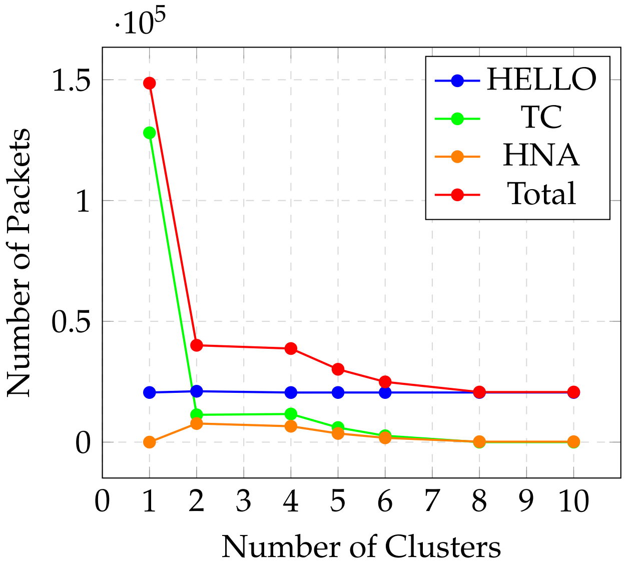


Figure 5.5: OLSR HELLO, TC and HNA Messages

The routing overhead as described in Section 5.1.1 counts the average number of maintenance messages the protocol incurs in carrying out topology maintenance and routing. As it is known that hierarchical network architectures are widely deployed to reduce routing overheads and increase scalability in ultimately large-scale MANETs [16, 7], we are interested in this metric because it gives an insight into the extent to which clustering reduces the overall routing overheads when compared to a flat MANET architecture. In Figure 5.5, the blue line represents the total number of HELLO messages, the green line represents the total number of TC messages, the orange line represents the total number of HNA messages and the red line is the sum of all the above-mentioned messages. In the flat network, there are no active gateways and so all nodes exchange HELLO messages every 2 seconds and MPRs propagate TC messages every 5 seconds. MPRs are an optimization strategy in OLSR to reduce the number of nodes who flood topology control information in the network. The topology control messages are rebroadcast by other MPRs once to ensure that all other nodes receive updated topology information. In the flat network, there are many MPRs because nodes are not within reach of each other. This accounts for the high number of TC messages flooded in the MANET as seen in Figure 5.5. As the number of clusters increase, the nodes are limited to smaller areas and more nodes are within reach of each other. The number of MPRs, therefore, begins to reduce more and more and completely vanishes in the ten-cluster architecture. The number of TC messages was about 130,000 messages in the one cluster architecture but reduced significantly to about 20,000 messages in the 2 cluster architecture and continued to reduce as the number of clusters increased. As the number of clusters increased, a new routing overhead was introduced, “HNA message”, which gateways use to inform their local cluster of their reachability to external clusters. The HNA messages are propagated every 5 seconds and forwarded only via the MPRs in the network. This slightly increases the number of control messages but with fewer nodes per cluster as the number of clusters increases, the total number of HNA messages reduced similarly to the number of TC messages. The HELLO message interval does not change at all through the simulation time because every node sends these messages every 2 seconds and these messages are never forwarded. Since the number of nodes remains constant for all scenarios, the number of HELLO messages remains unchanged. Overall, with the total number of HELLO messages remaining constant, the total number of TC and HNA messages reduced across the cluster architectures, the total OLSR control messages reduced significantly between the one and two cluster architectures; from over 150,000 in the one cluster architecture to about 40,000 in the two cluster architecture and reduced more for the remaining cluster architectures to about 38000, 35000, 30000, 25000 and 25000 respectively. As a result, for the intra-cluster statistics, the ping packet delivery ratio increased from about 80% in the one-cluster architecture to about 95% in the two-cluster architecture and 100% for the 4, 5, 6, 8 and 10 cluster architectures respectively. Similarly, the ping round trip time reduced from about 14 ms to about 1.5 ms which remained constant for the remaining cluster architectures. This was because with more clusters, there were fewer hosts per cluster which were now closer in proximity and only one hop away from each other in most cases. This clearly shows that large-scale MANETs benefit from hierarchical architectures in the sense that overall routing overhead is reduced and routing performance increases.

5.2 Mobility Scenarios

To conduct meaningful mobility-wise performance analysis of the routing algorithms deployed in MANETs, it is essential that the mobility models reflect realistic mobility behavior. Popular mobility models for MANETs are the Random Way-Point (RWP) Mobility model and Reference Point Group Mobility (RPGM) model [82, 83, 84]. In this thesis, we are particularly interested in nodes moving among themselves in their local clusters as well as whole clusters moving in the network. For our study, mobility models like RWP will not be sufficient to model the movement of people among themselves in their groups/teams as well as groups/teams moving as a whole [82]. In the RWP model, the nodes move along a zigzag path consisting of straight legs from one waypoint to the next [84, 82]. The model is insufficient to capture some realistic scenarios like group mobility for example. A better model, which more accurately captures the actual movement of people in a group and provides a better insight into the order of node movement expected from MANET scenarios, is the RPGM [82, 83]. In RPGM, each group has a 'logical center' whose movement defines the behavior of the whole cluster/MANET, including location, speed, direction, acceleration, etc. Therefore, the trajectory of the group is determined by providing a path to the center. In general, the nodes are evenly distributed in the geographic scope of a group. Each one is assigned a reference point that follows the movement of the group and nodes are placed at random near the reference point according to a specified maximum distance from the reference point. The reference point scheme allows independent random movement behavior for each node, in addition to the group movement [85, 83].

We study three mobility scenarios under which we evaluate the performance of our design. These scenarios are discussed as follows:

Intra-Cluster Mobility: First, we investigate the effect of mobility when all nodes are mobile within their clusters alone. This mobility scenario is a variation of what happens in the static scenarios, the difference being that the nodes now move within their cluster area. It then forms the baseline for the other mobility scenarios which we study.

Nodes Switching Clusters: In this mobility scenario, some selected hosts switch their cluster membership. Here, we have some hosts configured to start and stop moving at certain times, in a certain direction - a distance of its current position to the location of the intended cluster, to join that cluster. Since all clusters communicate with the same channel and interface locally, a host will easily join a different cluster by exchanging HELLO messages with the nodes in the cluster it joined.

