Distributed Hash Tables for Peer-to-Peer Mobile Ad-hoc Networks with Security Extensions

Grant P. Millar, Emmanouil A. Panaousis and Christos Politis
Wireless Multimedia & Networking (WMN) Research Group, Kingston University London, UK
Email: {g.millar, e.panaousis, c.politis}@kingston.ac.uk

Abstract— **Serverless distributed computing, especially Mobile Ad-hoc NETworks (MANETs) have received significant attention from the research community. Peer-to-peer overlay networks have the potential to accommodate large-scale, decentralised applications that can be integrated into a MANET architecture to enable peer-to-peer communication among different mobile peers. These overlay architectures must be very resilient and their utilisation, reliability and availability must satisfy the needs of mobile computing. One must also heed the fact that the wireless nature of the medium introduces security vulnerabilities. The aim of the work described in this paper is twofold. First, we describe our peer-to-peer distributed hash table (DHT) architecture entitled Reliable Overlay Based Utilisation of Services and Topology (ROBUST). This is designed to be efficiently applied to MANETs. We additionally propose security extensions to protect the ROBUST signalling messages against malicious activities. We evaluate the ROBUST performance as well as the security extensions under varying levels of mobility and network sizes by building a custom DHT module for the network simulator ns-2. The outcome of the results show negligible overhead introduced by the extensions giving credence to their application in security sensitive scenarios.**

Index Terms— **peer-to-peer, mobile ad-hoc networks, distributed hash tables, security**

I. Introduction

The increase in computing power over time has caused an evolution in every day personal computers and laptops making them lightweight and portable with capabilities to act as both routers and clients in the network. This has spawned the creation of Mobile Ad hoc Networks (MANETs), which encompass the peer-to-peer (P2P) ad hoc paradigm. In MANETs all peers are able to send and retrieve data independently as well as act as routers (or intermediaries) for clients whom need to send data over multiple hops as a result of the target peer being physically further away than the maximum transmittable range of the source peer.

The motive for using MANETs is that no additional infrastructure is needed other than the devices themselves. Therefore to exploit the synergy of the peer-to-peer paradigm of MANETs, one must look towards an integrated solution to applications and information sharing, such as Distributed Hash Tables (DHTs). The motive for using DHT in MANETs is due to an extremely quick setup time in both application and network layer in addition to the fact that no additional infrastructure is needed in either layer other than the devices themselves. DHTs allow us to find the exact location of a party or piece of information stored within the network, using a piece of simple meta-data for example a name and domain, as proposed in Peer-to-Peer Session Initiation Protocol (P2PSIP). However the use of DHTs is not limited to simple name resolution and their distributed structure also allows for fast propagation and high availability of information through the network. When applied to MANETs which have no central authority, DHTs could provide the answer to distributed services such as DNS (Domain Name System), P2PSIP, distributed storage and information sharing, whilst aiding service lookup and discovery. Last but not least, all types of data could be stored redundantly and accessed easily and quickly by any peer.

A. Related Work

The area of research dedicated to aid the reduction of overhead from application layer architectures such as DHTs has seen significant activity in the recent past. A great amount of said research falls within the realm of static networks and Internet architectures, however a moderate amount of research can also be attributed to solutions in MANETs. We will have a overview of the aforementioned works in this section.

The paper [1] describes an algorithm designed to create a *perfect-ring* overlay in order to distribute messages to peers within a certain overlay group. A perfect-ring overlay is denoted as an overlay where sent messages are unidirectional and traverse all peers within a group once, before returning to the originator of the message. This can be useful when peers form groups or *clusters* which can be linked together in a ring structure. Whilst most DHT-overlays use a single-ring structure, the architecture proposed in [1] uses disjoint paths and routes for connecting overlay peers whom do not form part of the main ring. The algorithm is not reliant on any underlying infrastructure thus the overlay can function independently of the network layer and still be efficient. Another scientifically interesting issue is that the topology mismatch problem where the overlay network topology does not match that of the underlying network causing the overlay network to *stretch* out over the physical network. This issue must be considered in all DHTs designed for MANETs. In [1] the authors propose to

doi:10.4304/jnw.7.2.288-299

negate the problem by sending messages with very short Time To Live (TTL) when setting up the overlay. This creates an overlay with very strong proximity between the peers. The size of an optimised ring transmitted by a member of a group containing N members is $O(2N) \sim O(N)$.

The authors of [2] specifically examine cross-layer DHT MANET protocols. The examined architectures are Etka [3], Mobile Peer-to-peer Protocol (MPP) [4], a Gnutella optimisation for MANETs [5], FastTrack over AODV (Ad hoc On-demand Distance Vector) [6], and MADPastry [7]. Amongst these architectures, Etka and MADPastry are structured peer-to-peer overlays whilst the rest are unstructured. The Etka [3] architecture tightly integrates the structured P2P protocol based on DHTs with the routing architecture of MANETs by mapping logical DHT peer IDs to their MANET IP based counterparts causing the two separate architectures to merge into one structure. This is achieved by integrating the Pastry DHT with the DSR (Dynamic Source Routing) MANET multihop routing protocol at the network layer. MADPastry [7] is a DHT substrate which acts by combining the Pastry DHT with AODV MANET routing at the network layer. This can lower the overhead needed to maintain the DHT. While the architecture utilises three different routing tables (One akin to AODV's routing table, another akin to Pastry's routing table, and a leafset table) the only table requiring proactive management is that of the leafset table, with peers pinging their *left* and *right* respective leafs. The additional tables are updated by overhearing data packets destined for other peers.

In the majority of the proposed DHTs for MANETs published in the bibliography little thought has been given to security considerations. Especially, in all of the papers regarding DHT MANET protocols there is no mention of security for the DHT signalling messages. The most of the DHT-based P2P protocols take for granted that the participating peers behave legitimately and abstain from implementing major security measures. The latter are expected to be employed by software implementations that defend the P2P network against potential adversaries.

