Efficient Backbone Routing in Hierarchical MANETs

Thomas Kunz(✉)

Systems and Computer Engineering, Carleton University, Ottawa, Canada
tkunz@sce.carleton.ca

Abstract. Hierarchical network architectures are widely deployed to reduce routing overheads and increase scalability. In our work, we are interested in large-scale Mobile Ad-Hoc Networks (MANETs) which are formed by interconnecting smaller clusters through a backbone. To support end-to-end routing in such networks, we employ a hierarchical approach as follows. The clusters are MANETs, running OLSR locally. Each cluster has a gateway, and the gateways are interconnected through a backbone. In this paper, we study four different solutions to provide end-to-end connectivity through the backbone: flooding all data packets through the backbone, modifying an ad-hoc routing protocol such as OLSR and AODV, or using a P2P overlay for routing purposes. Running extensive simulations in OMNeT++, our results highlight the strengths and weaknesses of each approach. Flooding, albeit a very simple approach, appears to be quite competitive with more complex routing solutions, with good performance and low overheads.

Keywords: MANETs · P2P overlays · Hierarchical routing

1 Introduction

Mobile Ad hoc Networks (MANETs) are finding applications in a range of areas, including emergency response networks, intelligent transportation systems, outdoor enterprises, small businesses etc. [10,22,23]. One important characteristic is that they are self-organizing and self-configuring wireless multi-hop networks which do not rely on any existing infrastructure to exist; as nodes are by themselves, servers and clients [4,14]. Each node must act as a router to forward traffic unrelated to its own use.

The number of users in MANET applications may vary from just a handful to hundreds of thousands of people and more [2]. As MANETs and mobile devices become increasingly popular and the ensuing networks grow larger, more research effort focuses on devising protocols for route establishment and maintenance in these networks. In a flat network of several interconnected mobile devices, and spanning a large geographical area, for instance, the network will typically incur increasing overheads for route maintenance and establishment

L. Foschini and M. El Kamili (Eds.): ADHOCNETS 2020, LNICST 345, pp. 147–163, 2021.
https://doi.org/10.1007/978-3-030-67369-7_11

and other network functions. This scalability limitation is true for almost any MANET routing protocol proposed for flat networks. Many routing protocols for ad-hoc networks are either proactive (table-driven) or reactive (on-demand) [18,27]. Proactive routing protocols like OLSR or DSDV originate from the traditional distance vector and link state protocols. They continuously maintain routes to all destinations in a network, whereas reactive (on-demand) protocols like AODV or DSR will only seek out routes to a destination when necessary. Both routing protocol approaches scale poorly [18,27]. This is true because of the inherent characteristics of these protocols [26]: on the one hand, the on-demand routing protocols are limited by their route discovery techniques because of the extensive use of flooding. Hop-by-hop flooding usually has a huge negative impact on network performance and often leads to large delays in route discovery [3,22]. On the other hand, proactive routing protocols have these routes readily available, but it comes at a cost of constant route discovery throughout the lifetime of the network. It is evident therefore that both protocols have scalability issues, which get even worse in the case that nodes are mobile and links become generally unpredictable [3,22].

A hierarchical routing architecture, when carefully planned, shows its advantage of simplifying routing tables considerably and lowering the amount of routing information exchanged [8,23], thus increasing search efficiency and increasing scalability. This is best exemplified by the global Internet, which employs a hierarchical architecture and routing structure. The Internet is divided into routing domains. A routing domain typically contains a collection of co-located networks connected by routers (who are nodes) and linked in a common routing domain called the backbone [23].

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

- Nodes do not necessarily belong to a single network throughout their lifetime. As nodes are mobile, they may change their cluster membership, clusters may merge or split. So, a more general hierarchical routing architecture that supports various mobility scenarios is desirous. This rules out the hierarchical solutions proposed for the Internet, where nodes belong to IP subnets, and routing information is aggregated to handle routes to specific subnets only.
- In order for hosts within a cluster to route packets destined for hosts in external clusters or domains, there is the need for a protocol or scheme which will be the standard for such applications. OLSR supports a HNA message scheme which is primarily for external access, standardized in RFC 3626 [12]. This scheme is relevant for the designs we explore, as we will use OLSR as intra-domain routing protocol within each cluster.

A simple scheme to provide routing in such a hierarchical network would be for gateways nodes to simply flood all data packets destined for nodes outside a cluster through the backbone. Once a gateway receives a packet, it consults its routing table to determine whether the destination node is in its cluster.

If this is the case, it forwards the packet to the destination, based on the intra-cluster routing protocol (OLSR in our work). While such a scheme is easy to implement and can handle various mobile scenarios, it can easily overburden the backbone, so more efficient routing schemes would be beneficial. We present three additional possible solutions, and compare them using extensive simulations in OMNeT++ [31].

A hierarchical MANET architecture finds application in, for example, a military environment. In such a scenario, there exist platoons which move in groups and each platoon typically consists of soldiers and, probably, a dedicated vehicle (armoured tank/truck). These platoons are MANETs which are then considered as clusters in our study. In such a setting, the number of or size of a cluster is typically not known a-priori, but cluster membership is defined a-priori by the application. Each platoon has a dedicated gateway (this can be the armored vehicle), and, through the gateways, the various platoons are interconnected in a backbone network. The gateways are more powerful devices equipped with capabilities which will enable them effectively support communication between members of their local cluster and the different clusters.

The remainder of this paper is organized as follows. The next section reviews related work. Section 3 introduces our 4 routing solutions. Section 4 discusses the comparative quantitative simulation results we obtained when implementing our schemes in OMNeT++, focusing on the protocol performance when nodes jump from cluster to cluster. Finally, the last section summarizes our work and suggests avenues for future research.

