INTEGRATING PEER-TO-PEER FUNCTIONALITIES AND ROUTING IN MOBILE AD-HOC NETWORKS

Grant Millar (grant@seednet.eu)[1], Alexandros Ladas (A.Ladas@kingston.ac.uk)[2], Olayinka Adigun (O.Adigun@kingston.ac.uk)[2], and Christos Politis (C.Politis@kingston.ac.uk)[2]

[1]Cylo Tech Ltd, Essex, United Kingdom
[2]Wireless Multimedia and Networking (WMN) Research Group, Kingston University London, KT1 2EE London, United Kingdom

ABSTRACT

Mobile Ad-hoc Networks (MANETs) impose strict requirements in terms of battery life, communication overhead and network latency, therefore optimization should be made to applications and services such as domain name service (DNS), dynamic host configuration protocol (DHCP) and session initiation protocol (SIP) if they are to be considered for use on MANETs. Due to the decentralized and self-organizing nature of MANETs, such applications could utilize a distributed name resolution/data storage service. Distributed Hash Tables (DHTs) enable these features by virtually organizing the network topology in a peer-to-peer (P2P) overlay. P2P overlays have been designed to operate on the application layer without knowledge of the underlying network thus causing poor performance. To address this problem, we propose and evaluate two different DHTs integrated with MANET routing in order to optimize the overall performance of MANET communications when P2P applications and services are used. Both architectures share the same functionality such as decentralization, self-reorganization, and self-healing but differ in MANET routing protocol. Performance evaluation using the NS2 simulator shows that these architectures are suited to different scenarios namely increasing network size and peer velocity. Comparisons with other well-known solutions have proven their efficiency with regard to the above requirements.

1. INTRODUCTION

P2P networks can be defined as decentralized 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. Distributed Hash Table (DHT) is a class of a decentralized distributed system and are widely used today on the Internet in various peer-to-peer systems for example providing name look-up services such as P2PSIP [1] or keeping track of peers in a file sharing system such as BitTorrent [2].

MANETs are physical networks consisting of wireless devices (normally Wi-Fi i.e. IEEE 802.11) 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 centralized infrastructure, instead rely on self-configuration. P2P overlays and MANETs share a common notion of rapid network setup, thus MANETs can be vital in scenarios where an instant network deployment is needed. Such scenarios include ubiquitous computing, providing instant network and disaster recovery during emergency scenarios. 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 spatial 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.

One can see that even on the surface both P2P networks and MANETs share some common properties such as being decentralized and self-organizing, 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 route to a specific destination key and retrieving the data associated with the key. To achieve the former, MANETs use routing protocols running on the network layer whereas P2P networks can use DHTs to achieve key look-up.

Many routing protocols have been proposed for MANETs, with most pertaining to a certain group determined on whether they proactively or reactively (or a hybrid) search the network for routes to a destination. Proactive MANET routing protocols build up a picture of the entire network by exchanging information about the network topology so that every node will eventually know the shortest path to any destination in the network. The main overhead in this case is disseminating this topology information as it must be flooded in the network to all nodes. Reactive MANET routing protocols will remain idle until a packet is to be sent from a node. Before transmission of the packet, the route to the destination will be found usually by utilizing a one-off flooding algorithm to find the shortest or quickest path.

Routes are normally cached at the source, destination and intermediate nodes for a defined period of time in order to speed up routing requests in the future.

On the other hand, DHTs are based on storing data using a unique ID called a key ID which is then mapped to the peer with the numerically closest peer ID in the DHT. In many DHTs, the key ID and the peer ID are computed using a uniformly distributed hash function such as Secure Hash Algorithm (SHA-1). This self-organizing algorithm creates a structure commonly known as a ring topology with the ring representing the peer ID space. If a peer wishes to find the value of a specific key, said peer will send a request to the closest matching peer ID that it is aware of, this will then be forwarded around the DHT until the destination is found. Due to the two infrastructures carrying out much the same functionalities, unnecessary overhead is created in the network. For example neighborhood discovery is executed in both the DHT and the MANET routing protocols. Therefore deploying DHTs on top of MANETs causes poor performance in the DHT both in terms of replication and searching for key IDs (known as look-ups) [3]. A logical step is to either accommodate the P2P functionality in the routing layer with protocols such as [3] [4] [5] [6] [7] [8][9] or to use a cross-layer approach to exploit common information in both the MANET and P2P architectures using protocols such as [5] [10] [11].