Whole Clusters Merging and Splitting: Just as with the nodes, whole clusters will also be mobile. The clusters will generally move in relation to the group mobility model we use and the nodes in the clusters will move in relation with their individual group centers, here the gateways [83]. Since all clusters communicate locally over the same communication channel, when two cluster come into proximity with each other, they overlap physically and logically to form one cluster.

Parameter	Value
Mobility Model	Reference Point Group Mobility (RPGM)
Network structure	4 clusters
No. of nodes per cluster	10 nodes (4 active gateways, 30 hosts, and 6 inactive gateways)
Max. distance to reference node	500 m
Minimum speed	5 m/s
Maximum speed	20 m/s
Packet Size	64 bytes
Traffic generated	ICMP (ping)
Number of ping sources	30 hosts
Simulation time	3600 Seconds
Network Stabilization time	210 Seconds
Measurement time	3390 Seconds

Table 5.2: Simulation Parameters for the Mobility Scenarios

5.2.1 Traffic Generated and Evaluation Metrics

The traffic and the evaluation metrics used in the mobility scenarios are the same presented in Section 5.1.1. Table 5.2 presents the parameters specific to the mobility scenarios while the rest of the parameters are summarized in Table 5.1.

We collected performance statistics for the same metrics we measured in the static scenarios. Just like the static scenarios, the DHT maintenance traffic does not include the traffic introduced by node churn as we did not consider this effect. By introducing mobility, the gateways retrieve information from the DHT periodically in order to obtain the most resent route advertisements in the DHT. Thus, the DHT maintenance traffic will largely comprises of the frequent look-up messages and the DHT ”keep-alive” messages. This traffic will be obviously higher than the traffic in the static scenarios but not as much as to bring about any significant issue. This traffic is captured in the total number of packets at the MAC layer as seen in Figure 5.7. Again, DHT protocols like OneHopOverlay4Manets, use a cross-layered approach and no separate DHT maintenance is usually required.

For the mobility model where hosts switch cluster membership, we focus more on collecting performance statistics for the mobile hosts $X_0 ... X_X$ who switched their cluster membership and the hosts $Y_0 ... Y_Y$ who pinged hosts $X_0 ... X_X$ for the entire network lifetime. This way, we are able to determine the reachability of hosts $X_0 ... X_X$ throughout the network lifetime. Successful pings will reveal the following:

1. A host X_X who was initially a member of cluster X was reachable by a host Y_Y while it switched between clusters X, Y, Z.

2. Gateways GW_X, GW_Y, GW_Z discovered the arrival and departure of hosts $X_0 ... X_X$, and the DHT was adequately updated.

3. Gateways GW_X, GW_Y, GW_Z were able to route $X_0...X_X$'s packets to destination hosts $Y_0...Y_Y$ in clusters X, Y, Z for the period when hosts $X_0...X_X$ resided in their cluster.

5.2.2 Simulation Setup

To carry out our study based on the mobility scenarios described above, we use the four-cluster configuration. It comprises of 40 nodes in total: 4 participating gateways, 6 non-participating gateways which serve to support routing in the various cluster where they belong, and 30 hosts, each belonging to one of the four clusters. The four-cluster configuration is selected because, with this setup, it will be feasible to investigate the possibility of two clusters merging while the other clusters remain the same, or three clusters merging while the other remains the same, and when two or three clusters who merged, now split. It will also be feasible to instantiate the possibility of more than one host switching between clusters. For all the mobility scenarios, we used a ping packet size of 64 bytes while varying the average mobility speeds in increasing order from 0 to 20 ms^{-1} at intervals of 5. The mobility speed for each node in each of the variations is a random variable *N(m, d)* with a mean 'm' of 0, 5, 10, 15, 20 and standard deviation 'd' of 0, 2, 3, 4 and 5 respectively. We selected the 64-byte ping packet size for our analysis because, as we saw in the static scenarios, the trend was the same for all ping packet sizes as they all presented the same qualitative result. Therefore, it did not really matter as to which ping packet size we used.

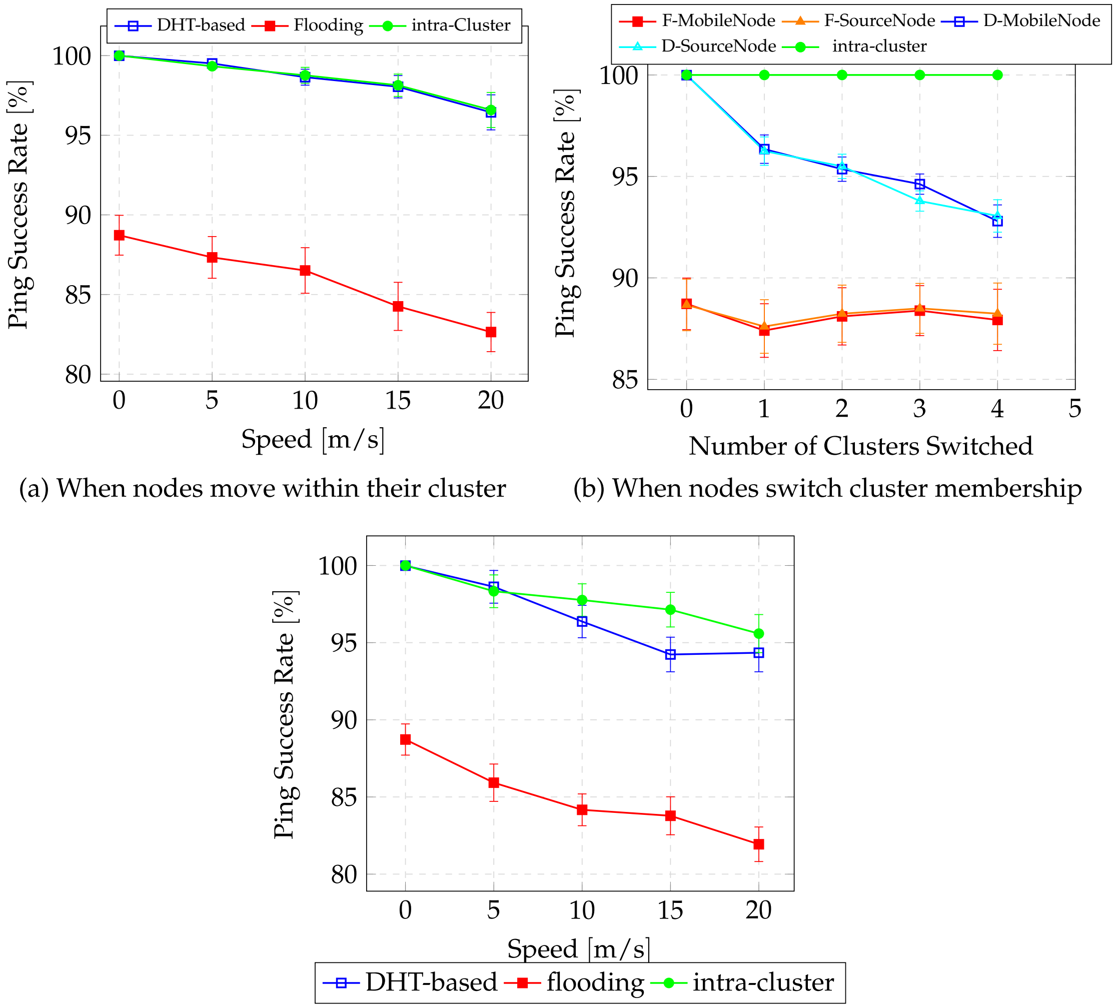
With the help of the Bonn Motion tool, which is a mobility scenario generation and analysis tool [86], we generated mobility traces based on RPGM which we used directly in *Omnet++* for our simulations. This catered for the scenarios where nodes were mobile within their individual clusters as well as for scenarios where clusters are moving among themselves. The mobility traces for hosts switching cluster membership were handcrafted. We allowed 210 seconds for network stabilization as was the case in the static scenarios, and the statistics were collected over the remaining 3390 seconds.