Adversaries within a P2P MANET network [8] are those peers which intentionally do not follow the protocol rules. For instance, a malicious peer might provide legitimate peers with erroneous lookup results or inoperative data. In addition, as far as P2P systems inherently rely on the relationships among the participated peers, trust issues arise with a need to be addressed by proper security extensions regarding signalling messages.

B. Contributions

In this paper we will describe our P2P DHT for MANETs entitled ROBUST (Reliable Overlay Based Utilisation of Services and Topology). ROBUST aims at decreasing the average path length and lookup time when sending DHT messages thereby decreasing *stretch*, while decreasing the maximum path length from the accepted $O(\log N)$ standard in DHTs designed for static networks to $O(\log C)$[1] complexity seen in most common DHTs used today.

In addition to the above, we will describe security extensions for ROBUST to protect the signalling messages against cryptanalysers. These extensions include (i) a *key exchange* phase where ROBUST peers exchange a pairwise symmetric key K_{pwk} material with their leafset peers as well as their super peers, (ii) a *key refresh* phase where peers generate new key material and exchange it with their leafset peers as well as their super peers, (iii) a *proximity synchronisation* phase where peers that wish to join a new cluster, to lower delay, have to accomplish a secure handshake and exchange key material with their new super peer and leafset peers.

C. Analysis and structure of this paper

In this paper we have first theoretically described the ROBUST DHT architecture as well as its security extensions. We have explained why ROBUST DHT is a suitable solution for MANETs and we have then discussed the total packet overhead of the architecture mentioning all the different packet types. To evaluate the performance of the architecture we have developed a module[2] for ns-2 (network simulator) which simulates the ROBUST DHT and its security features for MANETs at the packet-level.

The remainder of this paper is organised as follows. In section II we describe and discuss the Reliable Overlay Based Utilisation of Services and Topology (ROBUST) DHT scheme and its security extensions. In section III we describe the different types of packets of our model as well as the overhead introduced by these packets. In section IV we present the performance evaluation in terms of ns-2 simulation results. Section V concludes this paper discussing the benefits and the limitations of our solutions as well as our plans for future.

II. Proposed Architecture

Our proposed architecture is entitled ROBUST DHT [9] which stands for Reliable Overlay Based Utilisation of Services and Topology. 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 MANET peers.

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

[1] where N is the number of peers in the DHT and C is the number of clusters in ROBUST DHT.

[2] using C++ and TCL programming languages.

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 we need C clusters 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 $O(\log_2 C) \leq O(\log_2 N)$.

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. We propose to investigate two algorithms to increase the number of clusters. 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 we increase the number of clusters for every $C = \frac{N}{\log_2 N}$ and would need to decrease the size of each cluster by:

$$\frac{100}{C} - \frac{100}{(\frac{N}{\log_2 N})} \tag{1}$$

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:

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

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, there is no longer a need for so many clusters if $C < \frac{N}{\log_2 N}$ however to limit overhead and to maintain a threshold, the number of clusters will only be decreased when $\frac{N}{\log_2 N} < 2C$. 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, we do not currently foresee such possibilities in MANETs due to the restrictions of the routing protocols later discussed in this paper.

Peer proximity is central to our algorithm and we have designed the architecture 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.

The last factor taken into account in this paper 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; $u = w_1 t + w_2 b + w_3 l$. 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 some threshold denoted as $[v_1 v_2, \ ..., \ v_j]$ 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.

A. ROBUST Signalling

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 we can denote the upper bound on hops as $DHT_{hmax} = O(log_2 N)$ where n is equal to the total number of peers in the overlay network. However in ROBUST we reduce this by routing request through super peers and hence having an upper bound on hops described as $DHT_{hmax} = O(\log_2 C)$.

In the model we assume the underlying transport layer protocol is UDP as used in OpenDHT [10] 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, we expect reliable transfers by use of acknowledgement packets and sequence numbers in the DHT and the use of *round trip timers*. We can calculate the total packets (space) needed for DHT data gathering, and overlay maintenance for each peer denoted as:

$$\begin{aligned} DHT_{proc} = {} & DHT_{get}(no\_gets) + DHT_{put}(no\_puts) \\ & + DHT_{ack} + DHT_{sync}(t_{sync}) + DHT_{ls\_up}(t_{ls}) \\ & + DHT_{ping}(t_{ping}) + DHT_{clstr\_beacon}(t_{beacon}) \\ & + DHT_{prox\_sync}(t_{prox\_sync}) \\ & + DHT_{join}(no\_joins) \end{aligned} \tag{3}$$

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

$$DHT_{get} = 2DHT_{get\_req} \cdot DHT_{hmax} \cdot no\_gets \tag{4}$$

This equation has been derived based on the fact that there are two packet types sent for one get request (GET and GET_SUCCESS) therefore the total must be multiplied by a factor of 2, we then take into account each overlay hop denoted as DHT_{hmax} and multiply it for the total number of gets for the systems, in order to get the spacial overhead. Similarly we calculate all puts in the DHT as:

$$DHT_{put} = 2DHT_{put\_req} \cdot DHT_{hmax} \cdot no\_puts \tag{5}$$

We then calculate the packets required for data synchronisation across all peers for a particular given leafset:

$$\begin{aligned} DHT_{sync} = {} & (DHT_{sync\_vals} + \\ & DHT_{str\_keys}) \cdot DHT_{hmax} \cdot \frac{T}{t_{sync}} \end{aligned} \tag{6}$$

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

$$DHT_{ls\_up} = (DHT_{pull\_ls} + DHT_{push\_ls}) \cdot DHT_{hmax} \cdot \frac{T}{t_{ls}} \tag{7}$$

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

$$DHT_{join} = DHT_{join\_req} \cdot DHT_{hmax} \cdot C \cdot no\_joins \tag{8}$$

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. We can calculate all pings in the DHT as:

$$DHT_{ping} = DHT_{ls\_peer} \cdot DHT_{hmax} \cdot \frac{T}{t_{ping}} \tag{9}$$

where DHT_{ls_peer} is the leafset 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, we can describe this transaction as:

$$\begin{aligned} DHT_{clstr\_beacon} = {} & [(DHT_{beacon} + \\ & + DHT_{ping}) \cdot DHT_{1\_hop}] \cdot \frac{T}{t_{beacon}} \end{aligned} \tag{10}$$