The work discussed in this paper introduces absolute integration of DHT functionality at the network layer in MANETs. In this way, we improve the performance of DHTs in MANETs. In this paper, two new protocols, Proactive MANET DHT (PMDHT) and Reactive MANET DHT (RMDHT) are proposed based on both a proactive and reactive MANET routing protocol. These are then evaluated against other protocols proposed in literature in order to ascertain improvements in performance for different scenarios. The remainder of this paper is organized as follows. In section 2, we discuss the state-of-the-art relevant to this work. In section 3, we propose two different architectures for integrating MANET routing protocols and DHTs based on both proactive and reactive MANET routing. In section 4, we present simulation parameters, scenarios and simulation results. Section 5 concludes this paper by summarizing the main findings and mentioning some areas of future work.

2. STATE-OF-THE-ART

In this section, we first discuss the most relevant state-of-the art papers within the field of integrated routing and P2P DHT architectures for MANETs. We then describe our previous work which motivates the work undertaken in this paper.

2.1. Integrated Routing with P2P DHTs

Pioneering work published by Rhea et al. entitled OpenDHT (also known as Bamboo DHT) is used as the basis for our work. OpenDHT is based on Pastry [12] and it makes improvements over Pastry in the areas of reactive versus periodic recovery from failures, calculation of message timeouts during lookups and Proximity Neighbor Selection (PNS). The results from [13] show that OpenDHT is able to handle higher rates of churn, that is, peers randomly joining and leaving the network than Pastry on which it was based, and Chord DHT [14]. The main improvements of OpenDHT which we incorporate in our DHT architectures are; (i) reactive recovery from failures whereby peers replace a failed neighbor in a fixed periodic manner rather than a reactive manner and (ii) calculation of message timeouts during lookups where the authors show TCP-style timeout calculation performs the best when compared with other common techniques. Readers are encouraged to read [13] in order to better understand the full range of improvements OpenDHT is able to offer in networks with high churn rates. We believe OpenDHT's ability to handle churn makes it a good candidate for a basis of optimization 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 paper also extends to Pastry. However, Pastry has been used as the basis for a large number of DHT-based applications and hence has been studied and analyzed in many research papers such as [5] [6] [13] [15] [16]. This makes it one of the most thoroughly examined DHT which have been proposed. In [6], authors proposed MADPastry, a DHT substrate which acts by combining the Pastry DHT with Ad hoc On-demand Distance Vector (AODV) [17] MANET routing at the network layer. This can lower the overhead required to maintain the DHT. While the architecture utilizes three different routing tables (One akin to AODV's routing table, another akin to Pastry's routing table, and a leaf set table), the only table requiring proactive management is that of the leaf-set table, with peers pinging their left and right respective leafs. The additional tables are updated by overhearing data packets destined for other peers.

2.2. Our Previous Work

In the following, we mainly describe our previous work [18] entitled Common Group Aliasing (CGA). This is used in PMDHT as we will describe in the next section of this paper. Common Group Aliasing refers to the process of grouping devices which are close physically in the underlying physical network together in the P2P overlay network. To achieve this goal, we proposed that the 160-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 realized 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 close 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 as 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 key. Once a marker peer has been established, it then proceeds to send out a periodic marker beacon. When a peer becomes a marker, the peer primarily does not have any peers within its CG set. Therefore, we propose that initial marker peers send out a marker beacon with a two-hop 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, the peer will leave the overlay network and then rejoin the network with its new ID. The new ID is calculated as its original peer ID with the first x bits of the marker key replacing the first x bits of the current peer ID. 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, its participation will be reconsidered by examining with 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 beacon was sent, it will process the beacon information and then drop the packet.

3. DHT-ROUTING INTEGRATION

To evaluate the integrated DHT 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 Optimized Link State Routing version 2 (OLSRv2) [19], the other is a reactive MANET routing protocol, namely Ad hoc On-demand Distance Vector version 2 (AODVv2) [20]. The proposed DHTs which we will refer to as PMDHT and RMDHT, from herein have the overall aim of:

- Decreasing stretch when compared to similar protocols from the field, defined as (Physical hops)/(Logical hops)
- Maintaining lookup success rates when compared to similar protocols as defined as (Successful lookups)/(Total lookups)
- Decreasing the average physical path length for a lookup, thus reducing delay and reducing the possibility for errors.
- Reducing the number of packets sent in the network related to DHT operations (signaling traffic and DHT key-based lookup routing).

