The Optimisation of Peer-to-Peer Overlays for Mobile Ad-hoc Networks

Grant P. Millar

Wireless, Multimedia & Networking Research Group
Faculty of Science, Engineering & Computing
School of Computing & Information Systems
Kingston University London

This dissertation is submitted for the degree of
Doctor of Philosophy (Ph.D.)

2013

1. External Examiner: Professor Nikolaos Antonopoulos

Head of School of Computing and Mathematics

University of Derby

Kedleston Road

Derby DE22 1GB

United Kingdom

2. Internal Examiner: Dr Dimitris Tsaptsinos

Principal Lecturer

Kingston University London

Penrhyn Road

Kingston-Upon-Thames

London KT1 2EE

United Kingdom

Signature from Chair of PhD committee:

Faculty of Science, Engineering and Computing (SEC)

School of Computing and Information Systems (CIS)

Kingston University London

Penrhyn Road, Kingston-Upon-Thames

London KT1 2EE

ABSTRACT

The proliferation of smart devices and wireless interfaces has enabled the field of Mobile Ad-hoc Networks (MANETs) to flourish as the technology becomes realistically deployable. MANETs are created by a group of autonomous nodes that communicate with each other by establishing a multihop radio network and maintain connectivity in an infrastructureless and decentralised manner. Thus it becomes more important to examine how applications such as those used on the Internet can be deployed on top of such MANETs.

Peer-to-peer (P2P) networks can be defined as decentralised application-layer overlay networks where traffic flows on top of the physical network such as the Internet. Such networks are formed dynamically on-the-fly and rely on no fixed infrastructure such as servers. On the Internet a great number of applications have exploited the properties of P2P networks, thus they have been used to provide services such as distributed data storage systems, distributed instant messaging systems, publish/subscribe systems, distributed name services Voice over IP (VoIP) services and many more.

This thesis proposes three novel P2P protocols. Reliable Overlay Based Utilisation of Services and Topology (ROBUST), which minimises end-to-end lookup delay while increasing lookup success rate compared with current state-of-the-art and is usable on any MANET routing protocol. It achieves this by using a hierarchal clustered topology where peers are clustered together with other peers in close proximity in order to reduce P2P routing hops and create a more efficient network.

Proactive MANET DHT (PMDHT), which combines proactive MANET routing and DHT functionality in order to minimise end-to-end lookup delay while increasing lookup success rate compared with current state-of-the-art. This is achieved by heavily integrating the P2P functionality – by piggy-backing P2P messages on to routing messages – at the network layer using the proactive MANET routing protocol Optimized Link State Routing version 2 (OLSRv2). Using this method the P2P overlay topology exactly matches that of the underlying network, while all peers are fully aware of the state of that topology. This

means the P2P lookups can be completed in one logical step.

Reactive MANET P2P Overlay (RMP2P), which combines reactive MANET routing and DHT functionality in order to minimise end-to-end lookup delay while increasing lookup success rate compared with current state-of-the-art. in RMP2P we combine P2P lookup functionality with the MANET routing protocol Ad-hoc On-demand Distance Vector version 2 (AODVv2). In this case we piggy-back P2P lookups on to the routing request messages if possible, decreasing overhead and latency in the network.

We evaluate the performance of the proposed novel architectures by developing a custom made packet level simulator using ns-2 (network simulator-2), the results show that these architectures out perform the current state-of-the-art P2P overlays in specific scenarios. The ROBUST protocol is suited to scenarios where the underlying routing protocol cannot be modified. The PMDHT protocol performs best overall in networks which require more scalability. The RMP2P protocol performs best in networks with high mobility. We end this thesis with our conclusions and avenues for future work in the field of P2P networks for MANETs.

Acknowledgments

Firstly I must thank my supervisor, Dr. Christos Politis for his inspiration, advice, expertise and encouragement on the ideas contained within this thesis. It has been an honour to work with such a dedicated supervisor over the past four years. I cannot thank him enough for the freedom and confidence that he has given to me to pursue my own ideas and without his support and guidance this thesis would not have been possible. I am eternally grateful for the opportunity to work in the magnificent Wireless, Multimedia & Networking (WMN) research group, without which I would not have had the wonderful experiences of the past four years. For this I owe him my deepest gratitude.

I would also like to thank my friends and family for providing the support and encouragement needed to persist over the past four years. Special thanks must be given to my parents Tracey and Mike, Paul and Sheila for their unconditional moral support, my sister Keeley for her motivational abilities, and my uncle Kevin for his great advice and guidance. Without the support of my friends and family I would not be in the position I am today. Special thanks to my girlfriend Alexandra for her considerable support, patience and encouragement.

I should further express my thanks to my colleagues and friends that I have encountered over the past four years, especially those from the WMN research group. Both Emmanouil Panaousis and Arvind Ramrekha provided a great source of inspiration during our collaboration in the EU FP7 PEACE project and beyond, thus we have a number of papers together. Their expertise and discussions helped shape the insight needed to fulfil this thesis.

I would like to thank all collaborators from the EU FP7 PEACE — we had a great final review and the reviewers comments were very encouraging — and the European commission, for without their funding none of this work would have been possible.

Lastly I would like to thank all of my previous teachers and colleagues, along with all of the staff at Kingston University, all of whom have shaped my education.

Table of Contents

List of Figures

List of Tables

List of abbreviations

AODV Ad hoc On demand Distance Vector.

AODVv2 Ad-hoc On-demand Distance Vector version 2.

CAN Content Addressible Network.

CG Common Grouping.

CGA Common Group Aliasing.

CPU Central Processing Unit.

DHT Distributed Hash Table.

DREP Peer-to-Peer Reply.

DREQ Peer-to-Peer Request.

DSDV Destination-Sequenced Distance-Vector routing.

DSR Dynamic Source Routing.

E2E End-to-end.

EDSR Enhanced Dynamic Source Routing.

FR First Responder.

GPS Global Positioning System.

HSR Hierarchical State Routing.

HTTP Hypertext Transfer Protocol.

IEEE Institute of Electrical and Electronics Engineers.

IETF Internet Engineering Task Force.

IP Internet Protocol.

ISPRP Iterative Successor Pointer Rewiring Protocol.

MAC Medium Access Control.

MANET Mobile Ad hoc Network.

MPP Mobile Peer-to-peer Protocol.

MPR MultiPoint Relay.

NHDP NeighborHood Discovery Protocol.

ns-2 network simulator-2.

OLSR Optimized Link State Routing.

OLSRv2 Optimized Link State Routing version 2.

P2P Peer-to-Peer.

P2PSIP Peer-to-Peer Session Initiation Protocol.

PDA Personal Digital Assistant.

PMDHT Proactive MANET DHT.

PNS Proximity Neighbour Selection.

QoS Quality-of-Service.

RERR Route Error.

RFC Request For Comment.

RMP2P Reactive MANET Peer-to-Peer Overlay.

RO Research Objective.

ROBUST Reliable Overlay Based Utilisation of Services and Topology.

RREP Route Reply.

RREQ Route Request.

RTO Round-trip Timeout.

RTT Round Trip Time.

RWP Random Way Point.

SHA-1 Secure Hash Algorithm 1.

SPS Successor Pointer Solicitation.

SRS Successor Rewiring Solicitation.

SSR Scalable Source Routing.

TC Topology Control.

TCP Transmission Control Protocol.

TLV Type Length Value.

TTL Time-To-Live.

UDP User Datagram Protocol.

VAP Virtual Access Point.

VoIP Voice over IP.

VRR Virtual Ring Routing.

ZP2P Zone-based Peer-to-Peer.

ZRP Zone Routing Protocol.

Chapter 1

Introduction

Peer-to-Peer (P2P) networks can be defined as decentralised application-layer overlay networks where traffic flows on top of the physical network such as the Internet. Such networks are formed dynamically on-the-fly and rely on no fixed infrastructure such as servers. In contrast to the common networking paradigm where one participates as either a client or a server in a very defined role, P2P network participants or *peers* carry out the role of client, server and infrastructure. Peers in the P2P network can both serve other peers as well as act in a client fashion or as intermediaries for other peers to forward data to a certain destination. Unlike centralised networks peers have no central entity which they can query in order to find another entity in the network such as an object or other peer. In the case of P2P networks, distributed algorithms and protocols are relied upon in order to search the network for a given entity. There are numerous advantages to using P2P networks in certain situations owing to some of their unique characteristics, these are described below.

In a client-server scenario the client is generally limited in the number of simultaneous connections which can be made to the server, this is normally one, as a single stream of data is used in order for the bytes being transmitted to be reassembled in the same order in which they were downloaded. Due to the cost of transit agreements and the complex process of peering — where two ISPs decided to trade traffic — a route between two

ISPs on the Internet can often be congested due to many hops and many intermediate ISPs [1]. This results in packet loss and slower than expected throughput. The throughput could be increased by using multiple concurrent connections but this is implementation specific and dependant on the application due to the complexity of assembling data which was essentially received in the wrong order. P2P networks however have been built with multiple streams of data being received from their inception [2–6]. This results in more aggregate bandwidth available to the P2P network due to the accumulation of concurrent connections [7] and provides the network with more potential scalability as more peers will contribute more bandwidth in addition to computing power and storage.

The distributed nature of P2P networks means there is no single point of failure. If one takes the example of a simple web server, if for some reason the server is unavailable — due for example to a network outage — no clients would be able to retrieve data as no redundancy is built in. In P2P networks however, data can be accessed from any number of sources. If one source goes offline, the peer requesting the data can simply ask another peer. The redundancy of P2P networks grows with scalability, as more peers join the network, there is more possibility to store duplicates of the data. This is largely implementation dependant — and dependant on time factors of specific replication algorithms — but overall makes P2P networks highly resilient and fault-tolerant.

Another feature of P2P networks is their common use of self-configuration. Thus they take advantage of distributed algorithms so that each peer has a *job* which is fulfilled by carrying out certain tasks in the network such as replication, responding to queries or forwarding messages. Therefore central entities which control admission and global administration of the network are not required. New peers are bootstrapped to the network by existing peers, this enables the network to scale without the need for expensive dedicated hardware and bandwidth, also allowing the network to be created with minimal effort and investment. This dynamism and ease-of-setup of P2P networks can be seen as one of the major strengths which makes them so successful in large scale networks such as the Internet.

With the recent proliferation of social networks, privacy has become an area of interest in the global community due to the nature of sharing personal information en masse with the general public [8]. P2P networks provide a high level of anonymity when compared to centralised networks due to the lack of any authentication and distinguishing features of peers other than Internet Protocol (IP) address meaning a peer's actions cannot be easily monitored. This results in a highly resilient network which makes censorship extremely difficult and permanent shutdown of the network almost impossible due to its distributed nature. This has lead to some well known P2P services being labeled as havens for copyright violation [9] with some content providers going as far as creating false content in the system to cause what is known as *pollution* and lowering the usefulness of such a service [10].

In the past P2P networks have been used for a wide range of applications. Their scalability means that a vast number of these applications and services have appeared on the Internet, where using a central infrastructure would be complex to set up and economically expensive. Since their inception the most well known P2P services have been largely used for file sharing. The popular services [2, 4, 6] related to P2P file sharing can be categorised as *unstructured* P2P networks as they do not operate using a structured topology, rather the network's peers are organised largely at random. As a consequence of this, these services take advantage of typical flooding algorithms in order to search the network for the relevant content. This flooding — being broadcast in nature — creates a large amount of signalling traffic as each peer has to forward the request. The effect of this can be restricted by using a Time-To-Live (TTL) on the search query, however this has the drawback of not searching all of the available peers, leading to a trade-off between scalability and search-ability in large networks.

As previously mentioned the main hindrance of unstructured P2P networks is that of scalability. During the last decade an effort has been made in the research community to address this problem. Thus the area of structured P2P networks was born. Structured P2P networks consist of an overlay with an adherence to a structured topology such that a specific piece of data can be searched for and found within a certain number of

forwarded search messages. The most well known and heavily studied area of structured P2P networks is Distributed Hash Table (DHT) [11–15]. DHTs impose a structure on the topology of the overlay network and thus create an upper bound on the number of steps required to locate a piece of data in the network. The process of finding information involves using a key-based routing algorithm which can efficiently find a piece of data stored with a specific key among all of the participants of the overlay by locating the specific peer responsible for said data. This provides a big advantage over unstructured networks as no flooding of the network needs to take place. Due to this a great number of applications have exploited the properties of DHTs to provide services such as distributed data storage systems [16–19], distributed instant messaging systems [20], publish/subscribe systems [21–25], distributed name services [26–30], Peer-to-Peer Session Initiation Protocol (P2PSIP) [31–34] for Voice over IP (VoIP) services and many more.

Another field which has flourished in the past few years is that of Mobile Ad hoc Networks (MANETs). MANETs are physical networks consisting of wireless devices (normally WiFi i.e. Institute of Electrical and Electronics Engineers (IEEE) 802.11 [35]) or *nodes* such as smartphones, tablets, laptops, and sensor equipment among others. These devices are able to dynamically form wireless networks amongst themselves and thus akin to P2P networks, have no fixed or centralised infrastructure, instead relying on self-configuration. Like in P2P networks nodes will send requests (like clients), forward requests (like routers) and reply to requests (like servers). MANETs also have to deal with churn — where nodes can join and leave the network at any time without consulting others in the network — which is another feature shared with P2P networks.

P2P overlays and MANETs share a common notion of rapid network setup, thus MANETs can be vital in scenarios where an instantly deployable network is needed. Such scenarios include ubiquitous computing, proving a instant network infrastructure, and disaster recovery during emergency scenarios. The term *emergency scenarios* is considered to envelop a plethora of extreme disaster situations such as but not limited to; forest fires, terrorist attacks and major floods and earthquakes. In these catastrophic events normal

communication infrastructures have a high likelihood of being damaged beyond functional working conditions. Repairing such infrastructures can take a great amount of time, and emergency personnel (fire brigade, police, paramedics) whom need functional communications the second they arrive on scene in order to coordinate and put into action any disaster contingency plan they may have conceived as a forethought. In these stressful conditions reducing the setup time of the network can save countless lives — emergency workers refer to this critical window immediately in the aftermath of the disaster as the *golden hour*, thus emphasising its importance. The instantaneous setup of MANETs makes them an ideal candidate for use by those First Responders (FRs) on scene in disaster recovery. One of the main challenges related to the real-world deployment of MANETs is efficient *routing*. Due to nodes physically moving in a 3-Dimensional spacial area the most efficient route in terms of a specific metric (such as the fastest or most reliable route) may be constantly changing, therefore routes need to be re-evaluated regularly and route failures must be dealt with in a swift manner in order to reduce the impact.

1.1 Problem statement

One can see that even on the surface both P2P networks and MANETs share some common properties such as being decentralised and self-organising, as well as participants sharing their network resources to relay packets for others. The algorithms for routing in both technologies are focused on searching the network. In MANETs the focus is searching for a route to a specific destination IP, whereas in P2P networks the focus is on searching for a specific piece of data, and retrieving that data. DHTs are generally reactive in search and proactive in replication, in that a search query is executed with the destination being found on demand, while replication is executed at specific time intervals to ensure availability. In unstructured P2P networks search is again proactive, searching for data on the fly, where as traditionally no replication mechanisms exists.

The Pastry [13] overlay structure is an example where the physical topology of the underlying network (such as a MANET or the Internet) and logical topology (the P2P

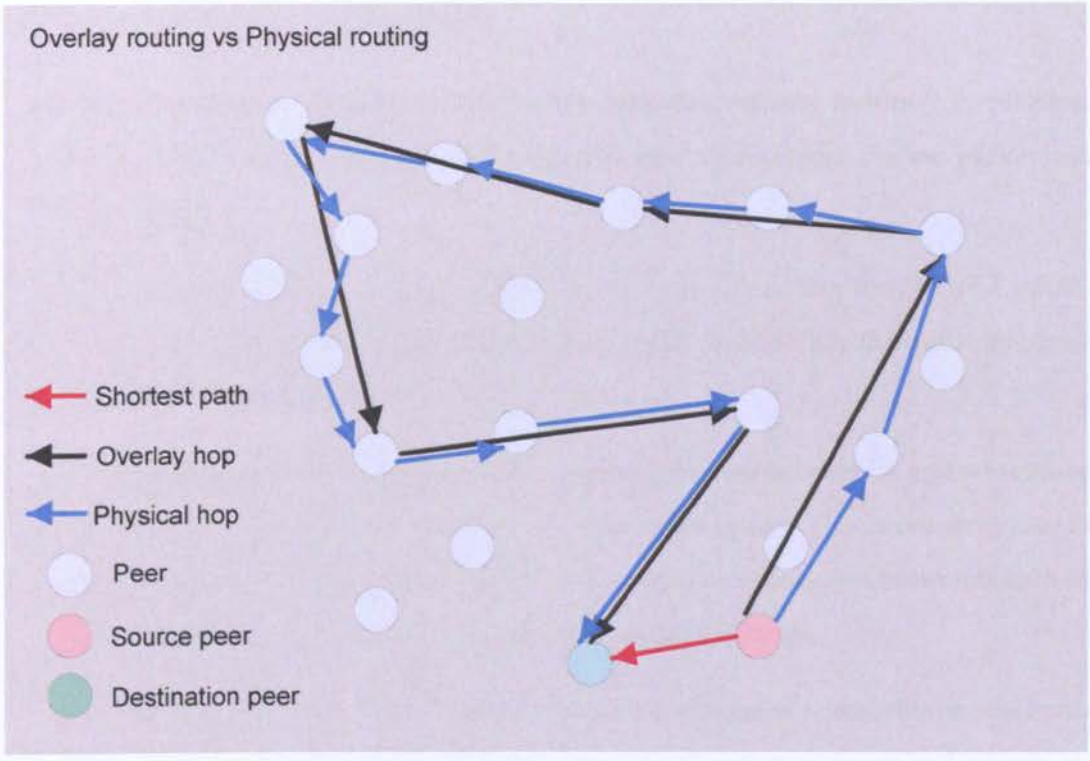

Figure 1.1: Overlay routing vs Physical routing issues.

overlay network) are not tied together. Despite the fact that Pastry has the potential to route to a given key within an overlay of N nodes in $\lceil \log_2 N \rceil$ steps, there remains the actuality that Pastry does not account for intermediate physical nodes. In this case the actual number of physical steps can increase greatly in large-scale networks, an exaggeration of this issue can be seen in Figure 1.1 where six overlay hops equates to twelve physical hops, meanwhile the source and destination are only one hop apart in the physical network. This may pose no problem when bandwidth is plentiful and the overhead — this refers to overhead meaning number of packets a given function transmits in the network — for routing such lookups is negligible when compared with the link bandwidth, such as the Internet where a neighbour in the overlay network could physically be located hundreds if not thousands of kilometres away. In the context of MANETs however, one cannot neglect

such overheads for the following reasons;

- The links between MANET nodes can be weak due to many factors (e.g. physical obstructions). This means the less bandwidth used in overhead, the less packet loss is induced.
- There are no backbones akin to those in fixed networks within the MANET infrastructure which can handle vast quantities of traffic. In MANETs the traffic is spread throughout the network.
- Links between nodes of the MANET are usually of lower bandwidth and more lossy in terms of packets compared to their Internet counterparts. This is primarily due to the bandwidth in MANETs being by no means guaranteed due to constraints such as distance, hardware, transmission errors and packet collisions.
- Device properties must be addressed. Overheads may seem reasonable on one hardware platform such as laptops, where plenty of processing power and energy resources are available, but may incur severely diminished battery life if the same routing operations are performed on devices such as a smartphones, where both processing power and energy resources are less.

These issues are further compounded when mobility of the MANET nodes is taken into account, routes from source to destination can be constantly changing as links break and new links are formed causing temporary loss of connections while new routes are discovered.

One can also see that having two routing protocols (MANET and P2P) rather duplicates the effort of finding data in the network. Due to their decentralised nature most MANET routing protocols involve some type of flooding, one could envisage a scenario where an overlay key lookup hop involved two physical hops, the destination of both hops needing to be discovered in the physical MANET, thus causing two iterations of flooding for just one overlay lookup hop, which is not an optimal solution.

Additionally one must consider that in order for a DHT to operate properly — all routing converges and consistency is achieved for redundancy — DHTs must periodically maintain their own overlay routing tables (or something similar). Depending on the DHT implementation this signalling traffic often accounts for a large portion of the overall DHT traffic, given the aforementioned technical constraints in MANETs the maintenance traffic could be too intensive and overwhelm the MANET causing congestion. On the other hand, unstructured P2P networks must search the whole network to find a specific piece of data, this flooding, even if temporarily can cause congestion in the MANET as a route to every node would need to be looked up.

Taking all of the above into consideration it is clearly not practical to use existing P2P networks on top of MANETs when their original design was intended for completely different network circumstances. Thus when designing an efficient P2P network for MANETs one must take all of the MANET characteristics into consideration. A number of studies have been made to address these problems in P2P networks designed for the Internet [36–38], however conventional P2P networks primarily do not consider the physical network, so their optimisation is limited.

1.2 Research aims and objectives

This thesis aims to design P2P networks for MANETs taking the previously mentioned characteristics into account in order to create more efficient P2P networks to be used in such scenarios. Another goal is to conduct network performance evaluation of said P2P networks using packet level network simulators. Thus enabling this thesis to guide development in future implementations of P2P networks designed for applications on MANETs. The main Research Objectives (ROs) of this thesis are summarised, as follows:

RO1. In order to be practical multiple methods should be investigated so that both P2P networks optimised to the full extent (by sharing MANET routing information) and P2P networks optimised based on the constraints of any MANET routing protocol

are considered.

RO2. Converge routing where possible. As discussed in Section 1.1 routing in P2P networks and MANETs often overlaps, this should be optimised where practical so that information can be shared between the P2P network and MANET routing table.

RO3. Proximity awareness in P2P networks. As described in Section 1.1 it is of paramount importance that P2P networks designed for MANETs take into account the physical network topology and optimise as far as possible the P2P network to use the same topology, therefore physical neighbours should also be P2P overlay neighbours in order to reduce overhead.

RO4. Key-based lookup for MANETs. To enable efficient key lookup for MANETs by exploiting the main functionality of P2P networks and enable existing applications in use on the Internet to be ported over with minimal effort.

RO5. Evaluate the performance of the above mentioned ROs using a suitably designed and implemented packet-level event based simulator.

Throughout this dissertation, these objectives will be referred to by using their acronyms RO1 – RO5.

1.3 Research contributions

The following summarises this thesis' research contributions with regard to the different research objectives:

- To address RO1, three novel P2P architectures have been proposed to cover the three scenarios, the two cross-layer — in the sense that application layer protocols are combined with the routing layer — protocols below were chosen based on proactive and reactive MANET routing protocols as it has been shown that the two types of MANET routing protocol are suited to different scenarios [39, 40] as confirmed later in this thesis:

1. The case where any MANET routing protocol is used for example AODV [41], OLSR [42] , DSR [43] or DSDV [44]. This to satisfy practicality as it is not always possible to use a specific routing protocol on any device, therefore a DHT architecture entitled Reliable Overlay Based Utilisation of Services and Topology (ROBUST) — based on OpenDHT from [11] — is proposed which is optimised as far as possible for MANETs but not dependant on any specific MANET routing protocol. This will be achieved by exploiting certain characteristics of MANETs, such as their broadcast nature, in order to create an overlay which closely matches the physical network. The work carried out in this thesis is novel and goes beyond previous work in this area as the author's search through the literature does not yield any in depth analysis — through thorough simulation and modelling — of similar DHT protocols based on optimising DHT routing and stricter neighbour proximity while allowing any MANET routing protocol to be used. - Related Publications: [45, 46].

2. Where a proactive MANET routing protocol will be used. A novel cross-layer solution entitled Proactive MANET DHT (PMDHT) is proposed — based on OpenDHT from [11] — to target the popular MANET routing protocol Optimized Link State Routing version 2 (OLSRv2) [47] in order to take advantage of routing information to further optimise the operation of the P2P network. The work in this area is novel as the authors work goes beyond the state-of-the-art by combining DHT functionality and proactive MANET routing, allowing key-based lookup with less packet overhead, which has not been thoroughly studied before. - Related Publication: [48].

3. Where a reactive MANET routing protocol will be used. A novel solution entitled Reactive MANET Peer-to-Peer Overlay (RMP2P) is proposed to target the popular MANET routing protocol Ad-hoc On-demand Distance Vector version 2 (AODVv2) [49] in order to take advantage of routing information to further optimise the operation of the P2P network. The work on this area goes be-

yond the state-of-the-art as the proposed protocols are compared to similar P2P networks which have been combined with reactive MANET routing, our simulation results show a marked improvement over the state-of-the-art. - Related Publication: [48].

- To achieve RO2, two P2P architectures have been designed for MANETs based on the two most widely used MANET routing protocols, namely AODVv2 and OLSRv2, these were chosen based on the fact they (their predecessors on which they are based) are the only two MANET routing protocols to be standardised as Request For Comments (RFCs) in the Internet Engineering Task Force (IETF). The two architectures take full advantage of the MANET routing characteristics in order to optimise the overhead caused by running the P2P network. This is achieved by exploiting the MANET routing table information for the discovery of P2P overlay peers among other optimisations. - Related Publication: [48].

- Regarding RO3, the author's proposed method entitled Common Group Aliasing (CGA) is employed which uses broadcast beacons to create proximity in the P2P network and map the physical topology to the overlay topology. To achieve this CGA creates clusters of peers with similar P2P overlay IDs using broadcasting. Thus physical neighbours have a greater chance of also being neighbours in the P2P network. - Related Publication: [46].

- To address RO4, all of the above proposed architectures are based on the main concept of key-based routing, the main principle behind DHTs. DHTs behave by routing a packet containing a key which the source wishes to retrieve the data (or value) for towards the peer which has the closest matching peer ID to the key. ROBUST, PMDHT and RMP2P utilise the concept of key-based data retrieval (using different routing mechanisms) and hence provide the functionality needed by such applications and services as described at the beginning of Chapter 1. - Related Publications: [45,48,50].

- RO5 was accomplished by building a unique P2P simulator on top of the network

simulator network simulator-2 (ns-2) [51]. The packet-level simulator encompasses all aspects of the P2P network including overlay routing, key replication and random lookups. Accordingly the MANET routing protocols are then simulated using the respective simulation models provided in ns-2. The outcome achieved is a very detailed analysis of the P2P network performance based on networking metrics such as end-to-end packet delay and packet loss, as well as more specific P2P overlay metrics such as lookup success rate and correlation between the physical and overlay topologies. - Related Publications: [45,46,48,50].

1.4 Thesis structure

The remainder of this thesis is organised as follows.

Chapter 2 contains an overview of the background required for understanding the concepts in this thesis. The different techniques used in P2P overlay networks, namely structured and unstructured are described in detail. The advantages and shortcomings of each method are examined. This chapter also discusses the different approaches to ad-hoc routing.

Chapter 3 examines related approaches to optimising P2P architectures for MANETs giving an overview of the area including both structured and unstructured approaches and additional service discovery mechanisms in the related area. The advantages and shortcomings of each proposed solution are evaluated.

Chapter 4 describes our DHT architecture for heterogenous MANET routing protocols entitled ROBUST which optimises the overlay architecture by utilising a clustering algorithm to create equal sized clusters of nodes brought together in a proximity aware fashion, routing is then carried out using a super-peer style architecture. Chapter 4 describes the architecture in detail specifically looking at the overhead caused by each function. The ROBUST architecture is then evaluated using our unique P2P simulator for MANETs which is described in detail.

Chapter 5 describes two proposed cross-layer P2P architectures tailored for the two most

popular MANET routing protocols (OLSRv2 and AODVv2) in their respective approaches of proactive and reactive route discovery entitled PMDHT and RMP2P. Each protocol is described in detail focusing on the overhead created in the network. Then each protocol is thoroughly evaluated using our customised unique packet-level network simulator. The simulations are given context by comparing them with ROBUST as well as two of the most cited alternative P2P architectures for MANETs.

Chapter 6 concludes this thesis summarising our findings and highlighting our main contributions with respect to the thesis' objectives. The research limitations are deduced and the main avenues for future work in optimising P2P architectures for MANETs are discussed.

Chapter 2

Background

This chapter serves as an overall background for the technologies which are discussed in the remainder of this thesis. Firstly an overview of unstructured Peer-to-Peer (P2P) networks is given followed by structured P2P networks. Next an overview of the general issues in Mobile Ad hoc Networks (MANETs) is examined. Finally this chapter will then conclude with an overview of optimisations for P2P overlay networks which take into account proximity in the physical network.

2.1 Peer-to-peer networks

As mentioned in Chapter 1, P2P networks are self-configuring and self-organising, meaning that individual peers are responsible for both joining the network, and carrying out searches or *lookups* as well as acting as part of the infrastructure for other peers. This means peers may act as clients, servers and routers all concurrently. The specific functions related to these roles include searching for data by issuing a request, issuing a response to another peers request, or forwarding a request to another peer if no match has been identified locally. As well as referring to said networks as P2P networks they can also be described as *overlay networks* this essentially refers to the fact that the aforementioned networks normally operate independently of the underlying infrastructure and sit on-top of the physical network on the application layer as a kind of virtual network.

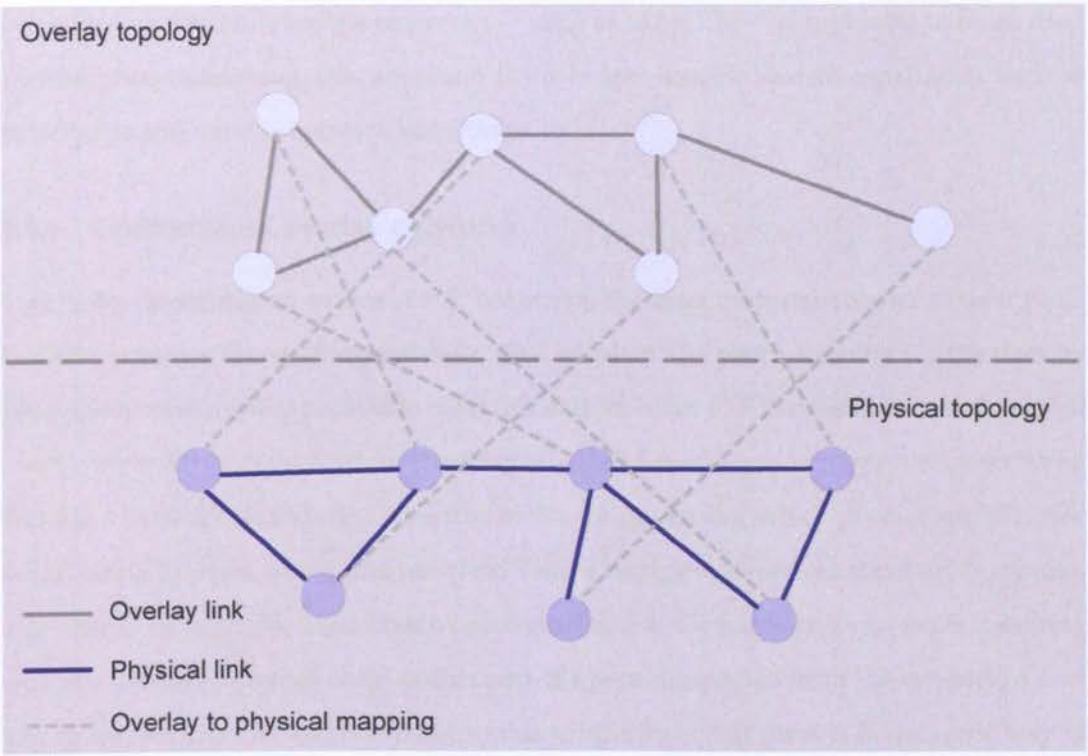

Figure 2.1: Overlay toplogy vs Physical topology issues.

