Remote Routing Approach to Restricted Devices in MANETs

Rodrigo Melo[1], Rafael R. Aschoff[2], Djamel Sadok[1], Eduardo Feitosa[3]

[1]Center for Informatics – Federal University of Pernambuco (UFPE)
Pernambuco – Brazil

[2]Pernambuco Federal Institute of Education, Science, and Technology (IFPE)
Palmares, Brazil.

[3]Federal University of Amazonas
Manaus – Brazil

{rodrigodma,jamel}@gprt.ufpe.br, rafael.roque@palmares.ifpe.edu.br,
∈feitosa@icomp.ufam.edu.br

Abstract. *Emergency rescue communication systems are designed to provide ubiquitous collaboration among mobile devices without the need for a fixed infrastructure by using mobile ad hoc networks (MANETs). In emergency and rescue operations, MANETs are naturally formed by devices with different processing and communication capabilities. In this scenario, low power devices may become overwhelmed with the control overhead and resulting additional processing required to provide reliable communications among these heterogeneous parties. In this paper we propose a strategy where resource constrained devices can offload part of the routing process to more capable devices participating in the network. Our proposal have been tested and validated in different scenarios. The results show how the proposed approach can successfully reduce the network overhead without significantly reducing the routing process efficiency.*

1. Introduction

The increasing interest in using mobile devices to establish ad-hoc communication systems (MANET) has enabled the development of a broad range of applications, particularly with the advent of new wireless access technologies for connectivity, such as 3/4G, LTE, and WiMax. However, these new applications typically assume that MANETs are a set of homogeneous devices, where each node has the same capabilities, which is usually unrealistic.

MANETs support several types of mobile devices and this is one of its success factors. This heterogeneity allows great flexibility, however, when devices with limitation in battery or processing power join the network new problems arise. In this paper, we concentrate on the routing process issues for restricted devices. For such devices, participating in large networks may not be possible due to the routing process that can be very resource consuming. For example, during the routing process, topology information must reach the whole network, which generates a great amount of overhead that can compromise the performance of these special devices.

The most popular strategy to address this problem is to reduce the routing protocol overhead, particularly, control packets that consume network bandwidth and other

resources. Works like [Pei et al. 2000] and [Younis et al. 2002] have been proposed with solutions focusing in performance improvements. However, these solutions aim at the overall network performance, while our concern is on allowing restricted devices in the networks. We focus on techniques to save the resources of restricted devices during the routing decision process so they can participate in the network.

This paper proposes the Distributed Remote Routing (DRR), a strategy to offload part of the routing decision process of restricted devices to more capable devices. Developed to work on networks running the HTR protocol [Souto et al. 2012], we introduce a new categorization of HTR nodes: the HTR-Lite (HTR-L) nodes, which indicates the restricted devices and the HTR-Outsourcing Router (HTR-OR), the more capable devices that will be responsible for helping the HTR-L nodes in the routing process.

2. HTR Overview

HTR is a routing protocol for MANETs that utilizes a 2.5 cross layer scheme [Souto et al. 2012]. It abstracts multiple and heterogeneous interfaces and constructs a self-organized heterogeneous communicating ad hoc network. An HTR node may have many interfaces, with similar or different technologies such as Wi-Fi, Bluetooth, WiMAX and LTE, but it has only one IP address.

HTR is based on the OLSR protocol [Clausen and Jacquet 2003] and similarly includes HELLO and Topology Control (TC) as control messages and defines a special node, the multi-point relay or MPR, for the control of traffic flooding. From a HELLO message, a mobile node receives information about its immediate, 2-hop neighbors and selects MPRs accordingly. A TC message originates at an MPR node announcing who has selected it as an MPR. In contrast to OLSR, HTR uses additional metric based on link quality information and node device capabilities to choose MPR nodes. Called HTRScore, the HTR cost metric is defined considering factors such as the awareness of link conditions and power efficiency in order to perform path computation.

The HTRScore formula can be seen at (1).

$$HTRScore(i,j) = \frac{e_{i,j}^{\alpha}}{(1-\rho_{i,j})^{\beta}} * \frac{E_{\gamma}^{i}}{R_i^{\Theta}} \tag{1}$$

Where i is the source node; j is the destination neighbor; $e_{i,j}$ is the transmission energy required for node i to transmit an information unit to its neighbor j; $\rho_{i,j}$ is the probability to lose a packet sent from i to j; R_i is the residual energy of node i; and E_i is the initial battery energy of node i.

The symbols α, β, γ and θ represent non-negative weighting factors for each described parameter. Note that if all weights are equal to zero, then the lowest-cost path is the shortest path, and if only γ, and θ are equal to zero, then the lowest cost path is the one that will require the least energy consumption, considering retransmission or not, regarding the value of β. If γ is equal to θ then normalized residual energy is used, while if only θ is equal to zero then the absolute residual energy is used. In case all three parameters α, γ and θ are equal to zero, then only the paths with best link stability are emphasized.