3.1. Proactive MANET DHT (PMDHT)

The proposed proactive DHT architecture integrates the DHT functionality of OpenDHT [13] 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 RFC [21] and now adhering to RFC 5444 message[22] 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. One of such adjustment in proactive DHTs such as OLSRv2 is utilizing a modified version of the CGA architecture as discussed above. 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 a peer leaves the network (due to battery life constraints or willfully) 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 [14], 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(logN) where N is the number of nodes in the overlay. In PMDHT, peer IDs are piggybacked on to the OLSRv2 [19] routing packets, so that signaling required to propagate routing information regarding IP addresses also propagates DHT peer ID information. Thus we avoid replication of data for much the same purpose and reduce overhead.

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 [18]. 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 cluster-key (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 we will refer to as the beacon packet denoted by b_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 as 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 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 realized when a node receives no DHT packets within a 2*Tb time period where Tb is the time period between sending beacon packets. If this is the case, the node will hash its IP address using SHA-1, generating a 160-bit ID hash. As this is the first node, it will then become a super-peer. We use the node's IP address because it is unique piece of information of the user and it gives a unique hash less likely for collision. We treat a change of IP as the peer leaving and joining the network again. In the case that an overlay node receives a DHT beacon in 2*Tb, the node will initiate a join request to the source of the beacon. The join request sent 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 destination of the request will then generate a new DHT peer ID for the joining peer by concatenating its own DHT prefix. 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 discussed, PMDHT tightly integrates lookup functionality into the routing layer of OLSRv2. 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, TLVs are defined. TLVs allow extensions to be added on to existing routing packets. PMDHT will introduce a new packet TLV in the HELLO messages of OLSRv2. The TLVs will contain the DHT peer ID of the source node. Multi Point Relay (MPR) nodes as defined in OLSRv2 will then collate this information. Another packet TLV will be used to extend the Topology Control (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 node and entered into an extended Topology Information Base. When a peer 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 we refer to a DHT key-based lookup request) 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 the OLSRv2 routing based on the source node IP address.

Replication in PMDHT is handled exactly the same as the replication algorithm described in [13] with regards to selecting a random peer from its leaf set table and executing a synchronization request. This has the effect that in O(logLS) 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 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, whereas OpenDHT leaf set peers are random due to the hashing of IP address to generate peer IDs. In a network without proximity between leafset peers, the synchronized data, which is the largest in terms of packet size can be relayed over many hops and even traverse over the same physical nodes.

3.2. Reactive DHT

The proposed reactive DHT architecture again integrates the DHT functionality of OpenDHT [13] in terms of all functionality except DHT routing and including the modifications below but within the MANET routing protocol AODVv2 and is therefore aptly named Reactive MANET DHT (RMDHT). AODVv2 was chosen due to it being the successor to the RFC [20] and adhering to the RFC 5444 message and TLV formats. As with the previously described proactive protocol, the reactive DHT protocol will also be integrated at the routing layer in order to minimize overhead and delay from the DHT application perspective. A description of the main integrated architecture follows. Unlike the proactive protocol, where routable addresses are flooded to every node using 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 caused by the DHT, it is necessary to integrate DHT lookups with routing lookups. If a DHT which uses PUTs (utilizing 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 neighborhood 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 DHT 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 we prove later in this paper using simulations. The other issue we will address is redundancy of the stored data. This will obviously incur an overhead and so should be optimized in order to efficiently replicate data with nearby nodes. Due to the fact that in AODVv2, the algorithm is purely focussed on finding a route on-demand, the signaling required when the network is idle or previous routes have been established is very small. This has the affect that when integrating the DHT with AODVv2, we only need to focus on a small number of functions, 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. One can see that RMDHT is not a DHT in the traditional sense, in fact it is comparable to an unstructured P2P networks as it utilizes flooding for lookups rather than conventional key-based routing. We still refer to it as a DHT however as each peer locally stores key value pairs in a hash table.

In RMDHT, there is no specific join procedure as such, because the lookup protocol is based on a flooding algorithm. Peers do not maintain leaf sets and hence do not need to notify peers in order for the lookup procedure to function. In order to provide redundancy, a node should however begin the replication procedure immediately after joining the DHT. Overall, not requiring a join function does not decrease overhead much compared to PMDHT as the replication procedure which utilizes the largest data packets is still required. It does however save time when joining the DHT as the transmission of the Join packet and subsequent waiting for the JoinREP in PMDHT does add some delay.