Figure 2.1 shows how current P2P overlay networks do not take the underlying physical network structure into consideration where the darker nodes show the underlying physical network topology, and the lighter nodes show the connections between the corresponding peers in the P2P overlay. The links between each peer in the overlay network can be referred to as virtual links as they do not exist in the physical network. Due to the fact that overlay networks are generally designed for the Internet, they take the physical routing for granted as routes are normally fixed and stable, this means a single overlay hop — that passing of a message from one overlay peer to its direct neighbour in the overlay — often translates to multiple physical hops. Additionally packets can end up traveling over the same hop multiple times. This does not generally hinder large-scale networks such as the Internet simply because of the vast resources available. However, when one

considers smaller-scale mobile networks — such as MANETs — comparable to those used in emergency situations, this approach is no longer feasible due to constraints such as battery life and volatile network conditions.

2.1.1 Unstructured overlay networks

Due to the decentralised nature of P2P networks, the most important aspect to their functionality pertains to searching for data. This involves the peer searching for the data by proactively transmitting packets to meet this end. In older P2P networks such as Kazaa [6] and Gnutella [2] this involved constructing what is known as *unstructured overlay networks*. In such a network no inherent structure exists for governing which peers a specific peer will connect to when joining the network. Thus a completely random topology is created. This allows for a simple algorithm when connecting to the network as no explicit routing table for the overlay needs to be maintained, if a peer disappears from the network, a new one simply replaces it. The downside to this simplicity is that there is no inherent way of knowing the best peer whom to forward a request, as the data is stored in a totally random peer, thus probabilistically any peer in the routing table is just as likely to provide the closest route to the data as any other.

Due to this lack of structure and knowing where a piece of information may reside, unstructured P2P overlays employ flooding techniques to search for data and when a peer wishes to execute a search. Thus, the query is forwarded to all of the peers in its routing table. Said peers will then check their local data repository and if no match is found, will forward the request again to each peer in their routing table. This act of forwarding will continue until either every peer in the network has been contacted or a specified Time-To-Live (TTL) parameter is reached. Clearly the flooding-based approach to searching for data in the P2P network does not scale well when considering increasing network size or increasing search volume [52]. As previously stated one could reduce the overhead — thus increasing scalability — by limiting the radius of the search algorithm within a certain TTL, however this has the obvious side-effect of decreasing the probability of finding the

specific data.

In order to address the above issues, topologies containing two tiers have been proposed such as the FastTrack protocol [53] and the super-peer network based on Gnutella [54]. The reoccurring theme through these works is to enable more powerful peers (known as super-peers) to handle more of the traffic. Such super-peers normally require faster connection speeds and more powerful hardware in combination with a greater uptime. Standard peers will contact a super-peer when joining the network and relay a list of data which the peer is storing locally. The super-peer will then index this data and act as a server for said peer. Thus when a peer wishes to lookup a piece of data in the network, it forwards its request to its super-peer, the super-peer then propagates the request among other super-peers. Said super-peers will then check their local index to find if any of the peers for whom they're responsible are storing the data and if so reply to the requester with the destination peer's address.

This super-peer architecture decreases the amount of flooding needed in the network by transferring the load to more powerful peers, however it does not tackle the problem of scalability directly in large unstructured P2P overlay networks, merely delaying the problem until the super-peers become saturated.

2.1.2 Structured overlay networks

In order to address the aforementioned issues relating to scalability and availability in unstructured P2P networks, Distributed Hash Tables (DHTs) were introduced [11–15, 55]. The main premise behind DHTs is the ability to route a packet containing a key to look up the value for to the destination peer whom is currently responsible for said key — so called indirect or key-based routing procedures. In order to achieve this DHTs form virtual overlay networks on the application layer. The virtual network consists of an overlay identifier space — such as 128 bit IDs. Each peer is assigned a unique ID within this space for example by hashing their Internet Protocol (IP) address. Conversely each object to be stored in the DHT will be given an ID in the same space, by hashing some metadata related

to the object such as the file name, or in the case of P2PSIP, the persons name. A peer will be said to be responsible for an object if said peer's ID is the closest among all the peers to the object's ID. When one refer to closeness in the DHT ID space, this refers to the numerical value of the ID, or in some cases Euclidean distance depending on the implementation.

As mentioned previously, unstructured overlay networks randomly choose the peers in their routing table and hence have no structured routing. Structured P2P networks on the other hand use a very structured routing method by not choosing routing table entries at random. One of the main advantages of this approach is that it allows for a guarantee that if the data exists in the network, it will be found. This is contrast to most unstructured P2P overlays where the lookup TTL may mean data in the farthest reaches of the network is not found. In structured P2P networks each routing table entry must meet some criteria as specified by the implementation. This inherent structure allows such P2P networks to impose an upper bound on the number of overlay hops needed to find the peer responsible for any object ID in the network. Most conventional DHTs have an upper bound of $O(\log N)$ routing hops where N is the total number of peers in the network [11–13, 15]. This upper bound is achieved differently depending on the implementation and the specific routing algorithm employed. Such algorithms include reducing the Euclidean distance in the P2P overlay ID space [14], halving the numerical distance in the overlay ID space with each hop [12], or increasing the length of the matching prefix of the targeted peer ID with each routing hop [11, 13, 15]. The upper bound on the number of routing hops to find an object can be seen as an advantage compared with unstructured overlay networks, however this comes at the cost of an increase in overlay traffic, as the overlay structure needs to be proactively maintained.

An example DHT entitled Pastry [13] will now be examined in order to have a clear understanding of the concepts behind how conventional DHTs operate. Pastry is chosen due to its wide proliferation in research and also as it maintains the same routing and maintenance functions of OpenDHT [11] which was based on Pastry. OpenDHT is used as the basis for Reliable Overlay Based Utilisation of Services and Topology (ROBUST)

DHT which is proposed in Chapter 3. The replication maintenance functions for Proactive MANET DHT (PMDHT) and Reactive MANET Peer-to-Peer Overlay (RMP2P) which are proposed in Chapter 4 are also based on Pastry.

Pastry uses a 128-bit overlay ID space which can be represented as a modulo ring so that all overlay IDs range from 0 to $2^{128}-1$. The overlay IDs in Pastry are normally assigned as digits of base 2^b where b is typically 4 as hex digits are normally used to represent the ID. A peer is responsible for an object ID in Pastry if the peer's ID is numerically closest to the object ID in the overlay ID space out of all of the peers in the network.

A Pastry peer will utilise two routing tables in order to efficiently route to any peer ID destination in the network. These two routing tables are known as the *leaf set* and the *routing table*. The routing table in pastry contains $\log 2^b N$ rows and $2^b - 1$ columns. An entry in the routing table consists of the peer's overlay ID along with its IP address. If row r in the routing table is taken as an example, each entry would have an overlay ID which has a matching ID prefix length of $r-1$ with the overlay ID of the local node. Consequently this means the entry in the first row of the routing table need share no matching prefix with the local node, entries in the second row would share the first digit with the local peer in their ID, the third row would share two digits with the local peer ID and so on.

As described above each entry in the Pastry routing table contains $2^b - 1$ columns. This correlates to one column for each possible overlay ID digit, thus in the case of a hexadecimal overlay ID the columns would consist of first entries where the matching prefixes would be $0, 1, 2, 3, ..., D, E, F$. Taking into account the already mentioned fact that each entry in row r shares $r - 1$ digits with the local peer, the value of the rth digit is deduced from its column index c where $0 <= c <= 2^b - 1$. The local peer will therefore adhere to this criteria for its own rth digit and thus is the reason for the Pastry routing table containing $2^b - 1$ entries at most.

The leaf set in Pastry contains l entries where l is implementation specific but usually set as 2^b. The leaf set contains two sets of peers. The first $l/2$ peers correspond to the peers whose overlay ID is the numerically closest but less than the local peer's ID. The second

set of $l/2$ peers corresponds to those peers whom are the numerically closest to the local peer's ID but more the local peer's said ID. Thus the peers in the leaf set are commonly referred to as the peers whom are *left* and *right* neighbours of the local peer in the overlay ID ring.

Overlay ID: 71654340 Pastry routing table

0	1	2	3	4	5	6	7
03757200	18573428	27573730	35322514	42657372	57146724	62734627	
70573520		72563512	73658264	74637273	75364626	76247637	77235747
71054305	71143520	71257363	71356357	71462537	71521367		71737463
71603251	71613310	71626573		71645263		71663673	71673276
71650573	71651637	71652642			71655632	71656372	
71654044		71654264			71654536	71654637	
		71654324				71654360	

Pastry leaf set

Smaller	71654037	71654124	71654233	71654326
Larger	71654440	71655101	71657402	71657754

Figure 2.2: An example Pastry routing table and leaf set with an overlay ID of base 8 where the leaf set contains 2^b entries and $b = 3$.

Routing in Pastry is based on the Plaxton prefix routing [56] algorithm. The main aim of the Pastry routing algorithm is to increase the length of the matching prefix of the next hop peer when compared with the object ID in the request packet. Ideally the matching part of the prefix will be increased by at least one digit per overlay routing hop, however where this is not possible the algorithm will try to reduce the numerical distance between the next hop and the object ID being requested. Thus the first thing each peer does when it receives a request packet is check if the object is stored locally — if the peer is closer to

the object ID than any of its leaf set peers. This would therefore equate that the peer is responsible for the object being that the peer's leaf set is correct and up-to-date. If however one of the peers in its leaf set is closer than any of the peers in its Pastry routing table, the peer will forward the object request on to that peer. If none of the peers in said peer's leaf set is responsible for the object the peer will consult its routing table. Firstly the peer calculates the length of the object ID which matches its own peer ID. Using this information the peer can establish which row to select from its routing table to select the next hop. As an example if the object ID and the local peer ID's first digits do not match, the first routing table row will be selected, if the first two digits match the third row will be selected and so forth. An example of a Pastry routing table and leaf set can be seen in Figure 2.2 where the overlay ID is base 8, the leaf set contains 2^b entries and $b = 3$. The greyed-out entries are where the selected entry would match with the local peer.

Once the routing table row to select from has been established, the peer then selects the column number where the object ID digit immediately follows the matching prefix — the matching part of the IDs between the local peer and object. If such an entry exists in the routing table the peer will then forward the object request to that peer, thereby increasing the matching part of the prefix by one digit with one overlay hop. If however no such entry exists the peer will simply forward the object request to the peer with the closest matching peer ID to the object ID from both its leaf set and routing table. Subsequently this behaviour will be repeated at every intermediate peer until the request reaches the peer responsible for the object. One can clearly see that this creates an expected number of overlay hops in Pastry of $O(log_b N)$ as in the worst case Pastry routing would be completely dependant on leaf set peers if no routing table entries ever matched.

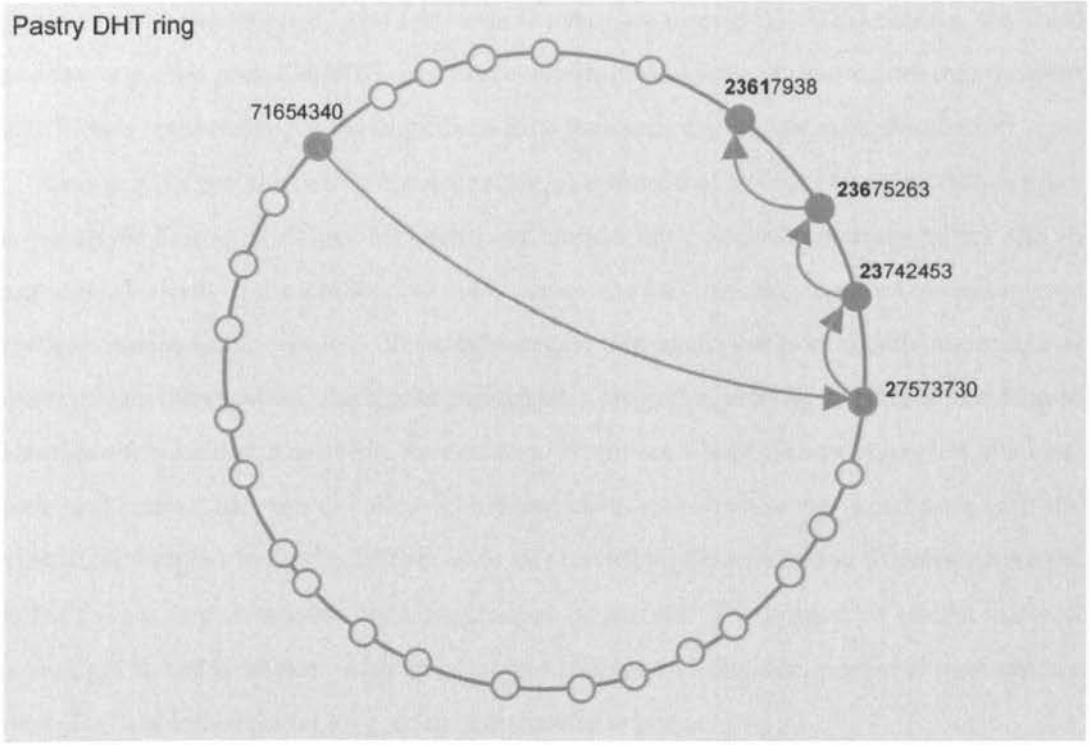

Figure 2.3: An example of Pastry's DHT overlay routing.

Figure 2.3 shows an example Pastry overlay ring where an object has been requested and the subsequent routing process. In this example the peer with an ID of 71654340 is requesting an object with the ID 23616070. As the peer 71654340 does not share the first digit with the object ID no prefix matching will occur. Therefore the peer consults the first row of its routing table and selects the peer which has a matching first digit with the key, this peer is 27573730. Thus the peer 71654340 sends a DHT packet to 27573730 requesting the object with the ID 23616070 this is the first overlay hop. Upon receiving the packet the peer 27573730 consults the second row if its routing table to find a peer which matches the first two digits of the object ID (assuming the responsible peer is not in it's leaf set). Thus peer 27573730 forwards the object request on to peer 23742453 equating the second overlay hop. Next the peer 23742453 consults its routing table for a peer matching the first

three digits of the object ID and forwards the request to peer 23675263 making the third overlay hop. The peer 23675263 then first consults its leaf set and determines that the peer 23617938 is responsible for the object and thus forwards the packet to its destination.

Taking all of the above into consideration, it is clear that in order to satisfy the overlay hops upper bound of $O(log_b\ N)$ each peer should have accurate routing tables and in particular leaf sets, if the leaf sets are not accurate, the DHT routing may not even converge and the overlay will be useless. In order to negate this, each peer periodically maintains its routing state information. Each peer periodically pings the peers in its leaf set to maintain whether the peer is still active in the network. Upon realising that a peer has left, the local peer will contact the numerically highest leaf in its set — when the failed peer is in the numerically higher half of its leaf set — or its numerically lowest leaf in the case where the failed peer is in the numerically lower half of its leaf set. The highest or lowest leaf will then reply to the local peer with its complete leaf set and the local peer will then replace the failed leaf in its leaf set with an appropriate new peer.

Periodically a Pastry peer will also query a random peer from its leaf set and request the leaf peer to send its complete leaf set to the local peer. The local peer will then inspect the remote peers leaf set and determine if there are any peers which have joined or parted the network which the local peer is unaware of and fall within the local peers own leaf set. In addition to maintaining leaf sets, each peer also needs to maintain its Pastry routing table. Thus at a periodic interval each peer will select a random peer from each row of its routing table and request the corresponding row from the routing table of the remote peer. When the local peer receives the routing row from the remote peer, it will then contact both the entries in the remote peers row, and the peers corresponding to same same row in its own routing table. Firstly this allows the local peer to check whether the peers in this specific routing table row are still active in the network and allows the local peer to learn about new peers which could possibly be inserted into its routing table in order to fill empty slots or replace failed peers. Secondly this allows the local peer to optimise its routing table with respect to the underlying network. As if the local peer finds two peers which could

fill the same slot, the local peer can compare both peers via a given network metric — such as latency or distance in hops — and insert the most preferential one into its routing table. This process is known as Proximity Neighbour Selection (PNS), it's important to note that this does not apply to leaf sets [11,13].

As previously mentioned OpenDHT [11] (also known as Bamboo DHT) is used as the basis for this thesis. OpenDHT is based on Pastry, and operates exactly the same other than in the following ways described below.

OpenDHT uses periodic recovery as opposed to the reactive recovery used in Pastry. Reactive recovery refers to the way a peer reacts to the loss of one of its leaf set peers or discovery of a new peer which should be in its leaf set by sending a copy of its new leaf set to every peer contained in it. Periodic recovery on the other hand refers to the process whereby peers periodically choose a peer from their leaf set with which to share said leaf set with. The peer with which the local peer has shared its leaf set will then respond in kind with its own leaf set. This change saves bandwidth while maintaining leaf set convergence of $O(logk)$ phases where k is the size of the leaf set.

The next difference of OpenDHT compared to Pastry is when computing peer timeouts (when to consider a peer has left the network based on how long it has failed to reply). In Pastry this is set as a configuration parameter. While in OpenDHT they follow the strategy of early Transmission Control Protocol (TCP) work [57] where each peer maintains an exponentially weighted mean and variance of the response time for each of its leaf set peers. The Round-trip Timeout (RTO) for each neighbour is thus computed as $RTO = AVG + 4 \cdot VAR$ where AVG is the observed average RTO and VAR is the mean variance of that time.

Additionally OpenDHT also makes improvements over Pastry in the area of PNS. This is achieved by using *global sampling* whereby the lookup function of the DHT is used to find new neighbours for the OpenDHT routing table. For a routing table entry which requires a neighbour with prefix p a random lookup with prefix p is performed. They state that the peer returned by the lookup will almost always have the desired prefix to satisfy the entry

of the OpenDHT routing table.

The results from [11] show that OpenDHT is able to handle higher rates of churn — that is peers randomly joining and leaving the network — than Pastry on which is was based, and Chord DHT [12]. This makes OpenDHT a good candidate for a basis of optimisation for MANETs as mobility and packet loss can cause high rates of what would appear to be churn in the network. The justification for using this DHT as a basis for this thesis also extends to Pastry. Generally, the modifications of DHTs in this thesis are practically independent of any specific DHT. However Pastry has been used as the basis for a large number of DHT-based applications and hence has been studied and analysed in countless research papers [11,21,58–62]. This makes it one of the most throughly examined DHTs which have been proposed.

2.2 Mobile ad-hoc networks

MANETs can be defined as mobile devices — such as smartphones, PDAs, notebooks and sensors — communicating amongst each other in an ad-hoc manner, usually using the Institute of Electrical and Electronics Engineers (IEEE) 802.11 WiFi standard. Such devices can create a decentralised network spontaneously without the need for any fixed infrastructure. Due to the fact that no infrastructure exists, coupled with the fact that MANETs have a highly dynamic topology (nodes are constantly moving in some scenarios) means that the most interesting challenge is that of routing. Routing determines how a packet can be sent from the source node to the destination node, along the optimal path, where optimisations can include least energy consumption, lowest latency, lowest jitter or highest throughput. This method of routing is known as unicast.

The area of routing for MANETs has been heavily investigated in literature and it is outside the scope of this thesis to present a comprehensive evaluation of all MANET routing protocols. Thus this section will give an overview of the main concepts behind different MANET routing protocols, focusing on hierarchical routing protocols, followed by flat routing protocols [63].

2.2.1 Hierarchical-based routing protocols

Hierarchical routing protocols try to exploit a specific node behaviour exhibited in certain scenarios, such as where instead of nodes moving around independently of one-another, nodes often move in clusters or are grouped together. Thus changes in topology within a certain group need only be broadcast within that group. Nodes within other groups are not affected by the change so long as they are able to contact any node in the remote group. This is often achieved by electing a specific group head that serves as the gateway to and from the group. The advantage of this technique is that route maintenance overhead can be greatly reduced. As an example, nodes can only maintain routes to nodes within their own group at specific regular intervals. Therefore only the group head will need to maintain routes to the heads of other groups, lowering the overall traffic overhead induced by the protocol. Hierarchical routing protocols can thus scale well to growing network sizes by for example forming larger groups containing smaller groups of nodes to subdivide the maintenance responsibilities.

There are however certain disadvantages to hierarchical routing protocols. For example due to the fact that group heads serve as gateways to and from their local group, consequently they will need to handle a larger load of traffic than regular nodes, thus depleting their energy resources more quickly. Therefore the group heads can become bottle necks to the routing process. Another issue can arise from the organisation and maintenance of the groups themselves. For instance due to mobility nodes may not always stay within the same group, as the nodes leave and join new groups, the hierarchy will need to constantly adapt in order to account for the changing topology. The worst case of this occurs when a node acting as a group head changes group which could trigger cascading group reorganisations. One of the most well known examples of a hierarchical MANET routing protocol is Hierarchical State Routing (HSR) [64]. Further reading on hierarchical routing protocols can be found in [65].

2.2.2 Flat routing protocols

Flat routing protocols expect that all nodes are able to have mobility, thus all nodes are able to behave equally — able to perform routing requests, as well as forward data. Flat routing protocols can be further subdivided into proactive and reactive routing protocols.

In the case of proactive routing protocols, state information about routes is updated periodically even if a specific route is not active. Link state based proactive routing protocols are quite common. Thus nodes periodically update their neighbours about the state of their current routes in order to inform the network of mobility changes and link failures. This gives proactive routing protocols the advantage of not having to wait for a route to be found before transmitting data, therefore lowering latency times. One can see a great advantage in proactive routing protocols in scenarios where packets are often sent to different destinations rather than only using a few routes. The obvious downside to proactive routing protocols is the constant maintenance traffic needed to keep network routes up-to-date. A large portion of the traffic in a proactive routing network can be the overhead caused by the routing protocol, this can become troublesome and lead to collisions with actual data packets. The fact that routes are maintained even when not in use means that proactive networks are not really suited to streaming scenarios where nodes may be constantly transmitting to just a few nodes which do not change. Proactive routing protocols could also be seen as wasting precious energy resources on routes that are never used and may be too heavy for low powered nodes such as sensors due to the impact on their limited energy resources. Popular proactive routing protocols for MANETs include Optimized Link State Routing (OLSR) [42], Optimized Link State Routing version 2 (OLSRv2) [47] and Destination-Sequenced Distance-Vector routing (DSDV) [44].

As OLSR [42] and its successor OLSRv2 [47] are a basis for this thesis their general concepts will be outlined in this section. OLSR is one of the most popular proactive routing protocols in literature and used in test-beds. The basis for OLSR is the conventional link state algorithm which has been tailored based on the requirements of MANETs. The main innovation of OLSR over conventional link state routing is the concept of a message

flooding reduction technique known as MultiPoint Relays (MPRs). The MPR set of each node is the minimum symmetrically connected 1-hop nodes that can symmetrically connect the local node to all 2-hop neighbours. Every node in the network will periodically flood HELLO messages to its neighbours within a certain radius in order to establish the MPR sets. Topology Control (TC) messages are then used to flood the routing information network wide, these messages are only flooded in the network via the designated MPR nodes in the network therefore optimising the flooding mechanism. Each node in the network will therefore receive this updated network information at periodic time intervals and thus update its routing state information while being able to compute routes to all possible destinations in the network. Additionally only MPR nodes generate link state messages thus further reducing the network routing overhead generated by the protocol. The OLSR routing protocol has been designed to operate independently of the lower layers. As previously mentioned OLSR is particularly well suited to scenarios where nodes are transmitting to a variety of different destination nodes often as no extra overhead is incurred for using many routes concurrently.

Work is currently being carried out on the next generation OLSR routing protocol by the Internet Engineering Task Force (IETF) MANET working group. The new updated routing protocol is entitled OLSRv2 [47]. OLSRv2 uses the same fundamental mechanics as OLSR, however OLSRv2 uses a more flexible and efficient method for control packet distribution while exchanging more simplified messages. OLSRv2 makes use of and extends NeighborHood Discovery Protocol (NHDP) [66] for neighbourhood discovery and uses the standardised message format to make improvements over OLSR. NHDP is further extended in OLSRv2 by adding MPR address block Type Length Values (TLVs) which contain the MPR selection of nodes in addition to the willingness metric for nodes to become MPR nodes. Using the willingness metric nodes can decline to become MPR nodes while still participating in the network as routers, source and destinations.

Reactive MANET routing protocols are another popular technique for routing in MANETs. Also known as on-demand routing protocols, routes between a source and destination are

established only when requested by the source node in order to transmit data to the destination. This has the obvious advantage of generating less overhead in the network, however this comes at the expense of added latency when a node is required to establish a route before being able to transmit data. The Dynamic Source Routing (DSR) routing protocol [43] is a well known reactive routing protocol for MANETs that uses route discovery and maintenance on-demand in order to route from source to destination. One advantage of DSR is that it maintains multiple routes to a destination which allows it to select a preferred route dependant on implementation, useful for load balancing or increasing fault tolerance.

The Ad hoc On demand Distance Vector (AODV) routing protocol is widely examined in literature and one of the well known routing protocols for MANETs. Like DSR, AODV utilises on-demand route discovery and maintenance for establishing routes on a unicast basis. AODV is based on a modified version of the Bellman-Ford algorithm [67]. In order to initiate a route discovery, the source node broadcasts a Route Request (RREQ) message. An intermediate node receiving the message will then cache information about the route back to the source node while again flooding the RREQ in the network. If a node receiving the RREQ has a valid route to the destination, they will then reply to the source node with an Route Reply (RREP) message. If however the RREQ makes its way to the destination node, said node will cache the reverse route to the source and reply with an RREQ. Upon receiving this reply the source node can then begin transmission of data through the path established by the RREQ and RREP. Such messages are sent via User Datagram Protocol (UDP) utilising IP header information. A TTL value is applied to the RREQ messages so that the broadcasted message is confined within a certain number of hops, as to not flood the entire network, limiting overhead. If a route is not discovered within a certain timeout, the TTL will be increased and the RREQ message will be sent again. Stored routes are valid for a certain time period and once expired, route discovery should be initiated once again. The timeout period for a route is reset each time data is successfully transmitted over said route. A Route Error (RERR) message is utilised in order to notify other nodes that a specific route has failed and that the destination is no longer reachable. The main

difference between AODV and DSR is that AODV acts on a hop-by-hop basis, where as DSR operates by utilising source routing (each node knows the whole route).

The Ad-hoc On-demand Distance Vector version 2 (AODVv2) routing protocol is the second generation of the original AODV routing protocol and is a work in progress in the IETF MANET working group. The route discovery and route maintenance processes of AODVv2 have remained essentially the same as in AODV. AODVv2 is rather suitable for use in scenarios where MANETs exhibit high mobility behaviour and traffic patterns. By establishing routes on-demand it is more suitable for networks where a few routes are used heavily. Thus, operating in this method requires little processing time from the Central Processing Unit (CPU) of the device. AODVv2 differs from AODV in that it utilises NHDP to establish bi-directional routes, representing an advantage over AODV. AODVv2 also benefits from the same standard message format of OLSRv2 as established by the IETF MANET working group, thus creating simpler messages and reducing overhead. AODVv2 also adds support for local route repairing and can accept improved routes even after the initial establishment of a route.

Concepts from hierarchical and flat routing protocols can also be combined. Such is the case the the Zone Routing Protocol (ZRP) [68] which is a hybrid of the two routing behaviours. Nodes in ZRP create a group of nodes of a specific radius (for example 2-hops) nodes within this radius then use proactive routing within the group. Thus these nodes then employ reactive routing for any node outside its local group. Therefore this protocol diminishes the cost of restructuring the hierarchy every time a node moves and lessens the formation of bottlenecks in the network. However when increasing network size the protocol suffers the same vulnerability as reactive routing protocols in that the number of nodes outside the group is increasing and the majority of nodes therefore lie outside the local group — causing more reactive lookups and expensive flooding.

2.3 Topology aware peer-to-peer networks

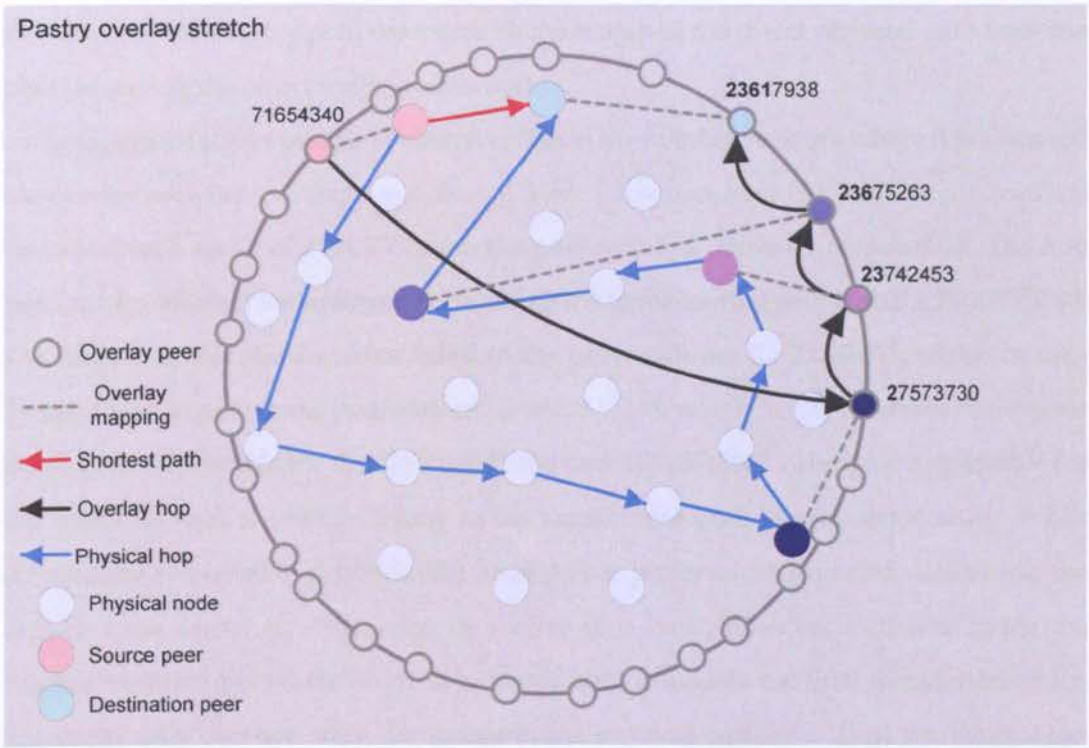

Figure 2.4: An example of overlay stretch in Pastry on MANETs.

An important aspect to consider is that a single overlay hop constitutes a route in the physical network which generally consists of multiple physical hops from the source of a request to the destination. Additionally one must remember that conventional P2P networks were designed for use on the Internet and generally ignore any aspect of the physical network routing. Due to this oversight, P2P networks are generally ignorant to the underlying physical network when constructing the topology of the overlay network. The consequence of this is that for example two neighbours in the overlay network — two nodes who have a numerically close overlay ID — will not generally be close in the physical network, as their overlay ID is generated using a unique parameter such as IP

address. The effect of this — the difference between the physical and overlay routing hops disparity — is referred to as *overlay stretch*. In other words, this refers to the ratio between the length of the overlay path compared to the length of the direct physical path from the peer requesting the object to the destination.

In Figure 2.4 is an example of stretch acting in the overlay network where it is assumed the overlay network architecture is Pastry. Here the source peer 71654340 which requests the object with an ID of 23616070 from the peer which is currently responsible. The first overlay hop shows the source peer forwarding the request to the peer with the ID 27573730. The request is then further forwarded to the peer with the ID 23742453, which in turn forwards the request to the peer with an ID of 23675263, which finally forwards the request to the peer responsible for the object with the peer ID 23617938. The peer responsible for the object ID then responds directly to the source peer with the requested data. When considering the overlay network, the architecture performs as expected, delivering the request from source to destination in a mere four hops, however in this example the request traverses the whole physical network until it reaches the final destination which is actually only one hop from the source in the physical network. Thus the request has traversed 13 hops in the physical network, where had the overlay known the route prior to the request, could have been reduced to just one hop. In this example the overlay stretch is 13/1 = 2.6. It should be noted that this is an extreme example used to illustrate the effects of overlay stretch. In reality most modern DHTs utilise a method known as PNS, to aid in reducing overlay stretch, which has been examined to be between 1.6 and 2.2 in most cases [69].