5.2.3 Simulation Results and Analysis

In this section, the flooding and DHT-based routing approaches are compared using the metrics discussed in Section 5.1.1, and for the different mobility scenarios, starting with the intra-cluster mobility to nodes switching clusters and finally clusters merging/splitting. The intra-cluster routing protocol performance is presented and analyzed as well. As with the static scenarios, we are particularly interested in the performance of the network protocol deployed in the backbone network. For the intra-cluster routing case, we measure the impact of mobility on the metrics measured, in comparison with the static scenarios discussed in Section 5, and we use this as a base case for the other mobility scenarios. Graphs for each metric and all mobility scenarios are then plotted and the margin of error displayed on all graphs represents 95% confidence intervals.

5.2.3.1 Ping Success Rate

In Figure 5.6 and Figure 5.7, the following abbreviations : $F-$ and $D-$, that appear in the legends, are used to denote the Flooding approach and the DHT-based approach respectively. Figure 5.6 shows the Ping Success Rates when nodes were mobile within their clusters, hosts switched cluster membership and clusters merged/split. The circles represent the intra-cluster ping successes and the squares in Figures



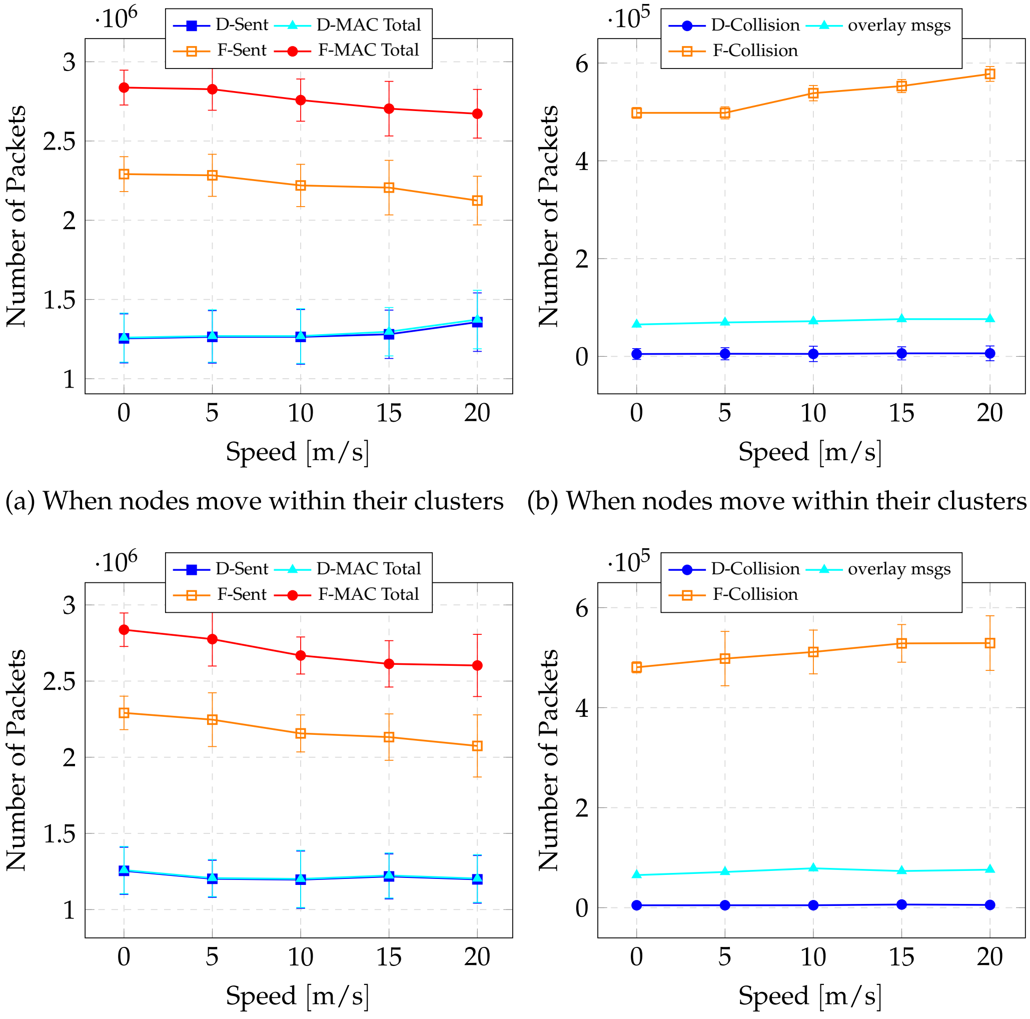
(a) When nodes move within their cluster

(b) When nodes switch cluster membership

(c) When clusters merge and split

Figure 5.6: Ping Success Rates for the Mobility Scenarios

5.6a and 5.6c represent the inter-cluster ping successes. In Figure 5.6b, the squares represent the ping success rate when the mobile host was the ping destination and the triangles represent the ping success rates when the mobile host was the ping source. In Figure 5.6b the number of clusters which each host switched to in the entire network lifetime, is varied from 0 to 4 in intervals of 1. Five (5) nodes switched their cluster membership but at different times from each other with no



