Hierarchy-Based Monitoring of Vehicular Delay-Tolerant Networks*

Ewerton M. Salvador
Daniel F. Macedo
José Marcos Nogueira
Universidade Federal de Minas Gerais
Computer Science Department
Belo Horizonte, MG – Brazil
{ewerton,damacedo,jmarcos}@dcc.ufmg.br

Virgil Del Duca Almeida
Instituto Federal de Minas Gerais
Betim, MG – Brazil
virgil.almeida@ifmg.edu.br

Lisandro Z. Granville
Universidade Federal do Rio Grande do Sul
Institute of Informatics
Porto Alegre, RS – Brazil
granville@inf.ufrgs.br

***Abstract*—Vehicular Ad Hoc Networks (VANETs) are mobile networks that extend over vast areas and have intense node mobility. These characteristics lead to frequent delays and disruptions. A solution is to employ the Delay Tolerant Network (DTN) paradigm. However, the frequent disruptions as well as the delay and reliability constraints of certain VANET applications hinder the employment of both conventional and DTN-based management architectures. This paper tackles monitoring, one of the tasks of network management. We describe a hierarchical architecture that copes with near real-time as well as non real-time monitoring tasks. The proposed solution is evaluated using simulations, where we measure the delay and delivery rates of the monitoring data. The results show that the proposed solution reduces the delivery delay and increases the chances that a notification will be delivered on time to its destination.**

I. Introduction

In the past decades, automotive systems experienced great improvements. Several sub-systems evolved into computer-based systems, and now these systems demand inter-vehicle communication for tasks such as improved collision avoidance or real-time traffic engineering. Vehicular Ad Hoc Networks (VANETs) are mobile ad hoc networks designed to provide communication among vehicles, and between vehicles and fixed equipment along roads [1]. We believe that vehicle-to-vehicle (V2V) communication is the preferred form of communication in VANETs, in order to avoid the unnecessary round-trips to the infrastructure, and also to reduce the reliance on the already crowded cellular and ISM bands.

Traditional networking approaches should not be applied to VANETs, since those networks suffer from frequent topology changes and node disconnections. This occurs because of the intense and high speed mobility of vehicles, together with the vast area of the network. Thus, the network may be sparse and subject to connection disruptions. The Delay Tolerant Network (DTN) paradigm [1] provides a solutions to those problems, enabling communications in environments with frequent disconnections and long transmission delays. Despite being originally conceived for the Interplanetary Internet, DTN is increasingly being applied in urban environments for opportunistic communication, such as vehicular networks [2]. A VANET employing the DTN paradigm is often referred as Vehicular Delay-Tolerant Network (VDTN) [3].

Vehicles carrying different types of devices and running different kinds of applications are constantly joining and leaving a VDTN. Because of this, VDTNs have a strong need to be properly managed in order to make sure that these devices and applications are performing as intended, even in face of such dynamic environment. However, existing DTN management architectures are not suitable for many monitoring needs of VANETs, since a number of applications, such as the safety-related applications, require monitoring operations with very strict delay constraints in order to ensure timely response for detected problems. Thus, the DTN management architectures should be adapted to vehicular scenarios.

We propose a hierarchical architecture for monitoring VDTNs. The proposed solution reduces the delivery delay in connected groups of vehicles and increases the chances that a notification will be delivered on time to its destination. Thus, this paper presents the following contributions: (i) the definition of a hierarchical organization for VDTN monitoring; and (ii) the proposal and evaluation of group detection and leader election algorithms for VDTNs.

The remainder of this paper is organized as follows. Section II presents the related work. The architecture of the proposed solution is introduced in Section III. Section IV describes the algorithms used in the architecture. Section V details the methods used for evaluating our proposals. Simulation results are discussed in Section VI. Finally, conclusions and future work are presented in Section VII.

II. Related Work

Birrane and Cole [4] investigated the DTN management problem using a system engineering approach. They proposed a mechanism that takes into consideration the needs of a network and proposed guidelines for the development of a scalable network management system for DTNs. Although the insights provided in this work are valuable to anyone designing management tools for DTNs, the authors do not propose finished solutions for the DTN management problem.

* The authors would like to thank CNPq, CAPES, and FAPEMIG, Brazilian research agencies, for their financial support in this work.

Torgerson [5] presented a suite of tools that provide network visualization, performance monitoring, and node control software for devices using the Interplanetary Overlay Network (ION) DTN implementation. This suite complements the DTN Management Protocol (DTNMP) [6]. The limitation of this work is the fact that its proposals are strongly tied to the deep space communication scenario, which is a type of network that tends to have a more static topology in comparison with highly dynamic vehicular networks.