Overlay stretch can be acceptable to a degree on the Internet due to fast links and abundance of fixed networks which will maintain constant power. However the issue can not be overstated in MANETs where every additional hops increases the chances that a packet will have a delivery failure and need to be retransmitted causing additional latency and load in the network. Thus several approaches have been studied in the literature to decrease the overlay stretch in DHTs.

One of the popular approaches proposed in [70] is known as *landmarking*. This approach involves a specific set of peers known to all peers in the network as landmark peers. These peers periodically measure the distance in hops between each other — by using a simple lightweight message such as a ping — without reliance on expensive positioning equipment such as Global Positioning System (GPS). Each peer will then order the landmark peers in terms of distance from them. Logically peers which are close to the same landmark peer should consequently be physically close in the network. Thus peers close to the same landmark peer are assigned an overlay ID with a prefix similar to the landmark peer.

The authors in [36] propose another approach. In this architecture a peer joining the overlay calculates its overlay peer ID by measuring the distance to its closest physical neighbours in the overlay network. The distance between the local peer and its neighbours is then used to create what are known as *virtual springs* between the peers. The energy value then needed in order to maintain the minimum energy state for the joining peer within that set of virtual springs is then used to calculate the joining peers overlay ID. This approach may have a disadvantage however in MANETs due to the fact that physical neighbours can change frequently when mobility is inherent, therefore creating the need for more overlay ID regenerations in order to maintain the system.

2.4 Summary

In order to address the inefficiencies and scalability issues of flooding-based unstructured P2P networks, DHTs have been proposed and studied heavily in the literature. The aforementioned DHTs create structure in the network in order to make requesting data more efficient. Due however to the fact that conventional P2P networks are largely unaware of the topology of the underlying physical network, the overall route length taken in order to converge a single overlay request can be significantly longer than the direct route from the source of the request to the destination. In order to address this problem a number of approaches have been developed to reduce the previously discussed overlay stretch issue by mapping physical proximity in the overlay ID space. The method of using landmark peers

as previously examined, can significantly lower the overlay stretch which occurs in the P2P networks of fixed or optimal networks. When deploying these approaches on top of MANETs however their apparent efficiency is no longer witnessed and the lookup success rate drops to unacceptably low levels due to factors such as MANET route discovery and packet collisions caused by the overhead of running two similar networks. Thus one can deduce that deploying conventional P2P networks on top of MANETs does not produce encouraging results, even when considering those P2P networks which optimise using PNS. Therefore this thesis will look at developing further optimisations to P2P networks in order to provide a reliable and efficient key-based overlay lookup architecture for specific scenarios taking into account, mobility and heterogeneity.

Chapter 3

Related work

The area of optimising Peer-to-Peer (P2P) overlay architectures for Mobile Ad hoc Networks (MANETs) has attracted a lot of interest in the research community in the last decade. This chapter will first give an overview of the literature behind the concepts used as the building blocks for optimising P2P networks for MANETs followed by literature directly related to the P2P overlay architectures proposed in this thesis.

3.1 Distributed hash tables

Conventional Distributed Hash Table (DHT) networks provide a stable and solid basis for their optimisation on MANETs. One could therefore simply run such networks on top of MANETs due to their successful proliferation on the Internet. As DHTs are simply a form of storage, they would then provide their services to the relevant applications without any further modification. As previously discussed however, conventional DHTs designed for the Internet are not well suited to be used directly on MANETs for the following reasons found during the course of developing this thesis:

- DHTs designed for the Internet may present the case where two neighbours in the DHT are physically far from each other, and hence are multiple hops apart in the physical network. This creates the situation whereby overlay routes incur unnec-

essarily long physical routes resulting in greater overlay stretch. As shown in the previous chapter, some architectures have been proposed to further optimise this issue on DHTs designed for the Internet however they do not go far enough in order to meet the strict demands of MANETs whereby due to transmission errors and packet collisions the probability of successful packet delivery can drastically decrease with each added physical hop.

- Due to mobility (sometimes high) in MANETs, their behaviour can be characterised as somewhat volatile and physical routes can change quickly. This creates a problem for conventional DHTs which take for granted the underlying links in the network, as they expect links to be static and always available. As an example there could easily be a situation where an overlay look up transverses over two overlay hops, thus if the routes are not already known to these peers, two route requests will need to be executed, as route requests in reactive MANET routing protocols are broadcast, the look up may have been more efficient if the peer had just broadcast the original request.

- As previously mentioned, in order for DHTs to maintain consistency and allow object look ups to execute successfully, DHT peers have to periodically carry out DHT routing table maintenance. As neighbours may not be physically close to each other, this can create a significant amount of extra overhead traffic in the network, further compounding the issue of packet loss in MANETs due to more possibility of transmission errors and packet collisions.

For these reasons one can come to the conclusion that DHTs which do not have forethought with regards to their optimisation for MANETs will not perform well when simply used unmodified on top of MANETs. Further evidence is provided for this in a later chapter of this thesis.

3.2 Unstructured peer-to-peer networks for MANETs

3.2.1 Flooding-based peer-to-peer networks

Previous work on optimising P2P overlay networks for MANETs had focussed on simply converting conventional unstructured P2P networks to be more efficient on MANETs by combining MANET routing and the unstructured flooding mechanism of the P2P overlay network in order to locate objects. In this section two such architectures are examined in detail.

First the P2P file sharing protocol for MANETs entitled ORION as proposed by [71] is investigated. ORION proposes to combine a Gnutella-based [2] file sharing network with the MANET routing protocol Ad hoc On demand Distance Vector (AODV), in order to locate files in the P2P network. In ORION each peer shares their own locally stored repository of files in the network. Thus when a peer then wants to locate any file, the request is broadcast throughout the P2P network. Upon receiving a request for a file, the peer will first check if the specific search string is matched by any of the files in its local repository. If any match is found the peer will then reply to the source via the reverse route in the same way as AODV sends a Route Reply (RREP) message. All peers along the reverse route will then cache the response from the peer storing the files, thus making searches quicker in the future. Once the source peer receives the message containing the list of files from the peer responding to the search, the source peer will then send out a download request to said peer in order to receive those files. The message related to download the files will be send over the previous path found to the peer. The peer receiving the download request and storing the files, will then split the files into chunks and send the data over to the source peer. If no match is found the peer will first store the reverse route to the requestor — just as AODV does when an Route Request (RREQ) is received by a node — and then forward the message on to its neighbours.

The authors in [72] propose the Mobile Peer-to-peer Protocol (MPP) protocol which is proposed to facilitate file sharing in MANETs. The protocol is similar to the aforementioned

ORION protocol in that central to the protocol in MPP is the MANET routing protocol known as Enhanced Dynamic Source Routing (EDSR) which combines Gnutella based flooding with the MANET routing protocol Dynamic Source Routing (DSR) [43]. Thus when a peer wishes to request a file, the request is flooded throughout the MANET network. The cross-layer nature of MPP means that when a search request is received by a peer, MPP will forward the request to the local application to see if the request matches any files. Any intermediate node adds its own address to the request packet thus creating DSR type routes, then retransmitting the search to its neighbours. If however the peer matches some files with the request, said peer will then reply directly to the source using the reverse route information contained in the search request. Once the reply has been received by the source it will proceed to download the file directly from the peer storing the files via Hypertext Transfer Protocol (HTTP) using EDSR which routes the packets normally akin to DSR.

In [73] the authors first test the existing Gnutella [2] protocol directly on top of the MANET routing protocol AODV. They discover that the cost of flooding the network with the search requests is quite high, thus increasing dramatically the overall percentage of dropped packets in the network, however they did find some advantages to the flooding such as creating routes which had previously not been looked up, therefore negating the need to send a RREQ if these routed are then used in the future, thus increasing resilience. Their proposed improvements to the Gnutella protocol in order to be more optimised for MANETs involves borrowing concepts from the classic MANET routing algorithm known as Gossiping such as [74]. In Gossiping, each node will forward requests to its neighbours based on a pre-defined probability, said to reduce energy consumption and network load. The architecture proposes an adaptive mechanism based on network load, whereby the forwarding probability for a given neighbour is calculated as $p(1 - u)$ where p is a fixed value and $u(0 <= u <= 1)$ is the utilisation of the neighbours queue. This process is then repeated at every peer which receives the query. This allows peers to send more messages to neighbours with a lower load while sending less messages to saturated nodes. While

the approach offers a more load balanced P2P network, it comes at the disadvantage of less discovery in the search process, as due to the selection of which peers to forward queries to some peers containing the searched for information may be overlooked.

The above described architectures which combine unstructured flooding-based P2P overlay architectures with reactive MANET routing protocols is a straight forward and simple approach. Minimal changes are needed to the underlying routing protocols as they already have the capacity to transmit broadcast messages and reply directly given the response. The data actually being broadcast is of little importance to the routing protocol, as their design is fundamentally just about finding a route to a destination. The downside to such approaches however is their inability to scale well. Clearly in larger networks broadcast-style messages combined with more frequent requests will cause an over-abundance of messages in the network leading to ever more packet collisions and transmit failures, however this could be negated by using heavy caching of P2P lookups and piggy-backing of P2P messages on routing lookups, something that will be further investigated later in this thesis.

3.2.2 Proactive search for mobile peer-to-peer networks

The authors in [75] describe the Zone-based Peer-to-Peer (ZP2P) protocol aimed at reducing the heavy load of broadcasting flood-based search requests. ZP2P is based on the concept of local zones, where each peer n is at the centre of its local zone. The radius of the zone is implementation specific and based on a distance in hops h. Thus all nodes which are within h hops of n are said to be within its zone. When a peer joins the network it sends an initial message advertising the information that the new peer is sharing in the network. Such a message will be broadcast within its h hops zone radius. Each one-hop neighbour of the peer will then reply with their own advertising information so that the peer can learn about information they are sharing in the network. Additionally one-hop neighbours also send messages regarding the information which more remote peers in the zone are sharing which the neighbour has previously received. Thus all members within a zone will know

about all information currently being shared within that zone. When the information a peer is sharing changes (i.e. due to addition or deletion) the peer will once again send an advertisement to its zone noting the changes.

When a peer wishes to find an object in the P2P network, first it will check its local cache to see if the information is being shared by a peer within its local zone. If a match is found the peer will contact the remote peer storing the information directly and request to download it. If however the information can not be found within its own zone, the peer will forward the request to the peers on the border of its zone radius h hops away, this process is known as a *bordercast* message. These peers will first check their own local repository for a match with the search query. If no match is found locally they will check their local zone cache to determine if any of the peers in their local zone contain the requested information, and if so forward them the request. In order to maintain a route back to the source peer, peers keep track of the last-hop (known as the reverse route) from which the message have been received, much like AODV. If however the information is not found in this peers zone, it will then again forward the request in a *bordercast* message to the peers on the border of its zone. This process will then be repeated until either a specific Time-To-Live (TTL) is reached or the whole network has been searched.

The concept of local zones may render some network wide broadcasts unnecessary, however the probability of the requested information being in a nearby zone is completely random. This is clearly the case in larger networks, where the chance for a request to be satisfied locally can be seen to be rare. Thus the usage of local zones does not scale well to larger networks. The notion of sending requests via the bordercast message can lower overall traffic as it acts as a kind of indexing system whereby the peers inside the zone need not know of the request. However in terms of scalability one can see that as the network size increases, and hence the number of zones increases, the number of zones needed to contact will cause issues in the network due to the volume of messages. Also the efficiency of the protocol can be dependant on the specifics of zone radius and density. Where a zone has a low density the number of physical hops needed to get from one side to the other

obviously increases, this has the effect of creating many more route requests in the network, and negates the supposed advantage of no longer needing flooding in the network. In such networks this fact combined with the additional regular advertisement messages may even create more messages than a traditional flooding-based protocol. Further adding to these issues are the matter of mobility, thus when a peers zone changes due to peer movement, advertisements will need to be reissued in order for the new peers to adhere to the protocol.

3.3 Structured peer-to-peer networks for MANETs

3.3.1 Pastry-based peer-to-peer networks

Ekta [60,76] integrates Pastry DHT functionality with DSR MANET routing at the network layer in order to create a DHT substrate specifically designed for MANETs.

Ekta peers still maintain the conventional Pastry routing table and leaf set. However instead of containing the normal DHT overlay ID along with the Internet Protocol (IP) address in the routing table, Ekta opts to store the DHT overlay ID along with the DSR style source route from the local peer to the destination. Upon a new peer p joining the network, it first assigns itself a unique overlay ID, then it contacts its bootstrap peer and requests it to route a lookup towards p's overlay ID. When the join request is received by the node x which has the closest peer ID to p, x then sends a message containing its leaf set to p. However unlike conventional Pastry, Ekta will not send any routing table information in its messages in order to decrease overhead in the network. The peer x will also broadcast a message to the peers in its leaf set to notify them of the new peer p joining the network.

The key-based routing mechanism in Ekta is the same as that in conventional Pastry. Thus when a peer p wishes to send a packet requesting object k, it first consults its Pastry routing table or leaf set for the peer which has the longest common prefix matching with k in its peer ID, or if none exists, the peer with the closest numerically matching ID to k's ID. Once the destination peer has been determined the source route stored in the Ekta routing table is used to send the object request directly to said node. This process is completed at

each intermediate peer until the packet reaches the peer whom is currently responsible for k.

If however peer p selects the next hop for a destination peer x and the physical route no longer exists, Ekta will behave in one of two ways:

1. If the peer was selected from p's leaf set, then no alternative peer to the destination will exists, and therefore a normal DSR route discovery procedure need to take place to find the route from p to x.

2. If the peer was selected from p's Pastry routing table, then p need not discover a route directly to x, instead p can select a route from its Pastry routing table which fulfils the same slot in the routing table entry as x, in said situation Ekta will then initiate a prefix-based route discovery for such a peer — unfortunately exactly how such a prefix-based route discovery process should be carried out is not described in Ekta, however any such method for doing so would generate considerable overhead in the network as either a network wide DSR route request would need to be initiated, or something akin to AODV's expanding ring search, both generating flooding in the physical network.

Ekta additionally tries to optimise overhead caused, by utilising overheard packets. Ekta will proactively update its routing table and source routes based on information sourced from packets overhead from its neighbours. Thus where possible Ekta will not engage in heavy Pastry routing table maintenance but instead when Ekta overhears a packet, it will use the overlay ID and source route contained in said packet to replace less optimal routes — in terms of number of hops to the destination — in the routing table, and update source routes. Thus Ekta significantly reduces the maintenance overhead observed when using Pastry directly on top of MANETs.

As mentioned Ekta integrates Pastry's key-based routing method with the MANET routing protocol DSR at the routing layer. The simulation results found in [60,76] show that this method significantly improves performance when compared to simply using

Pastry on top of a MANET routing protocol. However Ekta omits one important aspect of optimising DHTs for MANETs, that of physical location awareness. Thus in Ekta there is no correlation between physical and overlay proximity, and like Pastry, peers which are neighbours in the virtual overlay may be physically far apart. This can have the effect of increasing the overhead of functions such as neighbour data replication due to the data having to travel on physically long paths. Ekta tries to overcome this issue by overhearing packets and replacing suboptimal routes with those based on the information from the overheard packets. This issue is also further compounded by the fact that Ekta peers need to periodically communicate with their leaf sets in order to establish that they're still participating in the overlay network, due to the aforementioned physically long routes, these important maintenance messages may be lost. Another issue is that in larger networks the object ID lookup process could experience the effect of traversing the network multiple times before arriving at the destination. This is caused by a long shared matching prefix with the current peer and the object ID meaning less alternative destinations will exist — a further description of this can be found in [69]. Thus scalability can be said to be an issue in larger networks comprising of Ekta peers.

MADPastry [61] integrates the reactive MANET routing protocol AODV with the application layer Pastry [13] DHT to provide indirect routing functionality at the network layer.

MADPastry uses a concept known as *Random Landmarking* [77] to create clusters of peers in the physical network which share a similar overlay ID prefix. Therefore two peers which are close in the physical network should also be numerically close to each other in the DHT. This means that neighbours in the physical network have a higher chance of being in their physical neighbours leaf set. This is achieved by using a set of Landmark keys. Landmark keys are chosen in order to divide the overlay ID space into equally sized segments for example in hexadecimal ID these keys could be: 0800...000, 1800...000, 2800...000, ..., E800...000, F800...000. Thus to form clusters peers associate themselves with the nearest (determined by hop count) landmark node (the peer

closest to a landmark key which broadcasting its landmark key periodically) and thus adopts the same overlay ID prefix. A peer will adopt a new prefix if it moves closer to another landmark node.

MADPastry maintains three different routing tables, the standard AODV routing table, a stripped down Pastry routing table, and standard Pastry leaf sets. The standard Pastry routing table as described in Section 2.1.2 is used whereby the routing table consists of $log_{2^b} N$ rows with $2^b - 1$ entries for each row. To lower the overhead induced by routing table maintenance traffic MADPastry uses a degenerative Pastry routing table. The MADPastry routing table needs to contain $log_{2^b} K$ rows where K is the number of landmark keys. Thus it only needs enough entries to be able to contact a peer within each of the overlay clusters. The standard Pastry leaf set contains the $L/2$ closest peer IDs larger than the local peer, and the $L/2$ closest peer IDs but smaller than the local peer. In MADPastry a peer only pings its most immediate closest peers in order to reduce overhead, thus efficiency may be reduced in the case where leaf sets are not completely accurate. MADPastry utilises a standard AODV routing table whereby each hop along the route knows the next physical hop of a given route.

Routing in MADPastry is carried out by first checking if the local peer is the destination, by establishing if any peers in the modified Pastry routing table have a closer matching prefix than the local peer. The peer will also check whether any peer in the local peers leaf set is numerically closer than the current peer. If the peer establishes that it's an intermediate peer (a closer matching peer is found) it will determine the next physical hop towards the destination and forward the packet on. If an intermediate peers own overlay ID is numerically closer to the key than the next overlay hop it will intercept the packet and consider itself as having completed said hop. To further reduce overhead MADPastry will broadcast the lookup within the local peer's cluster if a lookup has a matching prefix with the local peer and the physical next hop is unknown.

Routing table maintenance in MADPastry is carried out as mentioned by pinging the immediately closest peers to the local peer but larger and smaller than the local peer. Thus

all other routing entries are gained by overhearing data from other packets. Thus AODV packets also contain the overlay ID of the previous physical hop.

Clearly MADPastry strips out a lot of overhead and in doing so diminishes the efficiency of the overlay, especially in cases where traffic is sparse and not much information is overheard from other packets. In cases where mobility is high, routing table entries to other clusters may be incorrect, as the peers have moved into other clusters, at that point leaf sets will become important and having only the correct two closest numerically closest peers in the overlay to the local peer would severely increase the number of hops. The authors in [61] do not mention the process for replacing the two closest leaf set peers in the case where both leaf set peers were unreachable as this is more likely to happen in MADPastry than Pastry and could result in failed lookups and incorrect leaf sets, severely hampering performance. The authors in [61] also suggest that the protocol would perform better in larger networks due to more chance of overhearing data from other packets, however this could incur greater energy consumption as each peer will need to inspect every packet. Additionally this means that more routes are available and hence, any peer has less chance of being along a given route.

3.3.2 Chord-based peer-to-peer networks

In [78] an architecture entitled Iterative Successor Pointer Rewiring Protocol (ISPRP) is described. ISPRP is aimed at integrating Chord's [12] structured overlay routing and the underlying MANET routing protocol. ISPRP proposes to create a Chord-like overlay ring while deducting the need to an underlying routing protocol in MANETs. The main premise is that each node upon joining the network, issued itself a peer ID in the overlay. Further, each peer then stores the overlay ID of its one hop neighbours, as well as the overlay IDs of its successor and predecessor peers in the DHT, while maintaining DSR style routes to its successor and predecessor peers. To this end, a peer p selects the node s with the smallest numerical overlay ID among its one-hop neighbours while the ID of s at the same time being larger than the peer p's ID. Thus the peer p therefore assumes that its successor peer is

among its one-hop neighbours. Accordingly peer p sends an Successor Pointer Solicitation (SPS) message to peer s containing information to tell s that peer p has selected it as its successor peer. The SPS message also contains the reverse route back to the source node in a DSR style route.

When a peer s receives an SPS message from any peer p it will check its local peer cache to determine if there are any inconsistencies. If no inconsistencies are found, peer s will store peer p as its predecessor peer along with the route to p which is contained within the SPS message. If however peer s does determine that an inconsistency is present, this would have occurred for one of two reasons:

1. It could be that peer s had been storing an incorrect successor peer. This could occur for example if peer s had noticed a peer x contained in the source route to peer p which had an overlay ID smaller than node s's current successor peer's ID but larger than peer s's current ID. If such a scenario occurs then peer x will become peer s's successor peer and peer s will send an SPS message to peer x using the source route from peer p's original message.

2. There is already another peer z pointing to peer s as its successor. Due to the fact that a peer cannot have two predecessors, peer s will determine which of peer p or z should be its predecessor. If one takes the example where z should remain the predecessor of peer s, peer s will then select the peer from its local cache which it believes should be the successor of peer p instead. Peer s will then send an Successor Rewiring Solicitation (SRS) message back to peer p in order to inform it that peer z should be its successor peer, the message will also contain the source route from p to z via s. Concurrently peer s will also send an SPS message directly to peer z on behalf of peer p in order to notify it that peer p should be its predecessor peer, and containing the source route from z to p via s. Whenever a peer receives an SPS or SRS message the peer will first determine if an inconsistency exists and the above process may repeat itself until consistency exists.

ISPRP can therefore create a two-way Chord style overlay ID ring without the need

for an underlying MANET routing protocol. In such a network, packets can therefore be routed based on the destination peers overlay ID instead of using a MANET routing protocol to find a route to a specific IP address. One can notice however that ISPRP leaves certain issues unaddressed. Firstly, it's clear that the source routes contained in the SPS and SRS messages will probably not be the shorted most efficient routes. Thus peers will often store routes to their successor and predecessor peers which are suboptimal routes when compared with MANET routing protocols such as Optimized Link State Routing (OLSR) or AODV. The authors of ISPRP also recognise this inefficiency and state that in order to reduce path lengths, peers will shorten their successor and predecessor paths using Dijkstra's algorithm [79]. However the exact method of how this will be achieved using the limited information known to peers is omitted in the paper. Secondly, due to the fact that peers only know the ID's of their successor and predecessor peers — as well as their random one-hop neighbours — this makes overlay lookups highly inefficient. Subsequently this could in the worst case mean traversing $N/2$ peers — where N is the number of peers in the network — in order for a lookup to converge. This issue is further compounded by the fact that physical proximity is not taken into account. This would therefore mean the protocol suffers from the same issue of overlay stretch as conventional DHT protocols, where a single lookup could traverse the physical network multiple times, further aggravated by the fact that as previously mentioned, routes are suboptimal. The biggest oversight however of ISPRP is the fact that mobility has been largely ignored. In MANETs routes often break and packets need to be retransmitted over a new route. ISPRP however does not mention how to detect route failures and how to repair them. Additionally it is not discussed how ISPRP will handle a node failure, and there is no consideration given to this situation, which would result in an incomplete overlay ID ring.

In [78] the authors propose a Scalable Source Routing (SSR) protocol very similar to ISPRP. Upon joining the overlay network each peer acquires the source route to its numerical successor peer. Each peer maintains a cache of routes found by processing the reverse routes and forward routes of packets received. The peer's goal is to add $O(log\ N)$

source routes in its cache to peers whose difference in overlay ID increases exponentially. Similarly to ISPRP, SSR has some disadvantages. Firstly as previously mentioned the acquired source routes from successor updates can be highly inefficient when compared with traditional MANET routing protocols. To tackle this problem peers use the source routes from their routing cache to prune long source routes, such as routes containing loops or where a short-cut may have been found. The effectiveness of this method however is hard to quantify as the outcome depends entirely on the available source routes in the cache, thus there's no guarantee on how well the routes can be pruned. The second disadvantage of SSR is due to the fact that physical proximity is not taken into account. As with ISPRP, an object lookup can traverse the physical network multiple times. Again this is further compounded by the fact that source routes may be unnecessarily long as mentioned with ISPRP, which is clearly a trait not suited to MANETs.

Another popular DHT-based architecture for MANETs is proposed in [80] known as Virtual Ring Routing (VRR). VRR works in a similar way to SSR. In VRR each peer maintains an AODV style route — where for a destination, just the next physical hop is stored towards the destination — to each of its v virtual neighbours — the $r/2$ peers with a numerically closest overlay ID but higher than the local peer's, and the $r/2$ peers with a numerically closest overlay ID but lower than the local peer's.

The routing method in VRR comprises of simply looking up the peer in the routing table whose overlay ID is closest to that of the requested object's ID. A packet is then transmitted to the corresponding next hop for that routing table entry. When a peer receives a request it again looks in its routing table for the peer with the closest matching peer ID to the object ID, and again forwards the message. Thus the process is repeated until the peer receiving the object request is the peer with the closest overlay ID to the object and is therefore responsible for said object.

Upon joining the network peer p will send a request to one of its physical neighbours n whom is already currently a member of the overlay network to send a request to the peer currently numerically closest to n's ID in order for n to join the network. The peer closest

to n's overlay ID, peer x will then add peer n to its routing table and reply to peer n with a message which contains the v virtual neighbours for n. On receipt of said message n will then add peer x to its routing table. Thus peer n will finally contact its other $v - 1$ virtual neighbours in order to discover the physical routes to them.

The authors in [80] do — unlike in SSR — discuss thoroughly how to handle link failures. However like SSR, VRR has a few significant disadvantages. Firstly because of the fact that peers setup routes to their virtual neighbours through existing physical neighbours, VRR will suffer from the same suboptimal routes which are inherent in SSR. Additionally there is no correlation between the physical location of a peer and its overlay ID. This creates the same issue whereby an object lookup can traverse the physical network multiple times, again being compounded by suboptimal routes. The impact of this can be seen in the results presented in [80]. Where the network size is over 150 peers, VRR's packet delivery ratio plummets drastically therefore bringing into question the scalability of VRR.

3.4 Hierarchical peer-to-peer networks

Another interesting area to investigate is that of hierarchical P2P networks. These networks generally delegate responsibility of P2P routing lookups starting from a subset of the network and increasing the lookup's reach should the initial set of peers not contain the requested information.

The authors in the paper [81] describe a DHT based hierarchical P2P routing algorithm entitled HIERAS. The HIERAS algorithm is proposed to decrease lookup latency in P2P networks, to achieve this they aim to create several lower level P2P DHT rings which span of the highest level DHT ring. Each lower level ring is a subset of the overall P2P overlay network. HIERAS creates said DHT rings by using a strategy whereby the average link latency is lower for all peers in each common ring. Another property of this algorithm is that latency between peers in lower rings is lower than those in higher rings. The P2P lookups are first executed in the lower level rings before falling back to the higher rings if

the search is not successful. The performance of the algorithm generally depends on much of the lookups completing within the lower rings. The algorithm contains multiple DHT rings which are based on the Chord DHT architecture, each ring is a subset of a larger ring, creating a hierarchical topology. The Chord finger table is used as the highest layer finger table, additionally each peer also creates $m-1$ (where m is the hierarchy depth) more finger tables for each lower level DHT ring it belongs to.

The HIERAS architecture uses landmark nodes spread across the Internet, these landmark nodes are responsible for designating each lower level DHT ring. How landmark nodes are chosen is not mentioned in the paper, and could be a worthwhile investigation for the future. Peers in the DHT use the latency information of landmark nodes to decide which lower level DHT ring to participate in.

For use in MANETs the performance of HIERAS has some characteristics which would be useful, such as more aspects of proximity awareness among peers. However its performance in smaller networks such as MANETs is questionable. As the authors do not mention how the number of landmark nodes scales with the number of peers in the system, it appears to be a configuration parameter and not real-time adaptive. The constraint of battery life and mobility in MANETs means that the number of peers in the network can vary greatly, thus having the number of landmark nodes adapt to the number of peers seems a logical step. The authors also mention that the protocol creates more overhead in the network, this requires more energy consumption and would need to be addressed if the protocol was developed for MANETs. Finally due to the protocol being Internet centric obviously no thought has been given to node mobility. This would pose a major obstacle for using HIERAS in MANETs as one would need to account for moving closer to another landmark peer. Said peer could simply leave the network and rejoin, but this may cause the network to become unbalanced in terms of load.

The paper [82] the authors present a P2P overlay network that maintains complete membership information at each node in order to lookup queries in just one hop, thus reducing lookup latency. The system design considers networks of n peers where n is

a large number like 10^5 or 10^6. Every peer in the overlay is assigned a unique 128-bit identifier and has a predecessor and successor in the modulo 2^{128} ring akin to the Chord protocol. Clients issue queries that try to reach the successor peer of a particular identifier in one hop. To achieve their goal every peer in the system should keep a full routing table containing information about every other peer in the overlay. In order to keep full state information at every peer, a hierarchy is imposed on the system to form dissemination trees, these are used to propagate event information.

Hierarchy is imposed on the system by dividing the 128-bit ring identified space into k equal sections called slices. Thus the ith slice will contain all peers in the overlay whose peer ID lies in the range $[i \cdot 2^{128}/k, (i+1) \cdot 2^{128}/k)$. The algorithm assumes a uniformly distributed ID space, thus each slice should have about the same number of peers at any time given the large network size they consider. Each slice will have a *slice leader* which is chosen dynamically as the peer that is the successor of the mid-point of the slice ID space. Therefore the ith slice would be given the slice leader which has the successor key closest to $(i + 1/2) \cdot 2^{128}/k$. In a similar way slices are divided into equal sized sub-slices called units. Each unit will have a unit leader which is dynamically chosen as the successor of the mid-point of the unit ID space.

When a peer detects a change in the membership of a slice (through keep-alive messages), it sends an event notification to the slice leader. The slice leader then collects all event notifications from its slice and aggregates them for a certain time period before sending a message to other slice leaders regarding any changes. These slice leaders then aggregate messages they receive for a certain time period before sending said event notifications to all unit leaders if their respective slices. Each unit leader will then piggy-back the event information on to keep-alive messages to its successor and predecessor. Finally each peer will then propagate this information to its successor if received from its predecessor and vice-versa if received from its successor.

The notion of a one-hop P2P network is indeed interesting in MANETs due to the lowered latency in lookups. However there is a fine balance between reducing latency,

and increasing congestion in the network due to the overhead induced by maintaining state information at all peers. On a large network such as the Internet, congestion is not so much an issue, however in MANETs links upon which many routes traverse can become congested, thus requiring more consideration. This could be alleviated by piggy-backing some traffic on to the routing packets, however as the routing protocol for proactive networks already has all state information about the network, using a cross-layer approach may be more efficient.

The paper [82] unfortunately does not investigate efficiency of smaller networks such as those expected to be the size of a MANET. A possible issue with smaller networks using this protocol pertains to load-balancing. There could be cases where there simply are not enough peers in order to have a uniformly distributed ID space. Thus there would not be enough peers to satisfy the number needed for every required slice leader and unit leader.

3.5 Service discovery for MANETs

Service discovery for MANETs is another related area which should be investigated due to its applicability as a distributed platform for applications on MANETs. This area has also received much research interest over the past decade. Thus this section will give an overview of the main concepts and proposed solutions in the field.

3.5.1 Broadcast-based service discovery

The simplest method to do service discovery in MANETs is to send a broadcast message throughout the network. Upon receiving such a request, a peer will check its local repository to establish whether it has the searched for data, if so the peer will sent its response back to the originator of the request. One of the first solutions akin to this is presented in [83]. In this protocol, when ever a peer successfully finds that they are storing the searched for data, the peer will reply directly using which ever underlying MANET routing protocol is being used. The originator of the request would then use said routing protocol for then connecting to the discovered service.

