Bachelor thesis
Computer Science

Radboud University

Improving Mobile Ad Hoc Networks in Realtime Situations

Author:
Jeroen van Alem
S4485211

First supervisor/assessor:
Dr. Ir. J.E.J. (Joeri) de Ruiter
`joeri@cs.ru.nl`

Second assessor:
Dr. G. (Greg) Alpár
`g.alpar@cs.ru.nl`

April 11, 2018

Abstract

In this thesis we propose a mobile ad hoc network protocol, based on SCALAR, called MARS (Message-priorized Adaptive-dynamic Resource-based Scalar), that is suitable for different realtime situations. SCALAR is shown to be a good solution for mobile ad hoc networks, due to its backbone approach and its data replication scheme. However we improved upon it by implementing an adaptive dynamic resource based backbone construction instead of a periodic backbone construction that is only based on the ID of the nodes. We also implemented message prioritization to provide for timely delivery of important messages, added timers to pending requests to be able to resend messages after a fixed period of time to increase the probability of succesful delivery and gave messages a TTL in the form of a maximum hop count to decrease the number of redundant messages in the network. We have tested our protocol by simulating its performance and that of SCALAR in three different scenarios. The results of the tests show that MARS is the better solution for those realtime situations than SCALAR.

Contents

Chapter 1

Introduction

A mobile ad hoc network (MANET) is a collection of autonomous nodes that has no structure predefined by wires and where the topology of the network can change unpredictably due to the mobility of the nodes. MANETs can be found all around us and in an increasing number. Think of all the drones, airplanes or self-driving cars that have to communicate with each other to prevent accidents. Moreover, MANETs are also used in military and emergency response situations. Military groups, tanks and emergency reponse vehicles, all communicate over a network that has no predefined structure, since every situation requires a different setup. Therefore it is essential that we keep improving these networks to get the best solution for realtime situations.

Protocols for mobile ad hoc networks are often rated based on the Quality of Service, sustainability and scalability of the resulting network. The Quality of Service (QoS) of a network can be measured by the ratio with which messages are successfully (and timely) delivered. The sustainability of a network can be measured by the message overhead of the protocol. The more messages the nodes need to send in a network to successfully deliver messages (efficiency), the sooner nodes run out of battery (durability). For a network to stay sustainably connected it is therefore crucial to keep the message overhead as low as possible. The scalability of a network can be measured by increasing the amount of nodes in the network and see if the change has a great effect on the performance of the resulting network. The smaller the effect is, the more scalable the network is.

In this thesis we propose a new MANET protocol called MARS (Message-priorized Adaptive-dynamic Resource-based Scalar), that is based on an existing protocol called SCALAR [1]. We have chosen SCALAR, because it is shown to be a better protocol than related protocols for mobile ad hoc networks. However we have made some improvements upon it to better suit the situations that we see in real life. The improvements we have made upon SCALAR are to increase the scalability, sustainability and Quality of Service of the network.

The second chapter of this thesis provides the necessary background information on the technologies we use in our protocol. In the third chapter we introduce our protocol and explain the changes we have made upon SCALAR. In the fourth chapter we test our protocol on three scenarios that can be found in real life and show that our protocol is suitable for those realtime situations. In the fifth chapter we compare our protocol to related work and in the sixth chapter we give our conclusions regarding our protocol.

Chapter 2

Preliminaries

In this chapter we discuss some of the basic protocols and technologies that we used to develop MARS. We start of with the MAC layer of IEEE 802.11 where both MARS and SCALAR are built upon. We then show the message prioritizing mechanisms in the MAC layer that we use in MARS. After that, we explain how SCALAR works, on which we based our protocol. Finally we explain how adaptive dynamic backbones are constructed in a mobile ad hoc network, which we implemented in MARS.

2.1 IEEE 802.11

IEEE 802.11 (also called WIFI) is a set of standards for implementing a wireless local area network [2]. It comes with two operating modes: an infrastructure mode and an ad hoc mode. In the infastructure mode, the wireless network has a certain structure and consists of at least one wired access point connected to the rest of the network. In the ad hoc mode, there is no network structure and every node communicates directly to the surrounding nodes without a wired access point or any kind of connection to a wired network.

The MAC layer in IEEE 802.11 is built so that multiple nodes can communicate over a shared medium. In principle, when a node wants to transmit data, it checks if other nodes are transmitting data before starting to transmit after a random backoff time followed by a distributed interframe space (DIFS). This is to minimize the chance of multiple nodes sending data simultaneously. When a collision occurs due to two nodes transmitting at the same time, the nodes will re-enter a contention state. The chance of a collision-free transmission is increased by increasing the size of the contention window repeatedly when a collision occurs. The receiver will send an acknowledgement (ACK) to the sender when the data is succesfully received. With this acknowledgement, indirect collisions are detected, since nodes will assume a collission has occured when a transmission is not acknowledged. Every message sent starts with an indication of the duration of the transmission of the message. A listening node then defers its contention window, until the message plus the acknowlegdement are sent and a DIFS has passed, before it tries to transmit a message of its own. The result of this is that a channel can be reserved by a node that has succesfully competed for transmission.

In ad hoc networks there is also the hidden node problem. This is a collision of two transmissions from two nodes that are unable to reach each other. This problem is solved in the MAC layer of the network by using the Request to Send/Clear to Send (RTS/CTS) procotol. Where the receiving node can reserve a channel for the sending node by broadcasting an Clear to Send packet. Time-bounded data applications, like video and voice, are also supported by the MAC layer of 802.11, through a point coordination function (PCF). When PCF is enabled, nearby nodes are systematically checked if they want to transmit data. The nearby nodes can only transmit data at the moment they are asked for it. This opens the oppurtunity to keep transmitting time-bounded data in time.

2.2 Message prioritizing mechanisms

The performance of a network is often given in terms of the Quality of Service and timely delivery of packets is a part of that. Since different types of packets are more or less tolerant of delays, the packets can be divided into different classes. Because not all classes of traffic can be transmitted simultaneously over the network due to the limited bandwidth, prioritization of these classes is required to increase the Quality of Service. There are three mechanisms to prioritize these classes in the MAC layer of IEEE 802.11 [7]:

Static priority scheduling assigns to each data traffic class its own first-in first-out (FIFO) queue and each queue is served in order of delay priority. The mechanism puts the data items from buffer into their respective priority queue, where the items will wait until they are transmitted by the node.

Prioritized waiting time is the mechanism that gives a node, with the highest priority packet in queue, the permission to transmit on a specific channel. It achieves this by defining different DIFS times for the different priority packets. Packets with the highest priority have the shortest waiting time, so the node, that wants to transmit the packet with the highest priority in the area, will have the highest probability for successfully reserving the channel. Respectively, packets with the lowest priority give a node the lowest probability for successfully reserving the channel.

Prioritized backoff time takes the priority of a packet into account when calculating the random backoff time. It makes use of different distributions per priority class, which increases the probability for higher priority packets to win contention during the backoff time. The standard distribution is randomly chosen to decrease the probability of nodes transmitting simultaneously. The prioritized backoff time mechanism also uses random distributions, but the maximum size of a contention window is smaller for higher priority packets.

2.3 SCALAR protocol

SCALAR [1] is a scalable data lookup and replication protocol for mobile ad hoc networks. The protocol consists of the following parts:

1. Virtual backbone algorithm
2. Scalable data lookup protocol
3. Reactive replication scheme

These parts increase the data accessibility and decrease the message overhead with an increasing number of nodes. This is because the virtual backbone minimizes the number of nodes involved when looking up data in the network. The protocol is suitable for large-scale mobile ad hoc networks where each node has a unique host identifier, denoted as $id(u)$ for node u. All n nodes are denoted in the set $M = \{M_1, M_2, ..., M_n\}$. All data items in the network are denoted as $D = \{d_1, d_2, ..., d_n\}$, where d_i is the initial data item of node M_i. Every node can request and save a replica of any data item at any time from any node in the network. It is assumed that every node has an equal memory capacity.

2.3.1 Virtual backbone construction algorithm

The virtual backbone construction algorithm is based on the connected dominating set construction problem from graph theory. A dominating set of a graph $G = (V, E)$ is a subset of the vertices (nodes), so that every node in the network is either in the subset or adjacent to a node in the subset. A connected dominating set is a dominating set whose induced subgraph is connected. Since a mobile ad hoc network has no predefined structure, every node only has local information about the network topology. That is, every node only knows its reachable neighbours. The (open) neighbour list of a node u is denoted as $N(u)$ and the closed neigbor list, which also includes the node itself, is denoted as $N[u]$.