RMDHT 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. RMDHT will extend the functionality of AODVv2 to support lookups within the routing protocol packets in order to lower overheads. A DHT 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 DHT Reply (DREP) packet in order to be processed by RMDHT. 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 DHT Request (DREQ) packet with an increased lifespan akin to the functionality of an RREQ but with the destination DHT key instead of an IP address. The DREQ will then be crosschecked against each DHT key at every node at which it is received.

Replication based on nodes with a numerically close DHT peer ID such as in PMDHT is not required as the DHT lookup algorithm is no longer based on the Plaxton prefix routing algorithm i.e. matching a DHT key to the relevant closest DHT peer ID. Therefore nodes can simply pick a random peer from its neighbor set which is populated by AODVv2 using the HELLO message defined in IETF RFC 6130 and send a SynReq message to the peer. DHT keys found at the random peer and not currently stored on the source node can then be requested akin to PMDHT.

4. SIMULATION RESULTS

In order to evaluate the two proposed protocols, we have developed an application for the network simulator ns-2 [23] implementing both PMDHT and RMDHT. The simulator implements all factors of the DHT including replication, pings, leaf sets, PUTs/GETs, clustering, caching, churn via mobility and joining. Table 1 shows the simulation parameters where the network area size is dependent on the number of nodes in order to maintain the node density. We have 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 MANET nodes. We have specifically chosen these scenarios as the differing attributes investigated represent two of the main challenges to be addressed in MANETs. All data points are an average of 10 simulation runs, and are presented with a 95% confidence interval. In our analysis, we have compared the results of our proposed algorithms with other known algorithms such as Etka, MADPastry and ROBUST which have all integrated P2P DHT on routing protocols. Table 1 shows the simulation parameters.

4.1. Network Overhead

In figure 1, the network overhead is shown as a function of MANET node mobility speed. It was straight away obvious that mobility seemed to have more of an effect on overhead per protocol than network size. Ekta handles mobility well, the overhead does seem to rise dramatically at 15m/s as route caching becomes less effective at higher speeds due to more frequent link breakages. RMDHT 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

Table 1: Simulation Parameters

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

sees a higher overhead due to considerably longer path lengths and more link breakages as a result.

4.2. Network Latency

Figure 2 (a) shows the average end-to-end delay of a DHT look-up as a function of network size. It is evident that as the DHTs 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 look-up path length. Subsequently AODVv2 has a lower delay when scaling due to the quick setup of new paths compared with the OLSRv2 whose MPR re-election takes place when an MPR node moves away

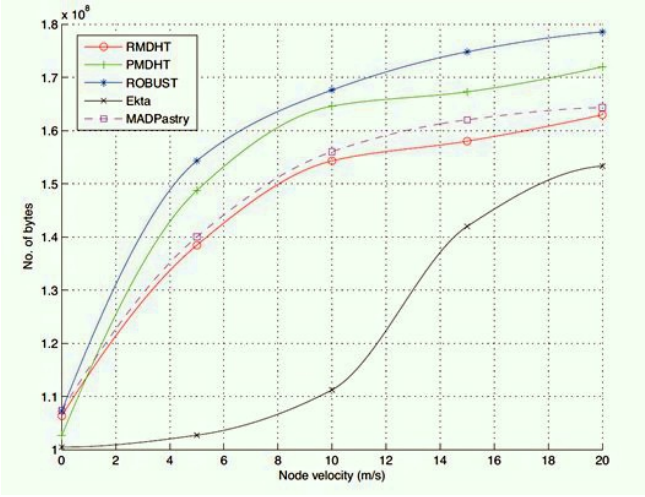
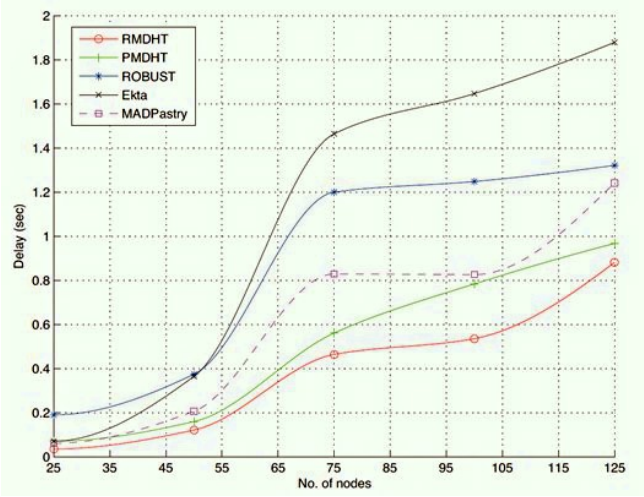