Integration the service discovery architecture with the routing protocol is another popular method discussed in the literature. Such a method is proposed in [84], whereby peers periodically broadcast their service advertisement within a certain hop radius. Due to there being no routing protocol used in this instance, each peer stores along with the advertisement, the last hop from where the service advertisement was received. Thus an AODV style route back to the originator of the service advertisement is constructed. In this proposed architecture, when searching for a particular service, these requests are also broadcast in the same manner, where peers forwarding the request store an AODV like reverse path back to the originator of the request. When a peer receives a request first it checked whether the service is provided locally, if so it responds back with its service advertisement. The idea presented is a simple solution, however there are a number of issued which are not addressed, such as how a requesting peer will know the route to the service provider upon receiving a reply to said service request, as no routing protocol is employed. If the requester would contact the service provider via the responding intermediate peer, this would obviously be highly suboptimal. The situation of how a peer will handle the breaking of a route to a provider upon having already constructed a route is also not addressed, which could be a common scenario in MANETs with high mobility.

Another more thorough solution is presented in [85]. In said paper the authors propose to create a new message type in the MANET routing protocol AODV specifically for service discovery. The message is handled exactly the same as a normal RREQ however instead of peers looking up a route, they check whether they can provide the requested service. If such a service is available they reply to the request in a similar way to a RREP. In all other functions such as accessing requested services, normal AODV is utilised to communicate.

Broadcast-based service discovery architectures for MANETs are very similar in operation to unstructured P2P networks designed to operate on MANETs as described in Section 3.2. One can go so far as to say that the two techniques are almost indistinguishable from each other except for the fact that service discovery looks up services and allows for their usage as apposed to finding files and downloading them. Thus broadcast-based service

discovery methods for MANETs exhibit the same issues as the unstructured P2P protocols designed for MANETs, namely those discussed relating to scalability.

3.5.2 Geographic service discovery

Another area of service discovery protocols can be classed as geographic-based service discovery protocols [86–90]. These protocols can be described as using geographic positioning, for instance using Global Positioning System (GPS) and geographic routing to provide efficient service discovery mechanisms in MANETs. In [89] for example, the concept of geographic hashing is introduced. In this case, each object is hashed into a geographical area of the network and thus the node closest to the hash therefore is responsible for being the directory for said object. Thus when a peer wishes to store or look up an object, said peer simply routes the request to the object's hash co-ordinates in the physical network using the geographic based ad-hoc routing protocol as described in [91].

Geographic service discovery protocols are able to provide efficient service discovery in MANETs. The feature of having available information on a peers geographic position can be seen as a great advantage when routing data between peers as the route should follow the shortest physical path. Providing further efficiency is that fact that route maintenance is no longer necessary as a peer only need know its one hop neighbours and the direction in which to forward the packet. Thus by combining geographical routing with geographic-based hashing, one can eliminate overlay stretch as the objects can be mapped directly on to the physical network.

These protocols however require the geographic position of the peer, by using technologies such as GPS which if often consume energy resources quickly, energy resources which are usually quite limited in MANET devices. In recent times GPS is common among devices which could be used in MANETs, even so there are many limitations on the places in which GPS works due to line of sight, and of course their complete failure when used indoors or underground. Thus this means the applicability of geographic-based solutions remains limited to a strict set of scenarios in which position can be guaranteed. As the focus

of this thesis should not be limited by these constraints — MANETs have a wide range of uses indoors and in underground scenarios such as emergency situations where GPS is either unreliable or not available — geographic-based service discovery and object lookup will not be further discussed. For further reading on the subject the readers are encouraged to examine a thorough study of such services as provided by the authors in [86, 92]

3.5.3 Hierarchical service discovery

The authors of [93] propose a system whereby peers dynamically assign themselves hierarchical addresses. The node addresses consist of l bits and the address space is conceptually organised as a binary tree with $l + 1$ levels. The leaves of said tree denote the the individual peers, whereas the inner tree node defines a sub-tree consisting of addresses with a common matching prefix. In order to guarantee routing converges, the system relies on what they describe as a prefix sub-graph constraint, where each sub-tree must form a connected sub-graph in the physical network. Thus one can derive that peers whom are leaves of a given sub-tree must be able to reach each other. The system also introduces level-k siblings in order to enable routing. The level-k sibling of a given peer's address is the sub-tree of said peer which contains the peer addresses which share a matching prefix of $l - k - 1$ bits with the local peer's address. As the addresses are binary based each level-k sub-tree therefore consists of exactly two $level - k - 1$ sub-trees. Thus each level-k address has one level-k sibling and a total of l siblings.

The routing table for each peer consists of one entry for each level-k sibling which contains the route information to any peer in the respective level-k sibling. Hence the routing table contains l entries. Upon a peer then requesting to route a packet to a given destination, the peer first needs to determine in which sibling tree the given address resides. Then it simply uses the given routing table entry for said sibling and routes the packet to the peer stored in such entry in the routing table. As the address allocation is dynamic, peers will often not be aware of the current address of the destination peer. In order for the peer to resolve the required address the system utilises a DHT-based lookup message. In

order to achieve this each peer hashes its static ID into the address space. The peer which has the closest address — where closeness is defined by the XOR value of the hash key and peer address — to said hash key will then store the peers current address.

This DHT-based name resolution could therefore be used for general purpose service discovery as an additional feature. In this case services and objects could be hashed into the address space and akin to the name lookup procedure, the peer closest to the service or object hash key would then act as its temporary directory peer. Thus because of the deterministic hierarchical routing employed the overhead incurred by service discovery would be significantly reduced when compared to broadcast-based or multicast-based service discovery protocols. The resulting service discovery protocol would be very well suited to smaller networks with less mobility. In larger networks however the system would incur a lot of address reorganisations, due to peers moving around more often causing frequent of the prefix sub-graph constraint previously mentioned. This would have the effect of peers having to reassign their addresses in order to satisfy the constraint. This is further compounded by the fact that when a peer acquires a new address the service advertisements for which it was previously responsible for would then need to be reassigned to the peer which is now responsible for said service hash keys. Additionally said peer will also need to acquire the correct service advertisements for which it is now responsible for from the current service directory peers. In dynamic networks, due to the strict relationship between the address tree and the peer address, such reassignments could occur frequently, therefore leading to a high overhead in the network from the resulting handover traffic.

3.5.4 Cluster-based service discovery

Another popular proposed service discovery method adapted for MANETs is that concerning cluster based service discovery. In [94] such a method is proposed. The authors propose local directory peers, which are selected so that every peer has at least one such directory peer within a radius of h hops. Normal peers register their services with the

designated directory peer within their h hops cluster. Such directory peers will periodically advertise their presence to the peers within their cluster. In order to reduce overhead the directory peer advertisements will be propagated within the cluster within a two hop radius, and again repeated another two hops until the advertisements meet the h hop limit. These advertisements can also be used by peers in order to update their routing information regarding the route to their directory peer. If a non-directory peer does not hear from any directory peer within a certain time period, it will initiate a directory election request which is again propagated over h hops at two hop intervals. Thus peers willing to become directory peers — which is dependant on their available capacity — will respond to the requesting peer. If more than one peer replies to the request, the requesting peer will then decide which peer to elect as its directory peer based on a given metric such as its available capacity, thus it will then register its service with said peer. Directory peers will also periodically exchange their indexes with each other, this is achieved by using a bloom filter representation of the service advertisements which are stored in their directory. To this end directory peers periodically advertise their bloom filters over a $2 * h$ radius to other directory peers. Such advertisements also act as ways for directory peers to maintain routes to their directory peer neighbours. In order for a peer to request a service, the request is first sent to the local directory peer. Having received the message, the local directory peer will then check its directory for any matching services. If a match is indeed found, it will reply directly to the requesting peer. However if no matching services can be found, the local directory will initiate a global service request. To achieve this, the local directory peer will first probe the bloom filters of known other directory peers in order to select directory peers which are likely to have a matching service in the respective directory, the message requesting such a service is then forwarded to the selected peers. The aforementioned process is then continued at each directory peer until the request reaches a peer which is able to fulfil the request.

The concept described in [94] has been shown to reduce service discovery overhead in relatively small networks with low mobility. However in larger networks with higher

mobility the efficiency of such an approach will soon diminish. As the network grows, it will become less likely that a directory peer forwarding a request will know of other directory peers likely to be able to fulfil the requested service requirement. Thus in larger networks which exhibit higher levels of mobility, the global service discovery process will be akin to a broadcast based service discovery process among the directory peers, bringing the scalability pitfalls of a broadcast approach to the described service discovery protocol. The selection process used for forwarding service discovery requests is not well described in the paper. The authors in [94] explain that discovery peers probe the bloom filters of other known directory peers in order to select those which are likely to have a matching service in their directory. The outline of the forwarding decision algorithm however only seems to take into consideration remaining battery life as well as distance. Another issue remains the fact that the size of the cluster radius h is not described, and calculating an optimal value is not investigated. The authors merely state that the value is dependant on the peer density. The value of h will obviously have a great impact on the overall protocol performance. As an example, if a too large value of h is used, the size of the clusters will become too large and the routes from non-directory peers to their local directory peers which were established during propagation of the directory advertisements will frequently fail. Then depending on the underlying MANET routing protocol used high overhead route discoveries will then need to take place. Meanwhile the distances between directory peers will also increase, meaning that inter-directory routes will also fail frequently further decreasing the performance of the global service discovery process and creating the need to incur again more route requests on the underlying MANET routing protocol. In the case where h is given too small however, the number of directory peers will increase significantly. Therefore when a global service discovery takes place, the chances of a directory peer knowing of other directory peers which are likely to be able to fulfil the service request will further decrease. In this case the global service discovery will again involve a large number of peers and the process of service discovery will again be akin to a costly broadcast based approach.

3.5.5 Multicast-based service discovery

Another method for service discovery on MANETs is the usage of multicast trees upon which services are advertised and can be requested. One of the most popular papers in this area [95] proposes a service discover architecture entitled Konark. In this architecture every peers maintains a topic-based register tree which is used to store known service advertisements. Peers will then periodically advertise their services in the network. Service discovery is achieved by a request being propagated throughout the multicast group where each peer receiving it will check its local registry for a service matching the request, if a match is found said peer will then reply. Several issues are left open with this architecture. Such as how multicast groups are formed and maintained. The example in the paper states that all participating peers will join the same locally scoped multicast group, this would equate to to a request being broadcast within the entire network. Additionally the categories at each level of the registry are not defined in the paper.

The authors in [96] present a more complete method for service discovery in MANETs. The authors propose to construct a virtual backbone of backbone peers known as Virtual Access Points (VAPs) which are two or three hops away from each other. Hello messages are used in order to establish routes between neighbouring VAP peers which periodically maintain the backbone. Normal peers register with the closest VAP peer which allows them to advertise their services. In order for a peer to request a service, VAPs construct multicast trees on top of the virtual backbone on demand. Thus when a peer wishes to request a service it sends such a message to its local VAP peer. Therefore each VAP is the root of its multicast tree, then for each multicast source, each VAP maintains a list of its neighbouring VAPs to which it should forward the said message as determined by the multicast source. As mentioned the multicast trees are constructed on demand. A VAP's forwarding list for each multicast source initially contains only the neighbouring VAPs. When a VAP receives a multicast message, it prunes the previous VAP from which it has received the message from its forwarding list for the given source, as such a VAP would have already received the message. If a VAP receives a duplicate message from another

VAP to which it had not forwarded the message, it will send a prune message to said VAP instructing it to remove the local VAP from its forwarding list for this particular multicast source. Due to this, the initial message sent from a VAP will resemble a regular broadcast to all VAP peers. However subsequent message will be sent started from the VAP source and able to use the constructed multicast structure.

VAP based multicasting can reduce service discovery overhead as seen in the results from the paper in static or slow moving networks. However in highly dynamic networks with more mobility the efficiency of the protocol can be expected to suffer greatly. In highly dynamic networks the virtual backbone topology will become unreliable as VAPs change regularly in order for the distance between backbones to be maintained (usually two or three hops). Furthermore, when an existing VAP is made aware of a new neighbour VAP it will need to reinitialise its forwarding lists as the old multicast trees would be significantly altered. Due to as previously mentioned, initial messaged resembling broadcast message between virtual backbone peers, this could severely affect performance.

It can be deduced from the results of the above papers that multicast based service discovery protocols for MANETs reduce advertisement and service request overhead when compared with simple broadcast-based approaches. Multicast based service discovery protocols can be seen to excel especially in static or slow moving networks where the multicast structure is not severely disrupted. However in networks with higher mobility and dynamism, the multicast structure can be constantly changing as peers move around. This in turn will cause the multicast structures to be difficult to maintain and cause significantly high overhead causing highly dynamic multicast based protocols to resemble their broadcast based counterparts in such circumstances.

3.6 Summary

In this chapter number of different approaches have been examined which could potentially be used as the basis for large-scale distributed network applications and services in MANETs. The various approaches evaluated has been designed for a wide variety

of different application scenarios. To elaborate, some have been designed with specific utilities in mind for their functional use, such as file-sharing, where as others have been presented more as the basis for applications to be built upon. In keeping with this idea, the different approaches also focus on optimising for different aspects of the network, such as maintenance complexity or service discovery overhead.

Chapter 4

ROBUST DHT

In this chapter the proposed architecture entitled Reliable Overlay Based Utilisation of Services and Topology (ROBUST) Distributed Hash Table (DHT) [45] is presented in order to satisfy the research objectives RO1, RO3, and RO4-RO5. The aim of the architecture is to decrease the average path length and lookup time when sending DHT messages thereby decreasing *stretch*, while decreasing the maximum path length from the $O(\log N)$ complexity seen in most common DHTs used today, where N is the number Mobile Ad hoc Network (MANET) peers. The proposed architecture should be heterogenous in that it should operate independently of the MANET routing protocol, thus providing flexibility of which MANET routing protocol a potential administrator could use. The advantage of this approach is due to the MANET community which is currently undecided on which MANET routing protocol will eventually become a standard, thus in this chapter ROBUST is evaluated and compared with OpenDHT [11] on top of the MANET routing protocols Ad-hoc On-demand Distance Vector version 2 (AODVv2) [49] and Optimized Link State Routing version 2 (OLSRv2) [47].

4.1 Overview of design

The concept central to our DHT is to use a clustered hierarchical topology. This means peers will be clustered together based on proximity in the underlying MANET. The peers will be

connected via a super peer which keeps track of peers within the cluster and also carries out cluster maintenance. Cluster peers will be able to communicate with one another, however peers within each cluster will forward their queries to their dedicated super peer, if the destination lies outside of the cluster, the peer will forward the query to the super peer responsible for the destination peer, the destination peer will then reply to the request.

At any given time C clusters are needed where $C = \lceil \frac{N}{\log_2 N} \rceil$ and N is the total number of peers in the network. This gives us a total routing complexity of $O(\log_2 C) + O(1)$ and $O(\log_2 C) \leq O(\log_2 N)$. The $O(1)$ is due to the fact that the peer has to forward the request to its super-peer in the beginning.

To address the issue of scalability and reduce any one super peer from being bottlenecked, more clusters would need to be factored into the DHT as the amount of peers increases. Two algorithms to increase the number of clusters will be investigated. The first of which is when $\frac{N}{\log_2 N} > 2C$ where C is the current number of super peers, each cluster will split their responsible ID space by 2. This maintains a balanced load and only creates more overhead due to moving keys at 2C. In the second algorithm the number of clusters is increased for every $C = \frac{N}{\log_2 N}$ and would need to decrease the size of each cluster by:

$$\frac{100}{C} - \frac{100}{(\frac{N}{\log_2 N})} \tag{4.1}$$

percent where C is the amount of current clusters. Decreasing the size of each cluster would leave ID space between each cluster, therefore each cluster would need to be moved in order to create a seamless ID space, resulting in a need to move a total of:

$$\frac{\log_2 N^2}{100} \cdot \frac{100}{(\frac{N}{\log_2 N})} \tag{4.2}$$

data keys when the number of keys each peer is responsible for is $O(\log_2 N)$ keys.

When the number of peers in the DHT decreases it is possible clusters may collapse, the nature of how peers join the DHT means that equal ID space is given to each cluster. However if all of the peers in a cluster leave the network, there is no longer a need for so

many clusters as $C < \frac{N}{\log_2 N}$, however to limit overhead and to maintain a threshold, the number of clusters will only be decreased when $\frac{N}{\log_2 N} < 2C$. In order to decrease clusters super peers will calculate the new ID space for which they are responsible. For example super peer i will calculate the new ID space for which it is responsible as $i \cdot 2^{128}/C$ given an ID space of modulo 2^{128}. Once this has been computed the super peer can assign ID space to each peer to keep the network and ID space load-balanced. This adaptive quality means that the overlay can maintain a lower number of total hops during lookups and decrease lookup latency whilst still being scalable and load balanced. When building the algorithm it has been specifically modelled to deal with a lower number of peers than that of DHTs created for use in massive networks. For instance DHT's used on the Internet can encompass thousands of peers, such possibilities are not foreseen in MANETs due to the restrictions of the routing protocols with regards to packet collisions and transmission errors causing many dropped packets as seen in our simulations later in this chapter.

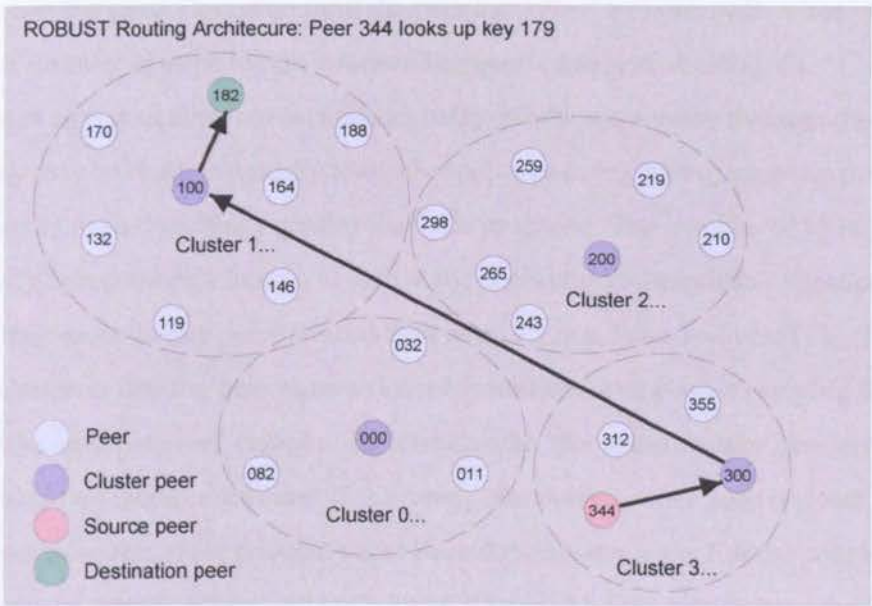

Figure 4.1: The ROBUST DHT routing architecture

Figure 4.1 shows the ROBUST routing mechanism. In this example peer 344 looks up

the overlay ID 179, one can note that first the look up message is sent to the local peers cluster peer, which then forwards it to the cluster peer responsible for that ID prefix, in this case 100. Upon receiving the lookup message, cluster peer 100 then forwards the message to the peer in its cluster whom is responsible — numerically closest — to the peer ID being looked up, in this case peer 182. The peer with ID 182 will then reply directly to peer 344 using the underlying MANET routing protocol.

Peer proximity is central to the proposed algorithm and the architecture has been designed with this notion in mind. From the start when peers join the overlay they contact the nearest bootstrap peer, which is the nearest super peer. If the number of peers within the cluster is less than $\log_2 N$ the peer will join that cluster after being given a *peerID* by the super peer in order to maintain ID space equality. If the the number of peers within the cluster is equal to $\log_2 N$ the super peer will forward the join request to its closest two super peers (the first numerically greater than and less than its own *peerID*) these super peers will then run the same algorithm until the peer has joined a cluster with a free space. The maximum number of steps to find a non-full cluster is denoted as $O(\log_2 C)$.

The next step is to take into account mobility. When peers move through the physical space, they may be closer physically to another super peer. In order to maintain proximity a function aptly named *proximity synchronisation* is proposed. This is achieved by super peers periodically broadcasting a beacon to each of the peers within their cluster, these peers then forward the beacon to any peers around them with a 1 hop Time-To-Live (TTL). The result of this function is that if a peer moves closer to another super peer, it can ping the newly discovered super peer and compare its latency with the current super peer with which it has established communications. If the newly discovered super peer is closer, the peer sends a move request to the relevant super peer, if the cluster is not full the peer leaves the overlay with its current ID and rejoins with an ID issued by the new super peer. This only happens periodically to limit overhead and the problem of a peer intermittently switching between two close super peers.

Another factor taken into account in this chapter for the proposed architecture is that

of super peer election, the super peer should be the peer whom is optimally suited to take the role. Therefore factors such as throughput, remaining battery power and latency should be taken into account. It is proposed that periodically the current super peer will poll all of the peers within its cluster for a reliability metric u every 300 seconds so that the network is not over saturated with super peer changes. The reliability metric is calculated as the following; $u = w_1t + w_2b + w_3l$. To allow flexibility in different scenarios, w_i values are weights given to increase or decrease the impact factor of a given metric. The values of t, b and l are measured by a threshold denoted as $[v_1v_2, ..., v_j]$ where a higher value equates to higher reliability. The data values used to calculate the reliability metric are as follows; maximum throughput t in bits per second, remaining battery power b as a percentage of remaining power, and latency l which is the average latency between the peer and every peer in the cluster. This metric allows us to simply select the peer with the highest reliability metric to be the super peer.

In order to calculate N to be used to define the cluster sizes, all super peers must agree on the size of the network. This is achieved by using a specific key in the DHT network to store information about cluster sizes. The specific key which would store the cluster sizes would be preconfigured. The data corresponding to the key is a list which each super peer either adds to or updates. The list contains the key of the super peer, and the corresponding number of peers in its cluster. Each super peer should update the list periodically. In order to confirm that the data is correct, upon retrieving the list — in order to recalculate N — the super peer will ping each other super peer on the list, the ping packet will also contain the number of peers said to be in its cluster as determined from the list. The pinged super peer will then reply back to the enquiring super peer with the difference between the list value and actual value if there is one. If no reply is received within a specified period, the super peer is deemed offline and it is removed from the list and not counted towards N.

Looking back to Section 3.4 where the literature review of hierarchical Peer-to-Peer (P2P) networks studies other hierarchical designs, the proposed protocol takes into account peer mobility by allowing peers to feely move between clusters. The proposed protocol is also

more suited to smaller networks as would be expected in MANETs as its load-balancing and scalable nature means that most of the DHT ID space should be covered as apposed to other architectures which require a certain number of peers in order to function correctly. It should also exhibit less overhead when compared to one-hop DHTs as state information about the entire network is not needed at every peer, in fact a regular peer in the worst case would only need to know its super-peer to be able to execute lookups.

4.2 Detailed description of protocol

This section describes the signalling packets needed to maintain the ROBUST DHT. The model consists of the list of inputs followed by the computational functions, followed by a practical example in the performance evaluation section. The formulas also indicate the packet overhead which provide insight into the efficiency of the proposed architecture due to the sensitivity of MANETs to network congestion when a high number of packets are being exchanged. This can be evidenced by the large number of duplicate packets sent and received by MANET nodes on the transport layer due to expiring of the round trip timer which is estimated using previous sent packets. Thus, the higher the amount of packets being routed around the MANET the higher the delay jitter occurs causing high variance in Round Trip Time (RTT); causing even more duplicate packets.

Within the DHT, single overlay hops can be attributed to multiple physical hops on the underlying network. Thus, in a typical overlay the upper bound on hops is denoted as $DHT_{hmax} = O(log_2 N)$ where n is equal to the total number of peers in the overlay network. However in ROBUST this routing request is reduced through super peers and hence having an upper bound on hops described as $DHT_{hmax} = O(\log_2 C)$.

In the model it is assumed the underlying transport layer protocol is User Datagram Protocol (UDP) as used in OpenDHT [11] this is due to the fact that the DHT must have access to transport information in order make decisions such as whether to remove a peer due to packet loss, reliable transfers are expected by using of acknowledgement packets and sequence numbers in the DHT and the use of *round trip timers*. The total packets (space)

are calculated which are needed for DHT data gathering, and overlay maintenance for each peer denoted as:

$$\begin{aligned} DHT_{proc} = & DHT_{get}(no\_gets) + DHT_{put}(no\_puts) \\ & + DHT_{ack} + DHT_{sync}(t_{sync}) + DHT_{ls\_up}(t_{ls}) \\ & + DHT_{ping}(t_{ping}) + DHT_{clstr\_beacon}(t_{beacon}) \\ & + DHT_{prox\_sync}(t_{prox\_sync}) \\ & + DHT_{join}(no\_joins) \end{aligned} \tag{4.3}$$

where the DHT functions are denoted below. First DHT gets (data retrieval) are calculated as follows:

$$DHT_{get} = 2DHT_{get\_req} \cdot DHT_{hmax} \cdot no\_gets \tag{4.4}$$

This equation has been derived based on the fact that there are two packet types sent for one get — a get is how a DHT key-based lookup request is referred to — (GET and GET_SUCCESS) therefore the total must be multiplied by a factor of 2, each overlay hop is then taken into account, denoted as DHT_{hmax} and it is multiplied for the total number of gets for the systems, in order to get the spacial overhead. Similarly all puts are calculated — a put is how a DHT request to store a particular key value pair in the DHT is referred to — in the DHT as:

$$DHT_{put} = 2DHT_{put\_req} \cdot DHT_{hmax} \cdot no\_puts \tag{4.5}$$

Then the packets required for data synchronisation across all peers for a particular

given leaf set are calculated:

$$DHT_{sync} = (DHT_{sync\_vals} + DHT_{str\_keys}) \cdot DHT_{hmax} \cdot \frac{T}{t_{sync}} \tag{4.6}$$

where DHT_{sync_vals} is the synchronisation request packet, DHT_{str_keys} are the storage keys held at the leaf set peer for which both the requesting peer and the leaf set peer are responsible and t_{sync} is the periodic synchronisation interval. The function for leaf set updating is denoted as:

$$DHT_{ls\_up} = (DHT_{pull\_ls} + DHT_{push\_ls}) \cdot DHT_{hmax} \cdot \frac{T}{t_{ls}} \tag{4.7}$$

where DHT_{pull_ls} denotes the leaf set pull request, DHT_{push_ls} denotes the keys of all the leaf set peers required and t_{ls} equates to the periodic leaf set update interval. The function for peers joining the overlay can be described as:

$$DHT_{join} = DHT_{join\_req} \cdot DHT_{hmax} \cdot C \cdot no\_joins \tag{4.8}$$

where DHT_{join_req} equates to the join request packet, which is forwarded to the nearest super peer with a free space in the cluster C this is multiplied by the number of peers which join the network *no_joins*. All pings in the DHT are calculated as:

$$DHT_{ping} = DHT_{ls\_peer} \cdot DHT_{hmax} \cdot \frac{T}{t_{ping}} \tag{4.9}$$

where DHT_{ls_peer} is the leaf set list for a given peer, and t_{ping} is the periodic ping interval. When a super peer (*SP*) broadcasts a cluster beacon every t_{beacon} time period containing

its own information to all of its 1 hop neighbours (DHT_{1_hop}) to determine whether the peer has a better like to the new super peer rather than its old super peer, this transaction can be described as:

$$DHT_{clstr\_beacon} = [(DHT_{beacon} + \\ + DHT_{ping}) \cdot DHT_{1\_hop}] \cdot \frac{T}{t_{beacon}} \tag{4.10}$$

Now the *Proximity synchronisation* function will be examined, which is called periodically at every t_{prox_sync} interval and acts when a peer moves closer to another super peer SP' and consequently should move to the new cluster by adopting a new ID in the DHT space in order to reduce stretch in the overlay and delay. The overhead of this function can be denoted as:

$$\begin{aligned} DHT_{prox\_sync} = [&DHT_{move\_req} + DHT_{move\_rep} \\ &+ (DHT_{part} \cdot DHT_{hmax}) \\ &+ (DHT_{broad\_part} \cdot DHT_{ls}) \\ &+ 3DHT_{ls\_up} \\ &+ 3DHT_{sync}] \cdot DHT_{move\_peer} \end{aligned} \tag{4.11}$$

where DHT_{move_req} is the request to join the new cluster, sent from the joining peer to the super peer called SP'. DHT_{move_rep} is the reply from the super peer SP' to the joining peer indicating whether there is room in the cluster for the peer to join. DHT_{part} is a packet sent from the joining peer to the super peer SP indicating that the joining peer is leaving the cluster for which SP is responsible. The super peer SP will then remove the joining peer from its leaf set if the peer is included and removes the peer from its cluster set which contains information about all of the peers within the cluster for which SP is responsible. DHT_{broad_part} symbolises the packets sent from SP to all of the peers within

the original leaf set (DHT_{ls}) of the joining peer relaying to them the information that the joining peer has left the cluster. These peers will then remove the joining peer from their leaf set. DHT_{move_peer} refers to the number of peers with high enough mobility in order to pass the mobility awareness threshold which is derived by comparing the round trip time (RTT) found by DHT_{clstr_beacon} of SP and SP'. Finally the total number of acknowledgement messages needed to acknowledge all the above mentioned packets is calculated as follows:

$$\begin{aligned} DHT_{ack} = {} & DHT_{get} + DHT_{put} + DHT_{sync} + DHT_{ls\_up} \\ & + DHT_{ping} + DHT_{clstr\_beacon} + DHT_{prox\_sync} \end{aligned} \tag{4.12}$$

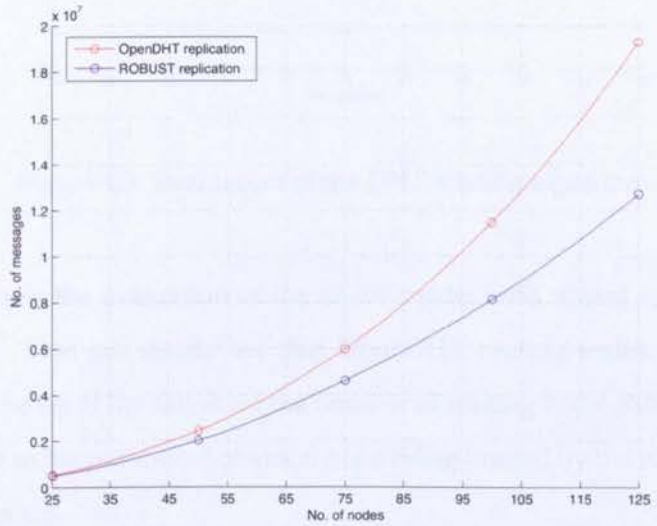

Figure 4.2: Evaluation of the DHT replication algorithm.

The figure 4.2 shows the evaluation of the above described model with regard to the DHT replication mechanism. The model shows a significant decrease in overhead messages of ROBUST when compared to OpenDHT. This is as would be expected as the proximity awareness caused by clusters of peers with similar peer IDs means that a leaf peer should

usually be within a few hops range. This is in contrast to OpenDHT where a leaf peer could be physically on the other side of the network due to the random way in which peer IDs are assigned.

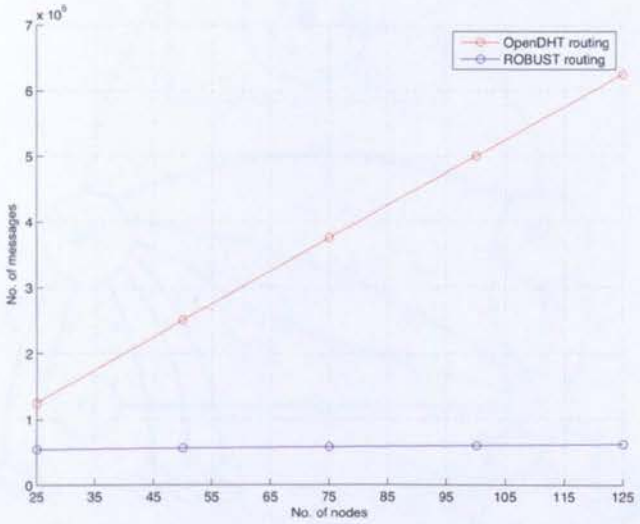

Figure 4.3: Evaluation of the DHT routing algorithm.