(a) When nodes move within their clusters (b) When nodes move within their clusters

(c) When clusters merge and split (d) When clusters merge and split

Figure 5.7: MAC Statistics for the Mobility Scenarios

particular order or interval. However, each node spent about 600 seconds in the first four clusters and 990 seconds in the fifth cluster. In order to determine that a host was reachable whenever it switched between clusters, the time a node X_X spent in cluster Y had to be large enough to both exceed the delay until a gateway GW_Y advertises reachability to X_X in the DHT, and allow for statistics to be collected to proof that it is reachable. For both protocols, it took about 10 seconds for a node to transit from cluster X_X to cluster Y_Y, a total of 40 seconds. In the DHT-based solution, it took an average of about 30 seconds more until a source gateway GW_X received a local update for a destination host X_X as described in Section 3.3.3. In both protocols, Ping packets were lost for the 10 seconds when a host was in transit between clusters and the delay until the intra-domain routing protocol updated the gateways routing table. In the DHT-based solution, ping packets were lost for an extra $\approx$ 30 seconds more. At 0 ms^{-1}, the network is a static network and the results are the same as discussed in the static scenarios.

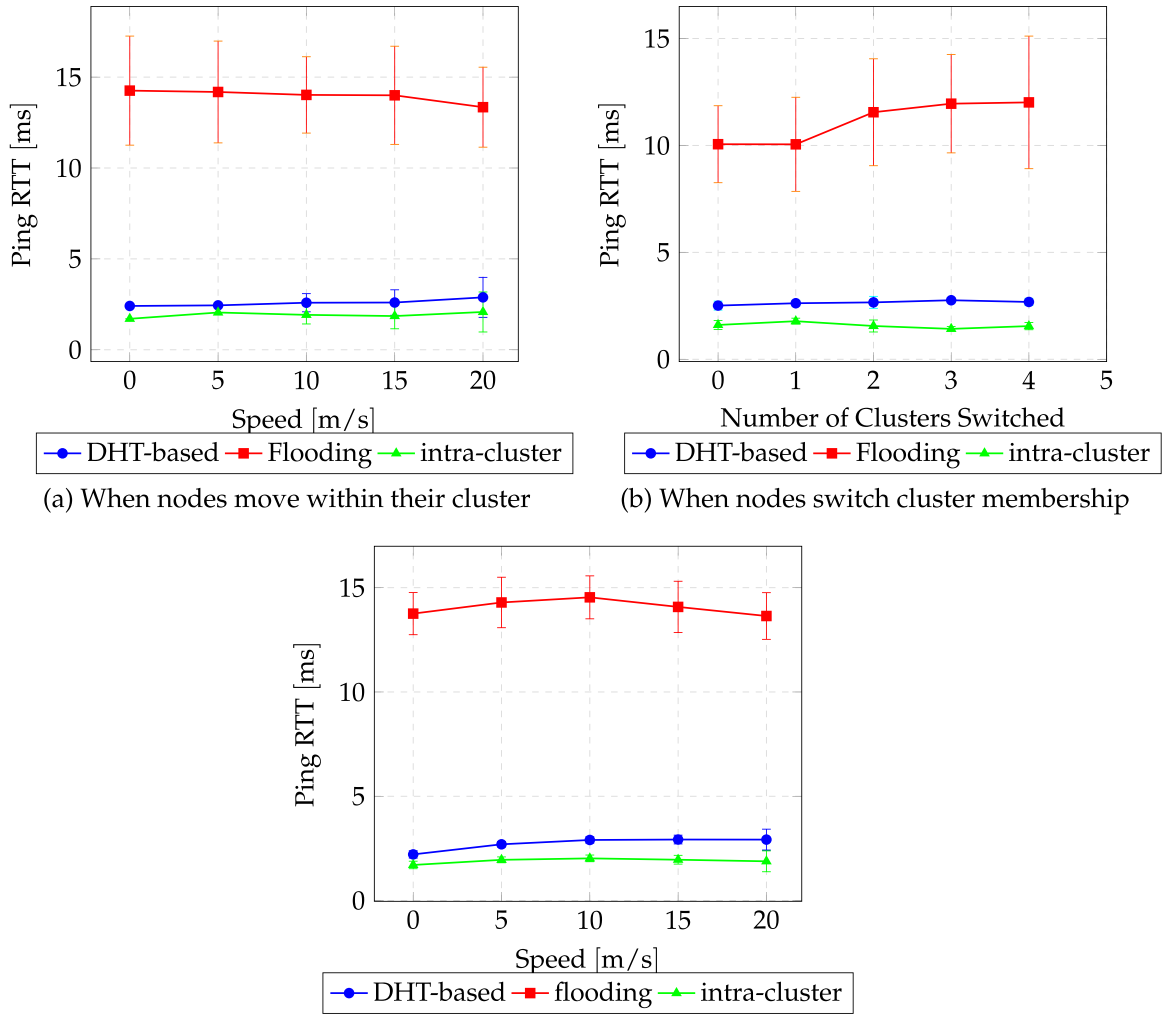
For both approaches, the presence of mobility brought about frequent link breakages which resulted in packet losses and as a result, declining ping success rates. In the DHT-based approach, when hosts began to switch between clusters and when clusters merged/split, success rates suffered because of the delay until the gateways discovered the new hosts they could now reach before advertising their reachability to these new hosts in the DHT. Similarly, the previous gateways needed to discover the hosts they could no longer reach and hence, update the DHT accordingly. As a result, the ping packets sent within this discovery/update period were unsuccessful. However, the time taken for a gateway GW_X to discover that a host X_X left or arrived in its cluster, is unrelated to our design. This delay is a function of the intra-domain routing protocol, which maintains individual node routing tables. When hosts began to switch between multiple clusters, the same set of events replayed for each switch, which contributed to the declining ping success rates. Figure 5.6b also shows that the success rate when the mobile node was the ping source and when it was the ping destination, are similar. This was because the average delay until a host X_X's current gateway GW_X advertises reachability to X_X in the DHT, and the average delay until the source gateway GW_Y retrieves the latest route advertisement for host X_X from the DHT, converge.