2 Related Work

As discussed earlier, treating the whole MANET as a single, flat routing domain runs into scalability issues, see also [8,23]. An alternative to flat MANETs is clustering or hierarchical routing. The motivation for exploring hierarchical routing is that it increases scalability, routing efficiency and potentially reduces routing table entries considerably. Typically, rather than assuming that node movement is independent, hierarchical ad-hoc routing protocols group nodes into clusters of nodes that follow the same movement pattern. These protocols are based on the idea that members of a group tend to move together and therefore a node will most likely remain within the same cluster. This allows a node to move freely within its cluster and only inform other cluster members, abstracting node movement within a cluster. Members of other clusters only need to know how to communicate with one of its members. These groups may have some sort of cluster leader, popularly known as gateways or cluster heads. Depending on the algorithm and the clustering technique, there might be gateways providing connectivity with other clusters, and cluster heads who coordinate routing within their clusters and with other clusters. Alternatively a single node acts as both cluster head and gateway, providing connectivity with other clusters through a core/backbone network. Clusters can then be organized into a hierarchy.

In [9,16] the authors present a review of current hierarchical routing protocols and clustering approaches. The authors first introduce fundamental concepts about clustering. Then they classify the proposed clustering schemes into six categories based on their main objectives, which are load balancing clustering, Dominating-Set-based (DS-based) clustering, low maintenance clustering, mobility-aware clustering, combined metrics-based clustering, and energy efficient clustering. They also grouped the clustering cost terms into five categories: the required explicit control message exchange, the ripple effect of re-clustering, the stationary assumption, constant computation round, and communication complexity.

One of the earliest clustering protocols is LCA [7], developed for packet radio networks. The LCA protocol organizes the nodes into clusters according to the proximity of the nodes. Each cluster has a cluster head and all nodes in a cluster are in the direct transmission range of the cluster head. The choice of the cluster head is based on node identifiers, where the node with the largest identifier in a given area becomes the cluster header. The gateways in the overlapping region between clusters are used to connect clusters. LCA specifies that there should only be one designated gateway to interconnect clusters at a given time. A pair of nodes within transmission range of each other can also be used to connect clusters if there are no nodes in the overlap region.

In GPSR [11], packets are routed alternately between the cluster leaders and the gateways. The authors define several extensions that can be added to CGSR, such as priority token scheduling and gateway code programming, to control access to the channel. In addition, they define a LCC (Least Cluster Change) algorithm, designed to reduce the number of changes in the cluster leader, since such changes can generate significant overhead.

The works of [6,15] take a different approach to clustering and present two clustering algorithms. The first of these is DCA, intended for "quasi-static" networks in which nodes are slow moving, if moving at all. The other algorithm is called DMAC, designed for higher mobility. Both algorithms assign different weights to nodes with the assumption that each node is aware of its respective weight. The weights are in turn used to determine the cluster leaders. In the DMAC protocol, if two cluster leaders come into contact, the one with the smaller weight must revoke its leader status.

Another approach is that taken by the CEDAR algorithm [28], which builds a set of nodes (i.e., a core) to perform route computation instead of creating a cluster topology. Using the local state information, a minimum dominating set of the network is approximated to form the core. CEDAR establishes QoS routes that satisfy bandwidth requirements using the directionality of the core path. Link state and bandwidth availability is exchanged to maintain important information for computing QoS routes.

Kleinrock was an early pioneer of hierarchical routing schemes for static networks. In [19], Kleinrock and Kamoun investigated a hierarchical routing scheme with the goal of reducing routing table size. The authors of [25] also adopt a similar approach. The authors determined that the length of the routing

table is a strict function of the clustering structure. Clustering generally has the unwanted side-effect of an increase in path length, and so the goal was to find an optimal clustering scheme that optimizes path length. It was determined that the number of entries in a node's forwarding table is minimized when the number of *level-i* clusters in each level *-(i + 1)* cluster is e, and the number of levels in the clustering hierarchy equals $\ln N$. In this case, the forwarding table contains $e \ln N$ entries.

The Landmark Routing technique [30] is a distinct approach to building a hierarchy as it is based on landmarks, as opposed to transmission ranges. A landmark is a router whose location is known by its neighboring routers up to some radius. All routers within that radius know how to reach the landmark. A hierarchy of such landmarks is built by increasing the radius of some of the routers. Nodes have hierarchical addresses based on the landmarks with which they are associated. A source node routes to a destination by sending the packet to the lowest level landmark with which both nodes are associated. As the packet approaches the destination, the granularity of routing knowledge about that destination improves, and so the packet can be accurately routed to the destination.

Advantages of hierarchical routing, alongside scalability, include the ability to reduce routing table sizes, to shield nodes within a cluster from mobility in other clusters and to use different routing protocols, with possibly different update frequencies, in different clusters. Disadvantages include the difficulty in maintaining the structure of clusters in the face of high mobility (which has a particularly adverse effect if cluster heads change groups), the possible bottleneck presented by gateway nodes (these nodes also suffer greater resource usage) and the use of suboptimal paths. Examples of hierarchical routing protocols can be found in [16].

In our research, scalability is a strong requirement and so, we try to limit routing knowledge and avoid flooding in the backbone. Thus, we apply a hierarchical approach. In the context we are working with, we do not use any special algorithm for cluster formation and maintenance. Rather we assume that clusters are externally given, as is the case when they flow from the application/use of the MANET. For example, we may have different platoons of soldiers (in a military context) or different first responder crews joining in as a group etc. We are focusing in the basic choices of routing through the backbone in this work. The following section outlines 4 quite distinct approaches to routing in the backbone that we are evaluating.

3 Backbone Routing Approaches

Our core network architecture assumes that the MANET is organized into a two-tiered hierarchy. Ordinary nodes are equipped with a single wireless interface and grouped into clusters, given by the nature of the application (or some other means). Each cluster has one (or potentially multiple) gateway nodes that are equipped with two wireless interfaces: one to communicate with other nodes in the same cluster, a second one (with potentially different characteristics such as

transmission range or bandwidth) for communication with other gateways. The gateways collectively form the MANET backbone. Nodes in one cluster should be enabled to communicate with nodes in the same cluster, but also with nodes in other clusters.