Nobre *et al.* [7] proposed the Opportunistic Management Contact Estimator (OMCE) for estimating the appropriate time to execute future management tasks. Results showed that it is possible to improve the performance of monitoring tasks through the usage of the authors' proposals. However this work relies on another overlay network, P2P, on top of the DTN. Unfortunately the authors did not analyze the computational costs of a P2P overlay over DTN.

Ferreira *et al.* [3] developed an SNMP-based management solution for VDTNs. Their solution supports load-related information collection, which was demonstrated on a testbed. Their work, however, encapsulates SNMP messages in bundles, without worrying about mechanisms for controlling the delays involved in such transmissions.

III. Proposed Monitoring Architecture

Monitoring DTNs deals with unbounded delays and unreliable links. Frequent link disruptions and long delays affect agents' ability to timely notify network managers of the current network state. A manager might receive a notification message several minutes, or even hours, after the occurrence of an event that may be critical to the correct operation of the network. Even worse, the manager may not receive such notifications at all. This affects the ability of a management system to make timely decisions based on the current status of the network, which may lead to incorrect decisions.

VDTNs are particularly sensitive to delays. The communication requirements of VDTNs can be organized into two classes: non real-time applications, and (near) real-time applications. Non real-time applications, such as traffic engineering, logging of failures, passengers and stops, are a perfect fit for the DTN paradigm. However, VDTNs must also support real-time or near real-time applications, such as collision avoidance and hazard alerts, which cannot rely on the unbounded delivery delays of the DTN paradigm. Thus, the specificities of VDTN require adaptations to the existing DTN management approaches.

The proposed monitoring architecture exploits *local communication* opportunities. Those opportunities happen when one or more nodes are within the transmission range of one another, as opposed to *remote communication*, which happens when one or more nodes have to make use of DTN protocols to communicate with distant nodes. To implement local communication, VDTN relies on a hierarchical organization, composed by a *top-level manager* (TLM), a group of *mid-level (local) managers* (MLMs) that are under the rule of the top-level manager, and finally *agents* that are monitored by either types of managers. MLMs also enforce management decisions delegated by the TLM.

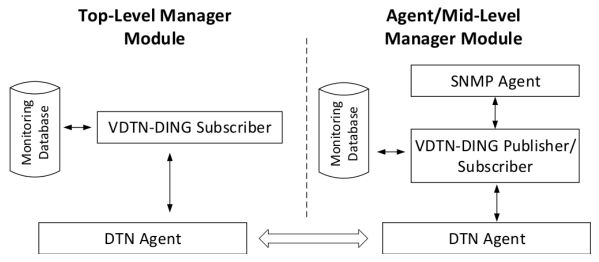

Fig. 1. Top-Level Manager and Agent/Mid-Level Manager architecture

An important benefit of hierarchical monitoring in VDTN is that messages are redirected to a manager node within local communication range, as opposed to sending monitoring messages to a distant manager. This results in monitoring messages getting to a manager quicker, allowing for a faster reaction to network events. A MLM also improves network reliability, since messages do not need to traverse many hops to reach a manager. Finally, MLMs are able to concentrate monitoring information of their managed local nodes in order to take contextual information into account whenever the need for making autonomous decisions arises.

This hierarchical approach is often referred to as *management by delegation* [8]. Our contribution lies in the application of this approach to VDTNs, which require algorithms and protocols to dynamically maintain communities of nodes, elect leaders, and detect node disconnections. The next subsections present those elements.

A. Top-Level Manager

The *Top-Level Manager (TLM)* is a static node that retrieves delay insensitive information from the entire VDTN, and it is responsible for all the management tasks that are not delegated to the mid-level managers. Figure 1 (left-hand side) shows the main components of a *TLM*. The *VDTN-DING Subscriber* requests information to agents and stores the received data in a *Monitoring Database*. The *VDTN-DING Subscriber* component interacts with a *DTN Agent*, using the DTN Bundle Protocol whenever remote communication is necessary.

B. Agent/Mid-Level Manager

Any vehicle can act as a mid-level manager or agent. As such, vehicles carry both software components. The architecture of the *Agent* is shown in Figure 1 (right-hand side). The *VDTN-DING Publisher* component handles subscription requests of network managers and sends monitoring information acquired through an *SNMP Agent* to the managers. Those entities also have a *VDTN-DING Subscriber* module similar to that found in the *Top-Level Manager*, which is activated when the vehicle is a *Mid-Level Manager* (MLM). The monitoring data is sent to the manager that requested such data, and it is also stored in a local *Monitoring Database* for redundancy or for future accesses.