We will now examine the *Proximity synchronisation* function, 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:

$$\begin{aligned} DHT_{prox\_sync} = {} & (DHT_{move\_req} + DHT_{move\_rep} \\ & + DHT_{part} \cdot DHT_{hmax} \\ & + DHT_{broad\_part} \cdot DHT_{ls} \\ & + 3DHT_{ls\_up} \\ & + 3DHT_{sync}) \cdot DHT_{move\_peer} \end{aligned} \tag{11}$$

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 leafset 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 leafset (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 leafset. 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'. We finally calculate the total number of acknowledgement messages needed to acknowledge all the above mentioned packets as follows:

$$DHT_{ack} = DHT_{get} + DHT_{put} + DHT_{sync} + DHT_{ls\_up} + DHT_{ping} + DHT_{clstr\_beacon} + DHT_{prox\_sync} \quad (12)$$

B. Assumptions and limitations of ROBUST DHT

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 leafset peers to route information.

Another limitation of ROBUST DHT is based on the restrictions of MANET routing protocols, at the time of writing IETF has two main 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. We try to counter this 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 [11]. This was later confirmed by the authors in [12] where using a real-world testbed implementation of 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.

In Fig. 1 we show 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 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 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 leafset.
5) Upon SP_1 receiving the DHT_{part} packet, it first sends a DHT_{broad_part} packet to P_1s previous leafset 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 leafset and reply to P_1 with the leafset peers which fall within P_1s leafset.
7) The leafset nodes of peer P_1 learn of his existence through the leafset update function which runs every t_{ls} period and chooses a random existing

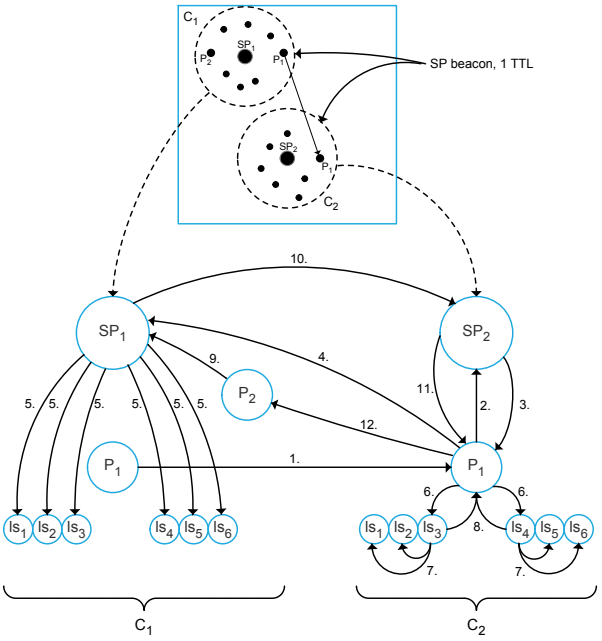

Figure 1. An overview of the described DHT architecture.

leafset peer to sync with.

8) The leafset peers of P_1 run the DHT_{sync} data synchronisation function every t_{sync} interval randomly choosing a leafset 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.

C. Security Extensions

According to ROBUST different types of signalling messages have to be exchanged amongst peers during the networks' lifetime. Thus potential attackers could find a way to exploit security vulnerabilities which appear due to the transmission of these messages. To give a clear picture of how harmful the existence of malicious peers against the ROBUST DHT can be we have derived the maximum length of an overlay (logical) route in hops between a source and a destination peer. This is equal to $\log C+2$ where C is the number of clusters in the overlay network. Further, $C = \lceil \frac{N}{\log_2 N} \rceil$ namely the maximum length of an overlay route is

$$\log N - \log(\log N) + 2 \tag{13}$$

Assuming that the probability of a peer to be malicious is m the probability to have a secure route namely a route that consists only of legitimate peers is equal to

$$P_{secure\_route} = (1-m)^{\log N - \log(\log N)+2} \tag{14}$$

From this we can derive that the impact of the attacker is severe even for a small fraction of malicious peers. Thus security provisions to maintain reliable communication under malicious activities must be provided.

To cope with external adversaries the ROBUST DHT needs to be extended in a way that peers will exchange cryptographic material. In this context we describe in the following how peers as well as super peers exchange cryptographic pairwise symmetric keys. These keys will then be used to encrypt ROBUST signalling information in a pairwise manner. This means that each pair of peers (including super peers) will use a specific symmetric key to encrypt the signalling information.

We assume that in the beginning of the networks life, a global 128-bit symmetric AES (Advanced Encrypted Standard) pre-shared key called a *network wide key* (K_{nwk}) has been installed in all the devices of the mobile peers. Similarly groups of users who wish to share data securely can use such a pre-shared key much like common security encryption standards use today for example Wired Equivalent Privacy (WEP) and 802.11i pre-shared key mode (PSK). The use of K_{nwk} defends a MANET against *man-in-middle* attacks during the exchange of the peers' pairwise symmetric keys. Another way of acquiring the K_{nwk} it could consider a secure side channel (e.g. bluetooth) communication or physical contact in the beginning of the network's lifetime. It is also worth mentioning here that the use of symmetric rather than asymmetric cryptography is due to asymmetric cryptographic algorithms being in the order of 1000 times slower than symmetric algorithms as well as they introduce higher energy cost [13].

On the other hand, the reason why we do not use the K_{nwk} for the duration of the network's lifetime is due to the ample opportunities for cryptanalysers to retrieve the key material. By exchanging pairwise symmetric keys and refreshing at a certain interval we minimise the risk of successful cryptanalysis activities whilst we prevent compromised peers to read information exchanged between other peers assuming that there is very limited probabilities a peer to be compromised before the exchange of the pairwise symmetric keys.