Routing within a single cluster is done using OLSR, as this allows us to easily explore different alternatives for the backbone interconnecting multiple clusters: each cluster gateway advertises reachability to nodes outside a gateway through OLSR's HNA message by advertising a default route. Regular nodes in the cluster therefore have host-specific routes to all other nodes in the same cluster, and a default route to the gateway for any other destination. This information gets updated locally as nodes join and leave a cluster. It also gets updated periodically to reflect changes in the network topology within a cluster, as nodes are free to move. Due to OLSR's pro-active nature, gateways know which nodes are within their cluster simply by observing their local routing table. To enable routing through the backbone (to enable inter-cluster communication), we designed and implemented a number of different approaches. This section briefly discusses these approaches and provides some quantitative comparison, the next section presents some quantitative results obtained via simulation in OMNeT++.

The 4 solutions differ in the routing approach taken by the gateway nodes to deliver packets to destination nodes in remote clusters. In the discussion below, we will use the term ingress gateway to denote a gateway that receives a packet from a local node in its cluster, destined to a node in a different cluster. Similarly, the egress gateway is the gateway managing the cluster to which the destination node belongs. A key question and differentiating factor for all routing approaches is how much routing information is distributed in the network, and how much effort is required to maintain this information in the presence of node mobility and clusters merging and splitting.

The simplest approach is to broadcast data packets trough the backbone and a simple implementation of that idea would be to flood the data packets through the backbone (i.e., each gateway, when receiving a data packet it has not seen before, rebroadcasts it). No backbone routing protocol is required in this case, and gateways only require knowledge about which nodes are in its local cluster. Once the ingress gateway receives a data packet that is not destined for a node in its cluster, it simply broadcasts that packet over the backbone interface. In case a data packet is received over the backbone interface that is destined to a local node, the egress gateway uses the information in its local routing table to forward the data packet accordingly. While this approach minimizes the amount of routing knowledge in each gateway, it may scale poorly as the backbone grows, as each data packet is transmitted multiple times (in the case of flooding, which we implemented, each data packet is (re-)broadcast by all nodes in the backbone except for the egress gateway).

Another relatively straightforward approach is to exploit the HNA capability in OLSR. In this solution, the backbone runs a second instance of the OLSR routing protocol, this time over the backbone interface. Each gateway advertises reachability to nodes in its cluster via host-specific HNA entries. When an ingress

gateway receives a data packet destined to a node outside the local cluster, it will have enough information in its routing table to forward this data packet to the correct egress gateway. As nodes join and leave clusters, gateways learn about the changes in their cluster membership and will propagate this information via the periodic exchange of HNA messages. Compared to the broadcast solution outlined earlier, data packets can be forwarded over a shortest-hop route through the backbone. On the downside, each gateway learns routing information for all nodes in the network: the local OSLR instance propagates routing information to all nodes in the local cluster, the backbone OLSR instance propagates the reachability of all remote nodes via the appropriate gateway. This information is maintained in a pro-active manner, whether needed or not.

A third solution uses a modified version of AODV as backbone routing protocol. Once the ingress gateway receives a data packet from a local node, it will trigger a RREQ through the backbone. Other gateways, upon receiving the RREQ, will check whether the destination node is a member of their local cluster and respond with a proxy RREP. As this information is propagated throughout the backbone, a forward path is set up in intermediate backbone nodes and the data packet can be forwarded over the shortest-hop path, similar to the OLSR-based solution discussed previously. To support node mobility, gateways will have to issue a RERR message when receiving a data packet to a node that has recently left their local cluster, causing the ingress gateway to re-issue a RREQ. To enable a gateway to distinguish a data packet that is meant to be delivered locally from a data packet it simply has to forward through the backbone, we use tunneling: the ingress gateway will tunnel a data packet (using IP in IP encapsulation) to the egress gateway. Unlike the OLSR-based solution, gateways learn (and maintain) much less routing information. Besides the routing information to nodes in the local cluster, a gateway only manages information about destination nodes its local nodes are currently communicating with. This information is than maintained only for the duration of the data flow.

Our final solution is similar to the AODV-based solution, in that ingress gateways learn about the egress gateways on-demand. However, where AODV uses flooding to determine this information (and to re-establish it when the routing path fails), we propose to use a P2P approach, using a Distributed Hash Table (DHT). All gateways join a P2P overlay, storing information about nodes reachable through them under the hash of a node's IP address. Each such entry, if it exists, will list one or multiple gateways through which a destination node is reachable. Once an ingress gateway receives a data packet, it will query the P2P overlay. With the information returned from the overlay, it then tunnels the data packet to the appropriate egress gateway (we are running OLSR as the backbone routing protocol, so an ingress gateway can select the egress gateway that is the closest, for example). The efficiency of this scheme will depend on how easy the P2P overlay resolves lookup requests. As nodes leave and join clusters, the gateways, detecting changes in their local routing tables, update the routing information in the P2P overlay. To support mobility, gateways periodically query the overlay to see whether the routing information has been changed. This solution will store a complete set of routing information, somewhat similar

to the OLSR-based approach. But it will only save a single copy of the routing information (instead of replicating it across each gateway), distributing the information across all gateways that joined the P2P overlay. The information is not maintained pro-actively, but updated as the topology changes. The costs of this approach are that the P2P overlay will have to be maintained, and that gateways will periodically re-confirm that the (cached) routing information they are currently using is still accurate/up-to-date.

4 Simulation Results and Discussions

We used OMNeT++ as our simulation platform [31]. The INET Framework, an open-source library for the OMNeT++ environment [17], provides implementations of MANET routing protocols such as AODV or OLSR, as well as an IEEE 802.11 MAC layer. OverSim [24] is a flexible framework for overlay network simulation and includes implementations of some structured P2P protocols (i.e. Chord [29], . . . *etc.*). We added an implementation of basic flooding through the backbone and implemented the modifications discussed above. While these were quite straightforward in the case of AODV and OLSR, using the DHT as a backbone routing protocol is more evolved, and the details are discussed in [13]. We selected Chord as the overlay DHT protocol in the results reported here.