Two main modules compose the HTR framework: the bootstrap module and the routing module. The bootstrap module is responsible for the start-up and configuration of

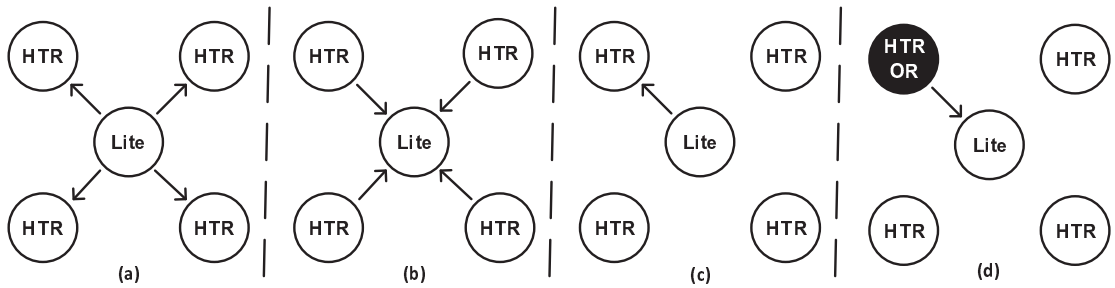

Figure 1. HTR Lite steps of the remote routing solution

a node (i.e. assignment of IP address and link layer adaptive configuration). The routing module manages the routing table and packet forwarding. It uses the Dijkstra Algorithm to perform path computation, however, the edges of the network graph have a weight equals to the HTRScore described above. Specific details regarding these modules can be found in [Souto et al. 2012].

3. Distributed Remote Routing

On a HTR-running network, each node is responsible for building its own routing table based on information received via the control messages. Each node knows its neighbors and is capable of deciding the best path to reach a destination. As introduced above, this may constitute a restricting factor for some devices.

The Distributed Remote Routing (DRR) is our approach to relax this restriction. It was developed to save resource from these nodes by offloading routing decisions (routing table calculation) to other devices within the network.

There are three fundamental roles in the DRR process as described below.

- **HTR nodes:** nodes that run native HTR protocol. They send and receive HTR control messages (HELLO and TC) and build their own routing table.
- **HTR-Lite nodes:** restricted devices. HTR-lite nodes do not send HELLO nor TC messages and do not act as routers. In other words, HTR-Lite nodes will not appear as a valid entry on the forwarding table of the network nodes. HTR-lite nodes mount their routing table by sending a table request to HTR-OR nodes.
- **HTR-OR nodes:** HTR-Outsourcing Router nodes are responsible to provide the routing table information requested by HTR-Lite nodes. There is no limit regarding the number of HTR-OR nodes participating in the network.

We embedded the DDR in the HTR framework but, as expected, DDR has its own set of messages and control states. Figure 1 illustrates the protocol operation.

As shown in the figure, the HTR-Lite initiates the operation as soon as it joins the network (managed by the bootstrap module). The node sends a broadcast request (discovery message) in search for a HTR-OR (a). When a HTR-OR node receives a discovery message (b), it replies with its HTRScore directly to the requesting node (HTR-Lite).

After having received replies from one or more HTR-OR nodes, the HTR-lite node uses the informed HTRScores as criteria to choose the node that will act as its HTR-OR. Having selected the HTR-OR the HTR-Lite node sends a Route Request message

to it (c). Finally, upon receiving a route request message, the HTR-OR node sends the Routing Table Reply message to the requesting node (d).

It is important to emphasize that there is no significant increase of resource consumption by the HTR node that becomes HTR-OR. This is because it will only send its already calculated routing table to the requesting HTR-lite node.

Since the HTR-OR only sends its own routing table, the HTR-Lite needs to perform minor adjustments to such table to adapt it to the point-of-view of the HTR-Lite itself. The algorithm to make this adjustments was designed to consider the cost of manipulating the route data without incurring in too much resource consumption. Firstly, during the second step of the remote routing process (see Figure 1), not just the elected HTR-OR but all nodes that replied to the Discovery Message are added to the neighbor table. Then, after receiving the routing table from a HTR-OR, the insert route table algorithm is started. The procedure is described in Algorithm 1. Simply put, the algorithm verifies the received routing table (line 4) and for every advertised entry (line 6) that has not already been added (line 7), it changes the next-hop field to the selected HTR-OR (line 8).

Algorithm 1 Insert Route Table Algorithm

```
 1: procedure INSERTTABLE(routeOR, addrOR)
 2:     size ← routeOR.size()
 3:     index := 0
 4:     table ← getRouteTable()
 5:     while index < size do
 6:         route ← routeOR.get(index)
 7:         if table.hasNoEntry(route) then
 8:             route.updateNextHop(addrOR)
 9:             table.insert(route)
10:         end if
11:         index := index + 1
12:     end while
13: end procedure
```

In order to comply with the unpredictable nature of the mobile environment, the received routing table is set to expire after a period and the whole process starts again. Figure 2 shows the state machine of the lite node in this proposed DRR protocol.

As shown in the figure, there are four states in the lite node: Start, Wait Reply, Wait Route Table, and Route Table Received. The Start state represents the initial phase, where the lite node wishes to initiate the remote routing process. After sending a Request Message, the lite node goes into the Wait Reply state. If the lite node receives no reply, it will constantly attempt to send another request after a timeout interval. If a reply is received, the node sends a Select Route message and goes into the Wait Route Table state. Similarly, at this state the lite node will attempt to retry to send the Select Route message after a timeout occurs, but only for a limited amount of time. If the route table is received, the node ends its configurations process; otherwise, it goes into the Wait Reply state to start the process again. Important to say that this state machine is always running when entry in network or when the table received expires.