Figure 1: DHT Overhead in Bytes vs Node Speed

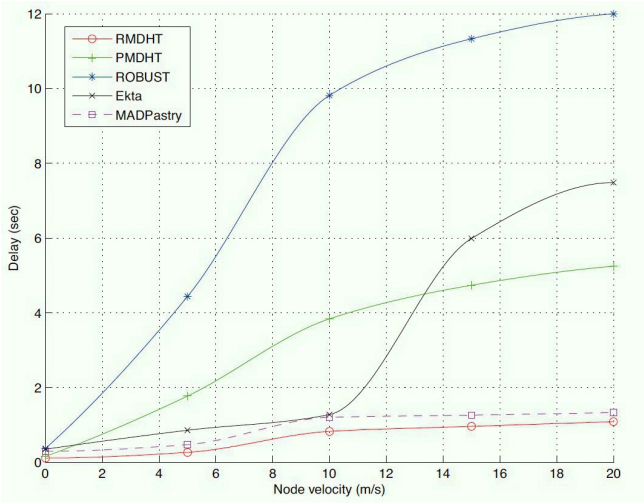
from its selectors. However, in the case of MADPastry this is countered by longer path lengths causing higher delays. Likewise, ROBUST further compounds this as the average path length is significantly higher. Ekta has the highest delay, due to the fact that a larger network requires DSR to store longer source routes, this gives a higher probability of link failures and leads to more route discoveries taking place. In figure 2(b) we investigate the effect that node speed has on delay. As in figure 2 (a), 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 RMDHT and MADPastry performing well. This can be attributed to AODVv2 incorporated within them the ability 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 15 m/s because of regular link breakages in routes. ROBUST performs badly here mainly due to the larger look-up 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.

4.3. Packet Loss

Figure 3(a) shows the number of dropped packets by each of the protocols studied with increasing network size. The higher packet loss for Ekta can be explained due to larger stored source routes meaning more potential link breakages. RMDHT, PMDHT, MADPastry and ROBUST perform similarly as AODVv2 with fast path discovery mechanism and OLSRv2 which has the potential to reroute around broken links. We can see that 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