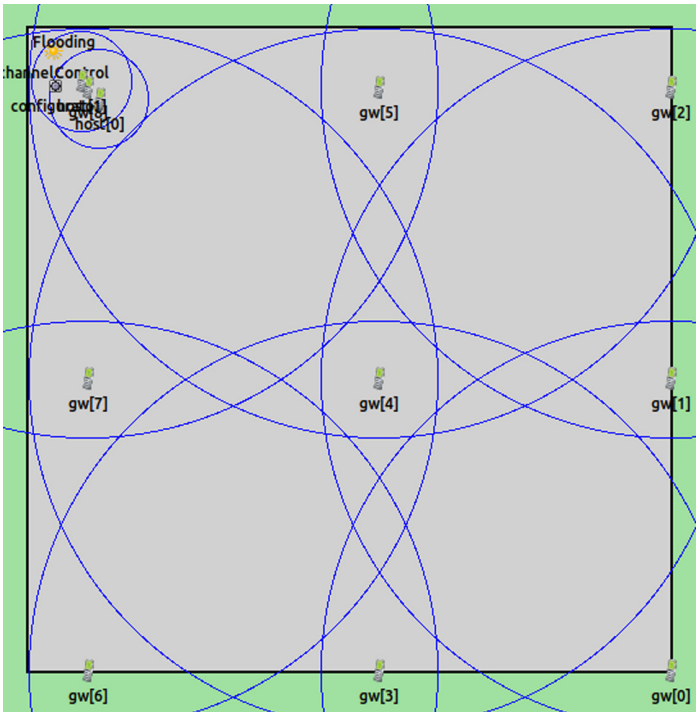

Fig. 1. Basic 3 × 3 grid layout

We focus in our evaluations on how well the proposed backbone routing solutions support nodes switching clusters. We set up a network with 9 clusters in a 3 × 3 grid layout as shown in Fig. 1, which shows a screenshot from the OMNeT++ GUI. We set the radio transmissions range for the backbone radio such that a gateway can only communicate with its direct neighbors in the grid. The local radio's transmission range is set such that nodes can only communicate with one gateway, forming a local cluster. Both local and backbone radios use IEEE 802.11 for the PHY and MAC layers, with a channel rate of 11 Mbps. Two nodes are initially in the same cluster (the top left one), sending 1 kByte UDP packets to each other once every second after an initial delay of 80 s to allow the various routing protocol instances to converge, the Chord ring to form, etc. We then explore two different mobility scenarios. In Scenario A, one node (called the stationary node) stays in the original cluster while the other node (called the mobile node) jumps to a new cluster every 100 s, starting at time 150 s. This continues until the node has visited all other clusters, staying in the last cluster for only 50 s. This results in a total simulation time of 900 s. In a more aggressive Scenario B, the mobile node jumps to a new cluster for 50 s before returning to its home cluster for 50 s. This is repeated until the node again visited all other 8 clusters. Other than the modifications alluded to above, we use all default values that come with the various routing and overlay protocol implementations in OMNet++, for example the Hello and TC message intervals in OLSR, the expanding ring search parameters in AODV, or the finger table maintenance intervals in Chord.

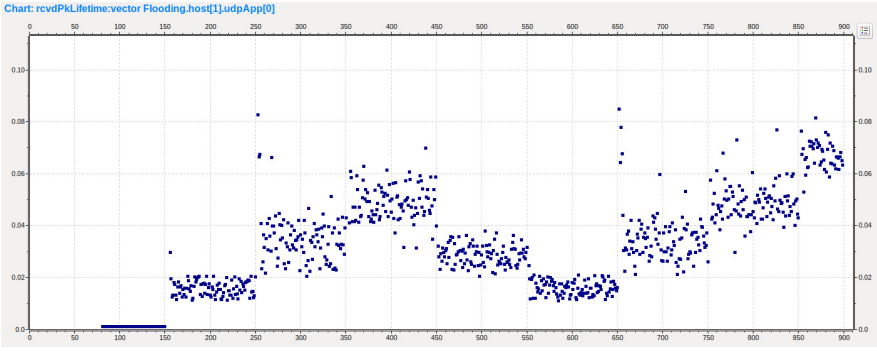

Fig. 2. Flooding: delivering packets to a mobile node

Figures 2 and 3 give an idea of the dynamic nature of packet delivery as time advances. The stationary and the mobile node start sending data packets to each other at time t = 80 s, the mobile node jumps to a new cluster at times 150 s, 250 s, etc. It visits the clusters in the following order, identified by the gateway index in Fig. 1: 5, 2, 1, 4, 7, 6, 3, 0. To reduce collisions in the backbone, packets that are re-broadcast are jittered uniformly random by up

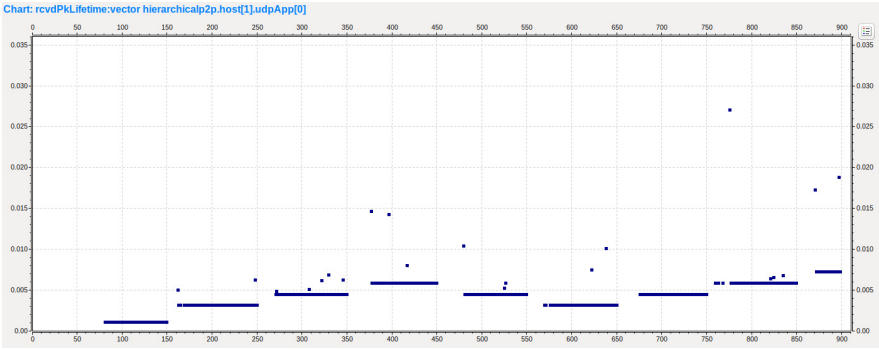

Fig. 3. OLSR: delivering packets from a mobile node

to 10 ms. Consequently, as shown in Fig. 2, the packet latency under flooding is somewhat noisy, but clearly shows a relation with the distance between the two nodes: the latency is lowest when the mobile node is in the same cluster as the stationary node (at the beginning of the simulation, followed by the two cases where the mobile node is in a cluster one hop away: Cluster 5 (from 150 s to 250 s) and Cluster 7 (550 s–650 s). Packet latency is the highest towards the simulation end, when the mobile node is in Cluster 0, 4 hops away from Cluster 8 (which contains the stationary node). It is also worth noticing that packets are almost constantly delivered. The stationary node learns about the default route early on. Once a gateway learns that the mobile node is in its local cluster, it will forward data packets from the backbone, no additional backbone-related routing information needs to be acquired.