Figure 4.3 shows the evaluation of the above model with regard to overhead caused by DHT routing. One can clearly see that OpenDHT routing scales logarithmically as expected due to the $log\ N$ limitation on the number of routing hops. ROBUST on the other hand is static due to the number of physical hops being limited by the routing of messages between cluster peers.

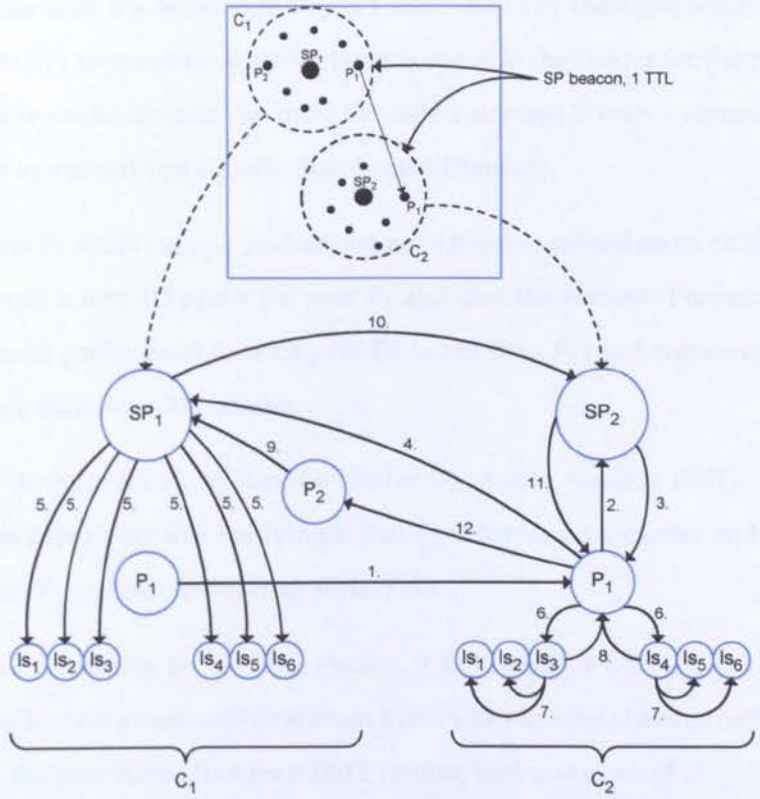

Figure 4.4: An overview of the described DHT architecture.

Fig. 4.4 shows an overview of the DHT architecture. The box at the top of the figure represents a top down overview of the network containing two clusters where C_1 represents cluster 1 and C_2 represents cluster 2. The dashed lines around these clusters represent the broadcast radius for the super peers for each cluster SP_1 and SP_2 respectively. Each step of the diagram is explained in detail below:

1. Peer P_1 moves out of transmission range of super peer SP_1 and subsequently into transmission range of super peer SP_2, therefore receiving the DHT_{clstr_beacon} packet from said super peer and gaining knowledge of its presence.

2. P_1 then compares SP_1RTT — a peer will ping a super-peer immediately after receiving a beacon to establish the RTT — with SP_2RTT over a period of $3t_{beacon}$ to confirm

the results with the realisation $SP_2RTT < SP_1RTT$. P_1 therefore sends a DHT_{move_req} packet to SP_2 to ascertain whether there is space in the cluster for the peer to join (as previously stated clusters can must be within size log_2N with a variance of +2 peers in order to maintain an equally distributed ID space).

3. SP_2 sends P_1 a DHT_{move_rep} packet stating if there is indeed room in the cluster and if so, sends a new ID prefix for peer P_1 and also the Internet Protocols (IPs) of P_1s new closest predecessor (closest peer ID lower than P_1) and successor (closest peer ID higher than P_1) in the cluster.

4. If there is space for P_1 to join the cluster C_2, it then sends a DHT_{part} packet to its previous super peer SP_1 notifying it that P_1 is leaving the cluster and subsequently removes all previous nodes from its leaf set.

5. Upon SP_1 receiving the DHT_{part} packet, it first sends a DHT_{broad_part} packet to P_1s previous leaf set peers notifying them that P_1 has left the cluster, they subsequently remove the peer from all of their DHT routing tables as does SP_1.

6. P_1 then sends a DHT_{ls_up} packet to its predecessor and successor nodes, they add the node to their leaf set and reply to P_1 with the leaf set peers which fall within P_1s leaf set.

7. The leaf set nodes of peer P_1 learn of his existence through the leaf set update function which runs every t_{ls} period and chooses a random existing leaf set peer to sync with.

8. The leaf set peers of P_1 run the DHT_{sync} data synchronisation function every t_{sync} interval randomly choosing a leaf set peer to synchronise data values for which both the peers are responsible. In this way peer P_1 will receive all of the data values it is responsible for.

9. Peer P_2 of cluster C_1 submits a DHT_{get_req} to its super peer SP_1 whom then compares the ID lookup stored in the request with the ID's of all of the other super peers in the overlay which it knows of.

10. SP_1 then forwards the request to the super peer with the closest peer ID to that of the ID stored in the request which in this case is SP_2.

11. SP_2 receives the DHT_{get_req} and forwards it to the peer within the cluster with the closest peer ID to the ID stored in the request which is P_1.

12. Peer P_1 receives the request and sends the data stored under the specified ID directly back to P_2 using its IP address.

4.3 Limitations of the proposed architecture

One limitation of the ROBUST DHT appears when a super peer disconnects from the network. This causes temporary instability in the DHT as a new super peer will need to be elected. During this time DHT routing from and to the affected cluster will be lengthened from $O(\log C)$ to $O(\log N)$. This occurs due to peers falling back to traditional DHT routing using their leaf set peers to route information.

Another limitation of ROBUST DHT is based on the restrictions of MANET routing protocols, at the time of writing Internet Engineering Task Force (IETF) has two main Request For Comments (RFCs) in this area and work is ongoing. Due to the still experimental protocols being used in MANETs, with larger networks high delay and packet loss can be expected, which invariably affects the DHTs functionalities. This is countered by making the DHT topology as close as possible to that of the MANET in order to lower the overhead incurred. One of the main reasons for this discrepancy is due to frequent route changes which can cause multiple duplicate packets and network congestion as described in the paper [97]. This was later confirmed by the authors in [98] where using a real-world testbed implementation of Optimized Link State Routing (OLSR), the authors found mobility incurs a large delay even for a small sized MANET, in this specific case the authors used up to 5 MANET peers and experienced an end-to-end delay of over 3 seconds due to transport layer and interference issues.

4.4 DHT Simulator

Simulations are utilised to verify the integrity of the proposed models and showcase the benefits of using our proposed solution. An event based simulator is used, customised with the author's implementation of ROBUST protocol, to validate the proposed model and optimisation solution. The author has developed a simulator module for the packet-level network simulator network simulator-2 (ns-2) [51]. The simulator incorporates all DHT packets and functions needed for a fully implemented DHT and the implementation is based on the ROBUST DHT clustered architecture with dynamic mobility considerations. Further lower layers are also simulated in the ns-2 simulator and these characteristics are taken into account such as the Medium Access Control (MAC) layer 802.11*b*, transmission errors, and packet collisions.

The process of packet initialisation to its definitive end is described below.

- packets are originated from the ROBUST protocol itself and then passed to the ROBUST sent agent which maintains connections and keeps track of packets RTTs using pings.
- subsequently when a RTT expires when a packet is sent the packet is resent since it is assumed that it has been dropped.
- the sent agent also keeps track of the packet sequence numbers so then the packet sent down the stack to the routing protocol (OLSR [42] is used).
- the latter then computes the best route to send the packet and forwards the packet over the intermediate peers of the MANET until it reaches the destination.
- the destination node pushes the packet up the stack thus it is then received by the ROBUST agent which sends an acknowledgement packet back to the source node and computes any information/ data stored in the packet.

4.5 ROBUST simulation results

Table 4.1: Simulation Parameters

Network size (Scalability)	25, 50, 75, 100, 125
Network size (Mobility effect)	50
Node velocity (Scalability)	1m/s
Node velocity (Mobility effect)	0m/s, 5m/s, 10m/s 15m/s, 20m/s
Node density	100 nodes per $1000m^2$
Mobility model	Random Way Point
DHT architectures simulated	OpenDHT, ROBUST
Routing protocols used	AODVv2, OLSRv2
Data packet payload size	512 bytes
MAC layer	802.11*b*
Link bandwidth	11Mbit/s
Maximum transmission range	250m
Types of traffic	UDP
Simulation time	1000 sec
DHT data distribution	Random
Number of DHT *GET* requests	10/sec
Data synchronisation interval	3 sec
Leaf set update interval	4 sec
Neighbour ping interval	5 sec
Super peer change RTT threshold	90ms
Cluster beacon interval	10 sec
Proximity synchronisation interval	60 sec

In order to evaluate the two proposed protocols the author has developed an application for the network simulator ns-2 [51] implementing both ROBUST and OpenDHT. The simulator implements all factors of the DHT including replication, pings, leaf sets, puts/gets, clustering, caching, churn via mobility and joining. Table 4.1 shows the simulation parameters where the network area size is dependant on the number of node in order to maintain the

node density. The author has simulated two scenarios in order to investigate how the protocols perform under different conditions. The first scenario — referred to as the *Scalability* scenario — is aimed at evaluating the protocol performance while increasing the number of peers in the network. The second scenario — aptly named the *Mobility effect* scenario — investigates how the protocols perform while increasing the velocity of the moving peers. The author has specifically chosen these scenarios as the differing attributes investigated represent two of the main challenges one must address in MANETs. All graphs show the standard deviation and 95% confidence interval.

The ROBUST DHT is fully simulated and compared with OpenDHT. Additionally each protocol was simulated on both the OLSRv2 [47] and AODVv2 [49] routing protocols in order to evaluate how the protocols are affected by different MANET routing protocols, thus also providing an insight into how truly heterogenous DHT architectures can be in terms out routing protocol support.

The author has run the simulations using the Random Way Point mobility model. The network size in the *Scalability* scenario starts at 25 peers and is incremented sequentially by 25 peers up to a maximum of 125. The specific choice of these numbers for network size should be elaborated. The authors of [99] in Chapter 1 discuss the scalability of MANETs. When discussing the nodes in the MANET competing for usage of the same channel they state that the unfortunate conclusion is that under certain reasonable assumptions, purely omnidirectional ad hoc networks cannot grow beyond certain fairly restrictive limits. This is due to the channel capacity per node of $O(\frac{C}{\sqrt{N}})$ where C is the total channel capacity and N is the number of nodes within transmission range [99]. An obvious point here is that the capacity can only be limited by the number of nodes within transmission range of each other. Given a node density of 100 nodes per $1000m^2$ and a transmission range of $250m$ 125 nodes is clearly reaching the upper bounds of what is possible given a standard 802.11 channel capacity of $20mhz$. A velocity of 1m/s was chosen to represent a brisk walking pace.

Calibration of the ns-2 [51] simulator has been carried out in the paper [100] where the

authors conclude that the packet delivery ratios, and network topologies are accurately represented in ns-2, once the simulation parameters are properly adjusted. Our simulation parameters for the DHT side of the implementation were taken from the real world OpenDHT source code [101] and are further calibrated in the paper [102].

The *Mobility effect* scenario was initially run using a static network with the peer velocity increasing sequentially with increments of 5m/s up to a maximum of 20m/s which is at the top end of a practical MANET — such as a car. The simulations assume *random* DHT data distribution. Table 4.1 also shows the

- number of DHT *GET* requests per second,
- *data synchronization interval*: indicates the period between choosing a random peer from a peer's leaf set and replicating its key value pairs,
- *leaf set update interval*: the period in which a peer chooses a random peer from its leaf set and it exchanges the updated IDs from the intersection of their leaf sets,
- *neighbor ping interval* : the period between a peer pings its leaf set peers (neighbours) to check if they are still associated with the network,
- *super-peer change RTT threshold*: indicates the maximum tolerable difference in delay — between a peer's current super-peer A and one newly discovered B — before which a peer will join the B's cluster,
- *cluster beacon interval*: is the period between broadcasts of cluster beacons from a super-peer and,
- *proximity synchronization interval*: is the frequency which a peer compares the round trip times of super-peers in order to ascertain the need to join another cluster.

4.5.1 Network overhead

4.5.1.1 Scalability

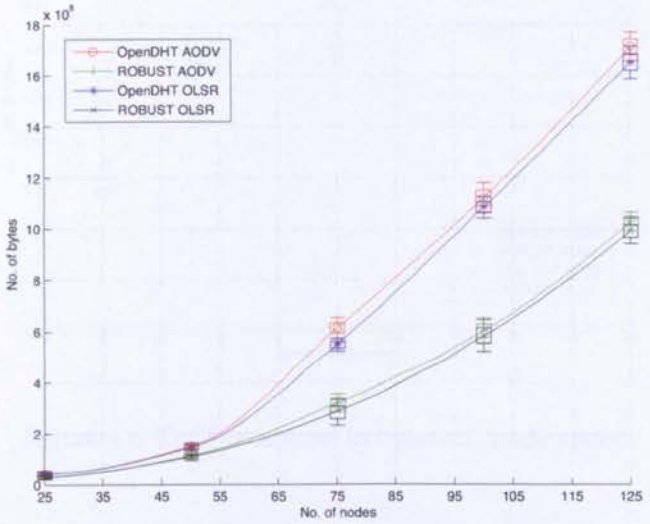

Figure 4.5: DHT overhead in bytes vs. scalability.

Figure 4.5 shows the network overhead caused by each of the protocols in terms of bytes. It can be clearly seen that as the network size increases, thus so does the overhead. This is mainly due to the increase of the number of replications being transmitted as well as longer routes due to each lookup having to traverse more physical nodes. One can see that OpenDHT has higher overhead in both the case of running AODVv2 and OLSRv2, this is due to the lack of proximity awareness in the architecture. Overall both DHTs perform slightly better on OLSRv2 as the physical nodes are moving meaning more route breakages, thus the additional overhead in AODVv2 needed to keep refreshing routes causes AODVv2 to behave like a broadcast-based flooding protocol, hence creating more overhead.

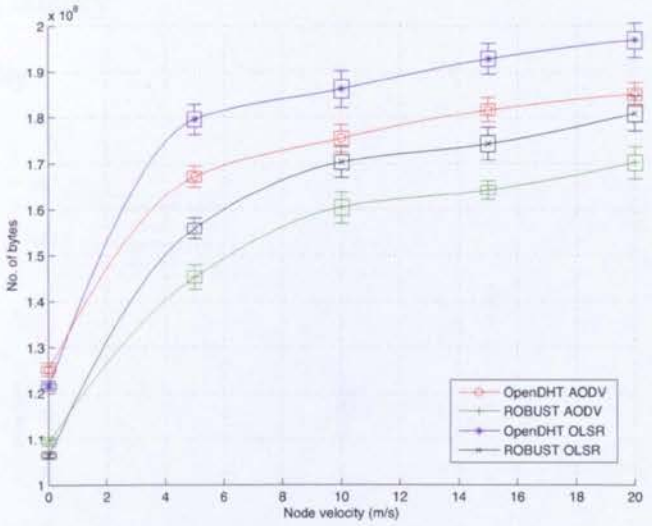

Figure 4.6: DHT overhead in bytes vs. node speed.

4.5.1.2 Mobility effect

In Figure 4.6 the network overhead in shown as a function of MANET node mobility speed. Straight away one can see that node mobility causes much more overhead in both protocols when compared with increasing network size. This is due to more frequent route breaks causing data having to be retransmitted multiple times due to packet loss. One can see that ROBUST outperforms OpenDHT by a significant margin, this is again due to the increased proximity of neighbour peers meaning less hops to traverse when replicating data.

4.5.2 Network latency

4.5.2.1 Scalability

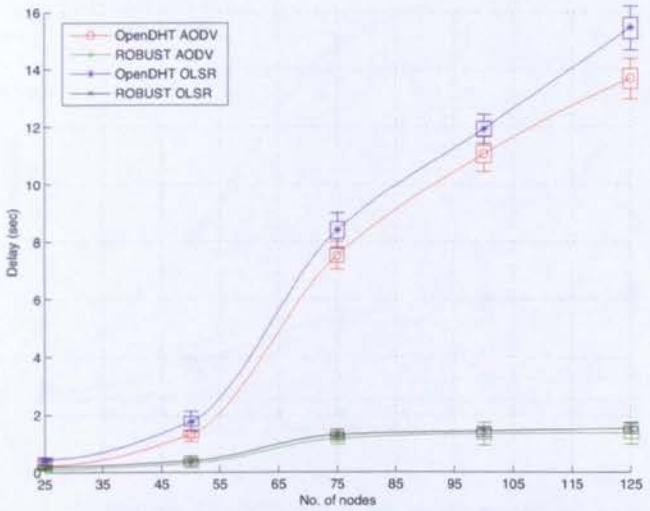

Figure 4.7: E2E delay vs. scalability.

Figure 4.7 shows the average end-to-end delay of a DHT lookup as a function of network size. In this scenario, OpenDHT has a much higher End-to-end (E2E) delay than ROBUST this is due to the combination of longer routes mixed with higher packet loss, thus meaning multiple retransmissions are needed for the packet to reach its destination, ending with the accumulation of delays. Here OLSRv2 has slightly more delay than AODVv2 when scaling due to the quick setup of new paths compared with the OLSRv2 MultiPoint Relay (MPR) re-election which takes place when an MPR node moves away from its selectors.

4.5.2.2 Mobility effect

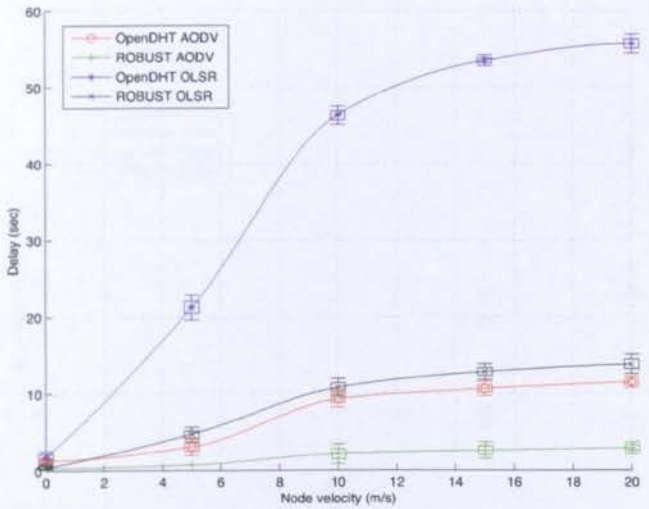

Figure 4.8: E2E delay vs. node speed.

Figure 4.8 shows the affect that node speed has on delay. As in Figure 5.7 the two factors of routing protocol and path length again seem to play a big role. AODVv2 appears to handle mobility the best in terms of delay with both protocols performing much better in terms of delay when used on the aforementioned protocol. This can be attributed to AODVv2 being able to immediately make use of routing information from intermediate nodes and thus learning new routes very quickly in addition to OLSRv2s MPR nodes moving outside of the connectivity range of their MPR selectors causing MPR re-electing and high delays. ROBUST DHT outperforms OpenDHT in this case for both routing protocols, this is due to the higher average path length for OpenDHT creating more chance that a packet needs to be retransmitted due to failure.

4.5.3 Packet loss

4.5.3.1 Scalability

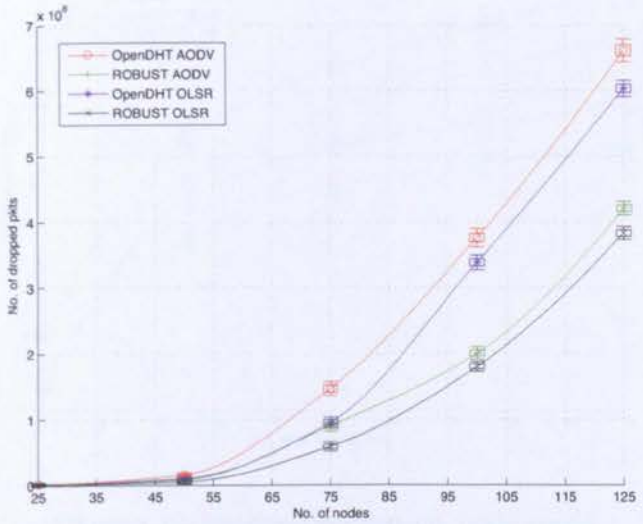

Figure 4.9: Dropped packets vs. scalability.

Figure 4.9 shows the number of dropped packets by each of the protocols studies with increasing network size. Both protocols perform similarly as AODVv2 has a fast path discovery mechanism and OLSRv2 has the potential to reroute around broken links. However OpenDHT has a higher number of dropped packets in both routing protocols due to its longer average path length when compared with ROBUST meaning more chance for a packet have a transmission failure along the route. ROBUST maintains shorter paths by taking advantage of the clustering mechanism described in Section 4.2.

4.5.3.2 Mobility effect

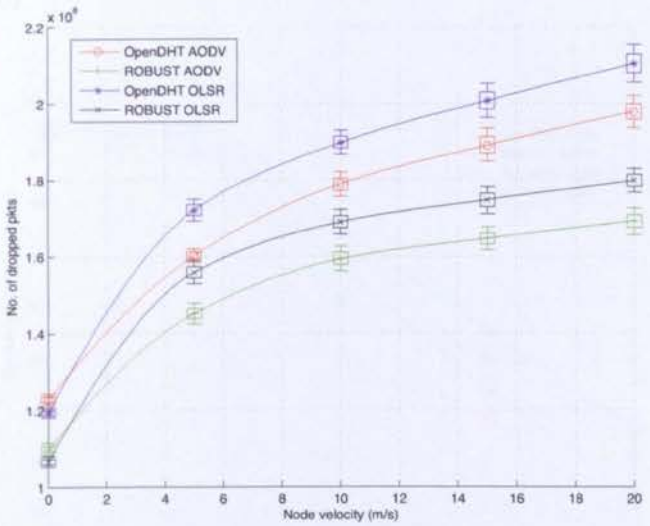

Figure 4.10: Dropped packets vs. node speed.

In Figure 4.10 the effects of node speed upon the number of dropped packets is investigated. When there is no mobility in the network there is a very low number of dropped packets. Here the affect of the chosen routing protocol really coming into play. OLSRv2 has significantly higher packet loss when compared with running the same DHT on AODVv2 which seems to handle mobility route breaks better than its proactive counterpart due to the quick discovery of alternative routes and dissemination of route breaks due to the Route Error (RERR) messages.

4.5.4 Lookup success rate

4.5.4.1 Scalability

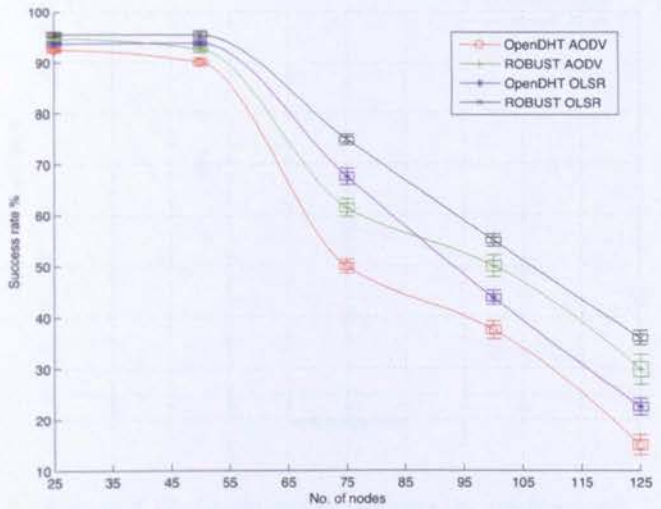

Figure 4.11: Lookup success rate vs. scalability.

Figure 4.11 shows the percentage of successful lookups against increasing network size. Both of the protocols perform well for up to 50 nodes network size, however as the network size increases, this causes longer average path lengths, giving a greater probability for a packet to be dropped along a route, thus creating a higher chance for a lookup to fail. AODVv2 performs worse here due to its greater percentage of dropped packets in this scenario as previously discussed in figure 4.9.

4.5.4.2 Mobility effect

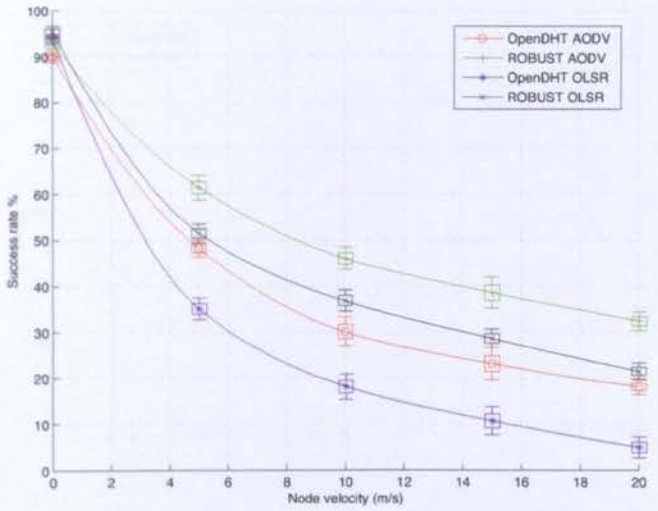

Figure 4.12: Lookup success rate vs. node speed.

Figure 4.12 investigates lookup success rate while increasing node speed. OpenDHT performs the worst here due to higher delays and packet loss caused by longer path length and route breakages meaning a high probability for a lookup to fail. OLSRv2 performs worse in both DHT protocols for the reasons previously discussed in figure 4.5.3.2.

4.5.5 Average path length

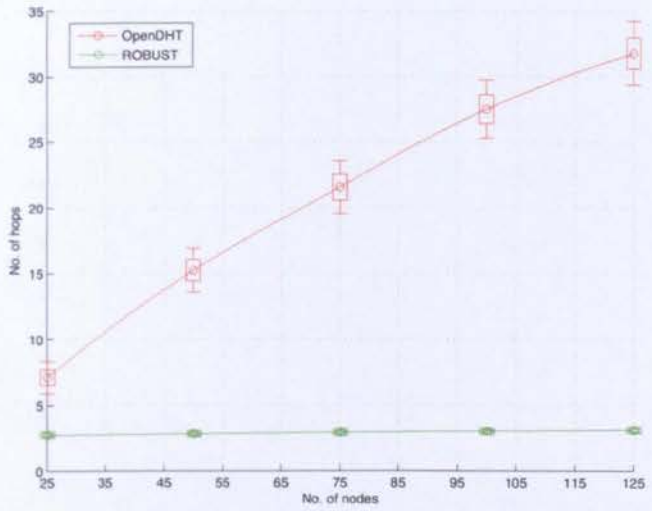

Figure 4.13: Lookup average path length.

In Figure 4.13 the average path length for each protocol is shown. Both protocols behave as expected here based on the model from Section 4.2, in ROBUST the path length will stay the same due to its hierarchical topology and strict cluster routing. However due to the algorithm employed in OpenDHT, path length scales logarithmically when increasing the number of peers in the overlay network.

4.5.6 Overlay Stretch

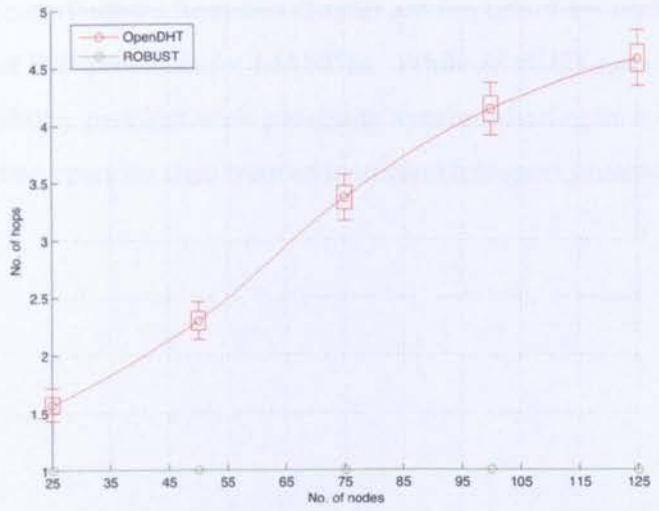

Figure 4.14: Logical path vs. physical path stretch.

Figure 4.14 displays the DHT lookup path stretch as computed previously discussed in the background of this thesis — namely overlay path length vs physical path length. ROBUST relies on broadcasting and obtaining IPs to DHT IDs to create an exact copy of the physical network at the DHT layer meaning no overlay stretch is present. However OpenDHT employs negligible Proximity Neighbour Selection (PNS) therefore resulting in long path lengths as described in Section 2.3 of this thesis.

4.6 Summary

In this chapter a new DHT architecture entitled ROBUST has been proposed in order to share and disseminate data and information throughout MANETs based on the P2P paradigm which both networks share. To this end the author has simulated the proposed architecture in the packet-level simulator ns-2. The results show that ROBUST DHT out performs the conventional OpenDHT in all scenarios due to the optimisations applied,

meaning ROBUST DHT could be used in scenarios where the underlying routing protocol cannot be modified.

The resulting contributions from this chapter are important for the future design and implementation of P2P protocols for MANETs. While ROBUST goes some way to addressing the scalability problem with proximity synchronisation, it is clear that a lot of the encountered discrepancies stem from an inefficient transport protocol which should be addressed.

Chapter 5

Proactive and reactive crosslayer P2P networks

In this chapter two Peer-to-Peer (P2P) network architectures will be proposed in this section based on differing underlying Mobile Ad hoc Network (MANET) routing protocols namely Optimized Link State Routing version 2 (OLSRv2) [47] and Ad-hoc On-demand Distance Vector version 2 (AODVv2) [49] in order to satisfy the research objectives RO1-RO4 and RO5 from Section 1.2 of this thesis. The reasoning behind developing two protocols is to shed light on how differing underlying protocols, in terms of their basic routing functionality, behave when integrated with P2P overlay features. As previous work investigating the two main areas — in terms of standardisation by the Internet Engineering Task Force (IETF) — of MANET routing, namely proactive and reactive, appear to be suited to different scenarios [39,40]. Our previous simulation results from Section 4.5 show that simply layering an existing P2P network such as OpenDHT [11] on top of a MANET routing protocol yields undesirable results due to high delay times and increased packet overhead caused by replication and P2P routing which is still not using the possible shortest path as Reliable Overlay Based Utilisation of Services and Topology (ROBUST) is oblivious to the underlying MANET routing protocol and exact network topology. While the benefit of using an integrated P2P-routing approach should be a lower traffic footprint when considering P2P

specific traffic, an obvious downside of this approach is that of *flexibility*. When integrating application functionality at the routing layer one has to use said protocols exclusively in order to take advantage of the functionalities available. Thus switching between different MANET routing protocols on a whim is no longer plausible without further investigation due to the complexity arising from interoperability. Our aim for this chapter is to guide development in future implementations of P2P networks designed for applications on MANETs by presenting the pros and cons of each approach when combining P2P key-based lookup and MANET routing protocols. Our two presented P2P overlays are novel within the field and an improvement over the state-of-the-art as shown by our detailed simulation results.

5.1 Overview of design

To evaluate the integrated P2P networks and routing approach, one protocol from each of the two main areas of research on MANET routing protocols has been chosen. One is a proactive MANET routing protocol, namely OLSRv2, the other is a reactive MANET routing protocol, namely AODVv2. The proposed P2P networks which will be referred to as Proactive MANET DHT (PMDHT) and Reactive MANET Peer-to-Peer Overlay (RMP2P), from herein, have the overall aim of:

1. Decreasing stretch when compared to similar protocols from the field, defined as:

$$(\#Physical\ hops)/(\#Logical\ hops) \tag{5.1}$$

2. Maintaining lookup success rates when compared to similar protocols as defined as:

$$(\#Sucessful\ lookups)/(\#Total\ lookups) \tag{5.2}$$

3. Decreasing the average physical path length for a lookup, thus reducing delay and reducing the possibility for errors.

4. Reducing the number of packets sent in the network related to P2P operations (signalling traffic and P2P key-based lookup traffic).