The security extensions for the ROBUST signalling messages consist of three main phases as described in the following;

- *key exchange* phase: during this phase ROBUST peers exchange their pairwise 128-bit AES symmetric keys K_{pwk} with their leafset peers and their super peers by sending a DHT_{key_exch}. To this end, peers use the K_{nwk} to encrypt the DHT_{key_exch} as well as the ROBUST packet header.
- *key refresh* phase: the task of this phase is for any given peer to generate new key material and exchange with their leafset peers and super peer by sending them a DHT_{key_refr} every t_{key_refr} seconds. For the encryption and transmission of the new keys, peers use the previous established symmetric keys K_{pwk}.
- *proximity synchronisation* phase: during this phase peers move closer to a new super peer SP' and consequently should move to the new cluster by adopting a new ID in the DHT space. In this case, peers must use the K_{nwk} to send their symmetric keys to their new leafset peers in addition to the new super peer. Therefore before they join the new cluster they send a DHT_{part} packet to all of their previous leafset peers and super peer.

All phases consist of a 2-way handshake between two peers as illustrated in Fig. 2 and 3. It is worth stressing here the following;

- Since the security extension algorithm is distributed, there is a possibility both peers that participate in a handshake send their K_{pwk} to each other at almost exactly the same time consequently having two different K_{pwk} at the end of the handshake. To

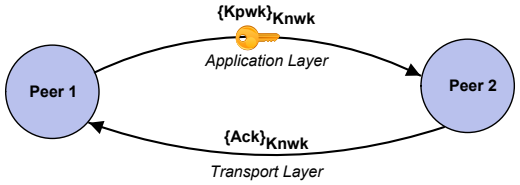

Figure 2. The 2-way handshake between two peers which are exchanging a pairwise symmetric key K_{pwk} during the *key exchange* or *proximity synchronisation* phase.

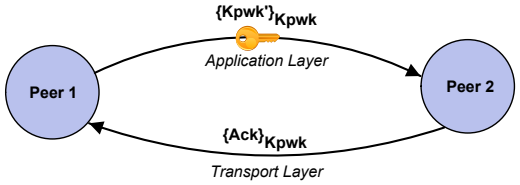

Figure 3. The 2-way handshake between two peers which are exchanging a pairwise symmetric key K_{pwk} during the *key refresh* phase.

avoid this issue, we have assumed that both peers keep track of the exact time they send their K_{pwk}. If they receive a K'_{pwk} from the target peer before they receive an acknowledgement for the K_{pwk} they previously sent, they will compare the send time of the received K'_{pwk} with the time they sent the K_{pwk} to the target node if the send time of $K_{pwk} < K'_{pwk}$ they will discard the key K'_{pwk} and use K_{pwk} for encryption with the target peer following suit.

- The 2 steps in the handshake combined with a round trip timer are adequate to guarantee that the sender of the K_{pwk} will know that the other peer has received the $\{K_{pwk}\}_{K_{nwk}}$[3], when it receives the acknowledgement in the second step of the handshake. If the sender of the K_{pwk} does not receive the aforesaid acknowledgement within the certain RTT (round trip time), it resends the key packet to the target peer assuming the original packet was dropped.
- The purpose of the key refresh is to harden the ability of a compromised peer to reveal any pairwise key material and information from any encrypted signalling packets. This is based on the fact that a compromised peer would need to overhear the key exchange of the first pairwise key which is encrypted with the K_{nwk} in addition to the subsequent keys thereafter making it extremely hard for an attacker even with the K_{nwk} to decrypt refreshed K_{pwk} messages.
- To satisfy confidentiality for the different DHT signalling packets, as previously presented, we use one of the K_{nwk}, K_{pwk} depending on the type of the ROBUST packet.

In Table I we summarise the different DHT signalling packets as well as the required keys per packet type regarding the different phases of the proposed security extensions. Authentication and integrity are both satisfied by ROBUST using HMAC (Hash-based Message Authentication Code). To this end, a hash function is applied to the cipher-text of the message using the proper symmetric key, depending on the DHT signalling packet type. The receiver of the signalling checks the message digest to verify the authenticity of the sender and to identify whether the message was altered compared to the one sent by the originator (due to intermediate MANET nodes routing the packet).

D. Assumptions and limitations of the Security Extensions

One of the limitations of our work, is the use of a common symmetric key called *network wide key* (K_{nwk}) shared amongst all the peers. Within this context, we assume that in the beginning of the networks life a K_{nwk} has been installed in all the devices of the mobile peers. Another way of acquiring the K_{nwk} could be established by peers initiating security association amongst each other by means of secure side channel (e.g. bluetooth) communication or physical contact in the beginning of the network's lifetime.

A further assumption of the security extensions for ROBUST signalling messages is that the beacon packet used to advertise super peers existence in order to estimate proximity to said super peer by a normal DHT peer must always be encrypted by the K_{nwk} due to its broadcast nature.

Regarding the *proximity synchronisation* phase we have assumed that peers joining a cluster communicate the new pairwise symmetric key for future transactions with their new super peer, as well as their new leafset peers. This is accomplished by using the K_{nwk}, as the overhead required to use existing secure channels[4] is deemed to outweigh the risks.

III. Performance Evaluation

In this section, we use simulations to verify the integrity of the proposed model and showcase the benefits of using our proposed solution. We next use an event based simulator, customised with our implementation of ROBUST protocol, to validate the proposed model and optimisation solution. The authors have developed a simulator module for the packet-level network simulator ns-2. 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. In addition, the different phases of the security extensions are implemented fully in ns-2 and are utilised in order to route ROBUST packets. Further lower layers are also simulated in the ns-2 simulator and these characteristics are taken into account.

The setup of our network comprises of randomly distributed peers throughout an area of $1km^2$. For smaller

[3]the notation $\{A\}_K$ means that A has been encrypted with the cryptographic key K.

[4]going through their previous super peer SP for which both the joining peer and the SP' have established pairwise symmetric keys.