The algorithm starts with the neighbour exchange phase, where every node sends its $N(u)$ to all its neighbours. It then proceeds with the marking process, where every node initially marks itself *true* as a backbone node if it has at least two non-adjacent neighbours in its neighbour list. The nodes that are marked *true* as a backbone node, send a backbone-announce packet to all their neighbours. There will not be a backbone node in a fully connected network after the marking process, which is good, since every node is already directly reachable from every other node. Then the nodes go into pruning phase to form the final connected dominating set of nodes in the network. In the pruning phase every node v checks the following two rules for itself:

Rule 1: If v has a neighbour u, u is still marked *true* as a backbone node, $N[v] \subseteq N[u]$ and $id(v) < id(u)$, return $false$ else return *true*.

Rule 2: If v has neighbours u and w, u and w are still marked *true* as backbone nodes, $N(v) \subseteq N(u) \cup N(w)$ and $id(v) = min\{id(u), id(v), id(w)\}$, return $false$ else return *true*.

When a rule returns $false$, the node is not a backbone node and sends a backbone-cancel packet to all its neighbours. After the pruning phase the nodes that are still marked *true* as backbone nodes will form the backbone.

2.3.2 Scalable lookup protocol

The scalable lookup protocol takes advantage of the backbone in a way that every node sends its data requests to a backbone node and all data requests are only forwarded among the backbone nodes. Because every node is either a backbone node or connected to a backbone node (called an end node), the number of nodes involved in the lookup process of a data item is at most the total number of backbone nodes plus two end nodes. In turn, this decreases the number of messages sent, which is crucial in large mobile networks. The whole lookup process is divided into two parts: search and data receive. There are four ways a backbone node can participate in the search part:

(1) Sending a self-generated request to adjacent backbone nodes.

(2) Forwarding a received request to adjacent backbone nodes.

(3) Receiving a request generated by an adjacent end node.

(4) Receiving a request forwarded by an adjacent backbone node.

A backbone node sends or forwards a request to a specific adjacent node when the destination of the request is either a neighbour or a neighbour of a neighbour. This is also called two-hop vicinity, since due to the neighbour exchange phase a node knows the local topology of a network up to two hops away. If the destination of the request is not in two-hop vicinity of that backbone node, the backbone node sends or forwards the request to its adjacent backbone nodes. End nodes can only send self-generated requests, either to a randomly selected adjacent backbone node or to a specific neigbour when the destination of the request is in two-hop vicinity. In the data receive part, backbone nodes only participate in the following 3 cases:

(1) It owns the requested data item.

(2) It receives a data item for which it forwarded the request.

(3) It requested the data item.

End nodes can only own the requested the data item or have requested the data item in this part of the lookup process. When a request is received by the destination node, the destination node will send the related data item via the path the request took. For case 1, the destination node of the request sends the requested data item to the node the request came from. For case 2, the backbone nodes on the path the request took check from which node the forwarded request came and forward the data item to that specific node. The backbone nodes keep pending requests in cache to be able to look up the previous node of the request, when the data item is received. For case 3, the node that requested the data item receives that data item and stores it in memory. When the data item of a request is received (and forwarded), the pending request is deleted from cache in that node. When that data item is then received again, without a request pending in cache, the data item will not be stored or forwarded. This decreases the amount of redundant messages sent in the network and the amount of duplicate data items stored in memory.

2.3.3 Reactive replication scheme

The distributed data replication scheme handles the caching of data in a mobile ad hoc network to increase data availability and decrease the message overhead. It runs in a passive mode, so there are no specific control packets needed for this. This way it eliminates the control overhead, that would be present in an active protocol, and increases thereby the scalability of the network. The goal is to make nodes eager to replicate data items that are requested more frequently and that are further away from them. This increases the probility that the requested data item is closer to the requesting node and therefore decreasing the number of message sent and increasing the scalability of the network. All nodes make a replication decision based on a cost function when a data item is received. The cost function is as follows:

$$cost(h_{ij}, \alpha, \alpha_i) = \frac{\alpha_i}{\sum_{\beta \in \alpha} \beta} * h_{ij}$$

Where h_{ij} is the amount of hops between M_i (where the data item d_i is received from) and M_j (the receiver) and where α is the set of local request frequency histories of data items that passed through node M_j and α_i is the specific local frequency history of the data item d_i. The data replication decisions are as follows:

- If a backbone node or an end node receives a data item that it requested and there is enough empty space in cache to store the data item, it will replicate the data item. If there isn't enough empty space to store the data item, it will check if the cost of the received data item is at least as high as the lowest-cost data item in cache. If this is true, it will replace this item with the newly received data item.
- If a backbone node receives a data item wherefore it forwarded the request, it is most likely to be closer to the requesting node than destination node of the request was. It will therefore make the replication decision as descripted in the first case.

2.4 ADB protocol

The adaptive dynamic backbone (ADB) [3] protocol, that is derived from VDBP [4], is based on the following requirements:

Adaptivity When a network is highly mobile, the backbone nodes in that network should decrease their hop vicinity. Which means that nodes should communicate only with their direct neighbours and the nearest backbone nodes, to decrease the need to keep up with the changes in the local topology of the network. In a highly mobile network it is not possible to maintain the neighbour lists of the neighbours, since the neighbour lists change constantly and exchanging the neighbour lists more frequently would flood the network. Respectively when a network is more static, the backbone nodes should increase their hop vicinity. Which means that nodes should communicate with the neighbours of their direct neighbours through those direct neighbours, to decrease the message flow through the backbone.

Dynamicity Nodes should keep track of what happens to the topology of the reachable surrounding network and decide for themselves when its time to change their role in the backbone of the network. This way the network is not periodically occupied as a whole when constructing the backbone, which delays the transmission of possibly highly important messages.

Because of this ADB is shown to be able to maintain connectivity in a highly dynamic network. ADB allows nodes to be a backbone node even though they are not in the dominating set of the network. It works with varying-depth trees that are all rooted at a backbone node, so that the hop vicinity of a backbone node can adaptively be changed depending on the mobility of the network. Static networks will have a higher amount of hops in the local groups (trees), where relatively highly mobile networks will have a lower amount of hops in the local groups. The *NLFF* (Normalized Link Failure Frequency) value in each node is the measurement of the mobility of its local group. When a node becomes fully operational, it sets itself to be a backbone node and starts its processes. All nodes in the network have to maintain the following data:

ParentID is the ID of the node that is upwards in the tree to the backbone node and which is set by the core selection process. A node stores its own ID when it is a backbone node itself.

NTab is the table with all the information of the neighbours of that node (see figure A.1). This neighbour table is updated by the neighbour discovery process.

NLFF shows the mobility of the local group. All nodes start with a NLFF of 0 and have an α set somewhere between 0 and 1, which is a smoothing factor. Every period of *NLFF_TIME_WINDOW*, the neighbour discovery process counts the number of link failures and then the NLFF is calculated as follows:

$$NLFF_{curr} = \frac{NUMBER\_OF\_LINK\_FAILURES}{NLFF\_TIME\_WINDOW * NUMBER\_OF\_NEIGHBOURS}$$

$$NLFF = \alpha * NLFF_{curr} + (1 - \alpha) * NLFF$$

Degree stands for the number of entries in the neighbour table and is thereby also updated during the neigbor discovery process.

FWTab is a table that maintains a list of shortest path entries to the nearby backbone nodes and is updated during the core forwarding process (see figure A.2).

2.4.1 Neighbour discovery process

The neighbour discovery process in a node ensures that the surrounding nodes know that that node is (still) there. Every node periodically broadcasts an *hello* message to let the surrounding node know its status. This way, all nodes know the information of the surrounding nodes that is needed for the core selection process. The *hello* message consists of the following fields:

NodeID is the unique identifier of the node broadcasting the *hello* message.

CoreID is the ID of the backbone node the broadcasting node is currently associated with. This is its own ID when it is a backbone node itself and $NTab(ParentID).CoreID$ when it is not a backbone node.