5.1.1 Proactive DHT

The proposed proactive Distributed Hash Table (DHT) architecture integrates the DHT functionality of OpenDHT [11] — in terms of all functionality except DHT routing and including the modifications below — within the MANET routing protocol OLSRv2 and is subsequently titled PMDHT. This specific proactive MANET routing protocol was chosen due to its predecessor being the IETF Request For Comment (RFC) [42] and now adhering to RFC 5444 message and Type Length Value (TLV) formats, allowing greater interoperability. With the aforementioned aims in mind a logical step in designing a new DHT for MANETs is integrating the DHT functionality with the IP routing protocol being used. The author has identified areas of DHT protocols where adjustments can be made in order to create DHTs more suitable for MANETs.

One such adjustment in proactive DHTs such as OLSRv2 is utilising a modified version of the Common Group Aliasing (CGA) architecture as proposed by the authors in [103]. CGA uses ID prefix concatenation to create clusters of peers with similar ID in the DHT peer ID space. The advantage of this over the normal operation of assigning random DHT peer IDs comes into play when considering replication. Each node needs to replicate its key value pairs with each of its leaf set peers, so that if the replicating peer leaves the network (due to battery life constraints or wilfully) another node can take responsibility for the leaving peers DHT key value pairs. The further away a leaf set peer is located in the physical space, the more overhead is created in terms of number of packets and delay, as more physical hops are needed to relay the information.

The second differing feature of PMDHT is the way DHT lookups are performed. In most DHTs such as [12] lookups are performed based on a ring-like algorithm where a lookup traverses multiple nodes in sequence until the destination is found by matching the lookup key to the closest DHT peer ID available at each node along the lookup traversal path. This

normally equates to a maximum routing path of length $O(\log N)$ where N is the number of nodes in the overlay. In PMDHT lookups are piggybacked on to the OLSRv2 routing packets, so that signalling required to propagate routing information regarding Internet Protocol (IP) addresses also propagates DHT peer ID information. Thus the replication of data is avoided for much the same purpose and overhead is reduced. This feature is further elaborated in Section 5.2.1.4.

5.1.2 Reactive P2P Overlay

The proposed reactive P2P architecture again integrates the DHT functionality of OpenDHT [11] — in terms of all functionality except P2P routing and including the modifications below — but within the MANET routing protocol AODVv2 and is therefore aptly named RMP2P. AODVv2 was chosen due to it being the successor to the RFC [41] and adhering to the RFC 5444 message and TLV formats. As with the previously described proactive protocol, the reactive P2P protocol will also be integrated at the routing layer in order to minimise overhead and delay from the P2P application perspective. A description of the main integrated architecture follows.

Unlike the proactive protocol, where routable addresses are flooded to every node using Topology Control (TC) messages, no such functionality exists in reactive MANET routing protocols such as AODVv2 . Instead the protocol awaits transmission of a packet, then if no existing route to the designation is known, the protocol in a reactive manner, tries to find a suitable route by using expanding ring search flooding of the Route Request (RREQ) messages. The protocol sets an initial lifespan on each RREQ in order to avoid flooding the entire network. If no Route Reply (RREP) packet is received within that lifespan, the lifespan is increased and the RREQ packet is again flooded into the network. This process is repeated until a suitable route to the destination is found.

In order to lower the overhead cased by the P2P network it is necessary to integrate P2P lookups with routing lookups. If a DHT which uses PUTs (utilising the numerical DHT ring to find a node responsible for a said DHT key to store the key value pair) is

overlaid on top of a MANET routing protocol such as AODVv2 , the DHT roughly mirrors the same functionality of part of the MANET routing protocol (namely neighbourhood discovery), except at the application layer. This creates an unnecessary overhead as the two protocols require similar information (IP address and routing information). There is an argument for simply adding P2P peer ID information in *HELLO* packets, then using this information to populate leaflets, it is intuitive however that further improvements can be achieved by tighter integration with AODVv2 as is proved later in this chapter using simulations.

One can see that RMP2P is an unstructured P2P network as it utilises flooding for lookups rather than conventional key-based routing. It still provides DHT-like functionality however as each peer locally stores key value pairs in a hash table.

The other issue the author will address is redundancy of the stored data. This will obviously incur an overhead and so should be optimised in order to efficiently replicate data with nearby nodes.

5.2 Detailed description of protocols

In this section a detailed description of the two proposed protocols PMDHT and RMP2P is given. The signalling behaviour of the aforementioned protocols will be described and the subsequent overhead cost shall be evaluated.

5.2.1 Proactive DHT

As previously mentioned the proactive DHT will be based on integrating OLSRv2 with DHT functionality at the routing layer in order to achieve the research objective RO2 set out in Section 1.2 of this thesis. The functionality of each of the procedures has been chosen in order to best fit the routing protocol while delivering the same functionality which a DHT such as [13] provides.

5.2.1.1 Common Group Aliasing (CGA)

Common Grouping (CG) refers to the process of grouping devices, which are also closed physically in the underlying physical network, together in the P2P overlay network,. To achieve this goal it is proposed that the 128-bit peer ID space is partitioned equally. For example peers beginning with the prefix ID 1..2..3..4..5..6..7..8..9.. will be partitioned into common groups based on their physical position in the overlay network. This is realised through the use of marker peers. These are peers whose own peer ID prefix is numerically closest to a set of marker keys. The marker keys in the above partitioned space would be 1000...2000...3000...4000...5000...6000...7000...8000... and 9000.

To establish marker peers in the creation of the overlay, peers initially send a marker request to the marker key closed to their peer ID, for example a peer with an ID of 2349... would send a marker request to the marker key 2000... This request would then be routed to the peer closest to the marker key using the normal OpenDHT routing method to distribute the keys. The marker peer is therefore determined by the peer, which does not have a peer closer to the marker key in its leaf set. Once the marker request has arrived at the peer with the ID closest to that of the marker key, the peer then declares itself as a marker peer. Initially this is accomplished by the peer changing its peer ID prefix to that of the marker peer.

Once a marker peer has been established it then proceeds to send out a periodic marker beacon. When a peer becomes a marker peer primarily it does not have any peers within its CG set. Therefore it is proposed that the initial marker peers send out a marker beacon with a two-hop Time-To-Live (TTL). The beacon is a basic packet, containing the ID of the marker peer, along with information relating to the distance the beacon has travelled in hops.

If a peer receives a marker beacon and it is not part of any CG will part the overlay network and then rejoin the network with its new ID. The new ID is calculated as its original peer ID with x digits added to the start to match the marker peer's first x digits. The digits are added instead of replaced to avoid peer ID duplicates.

Once a marker peer has a CG set of more than N peers it will then switch to only sending marker beacons to peers within its CG. peers participating in a CG will then forward the marker beacon to any peers within a one-hop radius in the physical underlying network. If a peer receiving a marker beacon in this situation is not already participating in a CG it will immediately join the CG in the aforementioned manner.

If a peer receiving a marker beacon is already participating in a CG however, it will consider by examining the hop count how far the beacon has travelled. If the hop count of the marker beacon is less than that of its hop count to its current marker peer, it will leave its current CG and rejoin the overlay in the new CG. Once the marker beacon has reached a peer not participating in the CG from which the peer was sent, it will process the beacon information and then drop the packet. This is a required because it is not desirable to broadcast a marker beacon throughout the whole network, hence one hop from the CG should suffice in keeping the CG peers physically close to each other both in the overlay network, and the underlying network.

Peers send periodic probes to their marker peers to check if the marker peer has left the network or has failed. If a marker peer does not respond for a predefined number of probes, the marker peer is regarded as having left the overlay P2P network. The current peer will then immediately send a marker request. The peer which then receives that marker request will become the new marker peer.

In CGA entities are defined as different classes of devices. In CGA three different types of entities are considered along with how they interact in the overlay network. These entities are

- Mobile devices with enough power to route and forward packets while maintaining minimum packet loss due to hardware constraints.

- Devices where power is limited and therefore resources are less abundant. In these cases overlay routing and packet forwarding is inefficient due to packet loss caused by hardware constraints.

- Intermediate devices, which do not participate in the overlay. However, these are still part of the underlying MANET and they will therefore be used to route packets using network layer routing protocols such as AODVv2 and OLSRv2.

In CGA these devices are differentiated between in order to serve queries more efficiently. It is assumed that when building an implementation for a specific platform, e.g. x86 or iPhone, the software itself will know if it is a low powered device like an iPhone or a device with more resources such as an x86 architecture based laptop.

To create more efficient overlay routing and to save vital battery power on low powered devices such as Personal Digital Assistants (PDAs) and smart phones, it is proposed that these devices are classed as selfish peers. Such peers would not route packets in the overlay network primarily to save precious battery power but also because of their small antenna which is prone to packet loss.

At the network layer, selfish peers could be regarded as unfavourable routing paths and routing protocols could be enhanced to choose alternative routes. However, if no alternative paths exist, the selfish peers could be used as relays. Such a scheme would preserve a higher overall network energy level prolonging network life as well as offering a better Quality-of-Service (QoS) for data communications.

On the other hand, devices which have sufficient resources will be enabled in the CGA architecture to forward and route lookup queries where necessary. This will be achieved using the OpenDHT implementation where lookups are routed to the peer with the nearest peer ID prefix to the specified key prefix. However there are differences, which are discussed in the following sections.

5.2.1.2 Clustering procedure

Clustering can be used to create physical areas in the MANET where nodes have similar DHT peer IDs by using prefix concatenation as described in [103].

CGA improves DHT replication overhead by electing super-peers for each cluster by using a pre-defined marker key function in order to elect super-peers based on which

node is numerically closest to a set of marker keys. CGA also defines different classes of peers which may or may not act as forwarders in the network based on battery life and computing power. In order to reduce the complexity of the protocol, instead of super-peers being elected, the closest nodes to an instantiation specific — in terms of an administrators configuration — set of pre-defined DHT peer ID keys, denoted as *Clstr_keys* would become super-peers. There is one cluster-key assigned to each cluster. Once a super-peer has been assigned it, will begin broadcasting its DHT peer ID at specific time intervals. Any peer within a one hop radius will receive the broadcast packet which will be referred to as the beacon packet denoted by β_i, where i is the corresponding cluster. The beacon packet will continue to be relayed by each peer until a peer closer to another beacon (in terms of packet latency — a peer will immediately ping a super-peer after receiving a packet) receives the packet, at which point the packet will not be forwarded. This method should create clusters of peers with similar DHT peer IDs at a relatively low cost in terms of packet overhead.

The overhead for creating clusters in PMDHT can be described as:

$$\sum_{i=1}^{Clstr\_keys} \beta_i * No\_RTT_i * T_\beta$$

Where No_RTT_i denotes the number of times a cluster beacon is forwarded in the network until super-peer $superpeer_i$ has a latency higher than $superpeer_{i+n}$ (any other super-peer) at a given node and T_β is the time period between sending beacon packets.

5.2.1.3 Join procedure

The join procedure can be described as a superset of two procedures, firstly the empty overlay join procedure, and secondly the existing overlay join procedure. The empty overlay join procedure occurs when no overlay currently exists. This is realised when a node receives no DHT packets within a $2*T_\beta$ time period as defined in Section 5.2.1.2. If this is the case the node will hash it's IP address using Secure Hash Algorithm 1 (SHA-1), generating a 160-bit ID hash. As this is the first node, it will then become a super-peer as described in

Section 5.2.1.2.

In the case that an overlay node receives a DHT beacon in $2 * T_\beta$ as defined in 5.2.1.2, the node will initiate a join request to the source of the beacon. The join request to the super-peer of cluster i is denoted by $Join_i$. This will include the 160-bit SHA-1 hash of the peers IP address. The super-peer whom is the source of the request will then generate a new DHT peer ID for the joining peer by concatenating its own DHT prefix as described in Section 5.2.1.2. The new DHT peer ID is then sent in a $JoinREP_i$ packet to the joining peer. The leaf set table can then be populated based on DHT peer information from the Topology Information Base in OLSRv2, and can be updated if any changes are detected. The peer can then begin replicating the data of peers from its leaf set as shown in Section 5.2.1.4 below. The total overhead of existing overlay join procedures, can be described as:

$$\sum_{i=1}^{N} Join_i + JoinREP_i$$

Where N is the number of peers in the DHT. The overhead is further evaluated using simulations in Section 5.3.1.

5.2.1.4 Lookup procedure

As discussed in Section 5.1.1 PMDHT tightly integrates lookup functionality into the routing layer of OLSRv2 [47]. As a prerequisite for looking up DHT keys, the peer must first find the peer responsible for the DHT key. In the IETF RFC 5444 [104] TLVs are defined. TLVs allow extensions to be added on to existing routing packets. PMDHT will introduce a new packet TLV in *HELLO* messages as also defined in OLSRv2 which will contain the DHT peer ID of the source node. MultiPoint Relay (MPR) nodes as defined in OLSRv2 will then collate this information. Another packet TLV will be used to extend the *TC* message with the DHT peer ID corresponding to each node for which a routable address is included in the original *TC* message. This information will then be flooded in the network using the controlled flooding mechanism known as MPR flooding in OLSRv2.

The information contained within the DHT peer ID TLV will then be processed by each

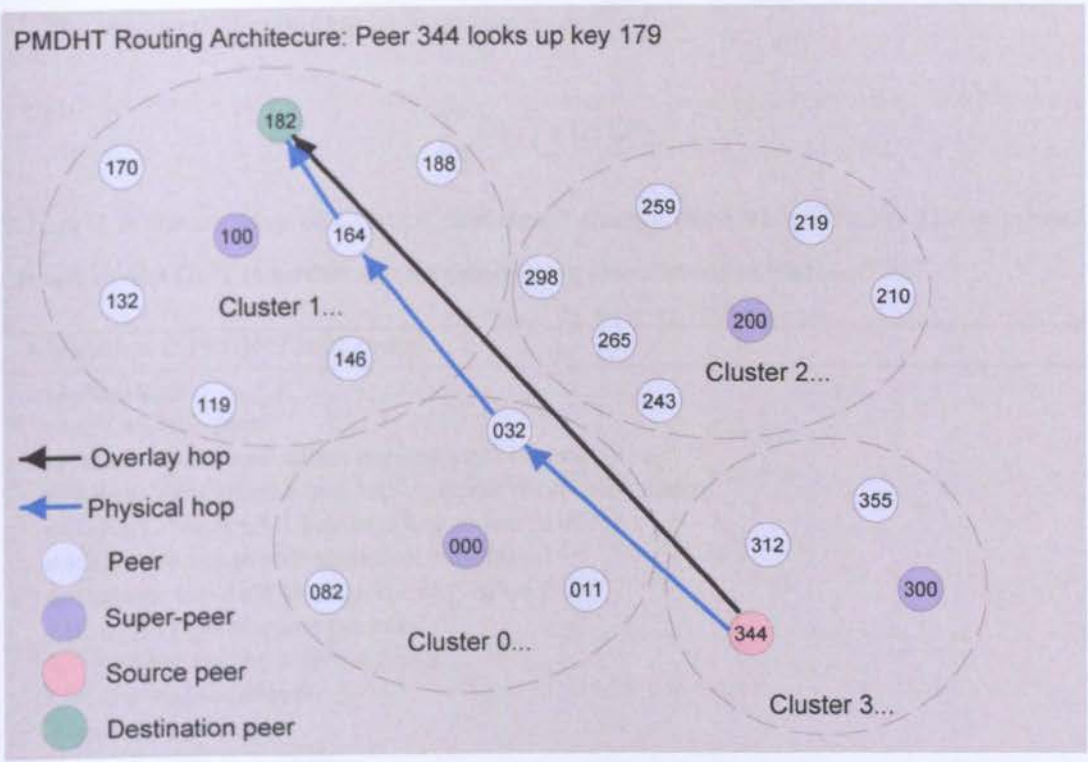

Figure 5.1: The PMDHT routing architecture

node and entered into an extended Topology Information Base. When a node then wishes to lookup a specific DHT key, the Topology Information Base can then be consulted to find the closest numerically matching DHT peer ID to the DHT key. The DHT lookup packet *GET* — a get is how a DHT key-based lookup request is referred to — can then be sent directly to that node via its IP address. The routing protocol OLSRv2 will then decide the most appropriate route for the packet to the destination. Once the *GET* packet has been received at the destination node, the value of the key can then be sent back via a GET Reply (*GREP*) packet directly to the source node using OLSRv2 routing based on the source node IP address. The resulting logical number of hops in this algorithm can be described as $O(1)$ as like OLSRv2 all routing state information is available at every peer. The described lookup procedure is graphically represented in figure 5.1.

The overhead of executing lookups can be defined as:

$$\sum_{j=1}^{\lambda} GET_j + GREP_j$$

Where λ is the number of lookups performed during network's lifetime. The overhead caused by the DHT is further investigated using simulations in Section 5.3.1.

Algorithm 1: PMDHT replication

```
Definitions
peer_i: a DHT peer
synReq: synchronisation request packet sent by peer_i
synRep: synchronisation reply packet received by peer_i
dataKEY: the SHA-1 key of a key value pair
data: a key value pair stored in datastore
datastore: the data storage for key value pairs
get: a DHT get request packet
T: synchronisation interval timer
LS_i: the leaf set of peer_i
Algorithm for peer_i
if time is equal to T_r then
    choose a random node peer_{i+n} from LS_i
    send synReq to peer_{i+n}
else
    if packet received type is synRep then
        foreach dataKEY in synRep do
            if dataKEY not in datastore then
                send get to peer_{i+n} for hash dataKEY
            end
        end
    end
end
Algorithm for peer_{i+n}
if packet type is synReq then
    foreach data in datastore do
        if data should be stored at peer_i then
            add dataKEY to synRep
        end
    end
    send synRep to peer_i
end
```

5.2.1.5 Replication procedure

Replication in PMDHT is handled exactly the same as the replication algorithm described in [11] with regards to selecting a random peer from its leaf set table and executing a synchronisation request. This has the affect that in $O(\log LS)$ — where LS is the leaf set size — number of iterations each peer will have updated its *datastore* based on information from its $leafset_i$ peers. The synchronisation operation consists of the algorithm shown in algorithm 1.

The main difference in replication between PMDHT and OpenDHT is the leaf set peers in PMDHT should be closer (in terms of hops) than those in OpenDHT as PMDHT will have leaf set peers within the cluster, where as OpenDHT leaf set peers are random due to the hashing of IP address to generate peer IDs. In a network without proximity between leaf set peers the synchronised data, which is the largest in terms of packet size, can be relayed over many hops and even traverse over the same physical nodes. The replication procedure can be described as:

$$\sum_{i=1}^{N}(SynReq_i + SynRep_i) + (\sum_{k=1}^{\phi} GET_k + GREP_k) * T_r$$

Where $SynReq_i$ is the synchronisation request packet sent by a peer $peer_i$ and $SynRep_i$ is the synchronisation reply packet received by $peer_i$. The number of lookups for keys in $SynRep_i$ not found in $peer_i$'s local storage, are represented by ϕ. Each GET_k is the subsequent request for the data[1], each $GREP_k$ is the key value pair to be stored at $peer_i$ and T_r is the time period between which replication occurs. Given that any $GREP_k$ contains the actual data, this procedure has a high overhead. The overhead caused by the DHT is further investigated in Section 5.3.1 using simulations.

[1]Not cached in the local storage.

5.2.1.6 Opportunistic caching

Due to intermediate nodes forwarding packets in MANETs this gives us the opportunity to cache DHT key value pairs along a route from the destination to the source of a lookup. In PMDHT when a peer receives a packet of type *GREP* it automatically caches that data into data storage separate from its main DHT datastore. This data can optionally have a timeout depending on implementation in case there is a desire for key value pairs to be replaceable. There would be some extra processing required to check each *GREP* and cache the data, however this would be negligible as the packet will need to be retransmitted to the next hop regardless, so the extra processing would simply be in storing the data which is a trade-off.

It is assumed that a lookup has been executed once, by using a route r, thus creating an opportunity for caching. This also represents a *worst case scenario* in the sense that the smallest number of peers have cached the replied data.

$\mathcal{P}_r := \{peer_{r_1}, peer_{r_2}, \ldots, peer_{r_{N_r}}\}$ is defined as the set of peers that r traverses where N_r is the number of peers along the route. Due to the broadcasting nature of the wireless medium, peers physically close to each other ("neighbours") can overhear transmitted packets. Therefore *GREP* packets are cached by such neighbours and if a neighbour receives the same lookup as the cached data, during the networks lifetime, they can reply to it directly. In this way, lookup latency and overhead are reduced. Thus, each peer $peer_{r_i} \in \mathcal{P}_r$, is defined a number of neighbours denoted by $v^{peer_{r_i}}$.

As there is only a finite number of peers in the DHT, the probability of getting a cache hit for a specific key value pair can be defined as:

$$(|\mathcal{P}_r| + \sum_{i=r_1}^{r_{N_r}} v^{peer_i})/N \tag{5.3}$$

where N is the number of peers in the network.

5.2.1.7 Churn considerations

When considering churn rates, a major factor is how recovery is dealt with. Recovery in terms of churn can be defined as how quickly changes are made to peers in the DHT so that the leaving peer's key value pairs are routed to the correct peer now responsible for said keys. In the paper [11] the authors investigate what they describe as proactive recovery vs reactive recovery. Proactive recovery consisting of notifying other peers in a peer's leaf set of the departure of the leaving peer, and reactive recovery consists of periodically pushing and pulling leaflets from random leaf set peers as is described in the previous Section 5.2.1.5.

In PMDHT recovery is dealt with the same way as when a node leaves the network according to the routing protocol OLSRv2. Thus when a peer leaves the DHT it will be removed from the Topology Information Base of neighbour nodes within a certain timeout specified by the implementation if a *HELLO* packet has not been received from the neighbour within that period. The peer will be removed from the wider network when the MPR node for the leaving peer does not receive a *HELLO* message within the previously mentioned timeout. Thus the leaving peer will no longer be included in TC messages propagated through the network by the MPR node.

The non-zero convergence time of OLSRv2 means that when a peer sends a *GET* for example, it cannot be sure that the peer still exists in the network. However as the packet is relayed closer in the physical network to where the leaving peer was last seen geographically, the closer peers are aware that the node is no longer in the network. The closer peers can then check the *GET* packet for the DHT key which is being looked up and reroute the *GET* packet accordingly based on the latest information from the Topology Information Base. In Section 5.3 below the effects of mobility on the DHT is investigated. Mobility is much like churn in that when a peer moves from one cluster to another — due to having a lower latency to a super-peer other than the local one — it has to essentially rejoin the DHT after leaving the original cluster, thus the mobility results are taken to give an indication of how well the DHT can handle churn.

Algorithm 2: PMDHT Cluster changes

Definitions
peer$_i$: a DHT peer
superpeer$_i$: a DHT superpeer
LS$_i$: the leaf set of *peer$_i$*
beacon: a super-peer DHT peer ID beacon
part: a DHT part packet
LSpart: a DHT leaf set part packet
sync: a DHT synchronisation packet
get: a DHT get request packet
GREP: a DHT packet containing key value pairs
noRTT: the number of RTTs received from a super-peer within $4 * T_\beta$
join: a packet indicating a peer joining the DHT
hashIP: the SHA-1 hashed IP of a joining peer
leafset: the leaf set of a peer or super-peer
clusterset: the peers whom are members of a specific cluster
RTT: *beacon* packet round trip time

Algorithm *for peer$_i$*
if *beacon is not from current_superpeer* **then**
  if *new_superpeer* **RTT** < *current_superpeer* **RTT then**
    new_superpeer noRTT += 1;
    if *noRTT* > 2 **then**
      send *part* to *current_superpeer*;
      send *join* to *new_superpeer* containing *hashIP*;
    end
  end
else
  current_superpeer RTT = *RTT* from *beacon*
end

Algorithm *for current_superpeer*
if *packet type is part* **then**
  foreach *peer$_{i+n}$ in peer$_i$_leafset* **do**
    send *LSpart* to *peer$_{i+n}$* containing *peer$_i$* DHT peer ID;
  end
  if *peer$_i$ in leafset* **then**
    remove *peer$_i$* from *leafset*
  end
  remove *peer$_i$* from *clusterset*
end

5.2.1.8 Cluster changes

These will occur in the DHT if the MANET is not moving in clustered groups. Depending on the speed of the nodes moving in the MANET cluster changes could be frequent or rare. When a cluster change does occur it is dealt as shown in algorithm 2.

The overhead for all cluster changes given a specific time period can be described as:

$$\sum_{\delta=1}^{\sigma} part + (LSpart * LS_\delta) + Join_\delta + JoinREP_\delta + SynReq_\delta + SynRep_\delta + (\sum_{k=1}^{\phi} GET_k + GREP_k)$$

Where σ is the number of cluster changes during the lifetime of the network. The *part* packet is sent to a super-peer denoting that the $peer_i$ is leaving the network. The *LSpart* packet is sent to each node in $peer_i$'s leaf set, which is of size LS_δ. The $Join_\delta$ packet communicates to the new super-peer that $peer_i$ is joining the new cluster. The $JoinREP_\delta$ packet contains the $peer_i$'s new DHT peer ID. $SyncREQ_\delta$ is the synchronisation request packet, ϕ is the number of lookups for keys in the $SyncREP_\delta$ synchronisation reply packet not found in $peer_i$'s local storage. Each GET_k is the subsequent request for the data and each $GREP_k$ is the key value pair to be stored at $peer_i$.

5.2.1.9 Super-peer election

As previously mentioned, a modified version of the CGA technique is used to define which nodes are super-peers. The main difference here being instead of election based on metrics such as; remaining battery, throughput and latency, super-peers are chosen based on their DHT peer ID. The reason behind this is due to the complexity caused during election, and the frequency which this could happen where nodes are constantly moving in a MANET.

Administrators can configure a set of pre-determined hexadecimal keys such as

κ{000.., 001.., 002... 00F..} the node closest to such a key would broadcast the *Beacon* packet defined in Section 5.2.1.2. The new super-peer would keep track of peers within its cluster by maintaining a Cluster set to be used when peers leave the cluster as shown in algorithm 2.

5.2.2 Reactive P2P Overlay

In this section the proposed reactive MANET routing based P2P overlay protocol is described in detail. As previously stated the reactive P2P protocol is based on the reactive MANET routing protocol AODVv2. Due to the fact that in AODVv2 the algorithm is purely focussed on finding a route on-demand, the signalling required when the network is idle or previous routes have been established, is very small. This has the affect that when integrating the P2P network with AODVv2, only a small number of functions need to be focussed on, namely *lookup* and *replication*. The added functionality seen in PMDHT such as clustering and DHT peer ID prefixing is not needed as will be explained later in this section.

5.2.2.1 Join procedure

In RMP2P there is no specific join procedure as such, because the lookup protocol is based on an unstructured P2P flooding algorithm peers do not maintain leaf sets and hence do not need to notify peers in order for the lookup procedure to function as described below in Section 5.2.2.2. In order for redundancy, a node should however begin the replication procedure immediately when joining the P2P network. The replication procedure is examined in detail in Section 5.2.2.3.

Overall not requiring a join function does not decrease overhead much compared to PMDHT as the replication procedure which utilises the largest (data) packets is still required. It does however save time when joining the P2P network as the transmission of the *Join* packet and subsequent waiting for the *JoinREP* in PMDHT does add some delay.

5.2.2.2 Lookup procedure

RMP2P nodes will store key value pairs which have originated from themselves, thus the peers will not have IDs identifying themselves, only the data. The reasoning for this is based on the expanding ring search algorithm which is used on AODVv2. RMP2P will extend the functionality of AODVv2 to support lookups within the routing protocol packets in order to lower overheads.

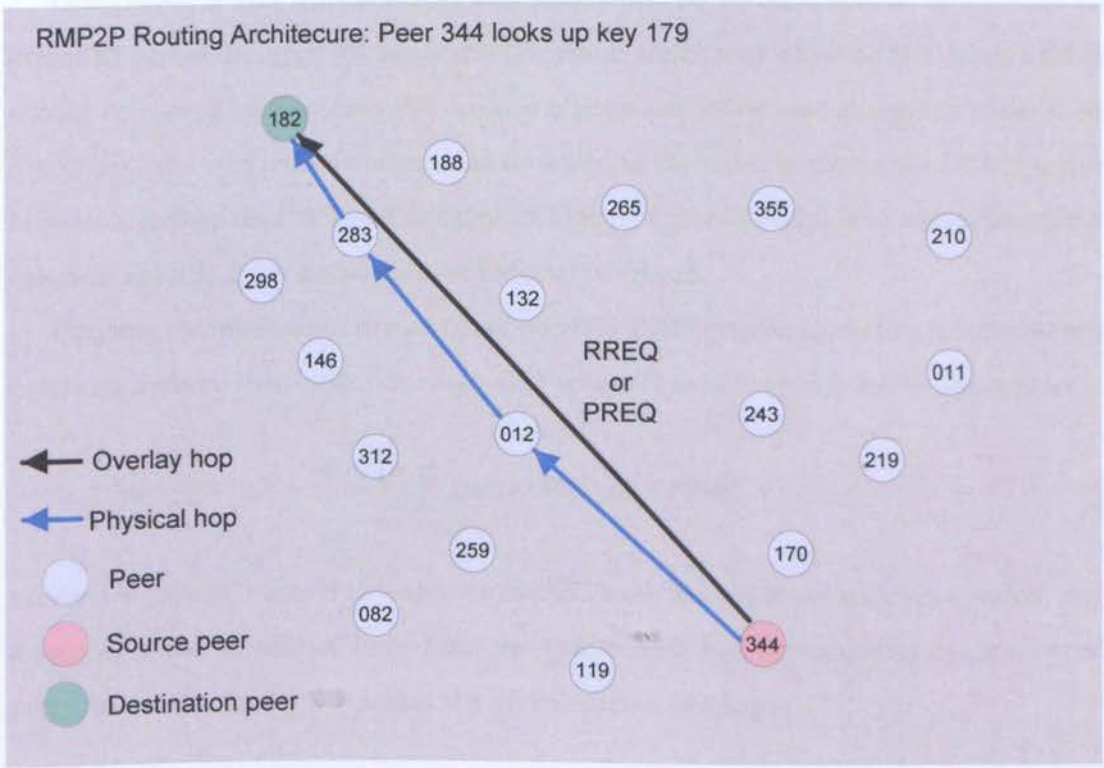

Figure 5.2: The RMP2P routing architecture

A P2P lookup TLV can be used to extend the RREQ packet. The lookup TLV will contain the key of the requested lookup. Upon a node responsible for said key receiving the RREQ, the matching value for the key will be sent back to the source node in a Peer-to-Peer Reply (DREP) packet in order to be processed by RMP2P. The DREP packet will be routed using the AODVv2 routing protocol the same as a normal data packet. If a RREQ is

received before the RREQ lifespan is expired, and no DREP packet has been received, the node will then issue a Peer-to-Peer Request (DREQ) packet with an increased lifespan akin to the functionality of an RREQ but with the destination P2P key instead of an IP address. The DREQ will then be crosschecked against each P2P key at every node at which it is received. An example of RMP2P routing can be seen in figure 5.2 where the overlay and physical routing follow along the same path.

One caveat of this feature comes into play when no RREQ has been sent within an extended period. In order to negate this limitation which may cause added delay, a timer should be started as soon as a P2P lookup is requested to be sent at a given node. If no RREQ has been sent within the required time period the node should send a DREQ packet in order to lookup the P2P key. The length of the timer could be implementation dependent based on specific delay demands over reduced overhead.

Defining l as the maximum number of hops the DREQ packet should be forwarded, the overhead induced by a lookup for each GET where l has not expired can be denoted as:

$$\sum_{i=1}^{\lambda} \sum_{r=1}^{l} (peers * PREQ_r) + PREP_r$$

Where λ is the total number of lookups in the P2P network given a specific time period, ring r equates to the number of hops from the source node and *peers* denotes the number of peers reached by the DREQ packet at a given number of r hops.