In the Flooding solution, in addition to the link breaks, which became more frequent as mobility increases, the performance deteriorated for the intra-clusters mobility model and when clusters merged/split because of the frequent collisions in the backbone as seen in Figure 5.7b and 5.7d. As a result, fewer packet transmissions were successful. However, when hosts switched between clusters, the results obtained were similar to nodes who stayed in their cluster for the entire simulation duration. This was true because only a few ping packets were lost in the delay until the intra-domain routing protocol updated the routing tables of the new destination gateways. Thus, the new gateways just pick up packets destined for the new node from the backbone after it learns about this node.

We can, therefore, conclude that our design supports routing efficiently under different forms of mobility and outperforms the flooding approach. The only case where we experienced decreasing ping success rates, from about 100% to about 92%, was the case where hosts switched between four different clusters and could only be discovered after a delay which was partly related to our design and the intra-domain routing protocol. However, this is not entirely different from what happens in other routing protocols that support mobility. When nodes move, they have to be rediscovered in order to be reached. Thus, messages exchanged within this rediscovery phase will be lost.

5.2.3.2 Ping Round Trip Time

Figure 5.8 shows the Ping Round Trip Time for all the mobility scenarios we study. The triangles represent the intra-cluster RTT for both the flooding and the DHT-Based approaches. The outcome shows that pings delivered within the clusters



(a) When nodes move within their cluster

(b) When nodes switch cluster membership

(c) When clusters merge and split

Figure 5.8: Ping Round Trip Time for the Mobility Scenarios

experienced low RTT because the nodes were within reach of each other most of the time and did not require the intervention of their gateways, except in cases where the gateways were MPR's or an intermediate relaying node. The squares and the circles respectively represent the RTT for the flooding and the DHT-based approaches when pings were directed externally. Just as seen in the static scenarios,

(F) = Flooding approach	**(D) = DHT-based approach**

in general, the RTTs in the flooding approach were higher than in the DHT-based approach. This was because the medium was most of the time used to re-broadcast redundant packets, which introduced queuing delays at the gateways. In the DHT-based solution, packets were only queued for the initial route discovery phase of our protocol and packets were delivered with a unicast protocol in single hops through the backbone for most cases. When hosts were mobile within their clusters and when clusters merged/split as seen in Figures 5.8a and 5.8c, the round trip times were similar for all the mobility speeds and the confidence intervals show that the results overlap. In both routing approaches, this was true because measurements were only recorded for successful pings. Ping packets were dropped as a result of route unavailability and some others were temporarily routed to wrong destinations, particularly in the case where clusters split. Hence, there was no queuing of packets in any of these cases which would normally impact the RTT. This was also true for the model where hosts switched between different clusters. Here, although the data points were not as close as they were in the other two scenarios, the confidence intervals overlapped and hence do not exhibit any statistically significant difference. Overall, the RTT in the DHT-based approach was very comparable to the intra-cluster RTT and was consistently and significantly lower than that of the flooding approach.

5.2.3.3 Routing Overhead

Finally, we measure the routing overhead incurred under mobility by the different protocols we deployed in the backbone. Figure 5.7 shows the MAC statistics we collected from both routing approaches. In the event that more and more nodes or whole networks join, the backbone will handle more and more traffic (user data packets and routing packets) and might become a performance bottleneck [19]. This metric will, therefore, reveal the extent of load experienced at the MAC layer of each gateway as a function of number of packets successfully transmitted and the total number of packets handled by the MAC layer. The total number of packets handled by the MAC layer is the sum of the number of successfully transmitted packets, the number of packets dropped by the queue and the number of collisions that occurred at the MAC layer. Overall, for both routing approaches, the statistics collected in the intra-cluster mobility model and when clusters merged/split, are closely comparable to each other. In the flooding approach, in addition to frequent link unavailability, the total number of collisions also increased as the mobility speed increased, which contributed to the decrease in the number of packets successfully transmitted by the MAC layer. In the flooding approach, the number of transmitted packets is represented by the no-fill squares in Figures 5.7a and 5.7c. The difference between the total number of packets handled by the MAC layer and the total number of successfully transmitted packets is the number of collisions which occurred at the MAC layer and is represented by the no-fill squares in Figures 5.7b and 5.7d. In the DHT-based approach, however, the MAC layer successfully transmitted almost all the packets it handled, and this is represented by the solid squares and rectangles in Figures 5.7a and 5.7c, which appear to be closely comparable. The circles in Figures 5.7b and 5.7d also show that the number of collisions at the MAC layer was almost zero. In the DHT-based approach, the only significant difference between the static and the mobility scenarios, however, is that the number of overlay maintenance packets increased as a result of the constant updates between the gateways and the DHT to support mobility. It is normal for these updates to occur and is not different from what happens in other routing protocols that support mobility.

In conclusion, although the total number of packets handled at the MAC layer increased in the DHT-based approach with cluster mobility, it is still a lot lower than the traffic we recorded in the flooding based approach. For all scenarios and mobility speeds, the backone traffic in the backbone exceeds the backbone traffic in the DHT-Based solution by a factor of between two and three.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, our goal is to provide a scalable and efficient routing solution in the backbone of hierarchical MANETs. As different forms and/or levels of mobility are expected in such networks, and considering that such networks have the tendency of growing both in the number of participants and/or covering large areas, a lot of concerns and constraints like link availability, scalability, resilience, among all others, ensue. While others proposed modifications to existing MANET routing protocols like Hierarchical OLSR (HOLSR), optimized AODV, hybrid protocols, etc. in order to support routing in larger MANETs, much focus was given to one or a few of the challenges which ensue in such networks. More so, some others only carried out research in the context of specific network scenarios, overlooking some other key issues that ensue in more general network scenarios. These approaches do not provide solutions that effectively handle the general context that we are interested in, ′providing a scalable routing solution for ultimately large scale MANETs which will support a variety of applications.'

In Chapter 3 we presented our solution, discussed in detail the context in which we are interested in providing a routing solution for MANETs, and explored various network conditions which our implementations would be subjected to. This includes different network architectures and mobility scenarios we aim at exploring as they support a variety of applications deployable in MANETs. In Chapter 4, we went into more details in discussing how we implemented our design and presented all the algorithms we implemented towards this study.

We carried out experiments on the two routing approaches we studied against the different network architectures and mobility scenarios, and presented our results and analysis in Chapter 5. As we expected, the flooding approach performed poorly under the different network conditions we considered when compared with our solution. Our solution, which leverages the storage and retrieval capabilities of a DHT, provides a scalable and efficient routing scheme for the different architectures and mobility scenarios we studied. In both the static scenario and when nodes were mobile, our solution incurred at most half of the total routing traffic the flooding approach incurred in the backbone network. Whereas the increased routing overhead negatively impacted the ping success rates in the flooding approach, as we varied the network architectures, success rates remained stable above 90% in our solution. The round trip times for messages delivered to destinations in different clusters were comparable to those for messages delivered locally within the clusters, which was not the case with the flooding approach.