IV. Dealing with disconnections and disruptions

The proposed solution defines a virtual management topology that improves reliability and bounded delivery delay of monitoring messages. This is accomplished by the definition of stable groups, using the group detection and local manager selection algorithms described in the following sub-sections.

A. Publisher/Subscriber Model

TCP/IP management employs Simple Network Management Protocol (SNMP), which is a polling based, "chatty", protocol. The transmission of many requests and responses is not suitable in DTNs, due to the long times for a round trip as well as the high probability of packet losses.

Thus, DTN management employs the publisher/subscriber model. In this model, the *subscriber* informs which data it wishes to receive. *Publishers* generate data, which is sent to the subscribed nodes. This operation reduces the overhead, since it requires one subscription for the reception of an arbitrary number of responses. The Diagnostic Interplanetary Network Gateway (DING) protocol [9] is a monitoring algorithm for DTNs based on publisher/subscriber. Our architecture employs DING for local and remote monitoring.

B. Group Detection Algorithm

Monitored nodes periodically execute the group detection procedure detailed in Algorithm 1. Using DING, every monitored node observes its current 1-hop neighbors, checking if those neighbors are reachable for a minimum amount of time ($time\_threshold$). This threshold is configurable, so the stability of the group can be tuned to the needs of each application. If the connection time is above the time threshold, the neighbor is kept in the local group. Otherwise, the neighbor is disconnected from the group. All members of the group are from a 1-hop distance from each other, so the communication within the group do not require the usage of routing algorithms for dealing with multi-hop transmission. Also, the overhead of routing algorithms in local groups might lead to the increase of the average communication delay, which can be harmful to delay-sensitive applications.

Algorithm 1 Group detection algorithm

```
1: procedure GroupDetection
2:     foreach $n$ in $neighbors\_list$ do
3:         if connection_time($n$) $\geq$ $time\_threshold$ then
4:             group.add($n$);
5:     foreach $n$ in $group$ do
6:         if connection_time($n$) $<$ $time\_threshold$ then
7:             group.remove($n$);
```

C. MLM Election

The MLM election procedure is executed whenever a local group is created or the MLM leaves its group. This procedure is described in Algorithm 2.

The first step of this process is every node announcing to the group whether they are a leaf (it is connected to only one neighbor) or a non-leaf (connected to more than a neighbor). Next, when the *get_neighbors* function is called, each node of the group sends its address and information about their current connections to the others (the *get_neighbors* function of Algorithm 2). Then each node defines which neighbor is more suitable to become a MLM using the *max_conn_score* function. This function attributes one point to all non-leaf neighbors and two points to all leaf neighbors, prioritizing the "most central" nodes within the group as candidates to become the local MLM. If there is a tie, the nodes select as MLM the candidate with the smallest address.

Once the new MLM is determined, all nodes of the group will issue a subscribe request message to the manager, and therefore send the subsequent monitoring messages to it until the node leaves the group or a new election process is started.

Algorithm 2 MLM election algorithm

```
1: procedure MLMElection
2:     start_election();
3:     $neighbors$ $\leftarrow$ get_neighbors();
4:     $most\_connected$ $\leftarrow$ max_conn_score($neighbors$);
5:     $node\_manager$ $\leftarrow$ min($most\_connected$);
6:     subscribe_request($node\_manager$);
```

V. Evaluation Setup

This work employs The Opportunistic Network Environment (ONE) simulator [10], a well established simulator in the VDTN community [11].

A. Node Setup

The simulations evaluate how the management proposals perform in a VDTN by monitoring two applications.

The first is a safety application called *Collision Avoidance*. It uses sensors in vehicles to monitor the surroundings. Beacon messages are sent every 100ms, containing data such as the vehicles' position, speed, and direction, used for the prevention of accidents. This application demands high data rates and low response times (hundreds of milliseconds), and in case of any problem affecting this application the network must react as quick as possible, so a collision between vehicles does not happen because of faulty software behavior.

The second application is an event logging system for the management of fleets. This is a typical application in fleets of vehicles, used to ensure safety and operational efficiency, emitting alerts of broken parts, unauthorized stops, or vehicles exceeding the maximum speed. This is a non real-time and low data rate application, which accepts delays in the order of hours or a few days. In our simulations, this application reports data concerning distances traveled and registered speeds for each vehicle, which is sent to a central management system in remote reports issued every 30 minutes.