When deploying OLSR as routing solution in the backbone, packet latency also shows a strong correlation with the number of hops through the backbone, as shown in Fig. 3, here for packets being sent from the stationary node to the mobile node. Different from the flooding case, packet latency is more uniform and lower, as no jittering is required and packets are delivered as unicast transmissions over a shortest path. Also, there are clear gaps in packet delivery each time the mobile node switches cluster: in addition to updating the intra-cluster routing information, backbone-specific routes need to be learned. Until the new cluster advertises reachability to the mobile node via HNA messages, backbone nodes will route data packets to a previous cluster, where they may be dropped or erroneously forwarded into the local cluster.

To more exhaustively study and compare the performance of our various solutions, Figs. 4 and 5 summarize PDR and average packet latency for 24 runs of Scenario A. For each routing solution, the graphs show the performance when sending packets from the mobile node to the stationary node (first bar), the stationary node to the mobile node (second bar), and the average of the two (third bar). Figure 4 also shows the 95% confidence interval for the average PDR. The results show that Flooding has the best overall PDR performance, in particular when sending data packets to the mobile node, for the reasons explained

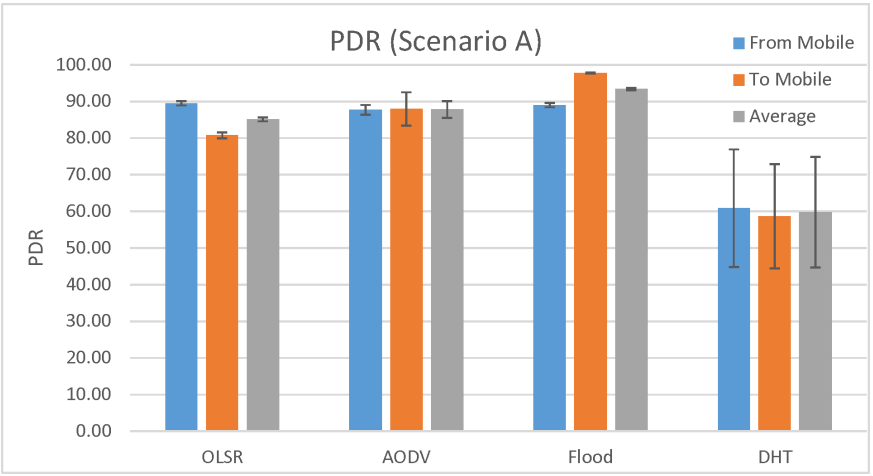

Fig. 4. Average PDR, Scenario A

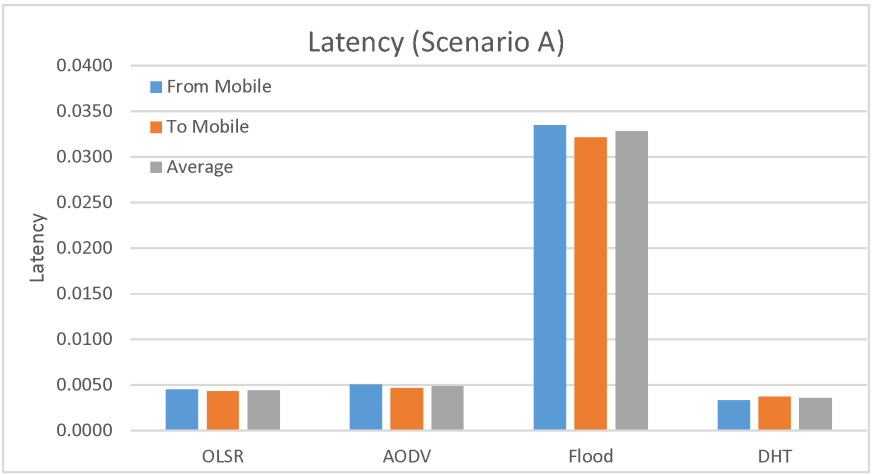

Fig. 5. Average Latency, Scenario A

above. AODV and OLSR have similar average performance, with OLSR somewhat biased towards better delivery of data packets send by the mobile node, while AODV performs similarly in both directions. Our DHT-based solution performs poorest overall, for reasons explained later. In the case of packet latency, the three routing solutions that send data packets via a shortest route perform better (and similar to each other) than Flooding.

In our second scenario, the mobile node only stays in a specific cluster for 50 s. It then returns to its home cluster, Cluster 8, for 50 s, before jumping to

another remote cluster. It follows the same order when visiting remote clusters as the mobile node in Scenario A. The next two figures summarize the results we collected in this case for 24 runs for each scenario.