We conclude that our DHT-based solution provides efficient routing supportive in both the static case and when mobility is involved, performing significantly better than the flooding approach.

6.2 Recommendations and Future Work

As future work, our DHT-based solution could be tested with much larger MANETs, i.e., increasing the total number of nodes N and splitting them into different number of clusters while maintaining the total number of nodes N in the network, to evaluate its effectiveness under the group mobility model we used in our study and other mobility models as well.

In this thesis, OLSR and a DHT application were deployed in the backbone in order to provide intra-domain routing. With OLSR, routes were readily available as per its proactive nature. One other protocol which could be explored is AODV. Therefore, future work could replace OLSR in the backbone with AODV to evaluate its effect in the presence of different mobility scenarios and for the different network architectures we used.

Another item for future work is to replace the location discovery that the DHT provides with one that mimics AODV protocol: broadcast a RREQ message through the backbone and have the gateway that contains the requested node respond. That will then build the route through the backbone and resolve the location of a node at the same time. In other words, using AODV's broadcasts for mobility management.

The backbone topology we explored in this study was a fully connected one where all the participating gateways were all within reach of each other. A more interesting scenario might be to design a multi-hop backbone network of gateways representing the different clusters they belong to, and particularly study the impact of mobility on our solution in such a scenario.

In our implementation, gateways schedule DHT queries every 30 seconds to obtain up-to-date route advertised from the DHT. As future work, the implementation could be modified such that ICMP Route Error messages will be returned to a source gateway GW_X from a destination gateway GW_Y for messages directed at a host Y_Y who is no longer a member of a destination cluster Y, or when ever an outdated DHT entry for a node Y is used, which will then trigger a route update from the DHT for destination Y.

With respect to the context of this thesis, clusters are given and so is the size of the clusters. We only considered evenly distributed clusters with equal number of nodes. It will be interesting also to study the performance of the network when the clusters are skewed. The interest here will be centered more on the ability of some of the gateway nodes to handle more traffic than the others.

We compared our DHT-based solution to blind flooding, which was easiest to implement. Also, as we scale up the number of clusters, overhead will grow similarly for all broadcasting-based solutions. There exist other flooding techniques like directed flooding which we discussed in the literature survey. As future work, any of these flooding approaches can then be used as a base case to evaluate the performance of our solution.

Finally, our DHT solution is designed to work with any routing and DHT protocol. We only experimented with OLSR and Chord, but it would be very interesting to see how the performance of this solution changes when used with DHTs that are particularly well suited for MANETs such as OneHopOverlay4MANETs. The use of a hierarchical network where the routing protocol and the DHT work together in a cross-layer fashion also seems apt for this task, especially for larger networks.

Bibliography

[1] Quang Hieu Vu, Mihai Lupu, and Beng Chin Ooi. *Peer-to-peer computing: Principles and applications*. Springer Science & Business Media, 2009.

[2] Marcello Caleffi and Luigi Paura. P2P over MANET: Indirect tree-based routing. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–5. IEEE, 2009.

[3] Ali Moussaoui and Abdallah Boukeream. A survey of routing protocols based on link-stability in mobile ad hoc networks. *Journal of Network and Computer Applications*, 47:1–10, 2015.

[4] Aisling O'Driscoll, Susan Rea, and Dirk Pesch. Hierarchical clustering as an approach for supporting P2P SIP sessions in ubiquitous environments. In *Mobile Wireless Communications Networks, 2007 9th IFIP International Conference on*, pages 76–80. IEEE, 2007.

[5] Mohammad Al Mojamed and Mario Kolberg. Structured peer-to-peer overlay deployment on MANET: A survey. *Computer Networks*, 96:29–47, 2016.

[6] Jamie R Furness. Optimising structured P2P networks for complex queries. 2014.

[7] Zhonghong Ou. Structured peer-to-peer networks: Hierarchical architecture and performance evaluation. *Dissertation*, 2010.

[8] Shahbaz Akhtar Abid, Mazliza Othman, and Nadir Shah. A survey on DHT-based routing for large-scale mobile ad hoc networks. *ACM Computing Surveys (CSUR)*, 47(2):20, 2015.

[9] Philippe Jacquet, Paul Muhlethaler, Thomas Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol for ad hoc networks. In *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International*, pages 62–68. IEEE, 2001.

[10] Gang Ding and Bharat Bhargava. Peer-to-peer file-sharing over mobile ad hoc networks. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 104–108. IEEE, 2004.

[11] Lu Yan. Can P2P benefit from MANET? performance evaluation from users' perspective. In *International Conference on Mobile Ad-Hoc and Sensor Networks*, pages 1026–1035. Springer, 2005.

[12] Harjeet Kaur, Varsha Sahni, and Manju Bala. A survey of reactive, proactive and hybrid routing protocols in MANET: A review. *network*, 4(3):498–500, 2013.

[13] Charu Sharma and Jaspreet Kaur. Literature survey of AODV and DSR reactive routing protocols. *ICAET, IJCA*, pages 14–17, 2015.

[14] Liliana Enciso Quispe and Luis Mengual Galan. Behavior of ad hoc routing protocols, analyzed for emergency and rescue scenarios, on a real urban area. *Expert Systems with Applications*, 41(5):2565–2573, 2014.

[15] Iftikhar Ahmad, Uzma Ashraf, and Abdul Ghafoor. A comparative QoS survey of mobile ad hoc network routing protocols. *Journal of the Chinese Institute of Engineers*, 39(5):585–592, 2016.

[16] Elizabeth M Belding-Royer. Hierarchical routing in ad hoc mobile networks. *Wireless Communications and Mobile Computing*, 2(5):515–532, 2002.

[17] Thomas Clausen and Philippe Jacquet. RFC 3626. *Optimized link state routing protocol (OLSR)*, 2003.

[18] András Varga. Omnet++ documentation and tutorials.

[19] Ngozi Silas, Thomas Kunz, and Babak Esfandiari. Evaluating chord over a hierarchical manet. In *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2017 8th IEEE Annual*, pages 608–617. IEEE, 2017.

[20] Joseph Macker. Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations. 1999.