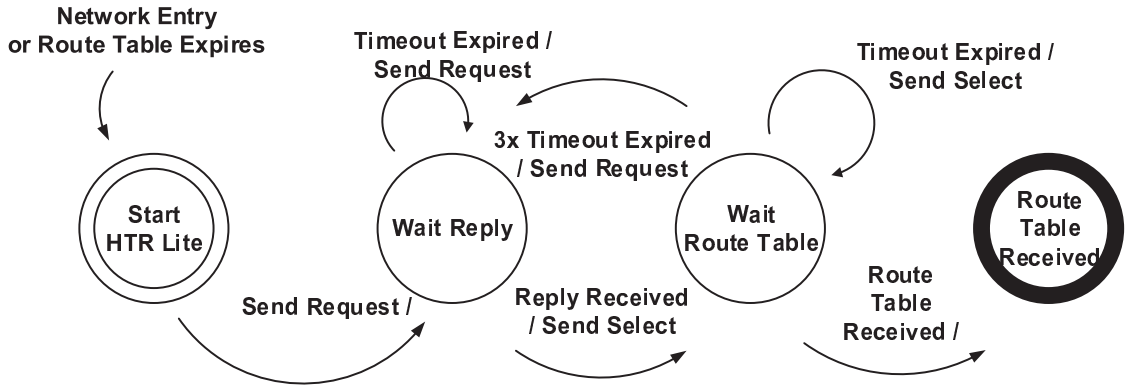

Figure 2. State Machine of the remote routing protocol for the Lite node

The server side works as a stateless process, as describe in Algorithm 2. The server continually listen to received messages (lines 2-3). When the server receives a Discovery Message (line 5), it sends back a Reply Message (line 6). If the server receives a Request Message, it sends a Route Reply Message.

Algorithm 2 Server side remote routing process routine

```
1:  procedure RECEIVEMESSAGE
2:      while true do
3:          message, srcAddress ← receiveMessage()
4:          type ← message.getType()
5:          if type = DISCOVERY then
6:              sendScore(srcAddress)
7:          else if type = REQUEST then
8:              sendReply(srcAddress)
9:          end if
10:     end while
11: end procedure
```

4. Evaluation Methodology

In order to evaluate the performance of the proposed protocol, we have used different scenarios and metrics. This section describes the methodology we used to perform the evaluation the proposed protocol.

4.1. Metrics

Given that the proposed routing protocol does not change the network behavior, apart from the process to obtain the routing table, only a few network metrics would have a possible change on their measurable values comparing to the standard approach. We chose to evaluate the routing delay, and message overhead, which are better described below.

Routing Delay. The amount of time a HTR-lite node takes to obtain the routing table shared by one HTR-OR, after the complete remote routing process was performed. More specifically, it constitutes the time between the discovery message and the routing

table reply message. As illustrated by Figure2, the process includes intermediate steps, such as the select and reply messages as well as possible timeouts and reattempts.

Message Overhead. The message overhead is a metric to measure the amount of network bandwidth save while using the proposed remote routing protocol. Nodes running the remote routing do not participate in the forwarding of messages nor send HELLO or TC messages, which saves resources. This metric compares the amount of traffic generated by networks with and without the remote routing protocol.

These metrics were evaluated using different scenarios, as described in the next subsection.

4.2. Scenarios

We decided to use two circular topology scenario to evaluate our work. The choice of such a simple scenario is motivated by the fact that our focus is on the performance of the HTR-Lite itself, since the approach has a negligible impact on the other components of the network.

For the first scenario, we configured one HTR-lite surrounded by a group of HTR-OR. We then vary the number of HTR-OR surrounding the HTR-lite (Figure 3). The objective of this scenario is to illustrate the viability of using the DRR protocol regarding the number of nodes available to share routing tables.

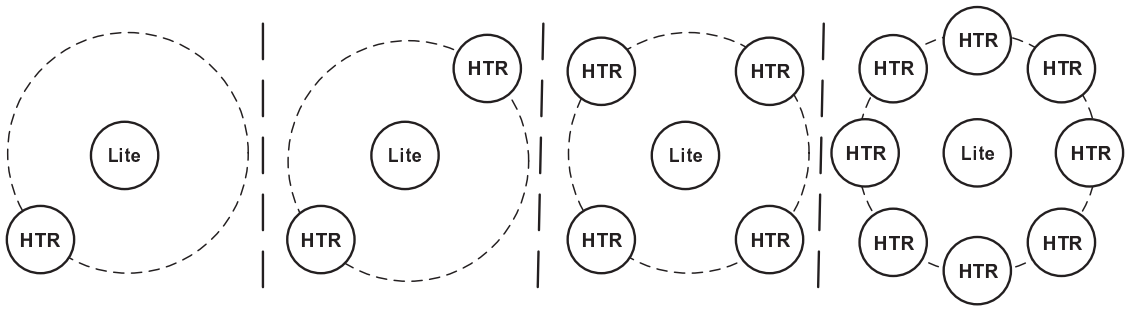

Figure 3. Topology of the first scenario

The second scenario is the inverse of the first one. In other words, the second scenario is configured with a HTR-OR surrounded by a group of HTR-Lite (Scenario 2). This scenario is important to detect the impact and scalability of the solution by detecting the number of HTR-lite nodes that overloads or decrease significantly the efficiency of the HTR-OR node. This scenario is illustrated in the Figure 4.

For both scenarios, in addition to varying the number of surrounding nodes, the mobility mode of the nodes (static or mobile) of the edge can be configured. These scenarios and metrics were implemented in NS-3 simulator which made it possible to arrange a circular topology to run the experiments a hundred times, for each scenarios. These results of the experiments executed with the presented scenarios are analyzed and discussed in the next section.