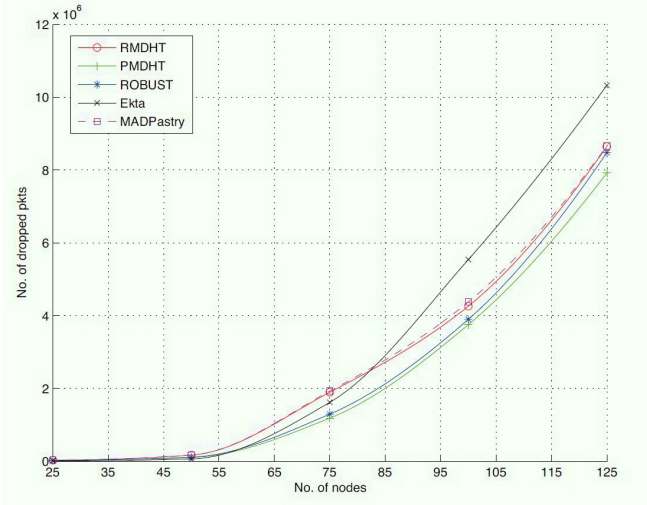


(a) E2E Delay vs Scalability

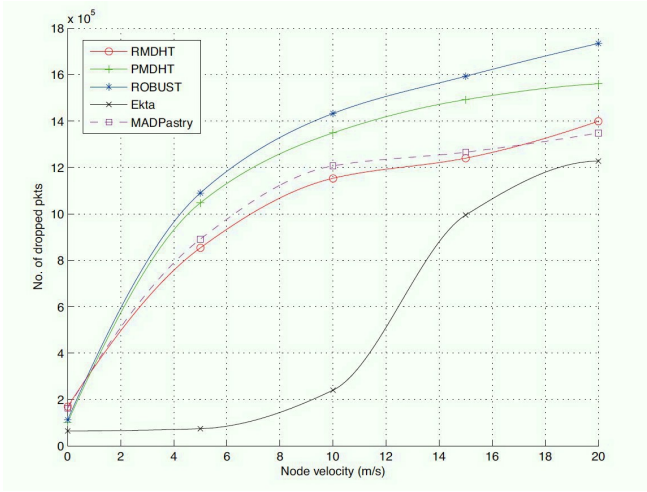


(b) E2E Delay vs Node Speed

Figure 2: Comparing End-to-End Delay for Different Protocols



(a) Dropped Packets vs Scalability



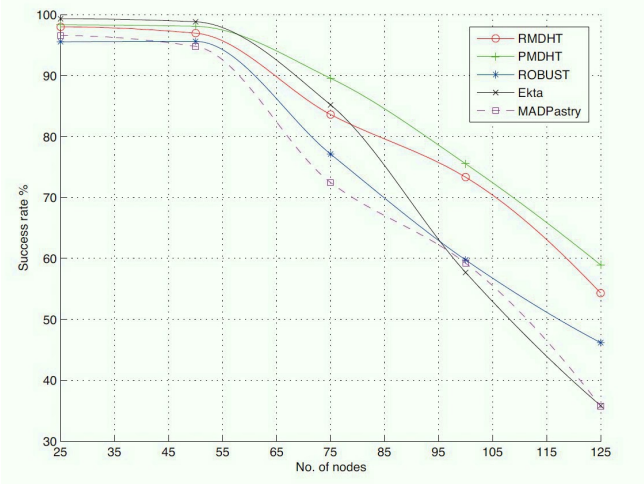
(b) Dropped Packets vs Node Speed

Figure 3: Comparing Dropped Packets for Different Protocols

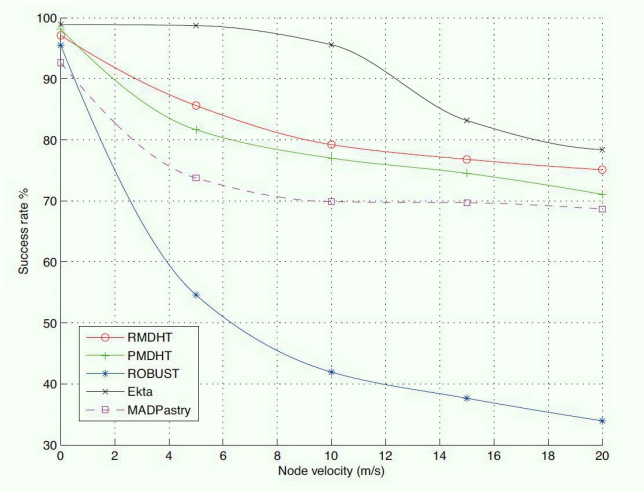
the protocol handles this packet loss by looking at the look-up success rate. In figure 3(b), the effects of node speed upon the number of dropped packets is investigated. When there is no mobility in the network we can see a very low number of dropped packets. Ekta maintains a lower number of packets dropped initially then increases dramatically at faster speeds. The performance of Ekta in this scenario can be attributed to over-hearing of new routes from intermediate peers passing by and datalink acknowledgement. The routing protocols seem to have a great effect here and distinguishing between the different protocols using the same routing approach is difficult mainly due to the high number of replication packets of each.

4.4. Look up Success Rate

Figure 4(a) shows the percentage of successful look-ups 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 look-up paths. RMDHT and PMDHT perform the best overall due to short paths and low delay. Figure 4(b) investigates look-up 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 RMDHT despite using the same routing protocol due to a longer average path length causing more possible failures, however this seems to stabilize around 5-10m/s. RMDHT 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.



(a) Look-up Success Rate vs Scalability



(b) Look-up Success Rate vs Node Speed

Figure 4: Comparing Look-up Success Speed for Different Protocols

4.5. Average Path Length

In figure 5, the average path length for each protocol is shown. RMDHT and PMDHT perform very similar due to the tight integration at the network layer meaning the DHT 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 Ekta. MADPastry relies on clusters to lower overhead, this doesn't however necessarily mean shorter paths in look-ups. 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 paper.

4.6. Overlay Stretch

Figure 6 displays the DHT look-up path stretch as described above. MADPastry is the only protocol which exhibits stretch as

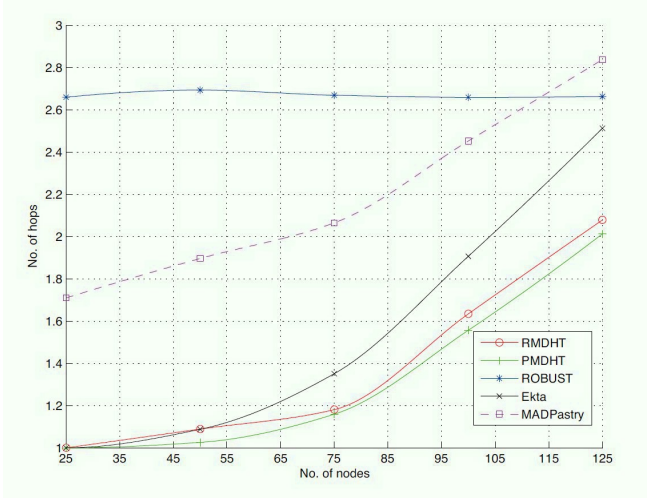


Figure 5: Look-up Average Path Length

the PMDHT, RMDHT and Ekta protocols are tightly integrated with their corresponding routing protocols at the network layer. ROBUST relies on broadcasting and obtaining IP to DHT IDs to create an exact copy of the physical network at the DHT layer.