[21] Rupali Mahajan and Rekha Patil. A review of enhanced blocking expanding ring search in mobile ad hoc networks. 2015.

[22] Azzedine Boukerche, Begumhan Turgut, Nevin Aydin, Mohammad Z Ahmad, Ladislau Bölöni, and Damla Turgut. Routing protocols in ad hoc networks: A survey. *Computer networks*, 55(13):3032–3080, 2011.

[23] Ozan K Tonguz, Nawaporn Wisitpongphan, Jayendra S Parikh, Fan Bai, Priyantha Mudalige, and Varsha K Sadekar. On the broadcast storm problem in ad hoc wireless networks. In *Broadband Communications, Networks and Systems, 2006. BROADNETS 2006. 3rd International Conference on*, pages 1–11. IEEE, 2006.

[24] Rajiv Misra and Chittaranjan Mandal. Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks. *IEEE Transactions on parallel and distributed systems*, 21(3):292–302, 2010.

[25] Dariusz R Kowalski and Andrzej Pelc. Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing*, 19(3):185–195, 2007.

[26] Wei Peng and Xi-Cheng Lu. On the reduction of broadcast redundancy in mobile ad hoc networks. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 129–130. IEEE Press, 2000.

[27] Hyojun Lim and Chongkwon Kim. Multicast tree construction and flooding in wireless ad hoc networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, pages 61–68. ACM, 2000.

[28] Anis Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'2002)*, 2002.

[29] Wei Peng and Xicheng Lu. Ahbp: An efficient broadcast protocol for mobile ad hoc networks. *Journal of computer science and technology*, 16(2):114–125, 2001.

[30] John Sucec and Ivan Marsic. An efficient distributed network-wide broadcast algorithm for mobile ad hoc networks. Technical report, CAIP Technical Report 248, Rutgers University, 2000.

[31] Brad Williams and Tracy Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 194–205. ACM, 2002.

[32] Reza Farivar, Mahdi Fazeli, and Seyed Ghassem Miremadi. Directed flooding: a fault-tolerant routing protocol for wireless sensor networks. In *Systems Communications, 2005. Proceedings*, pages 395–399. IEEE, 2005.

[33] Linliang Zhao, Gaoqiang Liu, Jie Chen, and Zhiwei Zhang. Flooding and directed diffusion routing algorithm in wireless sensor networks. In *Hybrid Intelligent Systems, 2009. HIS'09. Ninth International Conference on*, volume 2, pages 235–239. IEEE, 2009.

[34] Shuo Guo, Liang He, Yu Gu, Bo Jiang, and Tian He. Opportunistic flooding in low-duty-cycle wireless sensor networks with unreliable links. *IEEE Transactions on Computers*, 63(11):2787–2802, 2014.

[35] Adel Aneiba and Mohammed Melad. Performance evaluation of aodv, dsr, olsr, and grp manet routing protocols using opnet. *International Journal of Future Computer and Communication*, 5(1):57, 2016.

[36] Christopher Dearlove. Olsr developments and extensions. In *Proceedings of the 2nd OLSR Interop and Workshop*, 2005.

[37] Majda Omer Elbasheer Ali and Jaime Lloret. Performance evaluation of dsdv, dsr, aodv and tora manet routing protocols for body monitoring in free space environments. *SPWID 2016*, page 25, 2016.

[38] Sahabul Alam and Debashis De. Cloud smoke sensing model for aodv, rip and star routing protocols using wireless sensor network in industrial township area. In *Research in Computational Intelligence and Communication Networks (ICRCICN), 2016 Second International Conference on*, pages 51–56. IEEE, 2016.

[39] Pankaj Kumar Varshney, GS Agrawal, and Sudhir Kumar Sharma. Relative performance analysis of proactive routing protocols in wireless ad hoc networks using varying node density. *Invertis Journal of Science & Technology*, 9(3):161–169, 2016.

[40] Wafaa Ibrihich, Krit Salah-ddine, Jalal Laassiri, and Said El Hajji. Recent advances of hierarchical routing protocols for ad-hoc and wireless sensor networks: A literature survey. *International Journal Of Informatics Technologies-Ijit*, 9(2), 2016.

[41] Ravinder Ahuja. Simulation based performance evaluation and comparison of reactive, proactive and hybrid routing protocols based on random waypoint mobility model. *International Journal of Computer Applications*, 7(11), 2010.

[42] Avni Khatkar and Yudhvir Singh. Performance evaluation of hybrid routing protocols in mobile ad hoc networks. In *Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on*, pages 542–545. IEEE, 2012.

[43] Seung-Chul M Woo and Suresh Singh. Scalable routing protocol for ad hoc networks. *Wireless Networks*, 7(5):513–529, 2001.

[44] Anupama M and Bachala Sathyanarayana. Survey of cluster based routing protocols in mobile adhoc networks. *International Journal of Computer Theory and Engineering*, 3(6):806, 2011.

[45] Abdelhak Bentaleb, Abdelhak Boubetra, and Saad Harous. Survey of clustering schemes in mobile ad hoc networks. *Communications and Network*, 5(02):8, 2013.

[46] Ismail Ghazi Shayeb, AH Hussein, and Ayman Bassam Nasoura. A survey of clustering schemes for mobile ad-hoc network (manet). *American Journal of Scientific Research*, 20(2011):135–151, 2011.

[47] Doina Bein, Ajoy Kumar Datta, Chakradhar R Jagganagari, and Vincent Villain. A self-stabilizing link-cluster algorithm in mobile ad hoc networks. In *Parallel Architectures, Algorithms and Networks, 2005. ISPAN 2005. Proceedings. 8th International Symposium on*, pages 6–pp. IEEE, 2005.

[48] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *proceedings of IEEE SICON*, volume 97, pages 197–211, 1997.

[49] Stefano Basagni. Distributed clustering for ad hoc networks. In *Parallel Architectures, Algorithms, and Networks, 1999.(I-SPAN'99) Proceedings. Fourth InternationalSymposium on*, pages 310–315. IEEE, 1999.

[50] AbdelRahman Hussein, Sufian Yousef, Samir Al-Khayatt, and Omar S Arabeyyat. An efficient weighted distributed clustering algorithm for mobile ad hoc networks. In *Computer Engineering and Systems (ICCES), 2010 International Conference on*, pages 221–228. IEEE, 2010.

[51] Raghupathy Sivakumar, Prasun Sinha, and Vaduvur Bharghavan. Core extraction distributed ad hoc routing (cedar). In *Proc. of INFOCOM'99*, 1999.