5. Results and Discussion

As described in Section 4, we decided to evaluate our proposal against different network topologies and performance metrics. This section presents our finds and discusses the

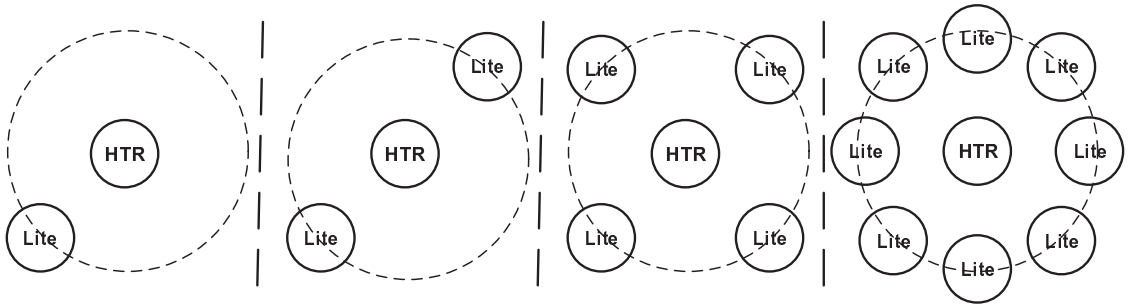

Figure 4. Topology of the second scenario

applicability of the approach giving the results.

We first present the results of the two metrics (Routing Delay and Message Overhead) for the scenario with one HTR-lite surrounded by a group of HTR-OR (Scenario 1). Next, and similarly, we present the results in the case where a HTR-OR is surrounded by a group of HTR-Lite (Scenario 2).

In order to present the results of the Message Overhead metric, we used a marked line chart with two data series. The data series represents the static and dynamic topologies. The x-axis presents the number of surrounding nodes (HTR-OR in case of Scenario 1 and HTR-Lite in case of Scenario 2). The y-axis shows the Message Overhead Reduction (MOR), which represents the percentage of reduction in the overhead when using the DRR protocol. More specifically, we have $MOR = 1 - \frac{O_{ddr}}{O_{htr}}$, where O_{ddr} is the global routing control messages overhead when using the DDR and O_{htr} is the global routing control messages overhead when using only the HTR.

On the other hand, to show the results of Routing Delay metric it was necessary to use two kind of graphs. The first one is a dispersion chart, which represents a grouping routing delay values of each scenario during one simulation timeline. This chart helps to visualize the behavior of the metric during the lifetime of a Lite node in the network. The second chart used to illustrate the routing delay results was a boxplot chart, which shows the minimum, maximum and average value of routing delay metric in each scenario. This representation goes to show the most limiting values achieved by this metric in each simulated scenario.

5.1. Scenario 1

Figure 5 illustrate the results for the Message Overhead metric of Scenario 1 when varying the number of surrounding nodes (HOT-OR) from one to 64.

As shown in Figure 5, the scenario with just two nodes in the network (one HTR-Lite and one HTR-OR) the Message Overhead Reduction was close to 30% for the static mobility model and slightly above that when the HTR-OR can freely move. The bandwidth save or reduction in the number of control messages can be observed with up to four or 16 HTR-OR nodes for the static and mobile configuration respectively. As previously explained, the DDR does not send the Hello or TC messages, which explains the overall reduction of the control sent in the network up to the above commented points. Since the wireless medium is shared amongst the participant nodes, the reduced number of messages sent over the network means a potentially better overall network performance.

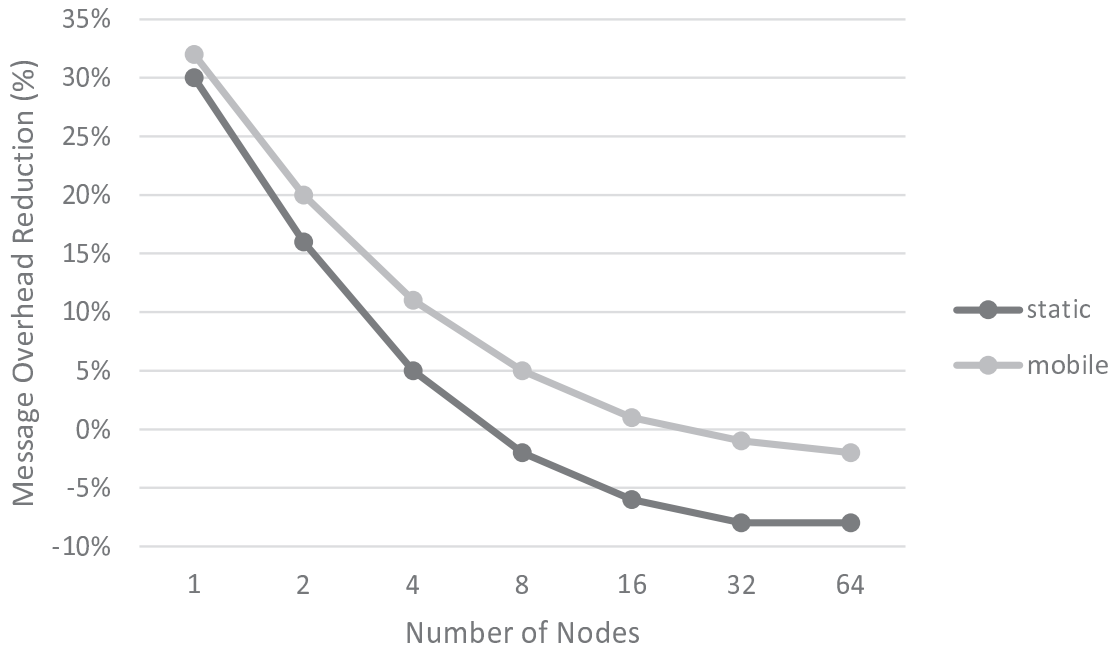

Figure 5. Overhead of messages of scenario 1

As the number of HTR-OR grows larger though, the DDR actually increases the total number of control messages generated in the network. We have to keep in mind, however, that this is observed only because we are increasing the number of HTR-OR while maintaining a single HTR-Lite. In other words, while every new additional HTR-OR have to deal with some additional control messages required by the DDR, there is only one node saving resources. Moreover, this increase would only be perceived in the neighborhood of the HTR-Lite node. Finally, this additional overhead generated in the neighborhood for larger network stabilizes between 32 and 64 nodes.