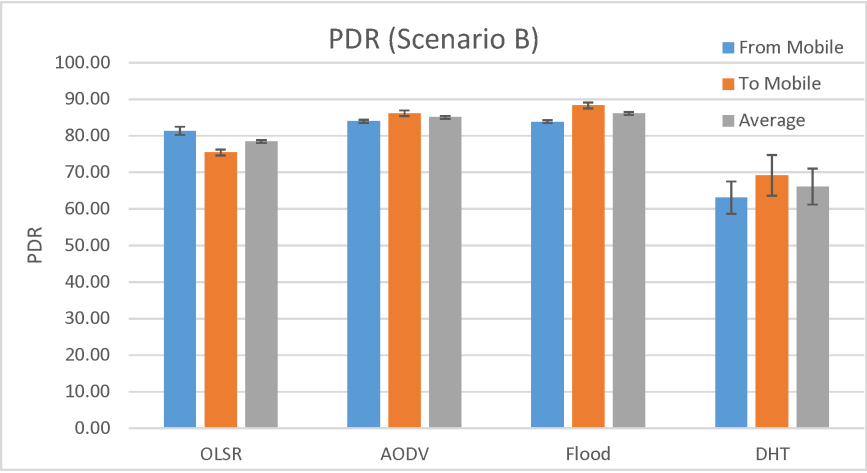

Fig. 6. Average PDR, Scenario B

Comparing the PDR performance of Scenario A (Fig. 4) with the results obtained in Scenario B (Fig. 6), we notice that overall the PDR suffered in the more aggressive Scenario B: as the mobile node changes cluster membership twice as often, more packets get lost due to inaccurate/outdated routing information. This is also true when the mobile node returns to its home cluster: until the stationary node learns that the mobile node has returned home, it will forward data packets to the cluster gateway, to be delivered over the backbone. This also explains why PDR drops in case of Flooding. AODV, on average, performs slightly better than OLSR, as AODV has an explicit route maintenance mechanism: if the incorrect backbone routing information causes data packets to be sent to the wrong cluster gateway, this gateway can at least inform the ingress gateway to stop forwarding data packets to it, triggering a new route request. OLSR, on the other hand, depends on the periodic update of such information, which will not prevent more data packets to be transmitted to the wrong egress gateway in the meantime. Of note here also is the fact that the DHT-based solution, while still performing worse than the other three routing approaches, actually performed better under this more aggressive scenario.

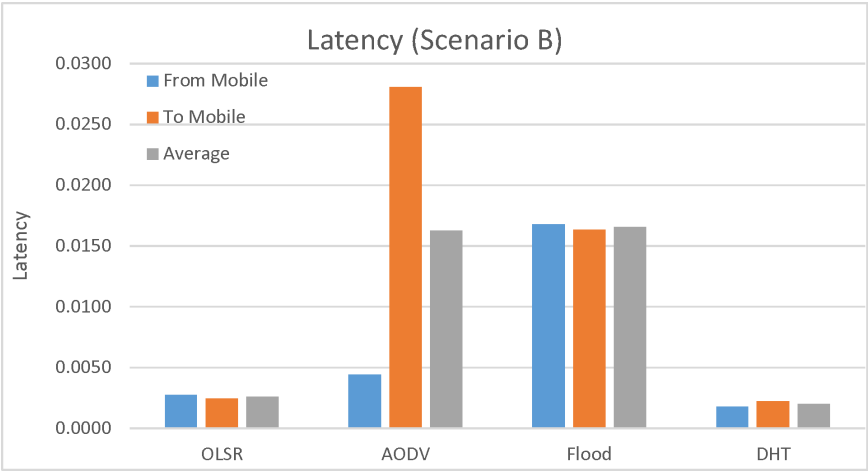

Fig. 7. Average Latency, Scenario B

In terms of latency (see Fig. 7), Flooding cut its end-to-end latency almost in half, as now more data packets (almost 50%) are delivered locally, i.e., when both nodes are in the same cluster. AODV sees its latency increase significantly, in particular for data packets destined to the mobile node. This is a consequence of the need to send more frequent RREQ messages, buffering data packets until a reply is received.

The last two figures focus on the traffic in the backbone network. Since the four routing solutions are very different, and we are interested in a fair comparison, we do not differentiate between control packets and data packets: in the case of Flooding, the protocol should be "penalized" for each unnecessarily retransmitted data packet, for example. And in the case of our DHT-based solution, we have actually two sets of control data packets: the OLSR routing protocol providing connectivity among the backbone nodes, plus the control packets imposed on the backbone by the formation and maintenance of the DHT-overlay, as well as the DHT queries. Figure 8 shows the total number of packet transmissions, averaged over 24 runs, in Scenario A, Fig. 9 similarly shows the average total number of packet transmissions for Scenario B. Flooding imposes the least amount of traffic on the backbone, in particular in the more aggressive Scenario B, where AODV has to issue many more RERR and RREQ messages. OLSR, due to its pro-active nature, causes more backbone traffic than either AODV or Flooding. The DHT-based solution stands out as generating the most backbone traffic. This helps to explain its poor performance overall, as the backbone starts getting congested. We collected the number of packet collisions and packet drops as well, not shown here, and saw that they were particularly high in the case of the DHT-based routing solution.