[52] Leonard Kleinrock and Farouk Kamoun. Hierarchical routing for large networks performance evaluation and optimization. *Computer Networks (1976)*, 1(3):155–174, 1977.

[53] Linet Özdamar and Onur Demir. A hierarchical clustering and routing procedure for large scale disaster relief logistics planning. *Transportation Research Part E: Logistics and Transportation Review*, 48(3):591–602, 2012.

[54] Paul F Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 35–42. ACM, 1988.

[55] Steven Glass, Tom Hiller, Stuart Jacobs, and C Perkins. Mobile ip authentication, authorization, and accounting requirements. Technical report, 2000.

[56] Ulf Jönsson, Fredrik Alriksson, Tony Larsson, Per Johansson, and Gerald Q Maguire Jr. Mipmanet: mobile ip for mobile ad hoc networks. In *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing*, pages 75–85. IEEE Press, 2000.

[57] Behrouz A. Forouzan. *Data communications & networking (sie)*. Tata McGraw-Hill Education, 2006.

[58] Anthony Yu and Son T Vuong. A DHT-based hierarchical overlay for peer-to-peer MMOGs over MANETs. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1475–1480. IEEE, 2011.

[59] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003.

[60] Thomas Zahn and Jochen Schiller. MADPastry: A DHT substrate for practicably sized MANETs. In *Proc. of ASWN*, 2005.

[61] Ben Y Zhao, Ling Huang, Jeremy Stribling, Sean C Rhea, Anthony D Joseph, and John D Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on selected areas in communications*, 22(1):41–53, 2004.

[62] George Porter, Kevin Lai, Ion Stoica, and Jeremy Condit. The Chord ad-hoc routing protocol. 2002.

[63] Donald Eastlake 3rd and Paul Jones. Us secure hash algorithm 1 (sha1). Technical report, 2001.

[64] Mian Ahmad Jan and Muhammad Khan. A survey of cluster-based hierarchical routing protocols. *IRACST–International Journal of Computer Networks and Wireless Communications (IJCNWC)*, 3(2):138–143, 2013.

[65] Chi-Kin Chau, Jon Crowcroft, Kang-Won Lee, and Starsky HY Wong. Idrm: Inter-domain routing protocol for mobile ad hoc networks. Technical report, University of Cambridge, Computer Laboratory, 2008.

[66] Danny McPherson, Vijay Gill, Daniel Walton, and Alvaro Retana. Border gateway protocol (bgp) persistent route oscillation condition. Technical report, 2002.

[67] Seung-Hoon Lee, Starsky HY Wong, Chi-Kin Chau, Kang-Won Lee, Jon Crowcroft, and Mario Gerla. Intermr: Inter-manet routing in heterogeneous manets. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 372–381. IEEE, 2010.

[68] Thomas Kunz, Babak Esfandiari, and Frank Ockenfeld. Efficient routing in mobile ad-hoc social networks. In *Internet of Things (iThings) and IEEE Green*

Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017 IEEE International Conference on, pages 216–222. IEEE, 2017.

[69] Hai L Vu, Sammy Chan, and Lachlan LH Andrew. Performance analysis of best-effort service in saturated ieee 802.16 networks. *IEEE Transactions on Vehicular Technology*, 59(1):460–472, 2010.

[70] Prasanna Murali Krishna, Mayya V. Subramanyam, and K. Satya Prasad. Investigation of Chord protocol in peer to peer-wireless mesh network with mobility. *WAS., ET, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, pages 934–938, 2015.

[71] Lokesh Pawar, Rohit Bajaj, and Geetika Sharma. Evaluation of mobile ad hoc network routing protocols and associated mobility models. 2017.

[72] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[73] Joe Suzuki and Yutaka Yamamoto. inet: An extensible framework for simulating immune network. In *IEEE INTERNATIONAL CONFERENCE ON SYSTEMS MAN AND CYBERNETICS*, volume 1, pages 119–124, 2000.

[74] Ingmar Baumgart, Bernhard Heep, and Stephan Krause. Oversim: A scalable and flexible overlay framework for simulation and real network applications.

In *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*, pages 87–88. IEEE, 2009.

[75] Simpson Timothy. Rfc 1853:" ip in ip tunneling", feb. 2001. *Disponible en (4-2010): http://www. faqs. org/rfcs/rfc1853. html*.

[76] Geoff Huston. Measuring ip network performance. *The Internet Protocol Journal*, 6(1):2–19, 2003.

[77] Rami Rosen. Internet control message protocol (icmp). In *Linux Kernel Networking*, pages 37–61. Springer, 2014.

[78] AG CoNe, Rechnernetze und Telematik, and Christian Schindelhauer. Internet control message protocol. 2008.

[79] John Postel. Internet control message protocol; rfc792. *ARPANET Working Group Requests for Comments*, (792), 1981.

[80] Josyl Mariela B Rocamora and Jhoanna Rhodette I Pedrasa. Evaluation of hierarchical dhts to mitigate churn effects in mobile networks. *Computer Communications*, 85:41–57, 2016.

[81] Xinghua Sun and Lin Dai. Backoff design for ieee 802.11 dcf networks: Fundamental tradeoff and design criterion. *IEEE/ACM Transactions on Networking (TON)*, 23(1):300–316, 2015.

[82] Deepak Kumar, Ashutosh Srivastava, and Suresh C Gupta. Routing in ad hoc networks under reference point group mobility. In *Modelling Symposium (EMS), 2013 European*, pages 595–598. IEEE, 2013.

[83] Radhika Ranjan Roy. Reference point group mobility. In *Handbook of Mobile Ad Hoc Networks for Mobility Models*, pages 637–670. Springer, 2011.

[84] Aniket Pramanik, Biplav Choudhury, Tameem S Choudhury, Wasim Arif, and J Mehedi. Simulative study of random waypoint mobility model for mobile ad hoc networks. In *Communication Technologies (GCCT), 2015 Global Conference on*, pages 112–116. IEEE, 2015.

[85] Cherry Ye Aung, Boon Chong Seet, Mingyang Zhang, Ling Fu Xie, and Peter Han Joo Chong. A review of group mobility models for mobile ad hoc networks. *Wireless Personal Communications*, 85(3):1317–1331, 2015.

[86] Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, and Matthias Schwamborn. Bonnmotion: a mobility scenario generation and analysis tool. In *Proceedings of the 3rd international ICST conference on simulation tools and techniques*, page 51. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.