The Figure 6 shows the results of the Routing Delay metric for the static configuration of our Scenario 1. More specifically, the figure illustrates the observed pattern for a single node in different network densities where we took the sequential readings of the routing delay for node one in a single simulation and varied the number of HTR-OR nodes.

During our preliminary analysis we found out unexpected moments where the delay were much larger then the usual collected data. We first thought that the mobility of the nodes could be causing this larger delays, but in the scenario with fixed nodes we could observe the same pattern. Next, we thought that some missing packets were causing these high delays. We found out, however, that even when everything went smoothly with our routing protocol, there would be instances of large delays.

As can be observed in the figure, independently of the network size, the time series of the routing delay of the node presented cyclical peaks. It turned out that the lower communication layers were causing this unexpected behavior. More precisely, the translation process between the network layer addresses into link layer addresses performed by the Address Resolution Protocol (ARP) was the culprit. The entries in the ARP cache con-

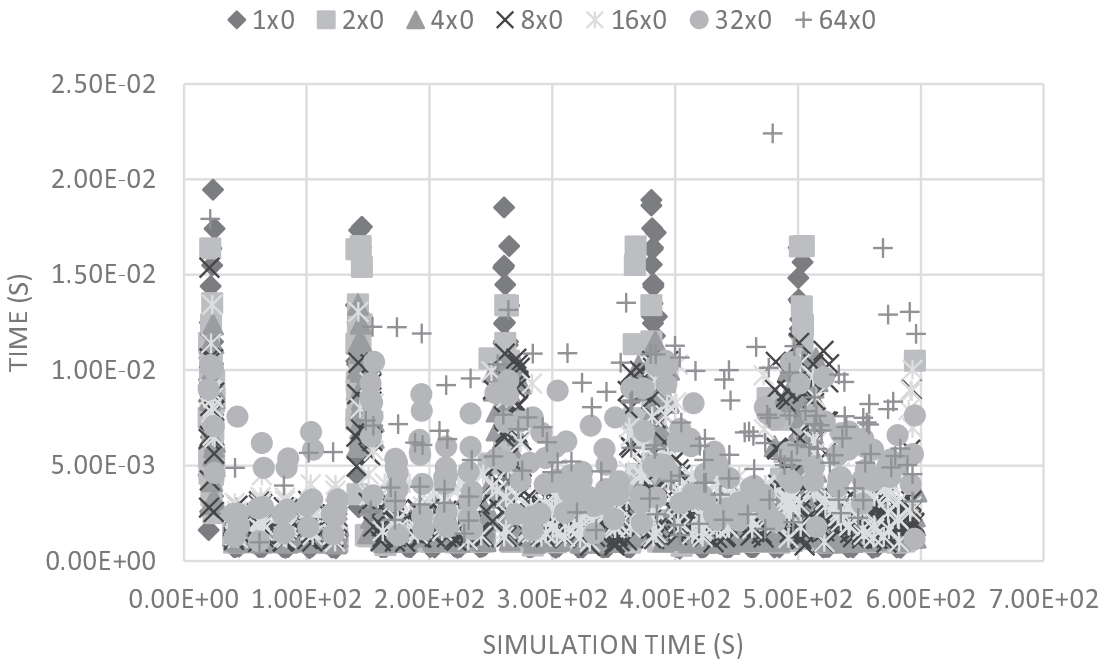

Figure 6. Routing delay of static scenario 1

taining the map between the IPv4 addresses and MAC addresses were expiring or absent, thus requiring an ARP request and reply messages prior to using the IPv4 address.

Giving the specific behavior observed for the static configuration and shown in Figure 6, we decide to verify if the pattern would remain the same when the nodes are configured to freely move. As shown in Figure 7, the same pattern can be observed early on the simulation, but when the nodes are farther apart, the behavior becomes a bit more chaotic. This behavior is to be expected due to the randomness introduced by the mobility of the nodes.

Overall, we can argue that the DDR protocol behaved quite well under the stressed circumstances created for the Scenario 1. It reduced the network overhead in the vicinity for low-density networks, while not greatly increasing the overhead for highly density networks. Moreover, we have to remember that we are potentially including nodes in the network that otherwise would not be able to participate.

5.2. Scenario 2

As previously presented, the Scenario 2 constitutes the inverse topology of the Scenario 1, with a central HTR-OR surrounded by a varying number of HTR-Lite nodes. Figure 8 illustrate the results for the Message Overhead metric of Scenario 2 when varying the number of surrounding nodes (HOT-Lite) from one to 64.