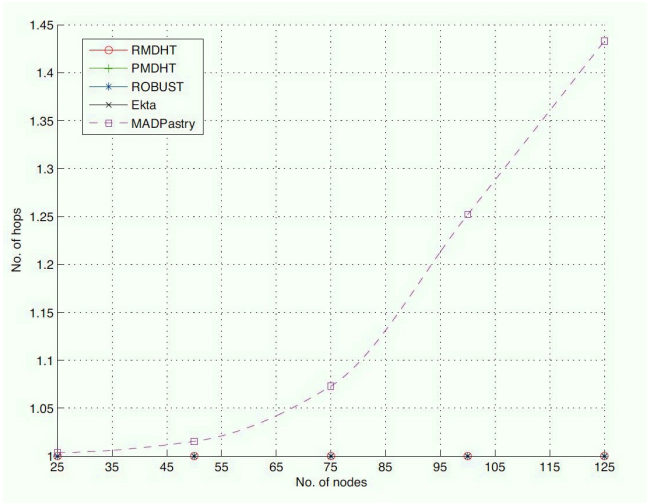


Figure 6: Logical Path vs Physical Path Stretch

5. CONCLUSION

In this paper, we presented two protocols that integrate peer-to-peer functionalities and routing. The DHT overhead in both protocols are similar to the DHTs optimized for MANETs namely Etka and MADPastry due to the similar aggressive PNS algorithms used. However, despite their similar overhead, we have shown that both RMDHT and PMDHT perform better than the rest of the protocols in terms of end-to-end delay when scaling the network size due to shorter average path length caused by utilizing MANET routing protocol routes. We have also shown

that RMDHT performs better than MADPastry in terms of end-to-end delay when increasing node speed due to shorter path lengths, whereas the delay caused by utilizing OLSRv2 due to the proactive routing protocol not handling high mobility as a result of constantly changing network conditions meaning routes not converging quick enough in PMDHT adversely affects performance in this case. We see that both proposed DHT protocols perform best overall in terms of look-up success rate when scaling the network due to shorter average path lengths, meaning less probability of dropped packets causing transmission failure across a route. Etka however outperforms both protocols in terms of success rate when increasing node speed due to lower packet loss; however Etka also has significantly higher end-to-end delay when increasing node speed in this scenario, thus RMDHT also performs well in this scenario. In PMDHT, we combine the DHT functionality with the routing at the network layer and piggyback DHT information inside the routing topology messages. Optimization is achieved in RMDHT by piggybacking DHT look-up information inside route request messages if such a packet is sent within a certain time period, else a DHT request packet is sent. The obtained simulation results have shown that both systems are suited to different scenarios. PMDHT 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 RMDHT seems more suited to scenarios with a high level of mobility when compared with the aforementioned protocols, it is slightly lower in terms of look-up 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 look-up applications when increasing node speed. The proposed protocols have each been evaluated while varying certain network conditions such as mobility or node speed. It would be of further interest to investigate how they 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. Another aspect not investigated in our simulations is that of node heterogeneity, currently all nodes have the same power. It would be interesting to see if any benefit can be gained from RMDHT and PMDHT in this case by optimizing the protocols for different aspects such as energy efficiency for example by using paths with the least number of neighbors and reducing the power needed for such neighbors 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 DHT data while other peers are able to perform look-ups whilst mobile.

6. ACKNOWLEDGEMENT

This work was partially funded by the EU FP7 Project; SALUS (Security and Interoperability for the Next Generation of PPDR Communication Systems) 2013 - 2016. Grant agreement no: 313296

REFERENCES

[1] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne. Resource location and discovery (reload) base protocol. RFC 6940, RFC Editor, January 2014.

[2] Bram Cohen. The BitTorrent Protocol Specification, Version 11031. http://www.bittorrent.org/beps/bep_0003.html, January 2008.