Hops is the number of nodes between the broadcasting node and its associated backbone node. This number is zero if the node is a backbone node itself and is $NTab(ParentID).HopsToCore + 1$ if not.

Degree is the number of neighbours the node has at that time.

NLFF is the calculated NLFF of the path to the backbone. Backbone node send their own NLFF and all other nodes send $NTab(ParentID).NLFF + NLFF$.

When a node receives an *hello* message, it updates its NTab on the entry with the corresponding NodeID. The process also periodically uses the *LastUpdated* field in NTab of a node to remove entries that are not updated for a certain period of time. After removing those entries that node updates its own NLFF value.

2.4.2 Core selection process

The core selection process determines if a node is a backbone node or not. It starts in each node after the node has become operational and a certain waiting time. This waiting time allows the node to receive at least one *hello* message from all surrounding neighbours. This process runs in every node, so that the nodes can decide for themselves whether they should (still) be a backbone node or not. Every node will begin this process by calculating its own current *status* value. This *status* value is a metric with the following values of the node: *(NodeID,* $NLFF^{-1}$*, number of neighbours).*

The nodes then calculate the *status* value for every neighbour in their NTab. When a node has a higher *status* value then all entries in their NTab, the node is considered a local optimal node and will therefore be consider itself a backbone node. When a node does not have a higher *status* value then all entries in their NTab, it will set the NodeID of the entry with the highest *status* value as its ParentID. Every *hello* message sent after this will contain the updated values.

The core selection process keeps the number of backbone nodes as low as possible, but has a constraint that limits this. The constraint is that the number of hops and the NLFF from every node to its associated backbone node may not exceed the HOP_THRESHOLD and NLFF_THRESHOLD, respectively. Once the local optimality check has been done, the constraint will be ensured in each node n by performing the following checks on all incoming *hello* messages:

- If the message came from the parent p of node n, node n checks if it violates the constraint by keeping node p as its parent. If it does violate the constraint, node n will try to find another parent in its NTab. If no parent exists in its NTab with which the constraint is satisfied, the node will set itself as its parent and thus becoming a backbone node.

- If node n is not a backbone node and the message came from node k that is not the parent of node n, then node n will set node k as its parent if node k has a smaller number of hops to the backbone than its current parent.
- If node n is a backbone node, the message came from node l and $NTab(l).CoreID$ is not its own ID, then node n will check if it violates the constraint when it sets node l as its parent. When it does violate the constraint it will remain a backbone node, otherwise it will set node l as its parent.
- If node n does not receive a message from its parent in time, it will try to find another parent in its NTab or be its own parent as with the first check.

2.4.3 Core forwarding process

The core forwarding process begins with every node updating its FWTab when it receives an *hello* message from a node associated with another backbone node. Then every non-backbone node constructs and sends a *core forwarding update* message to its parent every CORE_FORWARD_UPDATE period of time. This *core forwarding update* message is a list of all entries in the FWTab of the sending node. When the parent node receives this message, it updates its FWTab when it does not contain a certain backbone node in the message or when a matching backbone node has a smaller number of hops in the message. In the end, every backbone node will know the shortest path to all nearby backbone nodes. The process also takes care of entries that are not updated for a certain period of time by removing them from the FWTab. This way a backbone node becomes more proactive and less reactive when the coverage will get larger in low mobility networks and a backbone node becomes less proactive and more reactive when the coverage will get smaller in high mobility networks. It all depends on how much child nodes it has and how much direct contact there is with other backbone nodes.

Chapter 3

Research

We propose a mobile ad hoc network protocol, called MARS, that is suitable for different realtime situations, as shown in section 4. MARS is based on SCALAR, because it performs better than related protocols for mobile ad hoc networks and it is the most complete solution so far. Despite SCALAR performing well in mobile ad hoc networks, we have made some important improvements to perform better in the selected realtime situations. The improvements are as follows:

Adaptive dynamic backbone construction to improve the scalability of the network.

Resource based backbone construction to improve sustainability of the network.

Message prioritization, timers and TTL to improve the Quality of Service of the network.

3.1 Adaptive dynamic backbone construction

Backbone construction, that is based on the dominating set problem, lets the number of backbone nodes grow in the same order as the total number of nodes in the network and is not able to maintain connectivity in a highly dynamic network. SCALAR has a periodic backbone construction based on the dominating set problem and we wanted to make this an adaptive dynamic backbone construction to be able to maintain connectivity in highly dynamic networks and to prevent a backbone construction on crucial moments. The overhead of the backbone construction should be as small as possible to ensure that the probability of receiving important messages in the network is as high as possible. We therefore implemented an adaptive dynamic backbone construction based on the ADB protocol explained in section 2.4.

Every node starts as if it is a network on its own. Then the knowledge the node has of the surrounding network grows as more and more nodes broadcast their periodic *hello* message and their neighbour list. We added the neighbour list to the *hello* message with the broadcasting node as *NextHop*, just like in the FWTab. This way, we changed our ABD implementation from a multicast protocol to an unicast protocol. The neighbours will be added to the neighbour list as long as the *NLFF_THRESHOLD* and *HOP_THRESHOLD* are not exceeded. The nodes can then with this information decide for themselves if they could be seen as a backbone node. Changes in the network topology will be detected when certain *hello* messages are not being received anymore. Now it is possible for a node to send a highly important message in favor of backbone construction messages without being excluded from the network and its backbone. As long as the *hello* message will be send in time, all surrounding nodes know that that node is (still) there.

Besides the dynamic part of this construction, there is the adaptive part. We changed the two-hop vicinity part of SCALAR to a vicinity of an adaptively changing number of hops, just like the ADB protocol. The number of link failures and the *NLFF_THRESHOLD* and *HOP_THRESHOLD* are used in each node to determine the maintainable hop vicinity of that node. Neighbours, that are passed through to a node, that exceed the thresholds are not added to the NTab of receiving node. The direct vicinity, as we call it, increases when the mobility of the network decreases and respectively decreases when the mobility of the network increases. Such an adaptively changing vicinity makes the protocol suitable for both networks with for example quickly changing traffic as well as networks with for example more statically moving military lines.

3.2 Resource based backbone construction

In both SCALAR and the ADB protocol, the backbone selection proces is strongly based on the ID number of the node. In realtime situations, the ID number (or MAC/IP address for that matter) of a node can be seen as random and therefore does not give any important information about the node itself. We propose to base the backbone construction on useful information about the node that indicate the state of that node.

A backbone node will send more messages than an end node in most networks, due to the forwarding of messages. Because of this, the node will use up more battery life and memory space and therefore nodes with the highest battery life and the largest amount of memory space should be placed in the backbone. This will, in time, increase the durability of the (backbone) nodes and thus the sustainability of the network. We therefore take the state of the battery and the amount of memory space of the nodes into account when constructing the backbone.

Since our protocol is adaptive to the mobility of the network, we also take the reach and the speed (using GPS) of the nodes into account. A greater reach means that a node is more likely to stay visible for the surrounding nodes when they move around. Therefore a node with a greater reach is more suitable for being a sustainable backbone node. This also applies to the speed of the node. Nodes that move around relatively quickly change the network topology more than nodes that move around relatively slow. A backbone of nodes that move less relative to each other would thus be a more sustainable backbone.

We thus extended the *status* metric of the ADB protocol to included the battery life, memory space, speed and reach of the related node. When the situation occurs where the *status* values of two adjacent nodes are precisely the same, we still use the *NodeID* to break ties. The overall result of resource based backbone construction is that the probability increases of the backbone nodes being the nodes in the network with a stronger battery, stronger radio and that move relatively slow. In realtime situations, these nodes will be for example trucks on the highway instead of the passenger cars and tanks instead of the backpack radios of military groups. We calculate the *status* value as follows:

$$status = \alpha * NLFF^{-1} + \beta * degree + \gamma * battery + \delta * velocity + \epsilon * memory$$

Where $\{\alpha, \beta, \gamma, \delta, \epsilon\}$ are smoothing factors between 0 and 1 that can be adjusted as desired.

3.3 Message prioritization, timers and TTL

SCALAR is implemented upon the MAC layer in IEEE 802.11 and both do not take the priorities of the messages into account. Message prioritization is a must in realtime situations, because, simply said, some messages are just more important and need to have a greater chance of delivery. To implement message prioritization, we use the Message Priorization Mechanisms mentioned in section 2.2. The prioritized waiting time mechanism and the prioritized backoff time mechanism are implemented as they are explained in that section.