TABLE I.
LIST OF REQUIRED SIGNALLING ROBUST PACKETS AND ASSOCIATED CRYPTOGRAPHIC KEYS

ROBUST packet	**before the key exchange phase**	**after the key exchange phase before or during the refresh phase**	**proximity synchronisation phase**
DHT_{get_req}	K_{nwk}	K_{pwk}	-
DHT_{put_req}	K_{nwk}	K_{pwk}	-
DHT_{ping}	K_{nwk}	K_{pwk}	-
DHT_{beacon}	K_{nwk}	K_{nwk}	-
DHT_{join_req}	K_{nwk}	K_{nwk}	-
DHT_{ack}	K_{nwk}	K_{pwk}	K_{pwk}
DHT_{sync_vals}	K_{nwk}	K_{pwk}	K_{pwk}
DHT_{str_keys}	K_{nwk}	K_{pwk}	K_{pwk}
DHT_{pull_ls}	K_{nwk}	K_{pwk}	K_{pwk}
DHT_{push_ls}	K_{nwk}	K_{pwk}	K_{pwk}
DHT_{move_req}	-	-	K_{pwk}
DHT_{move_rep}	-	-	K_{pwk}
DHT_{part}	-	-	K_{pwk}
DHT_{broad_part}	-	-	K_{pwk}
DHT_{key_exch}	K_{nwk}	-	K_{nwk}
DHT_{key_refr}	K_{nwk}	K_{pwk}	

networks such as ten peers, the peers are considerably closer together in order to stay within transmission range of one another. Puts and Gets (Data transmission and retrieval) are called in the DHT at a rate of one request per second. The number of peers simulated ranges from ten peers to seventy peers with increments of twenty peers. The authors specifically chose this amount of peers to represent smaller networks and also investigate the scalability of the aforementioned approaches. During tests the authors found the threshold 90ms to be sufficient to allow peers to change cluster when SN RTT is less than $SN'RTT + 90ms$. The threshold is needed so that peers do not move cluster when even a small delay increase is experienced. In addition to the threshold we have also implemented the algorithm to always take the best RTT for the current super peer SN and the worst RTT for the new super peer SN' over three subsequent RTTs. This ensures that we can guarantee the super peer SN' to have a better RTT than SN for a total duration of $3t_{beacon}$. All simulations were run for a total of 1000 seconds simulation time. This was chosen in order for the DHT and network to stabilise. In the simulation experimentations regarding the ROBUST DHT we have assumed static super peers. One of the limitations of our simulations are the fact churn is not simulated per-se. This is due to the fact the authors have decided to only simulate mobility churn in this paper as adding churn would detract from the goal of investigating these results. We consider two main different scenarios in our simulations; one without security extensions (pure ROBUST) and one with security additions for signalling (secure ROBUST). The values for the intervals of the different DHT functions are based on those used in OpenDHT [10]. The list of simulation parameters can be seen in table II.

TABLE II.
SIMULATION PARAMETERS

Network size (number of peers)	10, 30, 50, 70
Number of clusters needed (C)	4, 7, 9, 12
Percentage of mobile peers	0%, 25%, 50%
Area size	1000m x 1000m
Data packet payload size	512 bytes
MANET Routing protocol	OLSR
MAC layer	802.11b
Link bandwidth	11Mbit/s
Maximum transmission range	250m
Node moving speed	1m/sec
Maximum number of overlay hops	$O(\log_2 C) + 2$
Types of traffic	UDP (All aforementioned DHT and security packets)
Maximum number of traffic connections	$ls + C$
Simulation time	1000 sec
DHT data distribution	Random
DHT node ID distribution	Random
Number of DHT get requests (no_gets)	1/sec
Data synchronisation interval (t_{sync})	3 sec
Leafset update interval (t_{ls})	4 sec
Neighbour ping interval (t_{ping})	5 sec
Super peer change RTT threshold	90ms
Cluster beacon interval (t_{beacon})	10 sec
Key refresh interval (t_{key_refr})	300 sec
Proximity synchronisation interval (t_{prox_sync})	60 sec

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

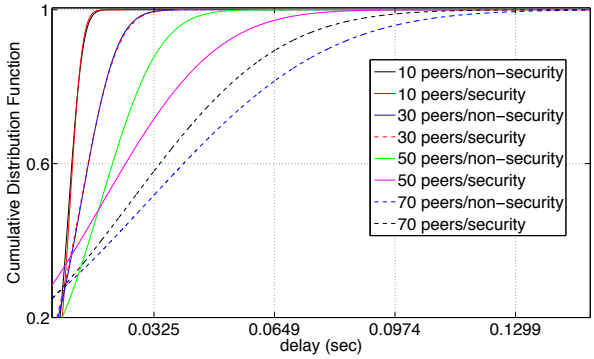

Figure 4. The cumulative distribution function of the DHT_{get_req} from transmission to completion of the request for the 8 scenarios where none of the peers are mobile.

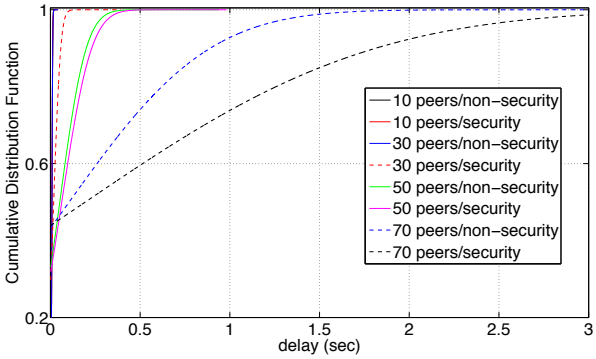

Figure 5. The cumulative distribution function of the DHT_{get_req} from transmission to completion of the request for the 8 scenarios with 25% of the peers mobile.

- 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 (we have chosen OLSR [14] (Optimised Link State Routing).
- 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.

The graph in Fig. 4 shows the cumulative distribution function of end-to-end DHT_{get_req} requests for 10-70 peers in a state with and without security when the network has no mobility (all peers are static). One can see from this figure that when the network is ad-hoc as apposed to mobile ad-hoc the delay experienced when getting data from the DHT is minimal as for 90% of all cases the end-to-end delay is less than 100ms. We can see that in almost all cases the security and non-security scenarios experience roughly the same delay (within a 5ms variance) except for 70 peers where the variance is extended to less than 20ms. The higher experienced delay here for 70 peers without security can be attributed to a slight variation of the distribution of peer and data IDs in the DHT, i.e. the data IDs are distributed more evenly in the non-security scenario creating more DHT_{sync} packets and higher redundancy, at the cost of slightly higher delay.