Contrary to what was observed in Figure 5, the Message Overhead Reduction increases as the network topology becomes denser. If in the previous scenario the ratio between HTR-Lite and HTR-OR decreases as the network becomes denser, causing a decrease in the Message Overhead Reduction; the reverse is observed for Scenario 2. In other words, the number of HTR-Lite nodes are increasing relatively to the single HTR-

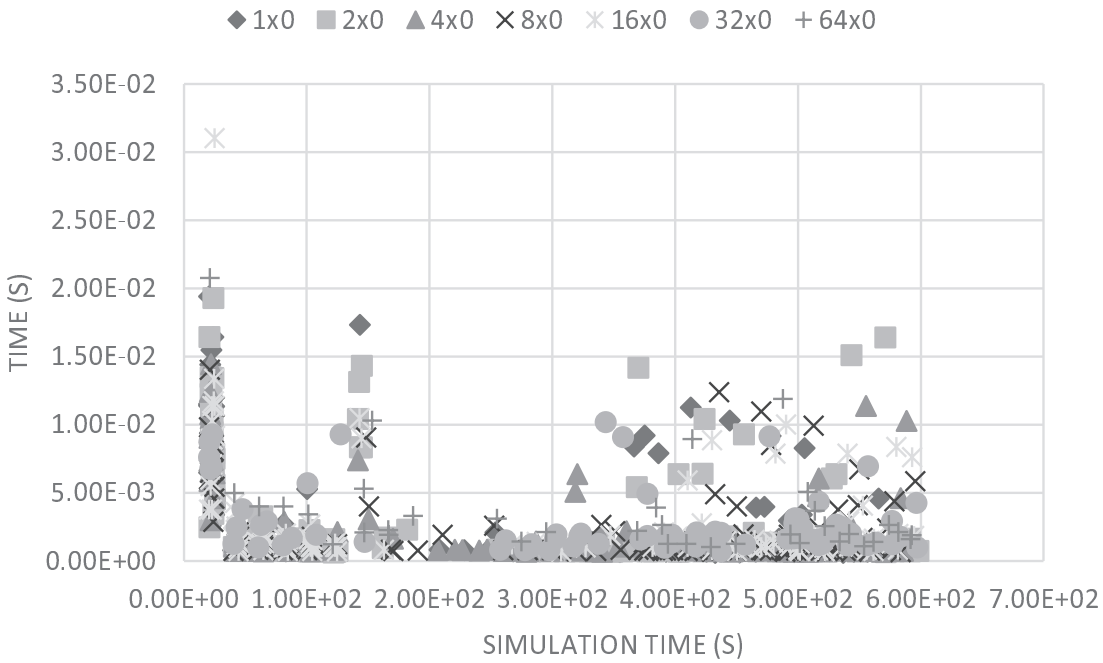

Figure 7. Routing delay of mobile scenario 1

OR and, thus, since the HTR-Lite does not send Hello or TC messages, it was to be expected the reduction of the number of control messages in the overall.

For the second scenario, we were also interested in investigating the unusual pattern (pikes) identified during our experiments. Once more, Figure 9 illustrates a behavior of a single node in different network densities where we took the sequential readings of the routing delay for such node. As shown in the figure, the same pattern can be observed, but it is clearer. The reason for this more deterministic behavior can be explained due to the fact that the HTR-Lite sends messages in a more expected and cyclic behavior.

In the same setup but with mobile nodes (Figure 10) the results are similar to what we found in Scenario 1. Once more, it is to be expected, since the the nodes may become out of reach and latter join again in the same cell, thus presenting a more random behavior.

Our results in both Scenario 1 and Scenario 2 show that our approach does note incur in significant impact on the network. It may be important to note that in all scenarios and configurations, the routing delay did not go above three milliseconds. Such value may not be practical in real environment, giving the additional time required by software and hardware related routines, but comparing with the values collected by the standard approach in the same simulated environment it proves to be satisfactory. Both Routing Delay and Network Overhead are maintained in moderate levels throughout the experiments, while we are able to ensure that a new class of restricted nodes become part of the network.

6. Background

Offloading is the ability to delegate the obligation over some task from one entity to another. It is usually employed to either free one processor that is already fully loaded or to

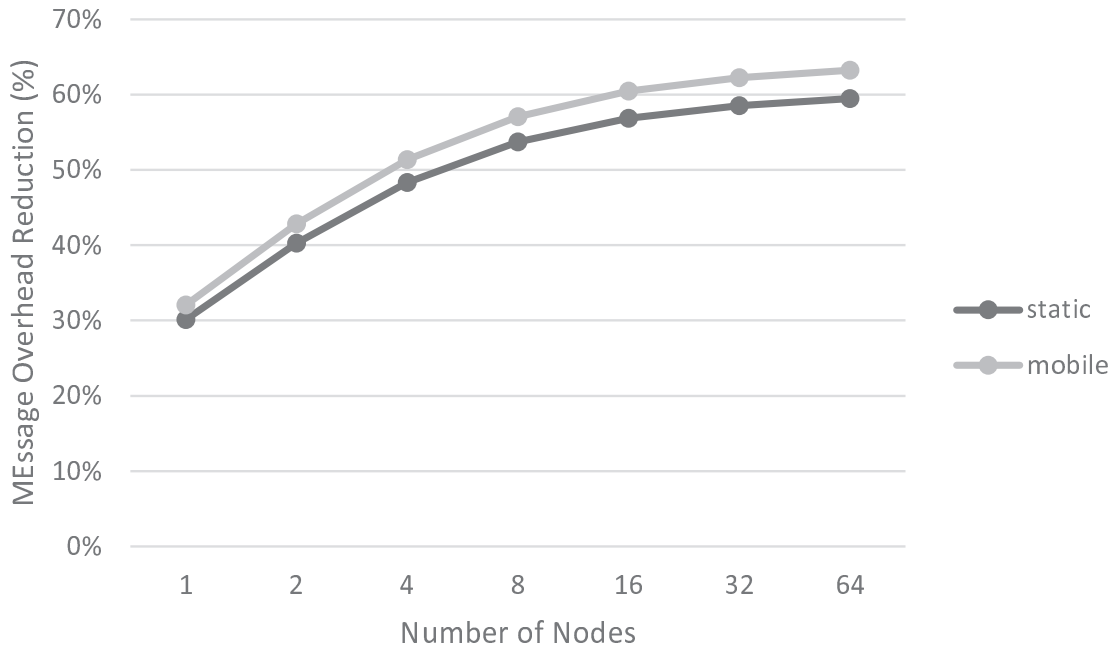

Figure 8. Overhead of messages of scenario 2

export some task from less capable devices to more powerful ones. It is possible to offload data traffic, applications, processing (among others) to any device or service available to handle such tasks. In this paper, we propose the offloading of routing decisions in ad hoc networks of mobile phones and tablets to servers.