In the static priority scheduling mechanism, we make use of 3 priority queues. The highest priority queue is for the most important messages, such as a *brake* message, between self-driving cars, when an accident has occured and an *attack* message, between military groups, when the front line is under attack from a certain direction. Basically, the messages that provide the safety of the nodes. The second priority queue is for control messages, which have a higher priority then normal messages for the reason that the backbone has to be updated for the protocol to keep working. The last priority queue is for normal messages, which are the remaining messages that nodes can individually send to each other. We will consider the case of one normal priority queue for the sake of simplicity, but if desired the normal priority queue can be divided in even more priority queues.

We also implemented timers in the nodes to keep track of how long certain requests are pending, which is not the case in SCALAR. When a timer exceeds a certain period of time, the sending node will send the request again. In the backbone nodes a timer is also used to remove forwarded requests that have been pending for too long to free memory. Due to this timer and the resulting resend of the request, the probability of receiving the data item increases, like with the TCP protocol [9]. We do not use timers on time-bounded messages, like with the UDP protocol [9], for the reason that delayed messages are not useful anymore.

After we implemented the timers, we concluded that a loop was possible where backbone nodes to keep sending the same message over and over again to each other as long as the loop took longer than the timer set for that message and thus filling the queues with messages that could already be received by the destination. We therefore included a TTL value to every message which works the same as TTL in the IP protocol [6]. The TTL value is decreased by one every time it passes through a backbone node. When the TTL value reaches zero, the message will not be forwarded anymore and the message will be deleted. This way, when a loop of backbone nodes sending each other the same message over and over again occurs, the loop will be cut off at a certain moment. The initial TTL value should be large enough, so that nodes which are separated by a great distance can still reach each other.

The last improvement we made upon SCALAR is on the last step of the data lookup process. The requested data item follows the path backwards the request for that data item took, but when the path is broken due to the mobility of the network the data item will not be received. We already solved this by setting a timer on the pending request and resending the request when the previous path may be broken somewhere. But with the adaptively changing hop vicinity of the nodes in a network, we could almost guarantee the delivery of a data item in the last hops of the path, by letting the nodes first check their neighbour lists for the requesting node before forwarding the data item to the node where the request came from. Now, in stead of following a specific path end, the data item could take the shortest path end to the requesting node, which improves the delivery time and decreases the probability of a broken path.

Chapter 4

Scenarios

Since our aim is to improve upon SCALAR we tested our protocol the way SCALAR is tested. We therefore used the same variables as mentioned in the paper discussing SCALAR. In the paper the two main types of simulations/tests, that they ran, are a density test and a business test. The density test is a simulation where the amount of nodes increases and the area size stays the same. In the density simulations the area size is $500m^2$, the number of requests sent per node is 2 and the number of nodes in order of iteration is $20, 50, 100, 200$ and 400 nodes. The business test is a simulation where everything stays the same, but the total number of messages per node increases. In the density simulations the area size is $500m^2$, the number of nodes is 100 and the number of requests sent per node in order of iteration is $1, 2, 3, 4$ and 5 requests.

We built a simulation environment based on scenario files to ensure that the nodes in both the MARS simulation and the SCALAR simulation will behave exactly the same. A scenario file for each test, each test iteration and each scenario is generated by the environment. We tested the five iterations of the two tests on the following scenarios:

1. Drone scenario
2. Highway scenario
3. Military scenario

These scenarios are explained in futher detail later on. The result of those tests are eight plots that show the following:

Messages sent which is the total amount of messages sent for the network to connect and for all requests to be sent. This plot represents the efficiency of the network the most.

Success ratio is the percentage of messages that are delivered succesfully. This plot represents the Quality of Service of the network the most.

Active nodes is the percentage of nodes that is still active at the end of a scenario to see if some batteries ran out.

Backbone nodes is the average amount of backbone nodes in the network. We also included parent nodes in MARS since they behave mostly like backbone nodes.

Average remaining battery life shows on average how much energy was left in the nodes at the end of the scenario. Sending messages costs 0.05% of battery life in our simulations (related to messages sent), so we can see how the backbone behaves when battery life is limited. This plot also represents the sustainability of the network the most.

Average memory space shows on average how much memory space is left during the duration of the scenario. Each node starts with a random memory size between 512 and 1024 kb and each pending and received request and each neighbour saved decreases this by 1 kb.

Average time frames per received message is the amount of time frames a request took on average to result in a received data item. Each time frame each node can transmit one message from its queue, these results thus show the average number of time frames a message was waiting in queue plus the average number of time frames a message was tranmitted (related to average hops).

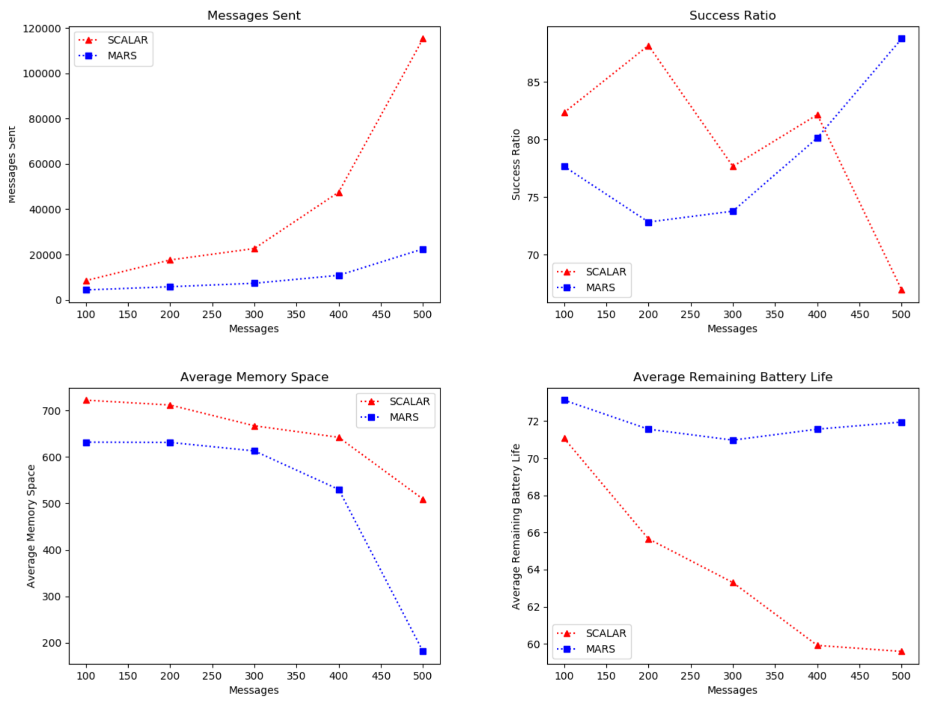
Average hops per received message is the amount of nodes on average a request plus the related data item passed through before it was received. This number also includes the nodes (hops) that are not on the shortest path between the requesting node and the data item, to see the efficiency with with a request is send to the node with the data item and back.

The scalability of the protocol is represented by the overall evenness of the plots. This namely shows how much effect the size of the network has on the performance of the network. Our simulation environment and the implementation of SCALAR and MARS can be found at the Gitlab repository: `https://gitlab.science.ru.nl/avalem/BachelorThesis`.