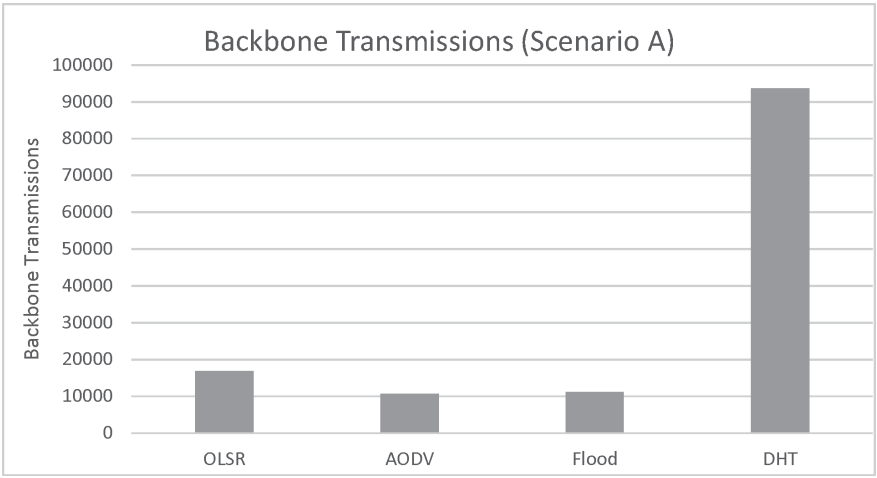

Fig. 8. Backbone packet transmissions, Scenario A

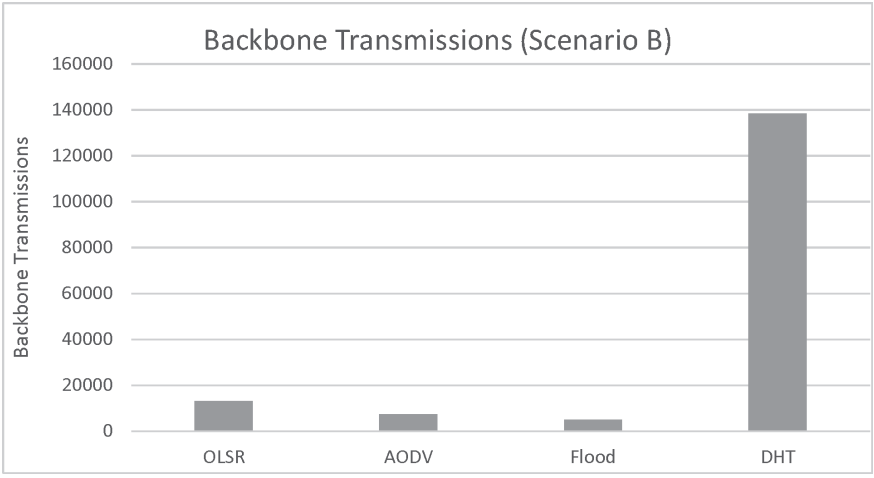

Fig. 9. Backbone packet transmissions, Scenario B

5 Conclusions and Future Work

In this paper we address the problem of routing data packets through a backbone in the context of a hierarchical MANET. While separating the routing domains is essential to ensure scalability, a proper solution needs to deal with various levels of mobility. This makes hierarchical routing protocols developed for the Internet not suitable. Rather, we proposed 4 quite distinct approaches for a suitable backbone routing approach, ranging from flooding over modifications to OLSR and AODV to an innovative proposal based on the use of a DHT. Qualitatively,

the approaches differ in how routing information is propagated among the nodes, in particular the gateway nodes providing connectivity between a cluster and the rest of the network. In flooding, no additional routing information is propagated. The OLSR-based solution has all gateways learn about all nodes in the network, increasing their routing table size considerably as the network scales up. The solutions based on AODV and the DHT learn routing information on-demand, with the AODV-based solution using flooding of RREQ messages while the DHT-based solution looks up the required information based on the P2P mechanism deployed (here, we used Chord, but many other structured P2P systems exist).

We implemented these solutions in OMNeT++ and run extensive simulations to evaluate the performance of these four approaches to support routing between a stationary node and a mobile node that periodically switches clusters. Overall, simply flooding the data through the backbone resulted in the highest PDR, but at the cost of increased packet latency. Flooding also generated the smallest amount of overall traffic in the backbone. Our proposed modification to OLSR worked reasonably well in both scenarios too (high PDR and low latency), whereas the proposed modifications to AODV suffered higher latency under the more aggressive scenario. However, AODV did impose less overall traffic in the backbone, compared to OLSR. The DHT-based solution performed the worst, with relatively low PDR and a high traffic load in the backbone. This is a consequence of our choice of Chord (whose implementation was available) to implement the DHT service. Chord (similar to many other overlay solutions) is agnostic of the underlay network, and a number of cross-layer solutions have been proposed to improve the DHT performance (and to reduce its overhead) particularly in the context of a MANET, such as [1,5], a study to evaluate how well such a cross-layered approach would work is one item of future work.

The results indicate that the flooding-based solution performs best overall: it is simple to implement, reduces the amount of routing information that needs to be propagated in the backbone, and achieves arguably the best performance in terms of PDR and traffic in the backbone. A concern could be the scalability of the protocol as traffic across clusters increases. We plan to explore in more depth at what point such a simple approach would deteriorate. However, we also want to point out that better broadcasting solutions (as opposed to flooding) exist. The broadcast problem has been extensively studied for multi-hop networks. The minimal number of nodes that need to transmit a data packet to ensure that all nodes receive a copy is known as the Minimum Dominating Set. To ensure that packet transmissions can propagate through the network, the nodes in this set should be connected, resulting in the Minimum Connected Dominating Set (MCDS) problem. Optimal solutions to compute a Minimum Connected Dominating Set [21] were obtained for the case when each node knows the topology of the entire network, but these solutions are NP-hard. When implementing an efficient broadcast protocol, many (though not all) solutions employ partial or local neighborhood knowledge, typically acquired through the periodic exchange of HELLO messages (not unlike the OLSR protocol). A good early classification and comparison of a number of proposed protocols is presented in [32].

An efficient broadcast protocol standardized by the IETF is Simplified Multicast Forwarding (SMF) [20], which is based on partial (local) topology information as well. In SMF, for our backbone topology, a subset of three nodes would be selected to re-broadcast data packets while ensuring that all gateways receive it: either gateways 3, 4, and 5, or gateways 1, 4, and 7. This drastically reduces the traffic in the backbone, cutting down the number of packet transmissions from 8 to (at most) 3. In addition, with less contention for access to the media, data packets would need to be jittered less, reducing the latency.