[3] A. Boukerche, A. Zarrad, and R.B. Araujo. A cross-layer approach-based gnutella for collaborative virtual environments over mobile ad hoc networks. *Parallel and Distributed Systems, IEEE Transactions on*, 21(7):911–924, July 2010.

[4] Diego N. da Hora, Daniel F. Macedo, Leonardo B. Oliveira, Isabela G. Siqueira, Antonio A.F. Loureiro, JosÃĺ M. Nogueira, and Guy Pujolle. Enhancing peer-to-peer content discovery techniques over mobile ad hoc networks. *Computer Communications*, 32(13âĂŞ14):1445 – 1459, 2009.

[5] Himabindu Pucha, Saumitra M. Das, and Y. Charlie Hu. Imposed route reuse in ad hoc network routing protocols using structured peer-to-peer overlay routing. *IEEE Transactions on Parallel and Distributed Systems*, 17(12):1452–1467, 2006.

[6] Thomas Zahn and Jochen Schiller. MADPastry: A DHT Substrate for Practicably Sized MANETs. In *Proc. of 5th Workshop on Applications and Services in Wireless Networks (ASWN2005)*, Paris, France, June 2005.

[7] Yuuta Nishihara, Yusuke Yokota, and Eiji Okubo. A technique for construction of overlay networks for manets with consideration of physical network topologies. In *Proceedings of the 2009 First International Conference on Advances in P2P Systems*, AP2PS '09, pages 66–71, Washington, DC, USA, 2009. IEEE Computer Society.

[8] Thirapon Wongsaardsakul and Kanchana Kanchanasut. A structured mesh overlay network for p2p applications on mobile ad hoc networks. In *Proceedings of the 4th International Conference on Distributed Computing and Internet Technology*, ICDCIT'07, pages 67–72, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] Thomas Zahn and Jochen Schiller. Designing structured peer-to-peer overlays as a platform for distributed network applications in mobile ad hoc networks. *Comput. Commun.*, 31(3):643–654, February 2008.

[10] Marcello Caleffi and L. Paura. P2p over manet: Indirect tree-based routing. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–5, March 2009.

[11] Bo Zhao, Yingyou Wen, and Hong Zhao. A d-tree based p2p-like routing algorithm for mobile ad-hoc networks. In *Intelligent System Design and Engineering Application (ISDEA), 2010 International Conference on*, volume 1, pages 14–19, Oct 2010.

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

[13] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a dht. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ATEC '04, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[14] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '01, pages 149–160, New York, NY, USA, 2001. ACM.

[15] AshwinR. Bharambe, SanjayG. Rao, VenkataN. Padmanabhan, Srinivasan Seshan, and Hui Zhang. The impact of heterogeneous bandwidth constraints on dht-based multicast protocols. In Miguel Castro and Robbert van Renesse, editors, *Peer-to-Peer Systems IV*, volume 3640 of *Lecture Notes in Computer Science*, pages 115–126. Springer Berlin Heidelberg, 2005.

[16] Adeep S Cheema, Moosa Muhammad, and Indranil Gupta. Peer-to-peer discovery of computational resources for grid applications. In *Grid Computing, 2005. The 6th IEEE/ACM International Workshop on*, pages 7–pp. IEEE, 2005.

[17] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[18] Grant P. Millar, Tipu A. Ramrekha, and Christos Politis. A peer-to-peer overlay approach for emergency mobile ad hoc network based multimedia communications. In *Proceedings of the 5th International ICST Mobile Multimedia Communications Conference*, Mobimedia '09, pages 59:1–59:5, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[19] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg. The optimized link state routing protocol version 2. RFC 7181, RFC Editor, April 2014. http://www.rfc-editor.org/rfc/rfc7181.txt.

[20] Charles Perkins, Stan Ratliff, John Dowdell, Lotte Steenbrink, and Victoria Mercieca. Ad hoc on-demand distance vector routing version 2 (aodvv2). Internet-Draft draft-ietf-manet-aodvv2-12, IETF Secretariat, October 2015. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodvv2-12.txt.

[21] Thomas Clausen and Philippe Jacquet. Rfc3626: The optimized link state routing protocol, 2003.

[22] T. Clausen, C. Dearlove, J. Dean, and C. Adjih. Generalized mobile ad hoc network (manet) packet/message format. RFC 5444, RFC Editor, February 2009. http://www.rfc-editor.org/rfc/rfc5444.txt.

[23] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2*. Springer Publishing Company, Incorporated, 1 edition, 2008.