5.2.2.3 Replication procedure

Replication based on nodes with a numerically close P2P peer ID such as in PMDHT is not required as the P2P lookup algorithm is no longer based on the DHT-based Plaxton prefix routing [56] algorithm i.e. matching a DHT key to the relevant closest DHT peer ID. Therefore nodes can simply pick a random peer from its neighbour set which is populated by AODVv2 using the *HELLO* message defined in IETF RFC 6130 [66] and sends a *synReq* message to the peer, P2P keys found at the random peer and not currently stored

Algorithm 3: RMP2P replication

Definitions
peer$_i$: a peer
synReq: synchronisation request packet sent by *peer$_i$*
synRep: synchronisation reply packet received by *peer$_i$*
dataKEY: the SHA-1 key of a key value pair
data: a key value pair stored in *datastore*
datastore: the data storage for key value pairs
get: a P2P get request packet
T: synchronisation interval timer
neighborset$_i$: the neighbour set of *peer$_i$* defined in [66]

Algorithm *for peer$_i$*
if *time is equal to T_r* **then**
  choose a random node *peer$_{i+n}$* from *neighborset$_i$*;
  send *synReq* to *peer$_{i+n}$*;
else
  if *packet received type is synRep* **then**
    foreach *dataKEY in synRep* **do**
      if *dataKEY not in datastore* **then**
        send *get* to *peer$_{i+n}$* for hash *dataKEY*;
      end
    end
  end
end

Algorithm *for peer$_{i+n}$*
if *packet type is synReq* **then**
  foreach *data in datastore* **do**
    add *dataKEY* to *synRep*;
  end
  send *synRep* to *peer$_i$*
end

on the source node can then be requested akin to PMDHT. The entire process is shown in algorithm 3. The overhead for this replication procedure can be calculated as:

$$\sum_{i=1}^{N} SynReq_i + SynRep_i + (\sum_{k=1}^{\phi} PREQ_k + PREP_k) * T$$

Where $SyncReq_i$ is the synchronisation request packet sent by $peer_i$ and $SynRep_i$ is the synchronisation reply packet received by $peer_i$. The number of lookups for keys in $SynRep_i$ not found in $peer_i$'s local storage, are represented by ϕ. Each $PREQ_k$ is the subsequent request for the data[2], each $PREP_k$ is the key value pair to be stored at $peer_i$ and T is the time period between which replication occurs.

Given that any $PREP_k$ contains the actual data, this procedure has a high overhead. The overhead caused by the DHT is investigated in Section 5.3.1 using simulations.

5.2.2.4 Opportunistic caching

Caching can be achieved in RMP2P much the same as described in Section 5.2.1.6, this is because both protocols rely on the same basic principle of broadcasting messages. Therefore when a DREP is sent back to the source of the DREQ the MANET nodes along the route can cache the key value pair and immediately use it to begin serving lookups. As in PMDHT, the cache can include neighbours of nodes along the route as MANET is a broadcast medium. Thus the neighbours along the route would also "overhear" the DREQ being forwarded. Finally, the probability to find the lookup in cache, equals to the one derived for PMDHT as given by formula (5.3).

5.2.2.5 Churn considerations

Due to the flooding nature of RMP2P the effects of churn on the P2P network should be negligible. As the protocol does not maintain leaf sets as such and rely on leaf set peers in order to lookup data one expects RMP2P to perform well under churn and mobility. The

[2]Not cached in the local storage.

case for churn affecting lookups can only be made when a peer has recently joined the P2P network and has not completed any replication iterations as described in Section 5.2.2.3. In this rare case any data stored on the leaving peer would be lost. This drawback applies to any P2P network where data is replicated between peers.

5.2.2.6 Evaluation of model

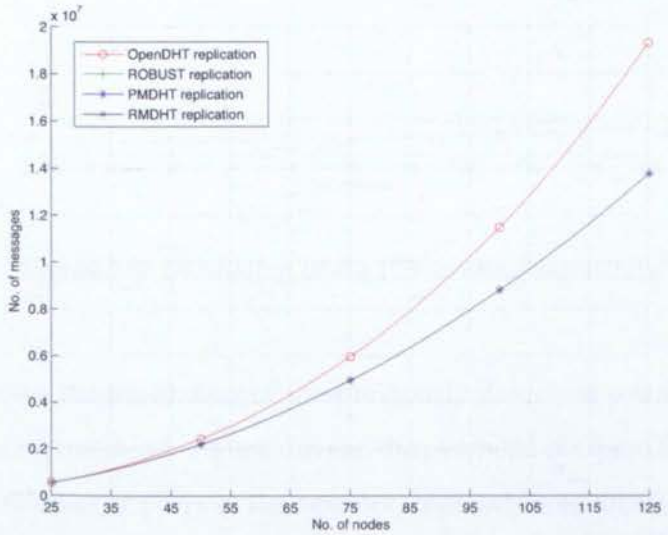

Figure 5.3: Evaluation of the P2P replication algorithm.

Figure 5.3 shows the outcome of the above models with regard to the P2P network replication mechanism. The figure shows the similarities of the ROBUST, PMDHT and RMP2P replication algorithms, this is due to the fact that these protocols generally choose neighbours within one or two hops to replicate with due to their proximity synchronisation, or in the case of RMP2P, purely because it replicates only with direct neighbours. The figure also presents the replication algorithm of OpenDHT, which has much higher overhead due to its lack of proximity awareness when selecting leaf set neighbours, meaning such a leaf could be physically far away in the network.

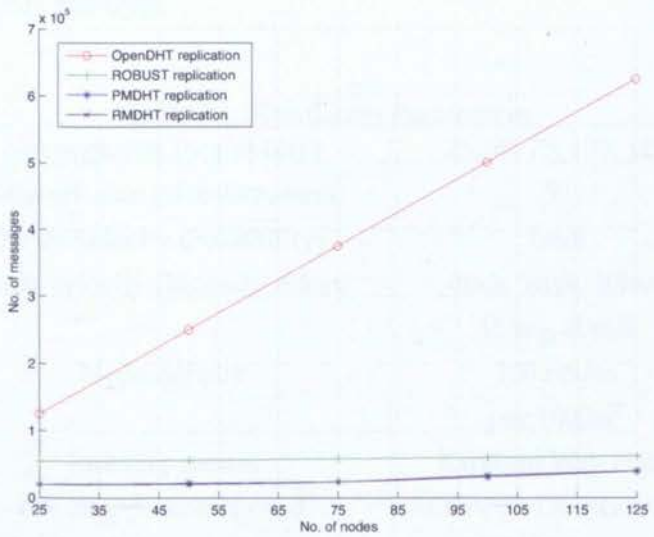

Figure 5.4: Evaluation of the P2P routing algorithm.

Figure 5.4 shows the evaluation of the previously described overhead models with regard to P2P routing overhead. As one can see, the overhead of OpenDHT scales logarithmically with the number of peers in the network, meanwhile ROBUST routing overhead remains stable when the number of peers is increased due to the limit on the number of hops introduced by using the clustering mechanism described in Section 4.2 of this thesis. PMDHT and RMP2P are considerably lower, due to the fact they take advantage of using MANET routing table information, thus the shorted path is usually traversed causing less overhead.

5.3 Simulation results

Table 5.1: Simulation Parameters

Network size (Scalability)	25, 50, 75, 100, 125
Network size (Mobility effect)	50
Node velocity (Scalability)	1m/s
Node velocity (Mobility effect)	0m/s, 5m/s, 10m/s 15m/s, 20m/s
Node density	100 nodes per $1000m^2$
Mobility model	Random Way Point
Routing protocols used	AODVv2, OLSRv2, DSR
Data packet payload size	512 bytes
MAC layer	802.11*b*
Link bandwidth	11Mbit/s
Maximum transmission range	250m
Types of traffic	UDP
Simulation time	1000 sec
DHT data distribution	Random
Number of P2P *GET* requests	10/sec
Data synchronisation interval	3 sec
Leaf set update interval	4 sec
Neighbour ping interval	5 sec
Super peer change RTT threshold	90ms
Cluster beacon interval	10 sec
Proximity synchronisation interval	60 sec

In order to evaluate the two proposed protocols an application for the network simulator network simulator-2 (ns-2) [51] implementing both PMDHT and RMP2P has been developed. The simulator implements all factors of the P2P network including replication, pings, leaf sets, puts/gets, clustering, caching, churn via mobility and joining. Table 5.1 shows the simulation parameters where the network area size is dependant on the number of nodes in order to maintain the node density. The author has simulated two

scenarios in order to investigate how the protocols perform under different conditions. The first scenario — referred to as the *Scalability* scenario — is aimed at evaluating the protocol performance while increasing the number of peers in the network. The second scenario — aptly named the *Mobility effect* scenario — investigates how the protocols perform while increasing the velocity of the moving peers. These scenarios have been specifically chosen as the differing attributes investigated represent two of the main challenges one must address in MANETs. All graphs show the standard deviation and 95% confidence interval.

The author has run the simulations using the Random Way Point mobility model as due to its random nature, it represents a difficult scenario to deal with compared to say, a nomadic model where MANET nodes mode in groups. The network size in the *Scalability* scenario starts at 25 peers and is incremented sequentially by 25 peers up to a maximum of 125. A velocity of 1m/s was chosen to represent a brisk walking pace.

The *Mobility effect* scenario was initially run using a static network with the peer velocity increasing sequentially with increments of 5m/s up to a maximum of 20m/s which is at the top end of a practical MANET — such as a car. The simulations assume *random* P2P data distribution. Table 5.1 also shows the

- number of P2P *GET* requests per second,
- *data synchronization interval*: indicates the period between choosing a random peer from a peer's leaf set and replicating its key value pairs,
- *leaf set update interval*: the period in which a peer chooses a random peer from its leaf set and it exchanges the updated IDs from the intersection of their leaf sets,
- *neighbor ping interval* : the period between a peer pings its leaf set peers (neighbours) to check if they are still associated with the network,
- *super-peer change Round Trip Time (RTT) threshold*: indicates the maximum tolerable difference in delay — between a peer's current super-peer A and one newly discovered B — before which a peer will join the B's cluster,

- *cluster beacon interval*: is the period between broadcasts of cluster beacons from a super-peer and,
- *proximity synchronization interval*: is the frequency which a peer compares the round trip times of super-peers in order to ascertain the need to join another cluster.

5.3.1 Network overhead

5.3.1.1 Scalability

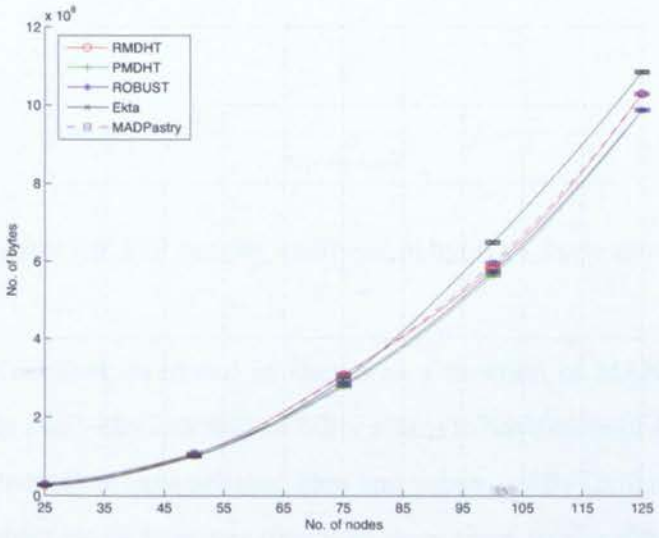

Figure 5.5: P2P overlay overhead in bytes vs. scalability.

Figure 5.5 shows the network overhead caused by each of the protocols in terms of bytes. As the network size increases, overhead scales quite dramatically. This is attributed largely to the replication procedure of each protocol whereby data is replicated at each of a peers leaf sets. Ekta slightly edges out the other protocols in terms of overhead down to more dropped packets requiring subsequent retransmits. However on the whole the protocols are quite similar, as Ekta and MADPastry both use aggressive Proximity Neighbour Selection (PNS) akin to the proposed protocols in this thesis.

5.3.1.2 Mobility effect

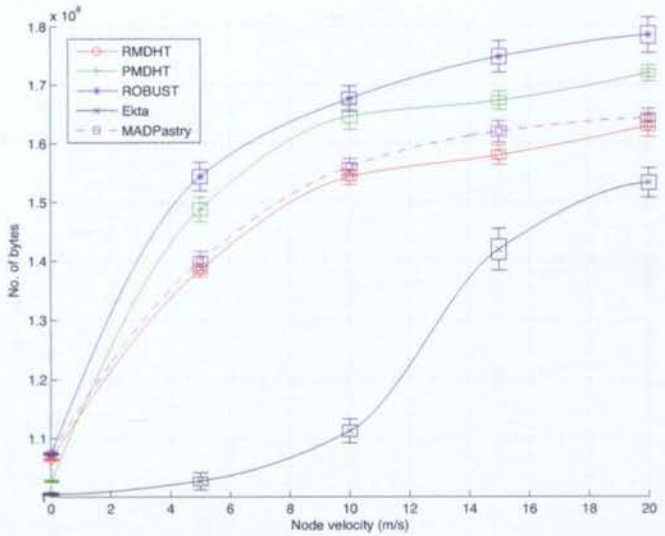

Figure 5.6: P2P overlay overhead in bytes vs. node speed.

In Figure 5.6 the network overhead in shown as a function of MANET node mobility speed. It is straight away obvious that mobility seems to have more of a differing effect on overhead per protocol than network size. Ekta and subsequently Dynamic Source Routing (DSR) handle mobility well however overhead does seem to rise dramatically at 15m/s as route caching becomes less effective at higher speeds due to more frequent link breakages. RMP2P and MADPastry have similar overhead as AODVv2 is quite consistent as routes are always found on demand, however MADPastry is slightly higher due to longer routes meaning more link breakages. PMDHT and ROBUST have the highest overhead as one would expect due to mobility meaning more frequent MPR re-election. In this case ROBUST sees a higher overhead due to considerably longer path lengths and more link breakages as a result.

5.3.2 Network latency

5.3.2.1 Scalability

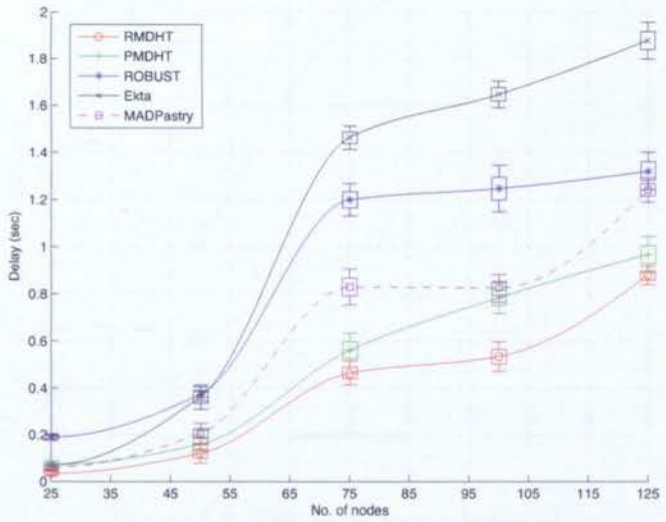

Figure 5.7: E2E delay vs. scalability.

Figure 5.7 shows the average end-to-end delay of a P2P lookup as a function of network size. It is evident that as the P2P networks scale in network size, delay is increasing as a result. Two factors seem to play a large role in delay caused by network size, the routing protocol, and the average lookup path length. Subsequently AODVv2 has a lower delay when scaling due to the quick setup of new paths compared with the OLSRv2 MPR re-election which takes place when an MPR node moves away from its selectors. However in the case of MADPastry this is countered by longer path lengths causing higher delays. Likewise ROBUST further compounds this as the average path length is significantly higher. Ekta and DSR have the highest delay, due to the fact that a larger network requires DSR to store longer source routes, this gives a higher probability of link failures and leads to more route discoveries taking place.

5.3.2.2 Mobility effect

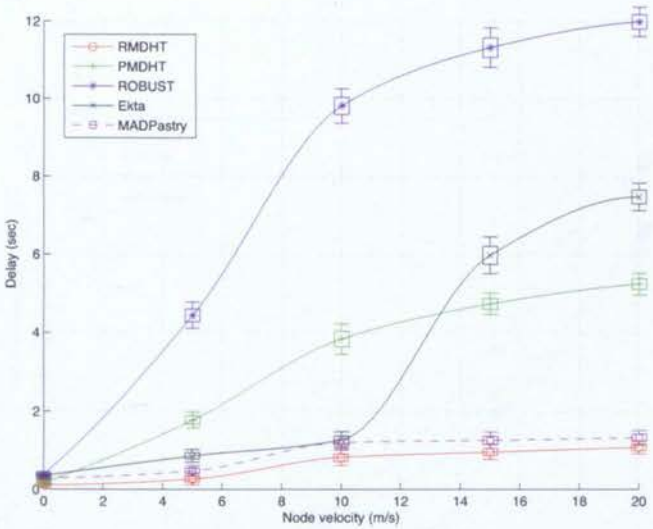

Figure 5.8: E2E delay vs. node speed.

Figure 5.8 investigates the affect that node speed has on delay. As in Figure 5.7 the two factors of routing protocol and path length again seem to play a big role. AODVv2 appears to handle mobility the best in terms of delay with both RMP2P and MADPastry performing well. This can be attributed to AODVv2 being able to immediately make use of routing information from intermediate nodes and thus learning new routes very quickly. The slightly higher delay in MADPastry is due to the average path being longer. The end-to-end delay in Ekta starts relatively low and increases dramatically at a node speed of 15m/s because of regular link breakages in routes. ROBUST performs badly here mainly due to the larger lookup path length, however PMDHT experiences quite high delay as well. This is caused by MPR nodes moving outside of the connectivity range of their MPR selectors causing MPR re-electing and high delays.

5.3.3 Packet loss

5.3.3.1 Scalability

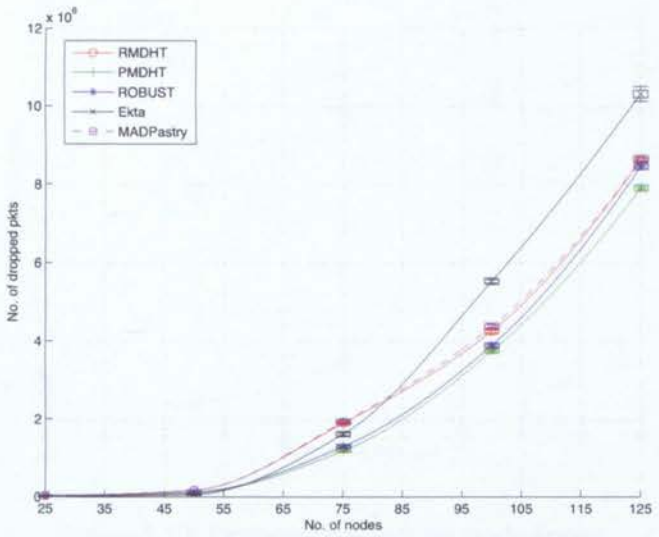

Figure 5.9: Dropped packets vs. scalability.

Figure 5.9 shows the number of dropped packets by each of the protocols studies with increasing network size. Like in Figure 5.5 higher packet loss for DSR can be explained due to larger stored source routes meaning more potential link breakages. RMP2P, PMDHT, MADPastry and ROBUST perform similarly as AODVv2 has a fast path discovery mechanism and OLSRv2 has the potential to reroute around broken links. For all protocols, the packet loss increases exponentially due to the high number of peers resulting in congested links and transmission errors, thus it is important to look at how the protocol handles this packet loss by looking at the lookup success rate examined in Section 5.3.4.1.

5.3.3.2 Mobility effect

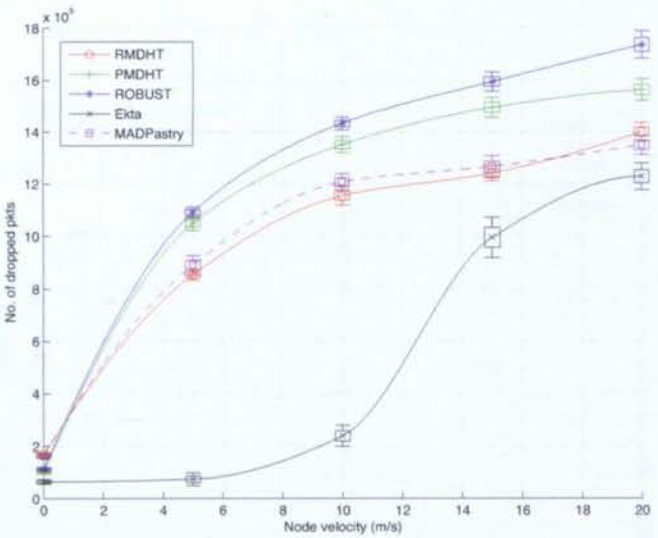

Figure 5.10: Dropped packets vs. node speed.

In Figure 5.10 the effects of node speed upon the number of dropped packets is investigated. When there is no mobility in the network there is a very low number of dropped packets. Ekta maintains a lower number of packets dropped initially then increases dramatically at faster speeds. The performance of Ekta in this scenario can be attributed to overhearing of new routes from intermediate peers passing by and datalink acknowledgement. The routing protocols seem to have a higher effect here and distinguishing between the different protocols using the same routing protocols is difficult mainly due to the high number of replication packets of each.

5.3.4 Lookup success rate

5.3.4.1 Scalability

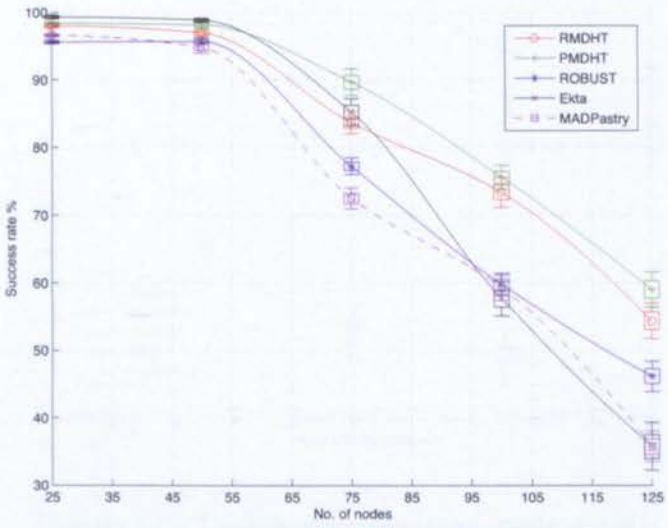

Figure 5.11: Lookup success rate vs. scalability.

Figure 5.11 shows the percentage of successful lookups against increasing network size. All of the protocols perform well for up to 50 nodes network size, with the protocols with a longer path length suffering more due to higher probability of a broken link. Ekta seems to take the most degradation of performance when scaling due to higher delay and packet loss. MADPastry and ROBUST both suffer degradation due to longer lookup paths. RMP2P and PMDHT perform the best overall due to short paths and low delay.

5.3.4.2 Mobility effect

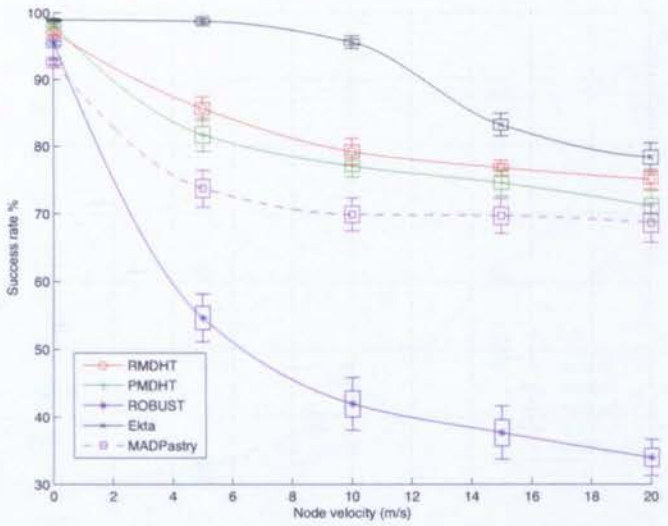

Figure 5.12: Lookup success rate vs. node speed.

Figure 5.12 investigates lookup success rate while increasing node speed. ROBUST performs by far the worst here due to higher delays and packet loss caused by a significantly longer path length and the effects of OLSRv2. MADPastry suffers compared to RMP2P despite using the same routing protocol due to a longer average path length causing more possible failures however this seems to stabilise around 5-10m/s. RMP2P and PMDHT perform similarly due to similar packet loss rates and short paths caused by tight routing protocol integration. Ekta however performs the best here due to the significant lower packet loss.

5.3.5 Average path length

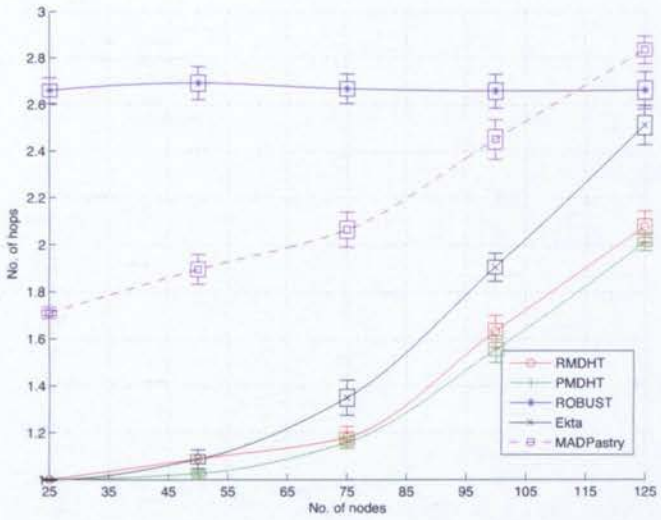

Figure 5.13: Lookup average path length.

In Figure 5.13 the average path length for each protocol is shown. RMP2P and PMDHT perform very similar due to the tight integration at the network layer meaning the P2P overlay paths mirror those of the routing protocol. Ekta has a slightly higher average path length as it still relies on pastry leaf sets and routing tables while sourcing much of the information from DSR. MADPastry relies on clusters to lower overhead this doesn't however necessarily mean shorter paths in lookups. ROBUST performs the worst as expected here, the path length will stay the same due to its hierarchical topology and strict cluster routing, this may be a benefit at larger network sizes not investigated in this thesis.

5.3.6 Overlay Stretch

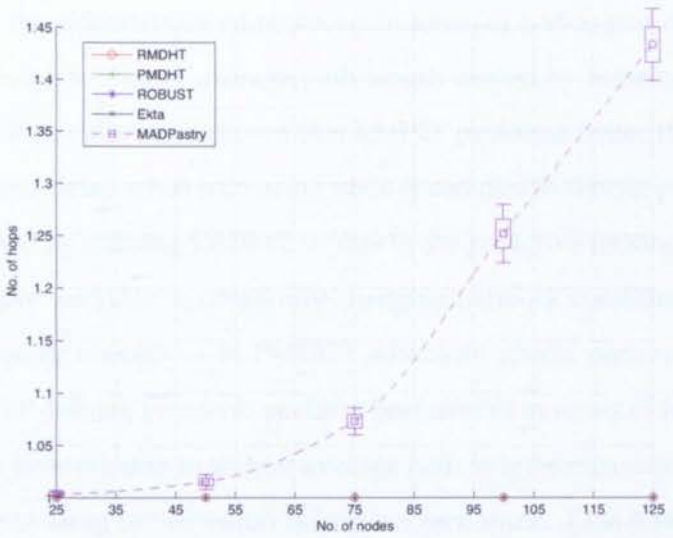

Figure 5.14: Logical path vs. physical path stretch.

Figure 5.14 displays the P2P lookup path stretch as computed using the expression in equation 5.1. MADPastry is the only protocol which exhibits stretch as the PMDHT, RMP2P and Ekta protocols are tightly integrated with their corresponding routing protocols at the network layer. ROBUST relies on broadcasting and obtaining IPs to DHT IDs to create an exact copy of the physical network at the DHT layer.

5.4 Summary

In this chapter, two cross-layer approaches to optimise P2P networks over MANETs were presented. Both protocols provide the same functionalities as widely used DHTs while reducing overhead and delay compared to conventional P2P networks being simply used on top of MANETs — when taking into consideration the results from Chapter 4, which show substantial improvements over OpenDHT. The P2P overlay overhead in both P2P networks presented in this chapter are similar to the P2P networks optimised for MANETs

studied in this thesis, namely Ekta and MADPastry, due to the similar aggressive PNS algorithms used. However despite their similar overhead both RMP2P and PMDHT perform better than the aforementioned protocols in terms of end-to-end delay when scaling the network size due to shorter average path length caused by utilising MANET routing protocol routes. It has also been shown that RMP2P performs better than MADPastry in terms of end-to-end delay when increasing node speed due to shorter path lengths, where as the delay caused by utilising OLSRv2 — due to the proactive routing protocol not handling high mobility well due to constantly changing network conditions meaning routes not converging quick enough — in PMDHT adversely affects performance in this case. Both proposed P2P overlay protocols perform best overall in terms of lookup success rate when scaling the network due to shorter average path lengths meaning less probability of dropped packets causing transmission failure across a route. Ekta however outperforms both protocols in terms of success rate when increasing node speed due to lower packet loss, however Ekta also has significantly higher end-to-end delay when increasing node speed in this scenario, thus RMP2P also performs well in this scenario.

In the proactive DHT the DHT functionality is combined with the routing at the network layer and DHT information is piggy-backed inside the routing topology messages. Optimisation is achieved in the reactive P2P network by piggybacking P2P lookup information inside route request messages if such a packet is sent within a certain time period, else a P2P overlay request packet is sent. Obtained simulation results have shown that both systems are suited to different scenarios. The proactive DHT appears to be well suited to larger networks overall when compared with similar protocols and a non cross-layer DHT protocol, thus this protocol is better suited to this scenario when compared to the current state-of-the-art techniques. However the reactive P2P overlay network seems more suited to scenarios with a high level of mobility when compared with the aforementioned protocols, it is slightly lower in terms of lookup success rate, however this is balanced with less end-to-end delay, thus offering a better solution than the state-of-the-art for time sensitive lookup applications when increasing node speed.

Chapter 6

Conclusions and future work

This chapter provides the conclusion of this thesis, first looking over the work previously presented, and following up with future work and directions in the area of MANETs and P2P overlay network optimisations.

6.1 Conclusion

The proliferation of recent advances in the hardware and software of mobile devices, primarily due to their increased mainstream popularity, has made creating and deploying distributed networks based on mobile devices not only feasible but a necessary solution where no centralised infrastructure is available. Looking at services deployed on the Internet which could be candidates for using such distributed technologies makes for interesting research as overcoming the hurdles of mobile computing present a unique challenge. Such services include name services, messaging systems, or distributed storage systems. Due to the similarities between conventional peer-to-peer systems and MANETs, such as their lack of centralised infrastructure, dynamic network topology and self organisation, the applicability of MANETs and P2P overlay networks presents a promising way to build efficient distributed applications in dynamic mobile environments.

As discussed and studies in the literature review in Chapter 3 of this thesis, broadcast-based unstructured P2P networks generally suffer from poor scalability, in that, as the

number of peers in the overlay network increases, severe problems are faced due to the traffic needed to search every peer for content. This is further compounded in MANETs where peers need to preserve their limited battery life. In the Internet DHTs were introduced in order to tackle the scalability issues faced with unstructured P2P networks, these have been utilised as the building blocks for large scale distributed applications. At first one may speculate that simply deploying P2P networks on top of MANETs would suffice to extend their usage to such mobile networks, however this is not a practical approach due to the issues mentioned in Section 1.1 namely, P2P overlay stretch making P2P lookups traverse unnecessarily long paths creating lookup delays, physical route discovery which can cause further delays in the P2P service and maintenance caused by the P2P overlay needed to function correctly.