4.1 Drone scenario

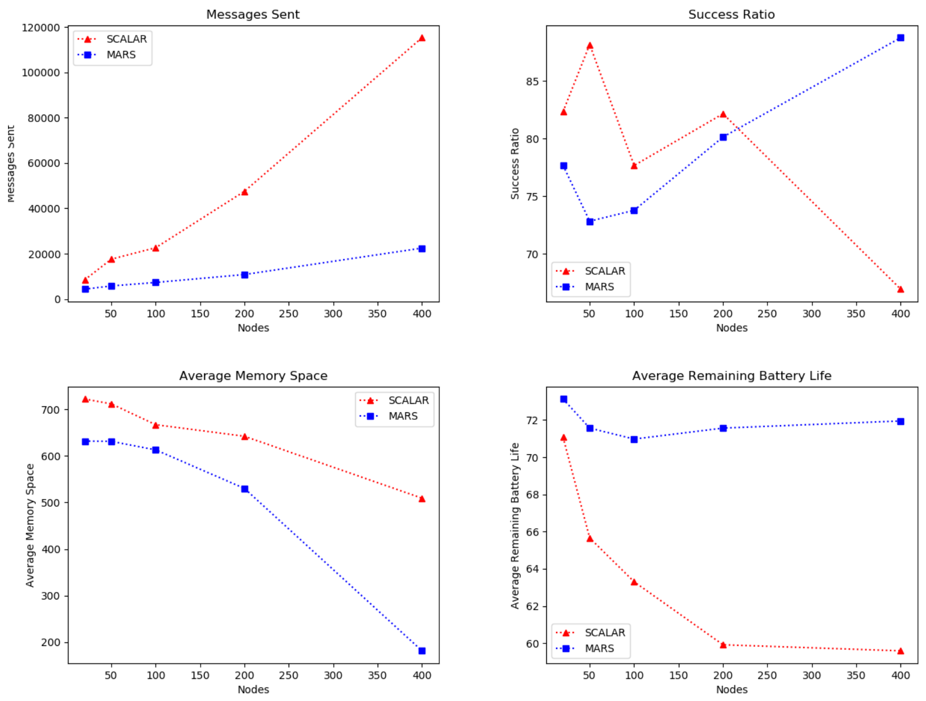
SCALAR is tested in a scenario where every node was randomly moving. This is very similar to a scenario where drones are flying randomly through a specific area and sending data to each other. This correlation is why we tested our MARS in comparison with SCALAR in a drone scenario. We allowed multiple drones on the same point in space, since drones can fly over each other. The strength and behaviour of the nodes is randomly generated to simulate different types of drones moving randomly in the area. The most important goal of this scenario is to see how our protocol performs in the scenario SCALAR is tested in. Below are some of the plots of the result, the rest of the plots for this scenario can be found in figure A.3 and figure A.4.

Figure 4.1: Drone Scenario - Business test:



Both the business as the density test show very similar results for this scenario. The first plot clearly indicates that MARS sends less messages than SCALAR. Which indicates that MARS sends the requests more efficient through the network. This can be explained by the rapid growth of the average number of hops a message goes through with SCALAR. In SCALAR, a request could be forwarded through the whole backbone despite the destination node being only a few hops away. And since SCALAR has also more backbone nodes on average, the number of hops a request goes through is also higher. This does seem to effect the average memory space, since MARS uses more memory space where SCALAR sends more messages. The amount of messages sent explains the difference in average remaining battery life between the two protocols. There are no nodes inactive (out of battery) at the end both tests for both protocols.

Figure 4.2: Drone Scenario - Density test:

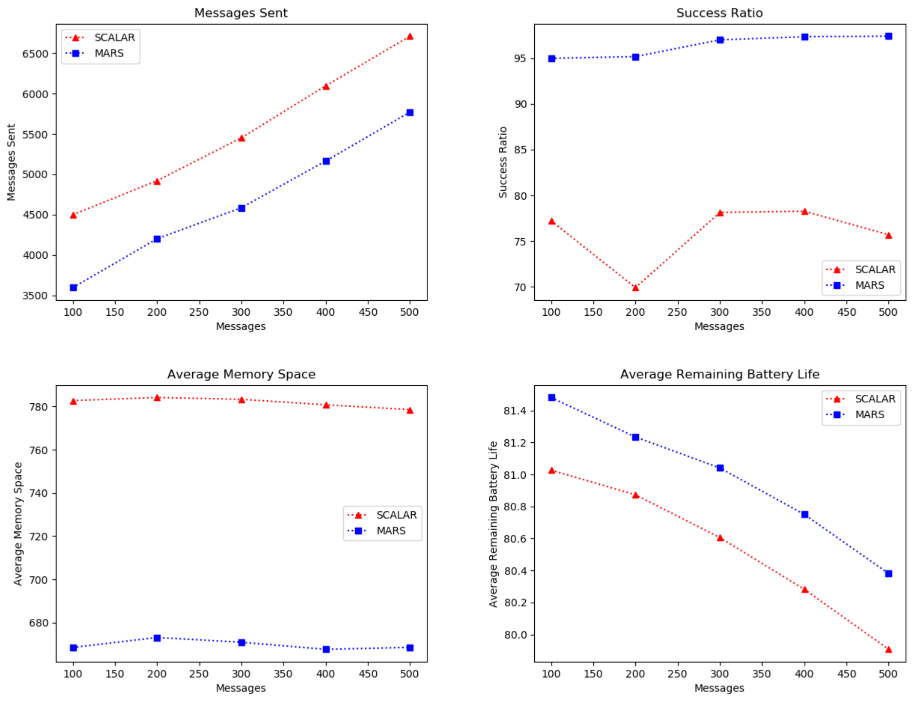


When we look at the success ratio, we see that SCALAR performs better than MARS with a small number of nodes in the network. However the success ratio of MARS increases as the number of nodes increases where the success ratio of SCALAR decreases significantly. The scalability of MARS is better than that of SCALAR, since SCALAR tends to diverge as the number of nodes increases. This is also visible with the average time frames per received message, which could explain the behaviour of the success ratio results, since a higher number of time frames decreases the chance of (timely) delivery. A high number of time frames namely indicates that the queues in the nodes are filled, probably due to the high amount of messages sent, which interferes with the delivery of the messages. The overall result of the density and business test, looking mainly at scalability, sustainability and Quality of Service, is that MARS is the better solution for this scenario.

4.2 Highway scenario

The highway scenario tests the realtime situation of a highway, like the name suggests. We simulate three lanes going west and three lanes going east with the slowest and strongest vehicles in the outer lanes and the faster and weaker vehicles in the inner lanes. This means that the trucks in the outer lanes contain the stronger radios and the passenger cars in the inner lanes contain the weaker radios. This is a far more structurized scenario than the drone scenario, since on the greater scale the cars follow a certain type of behaviour. The goal of this scenario is to see how our protocol performs on the highway with selfdriving cars, for example. Below are some of the plots of the result, the rest of the plots for this scenario can be found in figure A.5 and figure A.6.

Figure 4.3: Highway Scenario - Business test:



The plots show less divergent results than those of the drone scenario as expected, because the scenario is more structurized. In the business test we see that the results of MARS and SCALAR are somewhat parallel to each other. Where MARS comes out on top in every plot, except with memory space and average time frames per received message. Since the results, like messages sent, are somewhat parallel, we expect that the difference in memory space is mostly the data needed in each node to construct the backbone. The average time frames per received message may be explained by the fact that in MARS messages are resend after a fixed number of time frames, where that message would not succesfully be delivered with SCALAR. In this test we see that in both protocols the same amount of nodes did become inactive during the test.

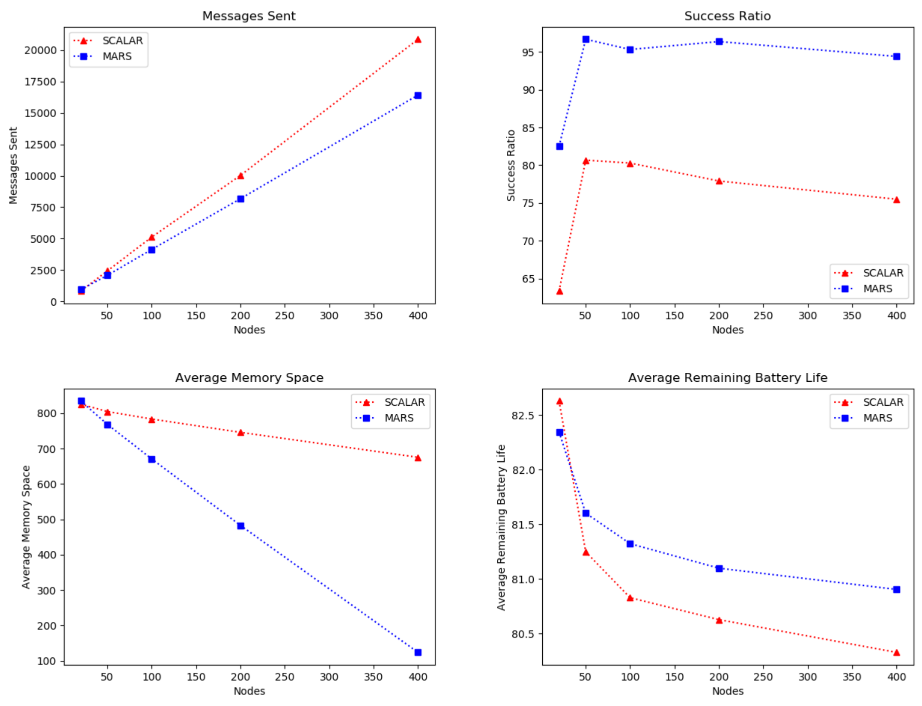


Figure 4.4: Highway Scenario - Density test:

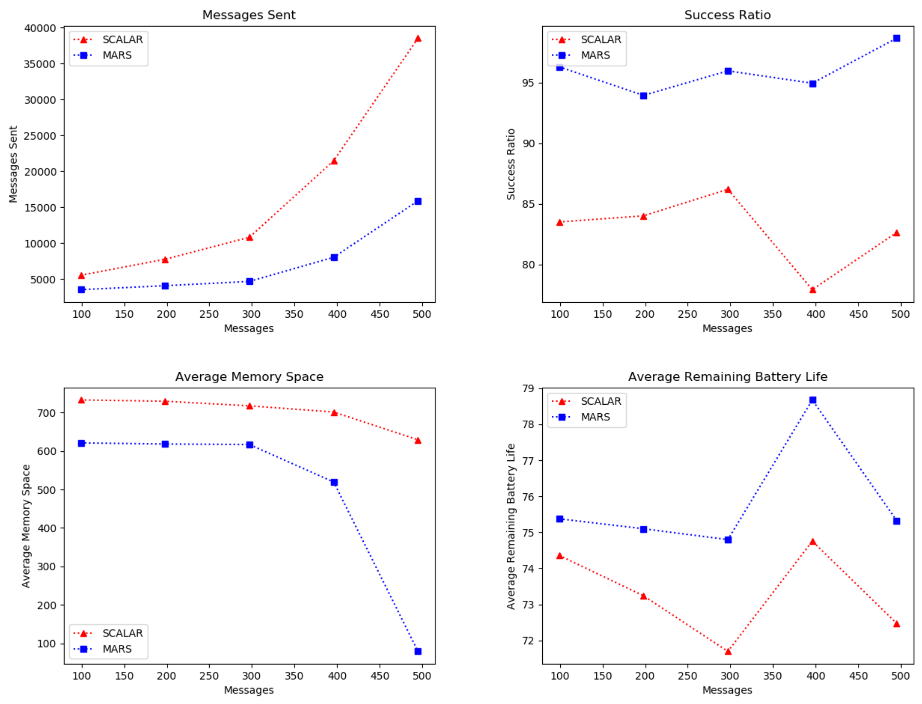
The results of the density test are similar to the results of the business test, although the protocols are less parallel to each other. The explanation for the difference in memory space and average time frames per received message from above can be given here again. In this test all nodes in both protocols stayed active till the end of the test as expected, since more nodes implies more choices for the backbone (dividing the load). Taking the scalability, sustainability and Quality of Service into account again, we see that MARS is also the better solution for this scenario.

4.3 Military scenario

In the military scenario there are three lines moving forward in formation. The front line consists of the backpack radios and thereby have less battery power, memory space and range. The middle line consists of the tanks that contain stronger radios than the front line. And the last line consists of the communication vehicles that contain the strongest radios in this scenario.

Although this is the most structurized scenario the nodes still move randomly in their respective line, but with an overall forward motion. Below are some of the plots of the result, the rest of the plots for this scenario can be found in figure A.7 and figure A.8.

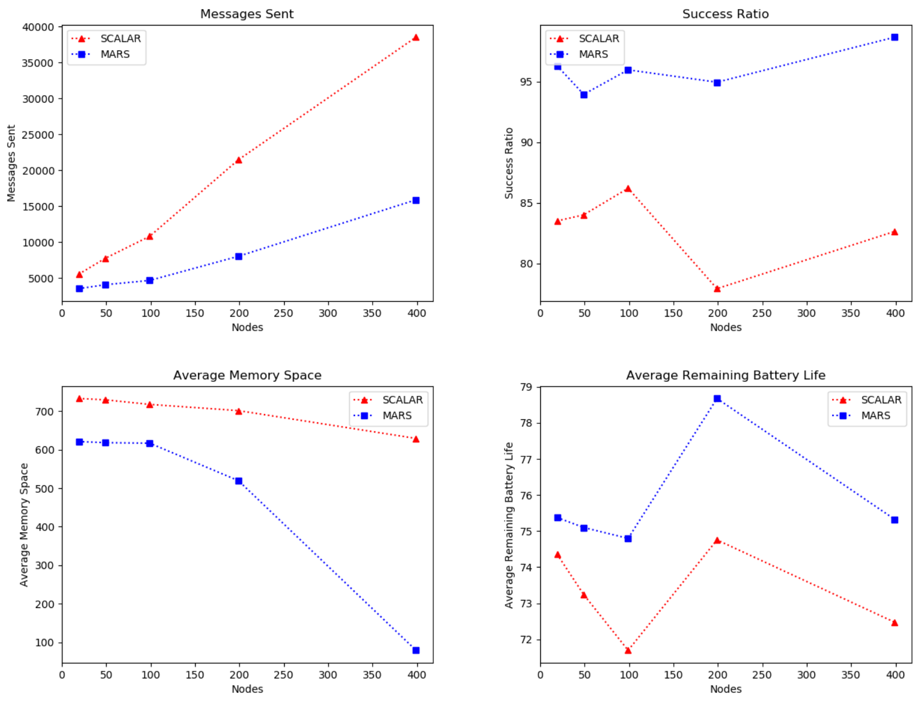
Figure 4.5: Military Scenario - Business test:



Since structure seems to effect the results, we expected the results to be similar as that of the highway scenario. In the business test we indeed see the parallel pattern from the business test of the highway scenario again, but this time SCALAR diverges with the time frames per received message as the number of messages increases. This is probably due to the high amount of message sent (large queues) in the network. In this test all nodes stayed active till the end of the test in both protocols.

With the previous results and explanations in mind, the result of the density test is as expected. MARS shows more even results and is in this scenario thus again more scalable than SCALAR. The Quality of Service and the sustainability of MARS is yet again better than that of SCALAR, only memory usage is again more than SCALAR. So we conclude that MARS is also the better solution for this scenario.

Figure 4.6: Military Scenario - Density test:



Chapter 5

Related Work

Besides SCALAR, there are other protocols that also try to ensure a scalable and sustainable mobile ad hoc network with a high Quality of Service. But none of them, discusses the backbone construction as well as the data lookup and replication scheme. When we break SCALAR up into those parts, we can compare them to related work.

When it comes to protocols that are based on a backbone construction, the most related protocols to ours are MAODV [5] and ODMROP [5]. Both protocols and SCALAR show that a backbone construction descreases the total amount of messages sent in a network and thereby increases the scalability of the network. MAODV is a multicast protocol like ADB that also follows the tree-based approach based on hard state information. ODMROP on the other hand follows a mesh approach based on softstate information. ODMROP is the better protocol of the two when it comes to success ratio, but the overhead of ODMROP is much higher. ADB was in both success ratio and overhead the better choice.

Another lookup and replication scheme that is scalable like SCALAR is DHTR (Distributed Hash Table Replication) [8]. A lookup and replication scheme caches data items that are frequently requested in intermediate nodes, which is shown by both protocols to increase the data availability of the network and thereby the Quality of Service of the network. DHTR clusters the nodes in hierarchical groups and uses hashtables for efficient data lookups in the groups. Due to this clustering, DHTR claims to decrease the communication overhead, which is an important constraint in a scalable mobile ad hoc network. Since SCALAR relates to DHTR and was the more complete choice, we chose to build upon SCALAR.

Chapter 6

Conclusions

We proposed a new MANET protocol called MARS, that is based on SCALAR. The protocol now includes an adaptive dynamic backbone construction and as the results show this increases the scalability of the network. It also now includes a resource based backbone construction which results in a stronger and more sustainable backbone, which in turn results in a more sustainable network. Last but not least, the protocol now includes message prioritization, timers on pending requests and message TTL, which further increase the Quality of Service of the network. Highly important messages have a higher chance of delivery in less time and the backbone construction does not interupt the transmission of such messages, because backbone construction has now the second most important priority.