Vehicles also run a monitoring module, called the *Agent/Mid-Level Manager Module*. When the **Agent Module** is employed, the application detects faulty messages issued by other vehicles near them running the Collision Avoidance application described above, in order to notify such faults to one of the network managers using monitoring notification messages. If the vehicle is a MLM, then it also runs the functions described in Section III.

The proposed monitored solution, as well as the baseline, are described below.

Hierarchical solution: The network topology consists of one TLM at the center of the simulation area, the MLMs and the monitored vehicles. The MLMs are elected using the algorithms described previously. Monitoring notification messages are issued to MLMs, using local communication, whenever a faulty message of the Collision Avoidance application is detected. The ability of MLMs to gather data from all the vehicles inside a group enables these MLMs to use this context information to better approach whatever problem is detected (e.g., MLMs can make distinction between isolated problems and problems that are affecting the group, partially or entirely). Also, MLMs aggregate the remote reports issued by the event logging system of the vehicles within the local group in order to forward to the TLM a single remote report. This reduces the overhead of the VDTN.

Simple VDTN monitoring: A Central Manager at the center of the simulation area monitors the vehicles. This solution employs DING [9]. All monitoring messages issued by the logging system are sent directly to the TLM.

Vehicles communicate using IEEE 802.11p, with a communication range of 300 meters. We present results for both Epidemic and Spray and Wait routing algorithms. The simulation time was one hour for all experiments, and the threshold for group detection (Algorithm 2) was set to 3 seconds.

B. Simulation setup

We implement the *commission fault model* to simulate corrupted messages, which can compromise the operation of safety applications. A commission fault occurs when an entity sends to another entity a message that should not have been sent (i.e., a message with invalid content, according to the rules of the system being used) [12]. The contents of the messages can be corrupted by a faulty sensor in a vehicle, a transmission error or interference. A faulty message can be detected by the receiver, but it cannot be corrected. Once a faulty message is detected, the receiver issues a monitoring notification message to a network manager, so this manager can take action to approach the detected problem. In our simulation we considered an arbitrary hostile communication scenario where messages have a failure probability of 10%, in order to produce higher numbers of fault notification messages and better evaluate the limits of our proposals.

In order to minimize the impact of routing techniques on the results, we employed the Epidemic routing algorithm with an oracle, so that messages are delivered instantly. In this variant, a message reaches its destination with the lowest possible time, since the size of the messages are not taken into account during transmissions in the Epidemic with Oracle, providing optimal results for transmission delays and delivery rates to be used in this study as a reference. Moreover, we also employed the Spray and Wait routing algorithm to the same scenarios simulated with the epidemic algorithm, in order to evaluate the performance of our proposal and the baseline in a more realistic environment, where the constraints of limited transmission speed and limited buffer size applies. The Spray and Wait was employed in the binary mode, and the initial number of copies for a message is 15% of the number of vehicles in each scenario, as considered in previous works [13].

Two types of vehicle mobility were evaluated: real mobility traces and a synthetic mobility model. The former assesses the proposed solutions in a more realistic scenario, while the latter was chosen to measure the impact of node density on the performance of the proposals.

Mobility traces: The traces are based on the movement of buses in the Seattle metropolitan area [14]. The measurements encompass the movement of 1200 vehicles between October and December 2001, over an area of $5100km^2$. The traces were sanitized as follows. All vehicles with sampling interval inferior to 90s were removed in order to avoid vehicles stopped due to malfunctions in their collection devices. After sanitization, we generated 4 sub-traces, with 108 up to 127 vehicles, and duration of one hour. The total simulation area is $1769km^2$ (29x61km), presenting zones with sparse and dense vehicle distributions.

Synthetic mobility model: We selected the Shortest Path Map Based Movement (SPMBM). It employs Dijkstra's algorithm to find the shortest path between two random points in city map. The map of downtown Helsinki was used in the simulations. To make the network denser, we reduced the simulation area to a rectangle of 4.5x3.4km. The number of nodes VDTN varies from 50 up to 250 vehicles. Nodes move with speeds from 10 up to 50 km/h, and the wait time varies from 0 to 120s, both using a uniform distribution. Each scenario was simulated 15 times.

We measured the notification delay (the amount of time required to perceive the occurrence of a fault) and the delivery rate (which quantifies the reliability of the messages). All results are plotted with a confidence interval of 95%.