To tackle the challenges of deploying P2P overlay networks on MANETs this thesis investigated three distinct approaches and evaluated their performance in order to establish their suitability in differing scenarios. The first approach which is entitled ROBUST was based on a scenario where the underlying routing protocol can not be chosen or modified — due to either technical constraints where the underlying operating system cannot be easily accessed, or where the underlying MANET routing protocol must adhere strictly to some RFC. ROBUST combines the conventional DHT architecture OpenDHT with a routing paradigm based around proximity-aware clusters of similar peer IDs in order to reduce overlay stretch and reduce routing complexity. Using this method ROBUST can benefit from the reduced overhead caused by the increase in physical locality without the need for additional energy intensive hardware requirements such as Global Positioning System (GPS). In order to avoid long overlay lookups in terms of logical hops, our routing imposes a strict bound on hops by using a hierarchical clustered topology. This limits both the overlay, and physical hops to a value less than the *logN* hops associated with conventional DHTs. Looking back at Section 3.4 the protocol proposed has tackled the challenge of adding mobility support to hierarchical DHTs by allowing peers to roam between clusters. The other pressing issue of supporting smaller networks such as the size

of MANETs is provided by making clusters scalable to manage larger and smaller networks. The allocation of specific clusters and peers to ID space means that load-balancing would not be an issue in ROBUST as compared to the other approaches.

Using the ns-2 [51] based P2P simulator developed during the course of this thesis, a number of differing scenarios were simulated for ROBUST comparing it with the popular OpenDHT conventional DHT architecture. ROBUST outperformed OpenDHT in all scenarios due to the latter's optimisation for MANETs. An interesting observation is that generally OLSRv2 handled network scalability better than AODVv2, whereas AODVv2 was more apt at handling scenarios with higher node mobility in the MANET.

While sacrificing flexibility when it comes to ROBUST in that it is run independent of the underlying MANET routing protocol, one can achieve significantly better performance by integrating the P2P architecture at the routing layer by combining the P2P routing and MANET routing, while sharing information from the MANET routing table with the P2P network. In order to fully investigate this approach, two P2P overlay architectures were proposed based on the popular MANET routing protocols, OLSRv2 and AODVv2. These specific protocols were chosen due to their standing as successors to the RFCs [41,42]. The proposed PMDHT reduces overhead created by the DHT by disseminating the overlay IDs of peers within the routing packets which are flooded throughout the network using the MPR architecture described in Section 2.2.2. Thus by utilising the qualities of OLSRv2 with regards to peer mobility and the holistic approach to OLSRv2 routing — whereby each peer knows prior to data transmission the route to any other peer — overhead can be dramatically reduced as two separate networks no longer need to be maintained and overlay stretch is eradicated because the DHT takes advantage of the underlying routing protocol to look up data. Additionally the breakdown of the overhead induced by the DHT was investigated, showing that replication is the most intensive mechanism, thus to decrease this overhead the concept of CGA was introduced whereby peers within a close proximity form clusters reducing the path length between DHT neighbours and thus reducing the overhead. In Section 3.3 two similar protocols were studied. Ekta [76] omitted

physical location awareness, thus peers in the overlay were not correlated to the underlying physical network. In PMDHT CGA was used to improve on this. Some aspects of Ekta were also borrowed increasing efficiency, such as aggressive caching. In MADPastry [61] the Random Landmarking concept was used as the foundation for CGA, however the notion of using Pastry routing tables on top of the MANET routing protocol increased overhead. PMDHT eliminated the Pastry routing table by simply relying on OLSRv2. This had the added benefit of enabling all peers to have state information about the entire network topology, thus decreasing latency.

The second cross-layer P2P network proposed in this thesis which integrates the P2P overlay with MANET routing is RMP2P. This investigated method uses the AODVv2 protocol as the MANET routing basis and integrates P2P functionality by adding P2P lookups within AODVv2 RREQ packets and searching the network using the same method AODVv2 uses to find routes as described in Section 2.2.2. If a RREQ is not sent within a certain time period a DREQ is sent in order to avoid high lookup latency times. The DREQ message is broadcast in the same way as a RREQ by using the expanding ring search algorithm with peers storing only the next hop to a destination and caching the reverse path. Thus the effect of this is to reduce overlay stretch completely as the P2P routing uses the same paths as deemed the most efficient by the MANET routing protocol in the physical network. RMP2P shows that it is possible to integrate unstructured P2P networks with MANET routing protocols to create an efficient P2P overlay. One of the main reasons the unstructured P2P network is able to perform so well in MANETs is due to the small network size when compared to Internet sized P2P networks, thus the scalability issue does not seem to arise when P2P packets are piggy-backed on to the routing protocol. The different P2P functions were modelled and show that replication is again creating the most overhead in the network. When compared with conventional P2P networks RMP2P performs much better with respect to this mechanism. This is due to peers from the local neighbourhood randomly being selected and their data replicated, rather than being a logical neighbour which could be physically far away in the network. In Section 3.2 unstructured peer-to-

peer networks for MANETs were studied. The conclusion was that these networks were unable to scale well in scenarios with many peers. The other downsides of flood-based protocols are their unreliability, in that finding a piece of data is not always guaranteed and data will be lost if the peer departs the network. This thesis has shown that RMP2P can scale well enough to provide service in practically sized MANETs by utilising aggressive caching and piggy-backing on to the network layer messages where possible. By utilising the underlying protocol, which will always find a route if one exists in the network, RMP2P is able to guarantee data will be found if it exists as the entire network is searched. RMP2P also tackles the issue of redundancy of data by using replication between local neighbours. Thus even if a peer departs the network, the data will still be available.

Using the ns-2 based P2P simulator developed specifically for this thesis, the two mentioned protocols were evaluated using a range of scenarios and measuring different metrics related to P2P overlay performance. The simulations show that the two P2P networks perform well in different scenarios. PMDHT seems to perform better in larger networks when compared with similar protocols due to lower packet loss caused by OLSRv2 in such scenarios as the routing protocol control flooding better than AODVv2 and DSR due to OLSRv2 employing MPR nodes and the latter two protocols behaving more like broadcast-based flooding protocols in larger networks as DHTs will have many route lookups to a large number of peers. The simulation results presented also suggest that RMP2P is better suited to scenarios where the network exhibits high levels of mobility i.e. peers are moving fast in the physical space. This is due to AODVv2 employing the Route Error (RERR) message which notifies nodes when a route has been broken, allowing for the protocol to reestablish a new route quickly by using alternative routes which could have been found.

Overall the proposed integrated P2P overlay protocols outperform their similar counterparts and ROBUST which is unable to share information with the MANET routing protocol in order to maintain flexibility. Such performance would indicate that any implementation of P2P networks for MANETs should be based on an approach which efficiently shares information from the routing protocol with the P2P overlay especially for routing

lookups and to aid in proximity replication.

6.2 Future work

The proposed protocols from this thesis have each been evaluated while varying certain network conditions such as mobility and node speed. It would be of further interest to investigate how the P2Ps overlays perform in more scenarios such as where nodes could lose battery power and are forced to leave the network, with more nodes joining at random points in time. It would also be interesting to investigate different topologies caused by nodes moving using different mobility models. This thesis mainly utilised the Random Way Point (RWP) model where nodes are moving randomly in space, however mobility models such as nomadic mobility whereby nodes move in groups would be worth investigating as in extreme emergencies First Responders (FRs) generally work in groups.

Another aspect not investigated in our simulations is that of node heterogeneity, currently all nodes have the same power, where ROBUST can differentiate between different node types it may perform differently in this scenario. It would also be interesting to see if any benefit can be gained from RMP2P and PMDHT in this case by optimising the protocols for different aspects such as energy efficiency — such as by using paths with the least amount of neighbours and reducing the power needed for such neighbours to receive these redundant packets. This could be further investigated in mesh networks, where only the static peers in the network would participate in storing P2P data while other peers are able to perform lookups whilst mobile.

The P2P overlays proposed in this thesis have been based on OpenDHT which is in turn based on Pastry. It would be interesting to find out how the performance would differ using other DHTs or P2P networks such as Kademlia [55], Content Addressible Network (CAN) [14] Chord [12] or Gnutella [2] in combination with different MANET routing protocols such as AODVv2 and OLSRv2 and then studying the performance improvements, if any, when integrating them at the network layer. Another approach which could be investigated would be combining the functionality of PMDHT and RMP2P in order to

create a hybrid network such as Zone Routing Protocol (ZRP) has in the MANET routing domain, such a P2P overlay could exhibit characteristics of each of the proposed protocols, such as ability to scale with also the ability to handle high rates of mobility.

This thesis did not study the specific applications of such P2P networks as it was deemed to be outside the scope of this thesis. However future work could look towards implementing real applications on top of the proposed P2P networks and studying their performance in real networks made up of nodes from a testbed environment and covering a range of different mobility scenarios. Such applications could include for example, distributed data storage systems [16–19], distributed instant messaging systems [20], publish/subscribe systems [21–25], distributed name services [26–30] and Peer-to-Peer Session Initiation Protocol (P2PSIP) [31–34] for Voice over IP (VoIP) services in purely MANET based networks.

6.3 My list of publications

6.3.1 Journal

1. G. P. Millar, E. Panaousis, and C. Politis, "Distributed hash tables for peer-to-peer mobile ad-hoc networks with security extensions," Journal of Networks, vol. 7, pp. 288–299, 2012.

2. E.A. Panaousis, G Drew, G. P. Millar, T. A. Ramrekha and C. Politis, "A Testbed implementation for securing OLSR in mobile ad hoc networks," International Journal of Network Security and its Applications, vol. 2, pp. 143–162, 2010.

3. E.A. Panaousis, T. A. Ramrekha, G. P. Millar and C. Politis, "Adaptive and secure routing protocol for emergency mobile ad-hoc networks," International Journal of Wireless and Mobile Computing, vol. 2, pp. 62–78, 2010.

6.3.2 Conference and workshop

4. T. A. Ramrekha, G. P. Millar and C. Politis, "A Model for designing Scalable and Efficient Adaptive Routing Approaches in Emergency Ad hoc Communications," in Proceedings of the Symposium on Computers and Communications. (IEEE ISCC 2011), pp. 916–923, 2011.

5. G. P. Millar, T. A. Ramrekha and C. Politis, "A cross-layer model to reduce control traffic overhead in peer-to-peer mobile ad-hoc networks," in Proceedings of the Wireless World Research Forum. (WWRF 10), 2010.

6. G. P. Millar, E. Panaousis, and C. Politis, "Robust: Reliable overlay based utilisation of services and topology for emergency manets," in Proceedings of the IEEE Future Network and MobileSummit, pp. 1–8, 2010.

7. G. P. Millar, T. A. Ramrekha, and C. Politis, "A peer-to-peer overlay approach for emergency mobile ad hoc network based multimedia communications," in Proceedings of the International Mobile Multimedia Communications Conference. (ICST MOBIMEDIA 09), p. 59, 2009.

8. G. P. Millar, C. Politis, "A Novel Architecture for Multiple Peer-to-Peer Overlays in MANETs in Emergency Situations", in Proceedings of the Wireless World Research Forum. (WWRF 09), 2009.

6.3.3 White paper

9. H Abramowicz et al., "Future Networks and Management White Paper," Net!Works European Technology Platform, Expert Working Group, 2011.

6.3.4 Standard

10. T. A. Ramrekha, E.A. Panaousis, G. P. Millar and C. Politis, "ChaMeLeon (CML): A hybrid and adaptive routing protocol for Emergency Situations," in Proceedings of

the Internet Engineering Task Force (IETF) Meeting 77, Mobile Ad-hoc Networks (MANET) Working Group, 2010.

6.3.5 Patent

11. G. P. Millar, E. Panaousis, C. Politis, and A. Ramrekha, "Framework for ubiquitous networking," Patent WO_2013_008 026_A2, [Publication date; 01-January-2014].

List of References

[1] W. B. Norton, "Internet service providers and peering," *in Proceedings of the North American Network Operators' Group. (NANOG 01)*, vol. 19, pp. 1–17, 2001.

[2] "Gnutella," http://www.gnutella.com, 2008, [Online; accessed 12-March-2014].

[3] "The bittorrent protocol specification," http://www.bittorrent.org/beps/bep_0003.html, 2008, [Online; accessed 12-March-2014].

[4] "Morpheus," www.morpheus.com, 2008, [Online; accessed 12-March-2014].

[5] "Jxta," http://jxta.dev.java.net/, 2008, [Online; accessed 12-March-2014].

[6] "Kazaa," www.kazaa.com, 2007, [Online; accessed 12-March-2014].

[7] R. Steinmetz and K. Wehrle, "What is this "peer-to-peer" about?" *Peer-to-Peer Systems and Applications*, vol. 3485, pp. 9–16, 2005.

[8] R. Gross and A. Acquisti, "Information revelation and privacy in online social networks," *in Proceedings of the ACM workshop on Privacy in the electronic society. (WPES 05)*, pp. 71–80, 2005.

[9] X. Rui-ping and Z. Zhen-zhen, "Legal problem and countermeasure of peer-to-peer networks," *in Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering. (ICIII 08)*, vol. 2, pp. 394–397, 2008.

[10] P. Zhang, N. Fotiou, B. Helvik, G. Marias, and G. Polyzos, "Analysis of the effect of inforanking on content pollution in peer-to-peer systems," *Security and Communication Networks*, 2013.

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

[12] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, 2001.

[13] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Middleware 2001*, vol. 2218, pp. 329–350, 2001.

[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," *in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. (ACM SIGCOMM 01)*, pp. 161–172, 2001.

[15] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," *Technical Report*, 2001.

[16] L. Li, C. Zhang, W. Mi, Y. Zhang, T. Ma, Y. Ji, and X. Qiu, "Sfdht: A dht designed for server farm," *in Proceedings of the Global Telecommunications Conference. (IEEE GLOBECOM 09)*, pp. 1–8, 2009.

[17] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer *et al.*, "Oceanstore: An architecture for global-scale persistent storage," *in Proceedings of the ACM Sigplan Notices*, vol. 35, pp. 190–201, 2000.

[18] G. Shi, J. Chen, H. Gong, L. Fan, H. Xue, Q. Lu, and L. Liang, "Sandstone: A dht based carrier grade distributed storage system," *in Proceedings of the International Conference on Parallel Processing. (ICPP 09)*, pp. 420–428, 2009.

[19] M. Karnstedt, K.-U. Sattler, M. Richtarsky, J. Muller, M. Hauswirth, R. Schmidt, and R. John, "Unistore: querying a dht-based universal storage," *in Proceedings of the International Conference Data Engineering. (IEEE ICDE 07)*, pp. 1503–1504, 2007.

[20] R. J. Desmarais, P. Lach, and H. A. Müller, "Yakit: a locality based messaging system using icon overlay," *in Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research. (CASCON 11)*, pp. 148–159, 2011.

[21] I. Aekaterinidis and P. Triantafillou, "Pastrystrings: A comprehensive content-based publish/subscribe dht network," *in Proceedings of the International Conference on Distributed Computing Systems. (IEEE ICDCS 06)*, pp. 23–23, 2006.

[22] R. Van Renesse and A. Bozdog, "Willow: Dht, aggregation, and publish/subscribe in one protocol," *Springer Peer-to-Peer Systems III*, pp. 173–183, 2005.

[23] A. Gupta, O. D. Sahin, D. Agrawal, and A. E. Abbadi, "Meghdoot: content-based publish/subscribe over p2p networks," *Proceedings of the ACM/IFIP/USENIX international conference on Middleware*, pp. 254–273, 2004.

[24] D. Tam, R. Azimi, and H.-A. Jacobsen, "Building content-based publish/subscribe systems with distributed hash tables," *Springer Databases, Information Systems, and Peer-to-Peer Computing*, pp. 138–152, 2004.

[25] W. W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. P. Buchmann, "A peer-to-peer approach to content-based publish/subscribe," *in Proceedings of the International Workshop on Distributed Event-based Systems. (ACM DEBS 03)*, pp. 1–8, 2003.

[26] I. Baumgart, "P2pns: A secure distributed name service for p2psip," *in Proceedings of the International Conference on Pervasive Computing and Communications. (IEEE PerCom 2008)*, pp. 480–485, 2008.

[27] B. Awerbuch and C. Scheideler, "Group spreading: A protocol for provably secure distributed name service," *Springer Automata, Languages and Programming*, pp. 183–195, 2004.

[28] V. Pappas, D. Massey, A. Terzis, and L. Zhang, "A comparative study of the dns design with dht-based alternatives." *in Proceedings of the International Conference on Computer Communications. (IEEE INFOCOM 06)*, vol. 6, pp. 1–13, 2006.

[29] V. Ramasubramanian and E. G. Sirer, "The design and implementation of a next generation name service for the internet," *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 331–342, 2004.

[30] M. D'Ambrosio, C. Dannewitz, H. Karl, and V. Vercellone, "Mdht: a hierarchical name resolution service for information-centric networks," *in Proceedings of the ACM SIGCOMM workshop on Information-centric networking. (ACM ICN 11)*, pp. 7–12, 2011.

[31] D. A. Bryan, B. B. Lowekamp, and C. Jennings, "Sosimple: A serverless, standards-based, p2p sip communication system," *in Proceedings of the International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications. (IEEE AAA-IDEA 05)*, pp. 42–49, 2005.

[32] D. A. Bryan, B. B. Lowekamp, and M. Zangrilli, "The design of a versatile, secure p2psip communications architecture for the public internet," *in Proceedings of the International Symposium on Parallel and Distributed Processing. (IEEE IPDPS 08)*, pp. 1–8, 2008.

[33] I. Martinez-Yelmo, A. Bikfalvi, R. Cuevas, C. Guerrero, and J. Garcia, "H-p2psip: Interconnection of p2psip domains for global multimedia services based on a hierarchical dht overlay network," *Elsevier Computer Networks*, vol. 53, pp. 556–568, 2009.

[34] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "Resource location and discovery (reload) base protocol, ietf internet draft (work in progress),"

http://datatracker.ietf.org/doc/draft-ietf-p2psip-base/, 2013, [Online; accessed 12-March-2014].

[35] Y. Xiao, "Ieee 802.11n: enhancements for higher throughput in wireless lans," *IEEE Wireless Communications*, vol. 12, pp. 82–91, 2005.

[36] M. Waldvogel and R. Rinaldi, "Efficient topology-aware overlay network," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 101–106, 2003.

[37] Y. Zhu and Y. Hu, "Efficient, proximity-aware load balancing for dht-based p2p systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, pp. 349–361, 2005.

[38] H. Shen and C.-Z. Xu, "Locality-aware and churn-resilient load-balancing algorithms in structured peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, pp. 849–862, 2007.

[39] A. Al-Maashri and M. Ould-Khaoua, "Performance analysis of manet routing protocols in the presence of self-similar traffic," *in Proceedings of the Conference on Local Computer Networks. (IEEE LCN 06)*, pp. 801–807, 2006.

[40] H. Xu, X. Wu, H. Sadjadpour, and J. Garcia-Luna-Aceves, "A unified analysis of routing protocols in manets," *IEEE Transactions on Communications*, vol. 58, pp. 911–922, 2010.

[41] C. Perkins, E. Belding-Royerand, and S. Das, "Ad hoc on-demand distance vector (aodv) routing, ietf rfc 3561," http://tools.ietf.org/html/rfc3561, 2003, [Online; accessed 12-March-2014].

[42] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr), ietf rfc 3626," http://tools.ietf.org/html/rfc3626, 2003, [Online; accessed 12-March-2014].

[43] D. B. Johnson, D. A. Maltz, J. Broch *et al.*, "Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks," *Ad hoc networking*, vol. 5, pp. 139–172, 2001.

[44] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, pp. 234–244, 1994.

[45] G. P. Millar, E. Panaousis, and C. Politis, "Robust: Reliable overlay based utilisation of services and topology for emergency manets," *in Proceedings of the IEEE Future Network and MobileSummit*, pp. 1–8, 2010.

[46] G. P. Millar, T. A. Ramrekha, and C. Politis, "A peer-to-peer overlay approach for emergency mobile ad hoc network based multimedia communications," *in Proceedings of the International Mobile Multimedia Communications Conference. (ICST MOBIMEDIA 09)*, p. 59, 2009.

[47] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "The optimized link state routing protocol version 2, ietf internet draft (work in progress)," http://tools.ietf.org/html/draft-ietf-manet-olsrv2-17, 2012, [Online; accessed 12-March-2014].

[48] G. P. Millar, E. Panaousis, C. Politis, and A. Ramrekha, "Framework for ubiquitous networking," Patent WO_2013_008 026_A2, [Publication date; 01-January-2014].

[49] C. Perkins and I. Chakeres, "Dynamic manet on-demand (aodvv2) routing, ietf internet draft (work in progress)," http://tools.ietf.org/html/draft-ietf-manet-dymo-25, 2013, [Online; accessed 12-March-2014].

[50] G. P. Millar, E. Panaousis, and C. Politis, "Distributed hash tables for peer-to-peer mobile ad-hoc networks with security extensions," *Journal of Networks*, vol. 7, pp. 288–299, 2012.

[51] "Network simulator ns-2," http://www.isi.edu/nsnam/ns/, 2011, [Online; accessed 12-March-2014].

[52] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," *in Proceedings of the Conference on Applications, Technologies,*

Architectures, and Protocols for Computer Communications. (ACM SIGCOMM 03), pp. 407–418, 2003.

[53] J. Liang, R. Kumar, and K. W. Ross, "The fasttrack overlay: A measurement study," *Elsevier Computer Networks*, vol. 50, pp. 842–858, 2006.

[54] B. Beverly Yang and H. Garcia-Molina, "Designing a super-peer network," *in Proceedings of the International Conference Data Engineering. (IEEE ICDE 03)*, pp. 49–60, 2003.

[55] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Springer Peer-to-Peer Systems*, pp. 53–65, 2002.

[56] C. Plaxton, R. Rajaraman, and A. Richa, "Accessing nearby copies of replicated objects in a distributed environment," *Springer Theory of Computing Systems*, vol. 32, pp. 241–280, 1999.

[57] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Computer Communication Review*, vol. 18, pp. 314–329, 1988.

[58] A. R. Bharambe, S. G. Rao, V. N. Padmanabhan, S. Seshan, and H. Zhang, "The impact of heterogeneous bandwidth constraints on dht-based multicast protocols," *Springer Peer-to-Peer Systems IV*, pp. 115–126, 2005.

[59] A. S. Cheema, M. Muhammad, and I. Gupta, "Peer-to-peer discovery of computational resources for grid applications," *in Proceedings of the International Workshop on Grid Computing. (IEEE/ACM GRID 05)*, pp. 7–14, 2005.

[60] H. Pucha, S. M. Das, and Y. C. Hu, "Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks," *in Proceedings of the Workshop on Mobile Computing Systems and Applications. (IEEE WMCSA 04)*, pp. 163–173, 2004.

[61] T. Zahn and J. Schiller, "Madpastry: A dht substrate for practicably sized manets," *in Proceedings of the Applications and Services in Wireless Networks Conference. (ASWN 05)*, 2005.

[62] W. Rao, L. Chen, A.-C. Fu, and G. Wang, "Optimal resource placement in structured peer-to-peer networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 1011–1026, 2010.

[63] L. Hanzo and R. Tafazolli, "A survey of qos routing solutions for mobile ad hoc networks," *IEEE Communications Surveys Tutorials*, vol. 9, pp. 50–70, 2007.

[64] G. Pei, M. Gerla, X. Hong, and C.-C. Chiang, "A wireless hierarchical routing protocol with group mobility," *in Proceedings of the Wireless Communications and Networking Conference. (IEEE WCNC 99)*, pp. 1538–1542, 1999.

[65] X. Hong, K. Xu, and M. Gerla, "Scalable routing protocols for mobile ad hoc networks," *IEEE Network*, vol. 16, pp. 11–21, 2002.

[66] T. Clausen, C. Dearlove, and J. Dean, "Mobile ad hoc network (manet) neighborhood discovery protocol (nhdp), ietf rfc 6130," http://tools.ietf.org/html/rfc6130, 2011, [Online; accessed 12-March-2014].

[67] R. Bellman, "On a Routing Problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[68] Z. J. Haas, "A new routing protocol for the reconfigurable wireless networks," *in Proceedings of the International Conference on Universal Personal Communications Record. (IEEE ICUPC 97)*, vol. 2, pp. 562–566, 1997.

[69] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron, "Proximity neighbor selection in tree-based structured peer-to-peer overlays," *Microsoft Research Technical Report MSR-TR-2003-52*, 2003.

[70] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," *in Proceedings of the International Conference on Computer Communications. (IEEE INFOCOM 02)*, vol. 3, pp. 1190–1199, 2002.

[71] A. Klemm, C. Lindemann, and O. P. Waldhorst, "A special-purpose peer-to-peer file sharing system for mobile ad hoc networks," *in Proceedings of the Vehicular Technology Conference. (IEEE VTC 03)*, vol. 4, pp. 2758–2763, 2003.

[72] A. S. Cacciapuoti, M. Caleffi, and L. Paura, "Mobile p2p: peer-to-peer systems over delay tolerant networks," *Delay Tolerant Networks*, 2012.

[73] D. N. da Hora, D. F. Macedo, L. B. Oliveira, I. G. Siqueira, A. A. Loureiro, J. M. Nogueira, and G. Pujolle, "Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks," *Elsevier Computer Communications*, vol. 32, pp. 1445–1459, 2009.

[74] X. Fan, J. Cao, H. Mao, and Y. Liu, "Gossip-based cooperative caching for mobile applications in mobile wireless networks," *Elsevier Journal of Parallel and Distributed Computing*, vol. 73, pp. 653—663, 2013.

[75] M. C. Castro, A. J. Kassler, C.-F. Chiasserini, C. Casetti, and I. Korpeoglu, "Peer-to-peer overlay in mobile ad-hoc networks," *Springer Handbook of Peer-to-Peer networking*, pp. 1045–1080, 2010.

[76] H. Pucha, S. M. Das, and Y. C. Hu, "Ekta+: opportunistic multiplexing in a wireless dht," *in Proceedings of the International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking*, pp. 69–71, 2006.

[77] R. Winter, T. Zahn, and J. Schiller, "Random landmarking in mobile, topology-aware peer-to-peer networks," *in Proceedings of the International Workshop on Future Trends of Distributed Computing Systems. (IEEE FTDCS 04)*, pp. 319–324, 2004.

[78] J. F. Buford and H. Yu, "Peer-to-peer networking and applications: Synopsis and research directions," *Springer Handbook of Peer-to-Peer Networking*, pp. 3–45, 2010.

[79] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Springer Numerische mathematik*, vol. 1, pp. 269–271, 1959.

[80] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: network routing inspired by dhts," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 351–362, 2006.

[81] Z. Xu, R. Min, and Y. Hu, "Hieras: a dht based hierarchical p2p routing algorithm," *in Proceedings of the International Conference Parallel Processing. (ICPP 03)*, pp. 187–194, 2003.

[82] A. Gupta, B. Liskov, and R. Rodrigues, "One hop lookups for peer-to-peer overlays," *in Proceedings of the Conference on Hot Topics in Operating Systems (USENIX HotOS 03)*, vol. 9, pp. 2–2, 2003.

[83] D. Doval and D. O'Mahony, "Nom: Resource location and discovery for ad hoc mobile networks," *in Proceedings of the First Annual Mediterranean Ad Hoc Networking Workshop*, 2002.

[84] C.-S. Oh, Y.-B. Ko, and Y.-S. Roh, "An integrated approach for efficient routing and service discovery in mobile ad hoc networks," *in Proceedings of the Consumer Communications and Networking Conference. (IEEE CCNC 05)*, pp. 184–189, 2005.

[85] J. A. Garcia-Macias and D. A. Torres, "Service discovery in mobile ad-hoc networks: better at the network layer?" *in Proceedings of the International Conference Workshops on Parallel Processing. (ICPP 05)*, pp. 452–457, 2005.

[86] A. Mian, R. Baldoni, and R. Beraldi, "A survey of service discovery protocols in multihop mobile ad hoc networks," *IEEE Pervasive Computing*, vol. 8, pp. 66–74, 2009.

[87] N. Le Sommer and S. Ben Sassi, "Location-based service discovery and delivery in opportunistic networks," *in Proceedings of the International Conference on Networks. (IEEE ICN 10)*, pp. 179–184, 2010.

[88] J. Kniess, O. Loques, and C. V. Albuquerque, "Location aware discovery service and selection protocol in cooperative mobile wireless ad hoc networks," *in Proceedings of the International Conference on Computer Communications Workshops. (IEEE INFOCOM 09)*, pp. 1–2, 2009.

[89] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hash table for data-centric storage," *in Proceedings of the International Workshop on Wireless sensor Networks and Applications. (ACM WSNA 02)*, pp. 78–87, 2002.

[90] S. Sivavakeesar and G. Pavlou, "Scalable location services for hierarchically organized mobile ad hoc networks," *in Proceedings of the International Symposium on Mobile ad hoc Networking and Computing. (ACM MobiHoc 05)*, pp. 217–228, 2005.

[91] S. Sun, J. Hu, X. Luo, and Q. Wang, "Improved gpsr in inter-vehicle communication," *in Proceedings of the International Conference on Communications and Mobile Computing. (IEEE CMC 10)*, vol. 2, pp. 259–265, 2010.

[92] C. Ververidis and G. Polyzos, "Service discovery for mobile ad hoc networks: a survey of issues and techniques," *IEEE Communications Surveys Tutorials*, vol. 10, pp. 30–45, 2008.

[93] J. Eriksson, M. Faloutsos, and S. V. Krishnamurthy, "Dart: dynamic address routing for scalable ad hoc and mesh networks," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 119–132, 2007.

[94] F. Sailhan and V. Issarny, "Scalable service discovery for manet," *in Proceedings of the International Conference on Pervasive Computing and Communications. (IEEE PerCom 2005)*, pp. 235–244, 2005.

[95] C. Lee, A. Helal, N. Desai, V. Verma, and B. Arslan, "Konark: A system and protocols for device independent, peer-to-peer discovery and delivery of mobile services,"

IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 33, pp. 682–696, 2003.

[96] M. E. Berezin, F. Rousseau, and A. Duda, "Multichannel virtual access points for seamless handoffs in ieee 802.11 wireless networks," *in Proceedings of Vehicular Technology Conference. (VTC Spring 11)*, pp. 1–5, 2011.

[97] D. Kim, H. Bae, and C. K. Toh, "Improving tcp-vegas performance over manet routing protocols," *IEEE Transactions on Vehicular Technology*, vol. 56, pp. 372 –377, 2007.

[98] L. Barolli, M. Ikeda, F. Xhafa, and A. Duresi, "A testbed for manets: Implementation, experiences and learned lessons," *IEEE Systems Journal*, vol. 4, pp. 243 –252, 2010.

[99] J. Wu, "Handbook on theoretical and algorithmic aspects of sensor, ad hoc wireless, and peer-to-peer networks," *Auerbach Publications*, 2005.

[100] S. Ivanov, A. Herms, and G. Lukas, "Experimental validation of the ns-2 wireless model using simulation, emulation, and real network," *in Proceedings of the Workshop on Mobile Ad-Hoc Networks. (WiMAN 2007)*, pp. 433–444, 2007.

[101] S. Rhea, http://www.opendht.org/, 2014, [Online; accessed 01-February-2014].

[102] O. Rensfelt and L.-A. Larzon, "A bandwidth study of a dht in a heterogeneous environment," *in Proceedings of the Swedish National Computer Networking Workshop. (SNCNW 06)*, 2006.

[103] G. P. Millar and C. Politis, "A novel architecture for multiple peer-to-peer overlays in manets in emergency situations," *in Proceedings of the Wireless World Research Forum. (WWRF 09)*, 2009.

[104] T. Clausen, C. Dearlove, J. Dean, and C. Adjih, "Generalized mobile ad hoc network (manet) packet/message format, ietf rfc 5444," http://tools.ietf.org/html/rfc5444, 2009, [Online; accessed 12-March-2014].

DECLARATION

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This thesis has not previously been presented in identical or similar form to any other UK or foreign examination board.

The thesis work was conducted from February 2009 to August 2013 under the supervision of Dr. Christos Politis at Kingston University London.

Kingston-upon-Thames, London, United Kingdom.