Since with the increasing of the Internet infrastructure routing has become a costly solution for routers, several solutions have been proposed for wired networks. The Path Computation Element (PCE) [Farrel et al. 2006] was proposed in 2006, a solution that provides centralized constraint-based path computation for large, multi-domain networks. Following, solutions offloading routing to the virtual devices in clouds were proposed [Wei et al. 2008][Zhu et al. 2008][Karaoglu and Yuksel 2013].Finally, with the advent of Software Defined Networks (SDN) [Gupta et al. 2014] offloading solutions quickly began to appear taking advantage of the separation of control plane and data plane. For example. RouteFlow, uses OpenFlow-based SDN to provide routing services through a single controller.

The world of mobile networks is even more restricted since mobile devices are often not capable of handling some processes, because of their limited resources such as batteries, processing power, storage and bandwidth capacity. In order to save resources from restricted devices, several solutions have been proposed, not only specific for routing, but to several other applications [Li et al. 2001][Chen et al. 2004]. Following the evolution of wired networks solutions, SDN emerged in mobile scenarios with solutions that allow control plane functions to be separated from the devices.

In 2006, a offloading technique for H.264 video encoder was proposed [Zhao et al. 2006] because video processing applications are very resource consuming. They modularize the H.264 video encoder and offload some modules or the whole appli-

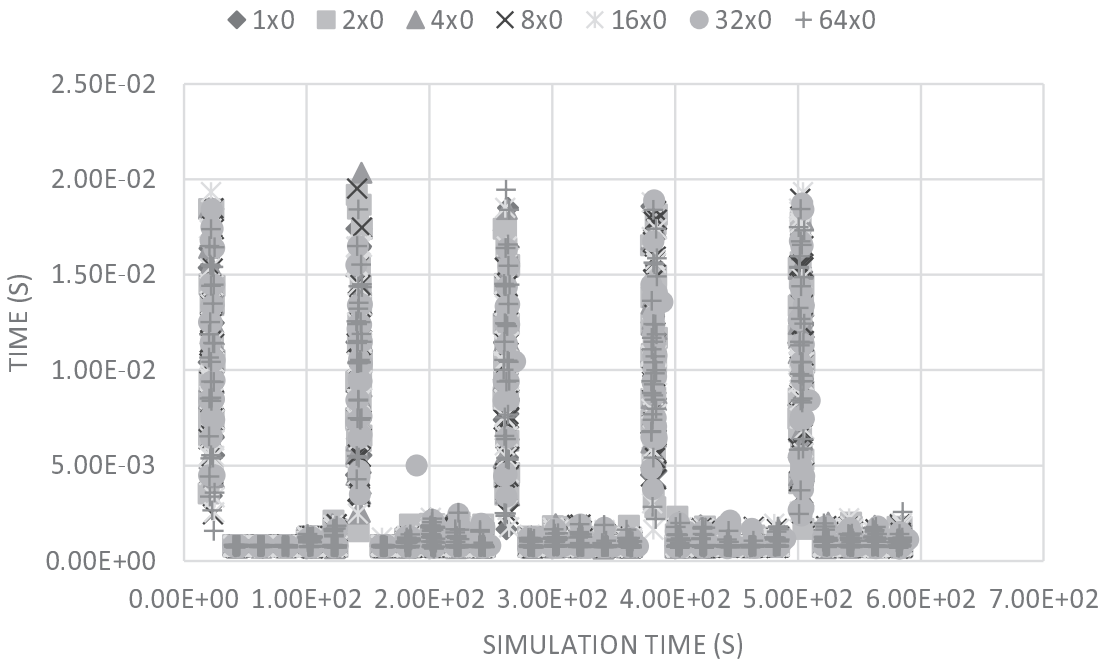

Figure 9. Routing delay of static scenario 2

cation to a nearby server. Results are shown in terms of energy saving, proving that nodes that offload video processing save energy. Later in 2009, the Stanford University released the OpenRoads [Yap et al. 2009], an open-source platform for wireless innovations based on OpenFlow. Specifically for offloading routing there is a solution that uses OpenFlow in wireless mesh networks [Dely et al. 2011], this paper proposes a centralized solution to provide routing decisions for all mesh nodes, the controller manages several networking functions, including handling mobility (handover).

Using the concepts of Offloading we propose DRR, a distributed strategy for offloading routing decisions of specific restricted devices, the HTR-lite nodes, to other devices. To the best of our knowledge, all the previously proposed solutions differ from ours, since ours is focused on distributed support to routing. Others, more capable, devices participating on the network assume routing decisions for the HTR-lite nodes.

7. Conclusion and Future Work

In this paper, we have presented a solution to the problem of routing table computation in heterogeneous ad hoc networks composed of special devices with limited resources. Our solution is based on the ability to offload the task of routing computation from restricted devices to more powerful ones.