Fig. 5 demonstrates the cumulative distribution function of end-to-end DHT_{get_req} requests for 10-70 peers in a state with and without security when the network has 25% mobility (25% of the peers are moving at 1m/sec). As one would expect the delay in smaller networks (10-30 peers) is very small with 80% of the round trip times (RTT) being less than 70ms due to no congestion, interference and the fact that peers hardly move out of a 1 hop range. When the network size is increased to 50 peers where 13 peers are moving, we see slightly higher delay due to broken links causing packet retransmition and more packets being sent over the network increasing congestion (primarily DHT_{prox_sync} packets when a node moves cluster). We see this evidently more for 70 peers (18 moving) where the delay increases greatly. The results here show a clear indication that adding the security packets increases delay due to the time the peer has to wait to establish keys before transmitting data, however for 30 and 50 peers this is less than 50ms, whereas for 70 peers this is increased to less than 600ms. The great difference here is due to a much higher rate of peers moving cluster causing higher delays due to packet loss and interference as confirmed in the paper [12]. Packet loss can cause very high delay times in get request RTT such as those experienced here due to the dropped packet timer implemented in ROBUST. Based on the OpenDHT implementation [10] when a packet is sent and a round trip timer is started with an expiry time of the RTT of the last successfully transmitted packet to the specific target peer, if the timer expires the packet is retransmitted and the expiry time is doubled. Due to this the RTT can increase exponentially when experiencing particularly high packet loss.

The graph Fig. 6 presents the results of the cumulative distribution function for end-to-end DHT_{get_req} requests for 10-70 peers in a state with and without security when the network has 50% mobility (50% of the peers are moving at 1m/sec). In keeping with the results for 25% mobility we can see that the smaller network sizes (10-30) experience less than 210ms delay for 80% of the requests, while this is significantly higher than the previous results, it is not unexpected due to the increasing volatility of the network. In this scenario the network with 50 peers has a sharp increase in the end-to-end delay compared with Fig. 5 due to the aforementioned factors, mainly resultant of peers moving more frequently. One can see

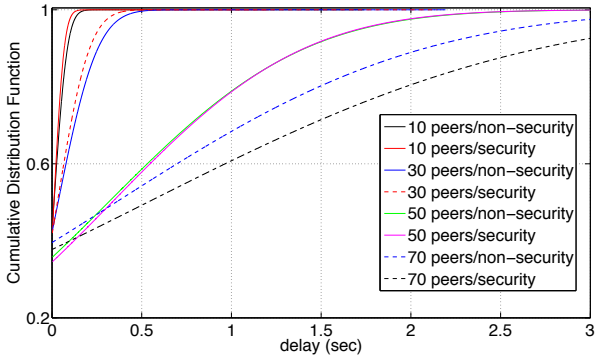

Figure 6. The cumulative distribution function of the DHT_{get_req} from transmission to completion of the request for the 8 scenarios with 50% of the peers mobile.

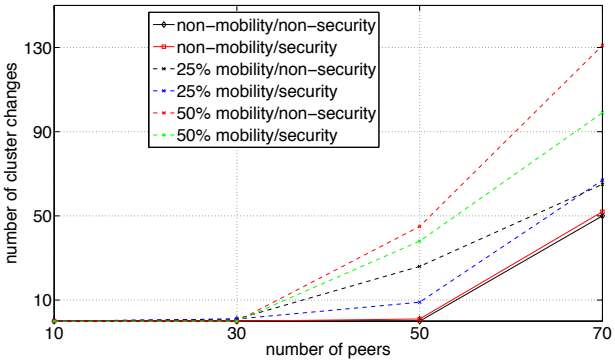

Figure 8. The number of peers whom change cluster due to the DHT_{prox_sync} function for the number of peers 10-70 for all the above mentioned scenarios.

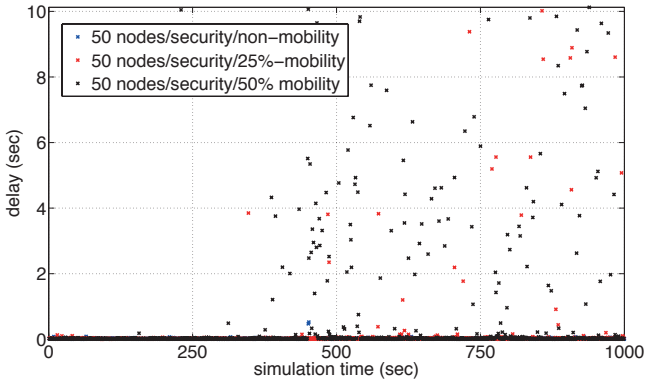

Figure 7. The end-to-end DHT_{get_req} delay for 50 peers for the scenarios with 0%, 25% and 50% of the peers mobile with security extensions enabled.

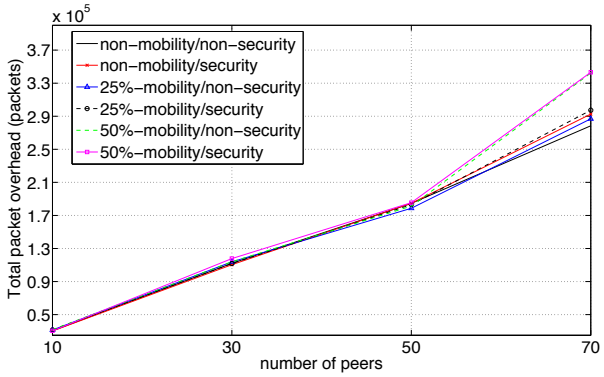

Figure 9. The total packet overhead for the 6 mobility and security scenarios for each number of peers.

again the overhead of security only marginally affecting the delay with a maximum different with 70 peers of 300ms.