As the results of our tests show, MARS is the better solution in the three selected scenarios in comparison with SCALAR. However, we also see that MARS takes up more memory space due to the adaptivity of the nodes and probably more computing power due to the complexity of the backbone construction and message prioritization. The effects of this on the nodes themselves should be tested in real life as future work. Also MARS should be tested with more scenarios to see if it is suitable for more realtime situations.

Bibliography

[1] Emre Atsan and Öznur Özkasap. Scalar: Scalable data lookup and replication protocol for mobile ad hoc networks. *Computer Networks*, 57(17):3654–3672, 2013.

[2] IEEE Computer Society LAN MAN Standards Committee et al. Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Standard 802.11-1997*, 1997.

[3] Chaiporn Jaikaeo and Chien-Chung Shen. Adaptive backbone-based multicast for ad hoc networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 5, pages 3149–3155. IEEE, 2002.

[4] Ulas C Kozat, George Kondylis, Bo Ryu, and Mahesh K Marina. Virtual dynamic backbone for mobile ad hoc networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 250–255. IEEE, 2001.

[5] Thomas Kunz and Ed Cheng. Multicasting in ad-hoc networks: Comparing maodv and odmrp. In *Proceedings of the Workshop on Ad hoc Communications*, pages 186–190. Citeseer, 2001.

[6] JE McGeehan, Saurabh Kumar, Deniz Gurkan, SMR Motaghian Nezam, Alan Eli Willner, KR Parameswaran, MM Fejer, J Bannister, and Joseph D Touch. All-optical decrementing of a packet's time-to-live (ttl) field and subsequent dropping of a zero-ttl packet. *Journal of lightwave technology*, 21(11):2746–2752, 2003.

[7] Xavier Pallot and Leonard E Miller. Implementing message priority policies over an 802.11 based mobile ad hoc network. In *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, volume 2, pages 860–864. IEEE, 2001.

[8] Pooja Sharma, Kamal Kant, and Naveen Chauhan. A comparative study of cluster-based data replication techniques for manets. *International Journal of Information Technology*, 2(2):665–667, 2010.

[9] Shie-Yuan Wang, Hsi-Lu Chao, Kuang-Che Liu, Ting-Wei He, Chih-Che Lin, and Chih-Liang Chou. Evaluating and improving the tcp/udp performances of ieee 802.11 (p)/1609 networks. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 163–168. IEEE, 2008.

Appendix A

Appendix

Figure A.1: Fields of NTab

Field	Description
NodeID	ID of the neighbour node
CoreID	ID of the backbone node with which that neighbour is associated
HopsToCore	Numbers of hops from that neighbour to the backbone
Degree	Number of neighbour nodes of that neighbour node
NLFF	Calculated NLFF by that neighbour to its nearest backbone node
LastUpdated	Timestamp of when this entry was last updated

Figure A.2: Fields of FWTab

Field	Description
CoreID	ID of the backbone node with which the destination is associated
NextHop	Next hop towards that backbone node
Hops	Total number of hops to that backbone node
NLFF	Calculated NLFF of the path to that backbone node
LastUpdated	Timestamp of when this entry was last updated

Figure A.3: Drone Scenario - Density test:

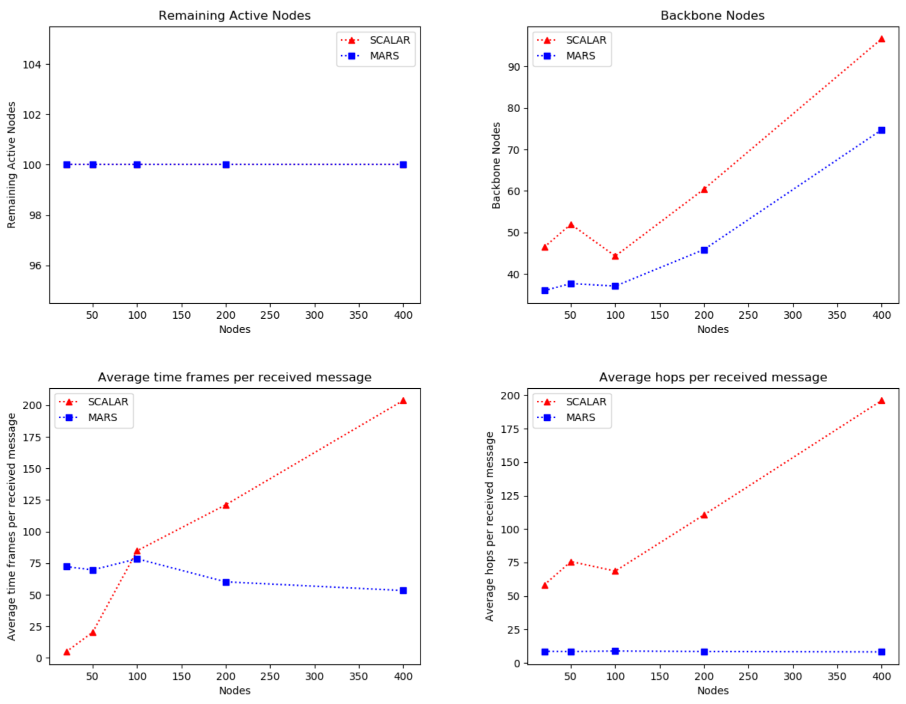


Figure A.4: Drone Scenario - Business test:

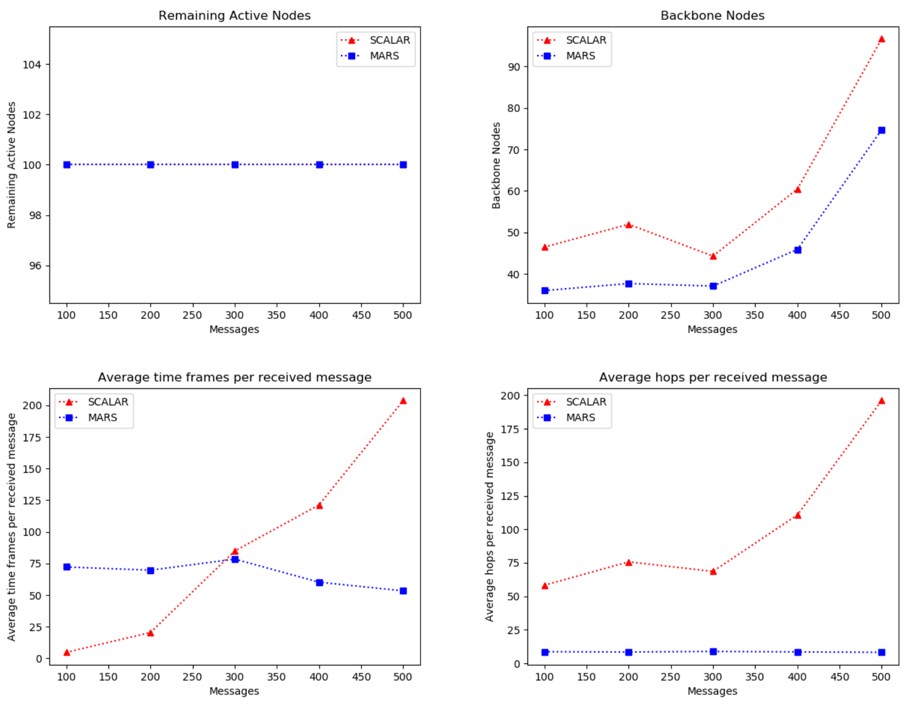


Figure A.5: Highway Scenario - Density test:

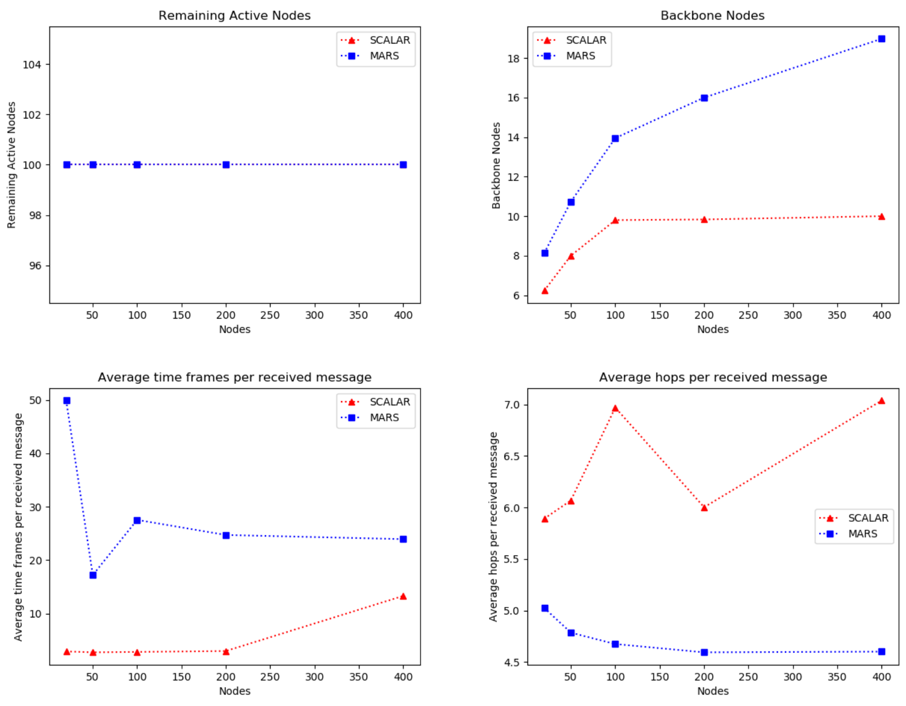


Figure A.6: Highway Scenario - Business test:

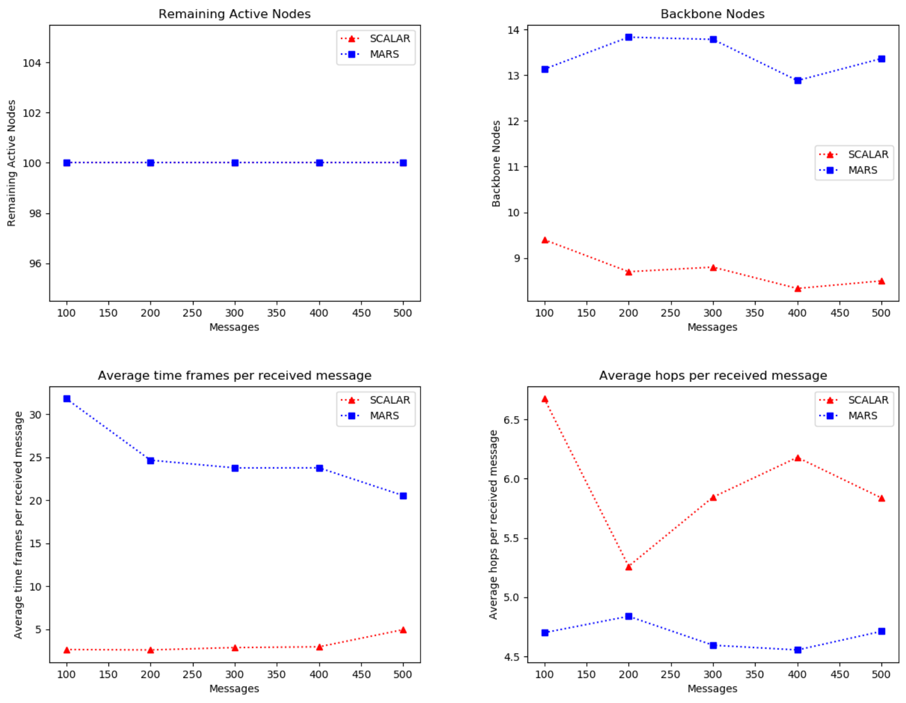


Figure A.7: Military Scenario - Density test:

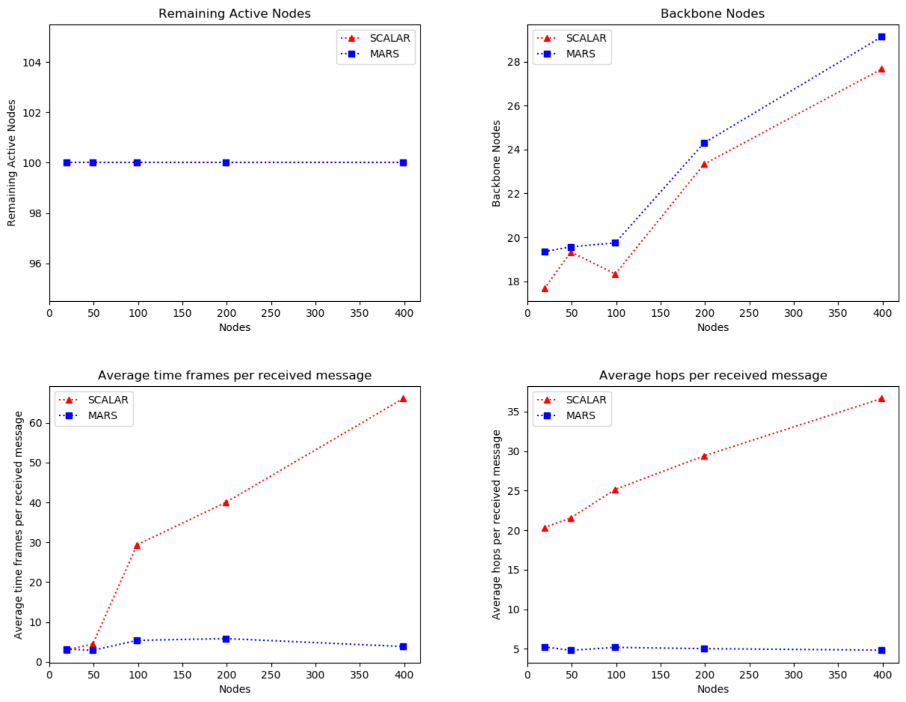
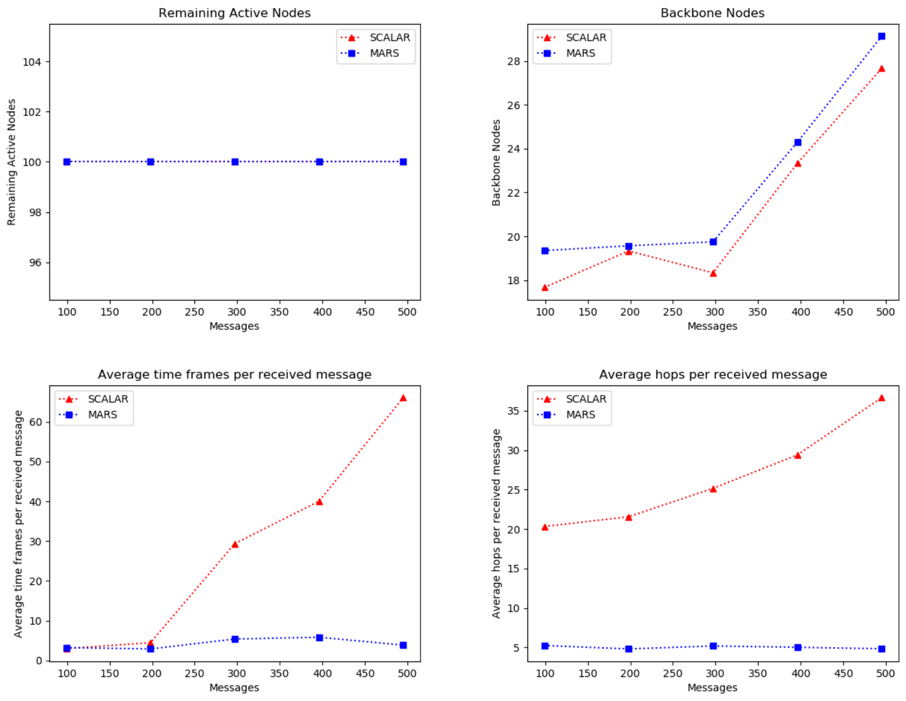


Figure A.8: Military Scenario - Business test:


Remaining Active Nodes
Remaining Active Nodes
Messages
Backbone Nodes
Backbone Nodes
Messages
Average time frames per received message
Average time frames per received message
Messages
Average hops per received message
Average hops per received message
Messages
SCALAR
MARS