The solution is successful in saving the resources such as processing power and network bandwidth. Even though we did not include a specific metric for processing power, since the simulator used does not provide the feature, we are offloading the routing table computation, which is a very demanding task. The experiments proved the bandwidth save at the special restricted nodes and the low routing delay resulting of the offloading process. Overall, the results of our evaluations were positive and confirm the

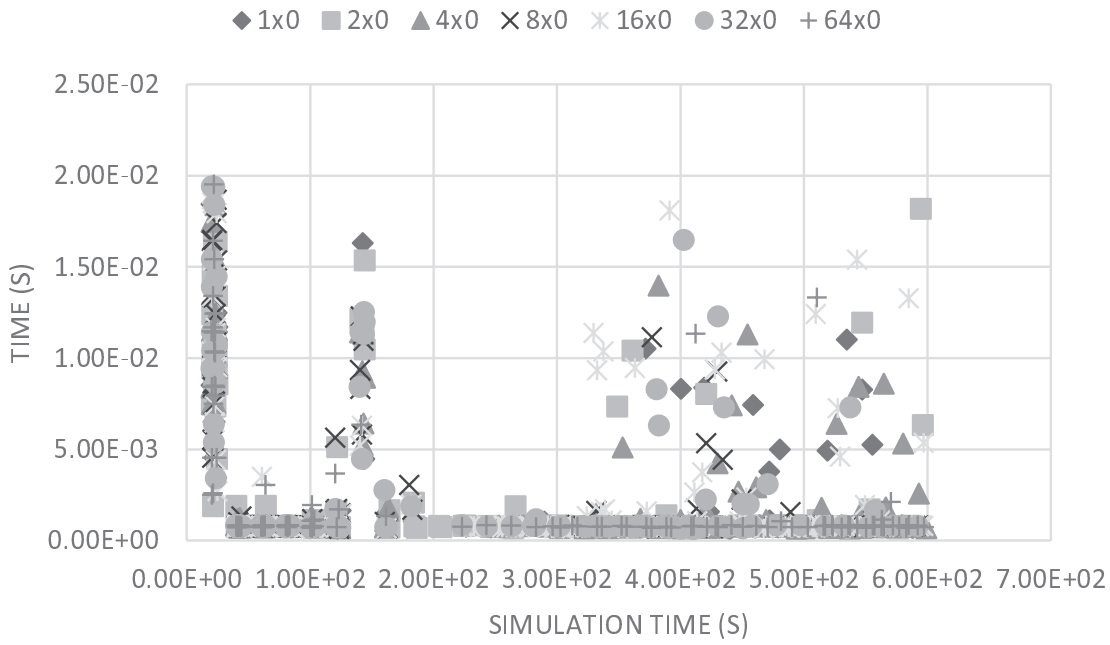

Figure 10. Routing delay of mobile scenario 2

potential applicability of the protocol.

We have also identified some limitations in our works, which are working to improve for a next version of the protocol. Currently, all non-restricted HTR nodes function as outsourced routers, which may not be the desired case. A centralized remote routing entity helped by proxies could be a better solution in certain scenarios.

We believe that by providing a routing offloading mechanism for ad hoc networks we may be doing a important step towards the definition of a software defined ad hoc network.

References

Chen, G., Kang, B.-T., Kandemir, M., Vijaykrishnan, N., Irwin, M., and Chandramouli, R. (2004). Studying energy trade offs in offloading computation/compilation in java-enabled mobile devices. *Parallel and Distributed Systems, IEEE Transactions on*, 15(9):795–809.

Clausen, T. and Jacquet, P. (2003). Rfc 3626. *Optimized link state routing protocol (OLSR)*.

Dely, P., Kassler, A., and Bayer, N. (2011). Openflow for wireless mesh networks. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE.

Farrel, A., Vasseur, J.-P., and Ash, J. (2006). A Path Computation Element (PCE)-Based Architecture. RFC 4655 (Informational).

Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E. (2014). Sdx: A software defined internet exchange. *Proceedings of the ACM SIGCOMM 2014 conference*. To Appear.

Karaoglu, H. and Yuksel, M. (2013). Offloading routing complexity to the cloud(s). In *Communications Workshops (ICC), 2013 IEEE International Conference on*, pages 1367–1371.

Li, Z., Wang, C., and Xu, R. (2001). Computation offloading to save energy on handheld devices: A partition scheme. In *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, CASES '01, pages 238–246, New York, NY, USA. ACM.

Pei, G., Gerla, M., and Hong, X. (2000). Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of the 1st ACM International Symposium on Mobile Ad Hoc Networking & Computing*, MobiHoc '00, pages 11–18, Piscataway, NJ, USA. IEEE Press.

Souto, E., Aschoff, R., Lima Junior, J., Melo, R., Sadok, D., and Kelner, J. (2012). Htr: A framework for interconnecting wireless heterogeneous devices. In *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pages 645–649.

Wei, Y., Wang, J., and Wang, C. (2008). Bandwidth guaranteed multi-path routing as a service over a virtual network. In *Intelligent Networks and Intelligent Systems, 2008. ICINIS '08. First International Conference on*, pages 221–224.

Yap, K.-K., Kobayashi, M., Underhill, D., Seetharaman, S., Kazemian, P., and McKeown, N. (2009). The Stanford OpenRoads Deployment. pages 59 – 66, Beijing, China.

Younis, M., Youssef, M., and Arisha, K. (2002). Energy-aware routing in cluster-based sensor networks. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE International Symposium on*, pages 129–136.

Zhao, X., Tao, P., Yang, S., and Kong, F. (2006). Computation offloading for h.264 video encoder on mobile devices. In *Computational Engineering in Systems Applications, IMACS Multiconference on*, volume 2, pages 1426–1430.

Zhu, Y., Zhang-Shen, R., Rangarajan, S., and Rexford, J. (2008). Cabernet: Connectivity architecture for better network services. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 64:1–64:6, New York, NY, USA. ACM.