One can see a sample of end-to-end delay for DHT_{get_req} requests for 50 peers with security extensions enabled in Fig. 7 following the trend of the previous graphs we can clearly see that he delay for 50% of the peers mobile is much more varied than the other two scenarios as expected. It is interesting to note that during the stabilisation period of 200 seconds we do not see high delay for any of the scenarios. However when the peers become mobile around 200 seconds the delay for a small percentage of the DHT_{get_req} requests increases rapidly.

Fig. 8 displays the number of peers whom change cluster due the DHT_{prox_sync} which compares the RTT of a new super peer with that of their current super peer. We can see a general trend here which shows that the peers without security change cluster more times than the peers with the security extensions enabled. If we compare this trend to what we see in Figs. 4-6 where the end-to-end delay for scenarios without security is lower, one can see that the DHT_{prox_sync} actually reduces overall end-to-end delay due to greater proximity of the overlay peers to their physical network neighbours.

Fig. 9 represents the total number of packets received during the networks lifetime during each scenario for a given number of peers. Here one can see that for each scenario the security extensions add a noticeable number of packets in the results, but still not enough to add any real difference in terms of congestion. It is interesting to note that for scenarios 10-50 peers the results are hardly distinguishable from each other, this shows that the threshold for DHT_{prox_sync} impacts more in the 70 peer network than all of the others due to higher RTT delay variance.

Confirming the notion in Fig. 9, in Fig. 10 one can determine the actual real number of total security packets received during the networks lifetime to be negligible, with the maximum number of security packets received at 70 peers with 50% mobility as expected due to DHT_{prox_sync}. One can say with clarity that adding 3600 packets to a total over 3×10^5 would not produce any noticeable difference in network behaviour, on the contrary any resulting impact from the security would have to be delay wise, while waiting for a packet to arrive due to the congestion caused by the total number of packets. This is more noticeable with the security extension as one has to wait for this procedure to complete before transmitting secure data. The results in fig. 10 do not include duplicate packets sent, which might result

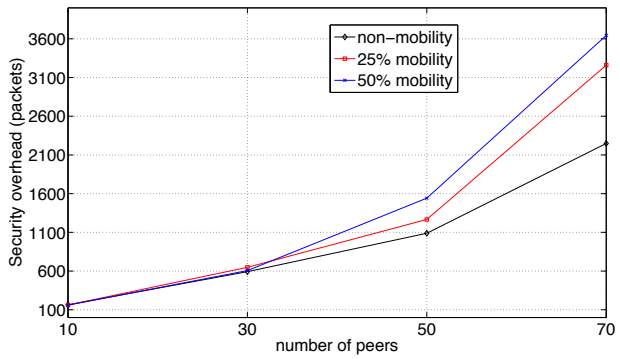

Figure 10. The packet overhead experienced due to security extensions for ROBUST for each of the different mobility levels and for all the number of peers.

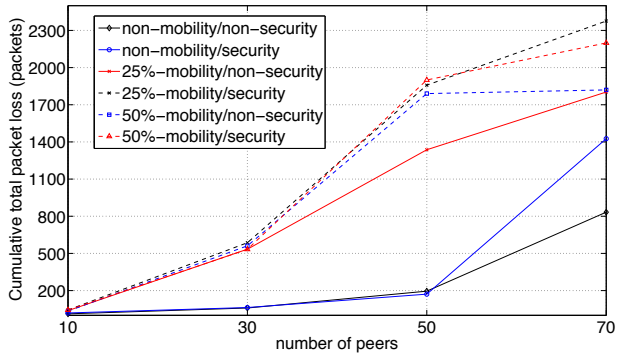

Figure 11. The cumulative packet loss experienced for each of the 6 mobility scenarios.

in more overall packets being sent from the security function.

The results in Fig. 11 provide insight into the cumulative packet loss experienced during the networks lifetime for any given number of peers simulated for each mobility scenario with and without security. One can gather that while there is packet loss, it is not experienced on a large scale. While the general trend shows that the security scenarios have a slightly higher packet loss, when compared with the total number of packets received this amounts to less than 1 percent. The reason for such a low packet loss can be explained by using the results from the paper [11] as a guide. Due to frequent route change multiple copies of a single packet can be received, this causes a lot of duplicate packets in the network and a phenomenon we experienced with a high impact when simulating 70 peers with high mobility due to congestion.

IV. Conclusion

In this paper the authors have proposed using a new DHT architecture entitled ROBUST in order to share and disseminate data and information throughout MANETs based on the P2P paradigm which both networks share. The authors have extended this notion to encompass security extensions in order for all DHT transactions to be private between only those participating in the overlay and appear encrypted to any intermediate MANET nodes. To this end the authors have simulated the proposed architecture and extensions in the packet-level simulator ns-2. The results show that while there is a slight difference in performance when applying aforementioned security extensions to the DHT, the impact of these extensions is negligible for the overwhelming majority of cases. It can be seen however that when increasing the number of peers so much as 70, higher delays can occur when many peers are moving. The authors attribute this to a number of different phenomenon which occur when mobility is introduced such as the transport issues highlighted in [11] where a high variance in RTT occurs when routes change often, leading to underestimation of RTT for many packets causing unnecessary duplicates to be sent, further congesting the network and increasing delay.

The resulting contributions from this paper are important for the future design and implementation of P2P protocols for MANETs, especially in the cases where data security and confidentiality is of high importance. Future work in the area of P2P for MANETs must address the scalability issues experienced when a high number of peers are mobile. While ROBUST goes some way to addressing this problem with proximity synchronisation, it is clear that a lot of the encountered discrepancies stem from an inefficient transport protocol which should be addressed. The authors main focus of future work will be within the area studying how mobility affects DHTs and solutions to the problems that entails, as well as improving the simulator by adding new transport protocols, adding super peer election, enhancing proximity synchronisation and investigating the effects of churn.

One can clearly see the advantages of having secure P2P overlays in MANETs, adding much functionality to an otherwise desolate landscape in terms of services and information sharing while maintaining secure communication between trusted peers. The advancement of solutions to the problems posed in the previously mentioned scenarios and the continued research into future services will ensure that MANETs do have functionality which goes beyond that of simply acting as gateways to more service rich architectures such as the Internet.