References

1. Abid, S.A., Othman, M., Shah, N.: 3D P2P overlay over MANETs. Comput. Netw. **64**, 89–111 (2014)
2. Abid, S.A., Othman, M., Shah, N.: A survey on DHT-based routing for large-scale mobile ad hoc networks. ACM Comput. Surv. (CSUR) **47**(2), 20 (2015)
3. Ahmad, I., Ashraf, U., Ghafoor, A.: A comparative QoS survey of mobile ad hoc network routing protocols. J. Chin. Inst. Eng. **39**(5), 585–592 (2016)
4. Al Mojamed, M., Kolberg, M.: Structured peer-to-peer overlay deployment on MANET: a survey. Comput. Netw. **96**, 29–47 (2016)
5. Al Mojamed, M., Kolberg, M.: Design and evaluation of a peer-to-peer MANET crosslayer approach: OneHopOverlay4MANET. Peer-to-Peer Netw. Appl. **10**(1), 138–155 (2017). https://doi.org/10.1007/s12083-015-0413-4
6. Basagni, S.: Distributed clustering for ad hoc networks. In: Fourth International Symposium on Parallel Architectures, Algorithms, and Networks, 1999 (I-SPAN 1999) Proceedings, pp. 310–315. IEEE (1999)
7. Bein, D., Datta, A.K., Jagganagari, C.R., Villain, V.: A self-stabilizing link-cluster algorithm in mobile ad hoc networks. In: 8th International Symposium on Parallel Architectures, Algorithms and Networks, 2005. ISPAN 2005. Proceedings, p. 6 IEEE (2005)
8. Belding-Royer, E.M.: Hierarchical routing in ad hoc mobile networks. Wirel. Commun. Mob. Comput. **2**(5), 515–532 (2002)
9. Bentaleb, A., Boubetra, A., Harous, S.: Survey of clustering schemes in mobile ad hoc networks. Commun. Netw. **5**(02), 8 (2013)
10. Caleffi, M., Paura, L.: P2P over MANET: Indirect tree-based routing. In: IEEE International Conference on Pervasive Computing and Communications, 2009. PerCom 2009, pp. 1–5. IEEE (2009)
11. Chiang, C.C., Wu, H.K., Liu, W., Gerla, M.: Routing in clustered multihop, mobile wireless networks with fading channel. In: proceedings of IEEE SICON, vol. 97, pp. 197–211 (1997)
12. Clausen, T., Jacquet, P.: RFC 3626. Optimized link state routing protocol (OLSR) (2003)
13. Echegini, N.: A DHT-based routing solution for hierarchical MANETs. Master's thesis, Carleton University (2018)
14. Furness, J.R.: Optimising structured P2P networks for complex queries. Ph.D. thesis, University of Stirling (2014)
15. Hussein, A., Yousef, S., Al-Khayatt, S., Arabeyyat, O.S.: An efficient weighted distributed clustering algorithm for mobile ad hoc networks. In: 2010 International Conference on Computer Engineering and Systems (ICCES), pp. 221–228. IEEE (2010)

16. Ibrihich, W., Salah-ddine, K., Laassiri, J., El Hajji, S.: Recent advances of hierarchical routing protocols for ad-hoc and wireless sensor networks: a literature survey. Int. J. Inform. Technol. Ijit **9**(2), 71–79 (2016)
17. INET framework user's guide, January 2019. https://inet.omnetpp.org/. Accessed 1 Oct 2019
18. Kaur, H., Sahni, V., Bala, M.: A survey of reactive, proactive and hybrid routing protocols in MANET: a review. Network **4**(3), 498–500 (2013)
19. Kleinrock, L., Kamoun, F.: Hierarchical routing for large networks performance evaluation and optimization. Comput. Netw. **1**(3), 155–174 (1977)
20. Macker, J.P.: Simplified Multicast Forwarding. RFC 6621, May 2012. https://doi.org/10.17487/RFC6621, https://rfc-editor.org/rfc/rfc6621.txt
21. Misra, R., Mandal, C.: Minimum connected dominating set using a collaborative cover heuristic for ad hoc sensor networks. IEEE Trans. Parallel Distrib. Syst. **21**(3), 292–302 (2010)
22. Moussaoui, A., Boukeream, A.: A survey of routing protocols based on link-stability in mobile ad hoc networks. J. Netw. Comput. Appl. **47**, 1–10 (2015)
23. O'Driscoll, A., Rea, S., Pesch, D.: Hierarchical clustering as an approach for supporting P2P SIP sessions in ubiquitous environments. In: 2007 9th IFIP International Conference on Mobile Wireless Communications Networks, pp. 76–80. IEEE (2007)
24. OverSim The Overlay Simulation Framework, January 2019. https://inet.omnetpp.org/. Accessed 1 Oct 2019
25. Özdamar, L., Demir, O.: A hierarchical clustering and routing procedure for large scale disaster relief logistics planning. Transp. Res. Part E Logistics Transp. Rev. **48**(3), 591–602 (2012)
26. Quispe, L.E., Galan, L.M.: Behavior of ad hoc routing protocols, analyzed for emergency and rescue scenarios, on a real urban area. Expert Syst. Appl. **41**(5), 2565–2573 (2014)
27. Sharma, C., Kaur, J.: Literature survey of AODV and DSR reactive routing protocols. In: ICAET, IJCA, pp. 14–17 (2015)
28. Sivakumar, R., Sinha, P., Bharghavan, V.: Core extraction distributed ad hoc routing (cedar). In: Proceedings of INFOCOM 1999 (1999)
29. Stoica, I., et al.: Chord: a scalable peer-to-peer lookup protocol for Internet applications. IEEE/ACM Trans. Netw. (TON) **11**(1), 17–32 (2003)
30. Tsuchiya, P.F.: The landmark hierarchy: a new hierarchy for routing in very large networks. ACM SIGCOMM Comput. Commun. Rev. **18**, 35–42 (1988)
31. Varga, A.: OMNeT++ simulation manual, January 2019. https://doc.omnetpp.org/. Accessed 1 Oct 2019
32. Williams, B., Camp, T.: Comparison of broadcasting techniques for mobile ad hoc networks. In: Proceedings of the 3rd ACM International Symposium on Mobile Ad hoc Networking & Computing, pp. 194–205. ACM (2002)