Acknowledgment

This work was undertaken in the context of the project ICT SEC-2007 PEACE (IP-based Emergency Applications and serviCes for nExt generation networks) with contract number 225654. The project has received research funding from the European 7^{th} Framework Programme.

References

[1] A. Banerjee and C.-T. King, “Building ring-like overlays on wireless ad hoc and sensor networks,” *IEEE Trans. Parall. and Distr. Syst.*, vol. 20, no. 11, pp. 1553 –1566, nov. 2009.

[2] M. Bisignano, G. Di Modica, O. Tomarchio, and L. Vita, "P2p over manet: a comparison of cross-layer approaches," *Proc. of Database and Expert Systems Applications (DEXA)*, pp. 814 –818, sep. 2007.
[3] H. Pucha, S. M. Das, and Y. C. Hu, "Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks," *Proc. IEEE WMCSA*, pp. 163–173, 2004.
[4] R. Schollmeier, I. Gruber, and F. Niethammer, "Protocol for peer-to-peer networking in mobile environments," *Computer Communications and Networks, 2003. ICCCN 2003. Proceedings. The 12th International Conference on*, pp. 121 – 127, oct. 2003.
[5] M. Conti, E. Gregori, and G. Turi, "A cross-layer optimization of gnutella for mobile ad hoc networks," *Proc. ACM MOBIHOC*, pp. 343–354, 2005.
[6] B. Tang, Z. Zhou, A. Kashyap, and T. Chiueh, "An integrated approach for p2p file sharing on multi-hop wireless networks," *Proc. of IEEE WIMOB*, vol. 3, pp. 268–274, 2005.
[7] T. Zahn and J. Schiller, "Madpastry: A dht substrate for practicably sized manets," *Proc. of ASWN (Applications and Services in Wireless Networks) Conference*, June 2005.
[8] D. Chopra, H. Schulzrinne, E. Marocco, and E. Ivov, "Peer-to-peer overlays for real-time communication: security issues and solutions," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 4–12, First Quarter 2009.
[9] G. Millar, E. Panaousis, and C. Politis, "Robust: Reliable overlay based utilisation of services and topology for emergency manets," in *Proc. IEEE Future Network and MobileSummit*, Florence, Italy, Jun. 2010.
[10] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a dht," in *Proc. USENIX*, ser. ATEC '04. Berkeley, CA, USA: USENIX Association, 2004.
[11] D. Kim, H. Bae, and C. K. Toh, "Improving tcp-vegas performance over manet routing protocols," *IEEE Trans. on Veh. Techn.*, vol. 56, no. 1, pp. 372 –377, jan. 2007.
[12] L. Barolli, M. Ikeda, F. Xhafa, and A. Duresi, "A testbed for manets: Implementation, experiences and learned lessons," *Systems Journal, IEEE*, vol. 4, no. 2, pp. 243 –252, June 2010.
[13] F. Anjum and P. Mouchtaris, *Security for wireless Ad hoc networks*. Wiley, 2007.
[14] T. Clausen and P. Jacquet, "Rfc 3626: Optimized link state routing protocol (olsr)." [Online]. Available: http://tools.ietf.org/html/rfc3626

Grant P. Millar is currently a Ph.D student at the Faculty of Science Engineering & Computing (SEC), School of Computing & Information Systems (CIS), Kingston University, London, UK. He works in Wireless Multimedia & Networking (WMN) Research Group as well as giving the occasional lecture on various subjects of the networking realm within the CISM faculty.

Grant received his M.Sc in Networking and Data Communications at Kingston University and his B.Sc in Computer Animation at the University of Portsmouth, UK. Grant is a student member of the IEEE and the IEEE Communications Society.

Emmanouil A. Panaousis is currently a research Ph.D. student at the Faculty of Science Engineering & Computing (SEC), School of Computing & Information Systems (CIS), Kingston University, London, UK. He works in the Wireless Multimedia and Networking (WMN) Research Group as well as giving the occasional lecture on various subjects within the TCP/IP and wireless networking field.

Emmanouil was born in Athens, Greece and he received his M.Sc., with distinction, in Computer Science at the Department of Informatics of the Athens University of Economics and Business and his B.Sc. in Informatics and Telecommunications at the National and Kapodistrian University of Athens. He has published more than 20 papers in international journals, conferences and standardisation bodies. He is also a patent holder. Emmanouil's previous work experience includes laboratory assistant in Athens University of Economics and Business, assistant in Foundation for Research and Technology - Hellas (FORTH) and Institute of Applied and Computational Mathematics (IACM). Emmanouil is a student member of the IEEE, the IEEE Communications Society and the IISP (Institute of Information Security Professionals).

Christos Politis is a Reader (Assoc. Prof.) of Wireless Communications at Kingston University London, UK, Faculty of Science Engineering & Computing (SEC), School of Computing & Information Systems (CIS). There he leads a research team on Wireless Multimedia & Networking (WMN) and teaches modules related to communications. Christos is the Field Leader for the 'Wireless Communications' and 'Networks and Data Communications' postgraduate courses at Kingston.

Prior to this post, he was the Research and Development (R&D) project manager at Ofcom, the UK Regulator and Competition Authority. There he managed a number of projects across a wide range of technical areas including cognitive radio, polite protocols, radar, LE Applications, fixed wireless and mobile technologies. Christos' previous positions include telecommunications engineer with Intracom Telecom in Athens and for many years he was a post-doc research fellow in the Centre for Communication Systems Research (CCSR) at the University of Surrey, UK. He is being active with European research since 2000 and has participated in several EU, national and international projects. Christos was the initiator and the project manager of the IST UNITE project. He is a patent holder, and has published more than 100 papers in international journals and conferences and chapters in two books. Christos was born in Athens, Greece and holds a PhD and MSc from the University of Surrey, UK and a B.Eng. from the Technical University of Athens, Greece. He is a senior member of the IEEE and member of the Technical Chamber of Greece.