VI. Results

This Section presents the simulation results of our work. All figures present a number of abbreviations that can be explained as follows: *BL* stands for the baseline (which is the Simple VDTN Monitoring solution) *Hier* stands for the Hierarchical solution, *Notif* means the monitoring notification messages, *RR* stands for the remote reports issued by the event logging application, *SaW* stands for the Spray and Wait algorithm, and *Orcl* means the Epidemic with Oracle algorithm. A logarithmic scale was used in Figures 2 and 3 due to the skewness of the lowest delay values found in the simulations.

A. Notification delay analysis

1) Trace Mobility: Figure 2 shows that a monitoring notification message takes more than 8 minutes in average to reach its destination in the Simple VDTN Monitoring scenario,

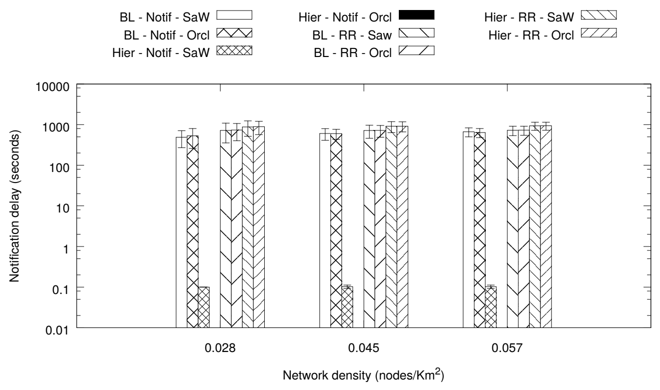

Fig. 2. Notification delay using trace mobility

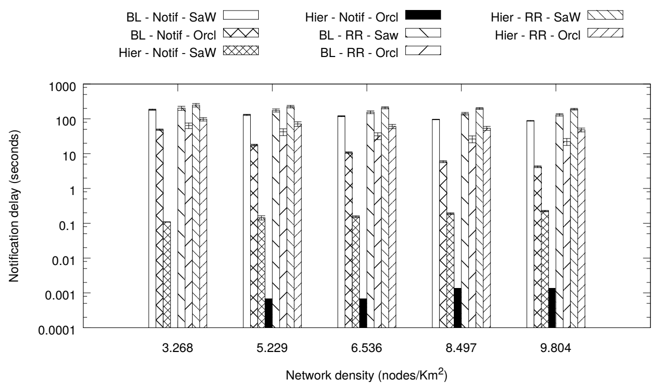

Fig. 3. Notification delay using the SPMBM mobility model

while this average delay is always close to zero for both the Spray and Wait and the Epidemic with Oracle algorithms in our Hierarchical solution. This is expected because the communication occurs within a connected group due to the hierarchical organization of the network. The performance difference in comparison to the optimal scenario (Epidemic with oracle) is less than 8% for monitoring notifications.

A remote report generated by the event logging application takes more than 9 minutes in average to reach its destination in the Simple VDTN Monitoring solution, when the Spray and Wait Algorithm is employed. In the Hierarchical solution this delay is between 30% and 40% higher than the ones observed in the Simple VDTN Monitoring scenario. This difference is due to the aggregation process that occurs in the MLMs, which aims to reduce the number of remote report messages in the VDTN. The difference between the Spray and Wait and the Epidemic with Oracle algorithms is less than 2% for both Simple VDTN Monitoring and Hierarchical scenarios.

2) SPMBM Mobility: Figure 3 show that the higher network density in the scenarios using the SPMBM model reduces the connectivity issues in DTN, making the network closer to a conventional ad hoc wireless network, which makes pure centralized management approaches more viable. Yet, with our Hierarchical solution the monitoring notification delay remains near 0 regardless of the network density and routing protocol, which is quite lower than the delays found in the Simple VDTN Monitoring scenario. Transmission delays of monitoring notification messages in the Simple VDTN Monitoring solution are significantly higher in comparison with the Epidemic with oracle, which is not observed when our Hierarchical solution is used.

The observed average delay in the transmission of remote reports issued by the event logging application, with the Simple VDTN Monitoring solution, is between 2 and 3 minutes in average when the Spray and Wait algorithm is used. As for our Hierarchical solution this delay is between 3 and 4 minutes in average with the Spray and Wait algorithm. The different performance between the two scenarios is explained by the aggregation of reports in the MLMs. The usage of better routing algorithm for disseminating messages in dense networks should narrow the difference between the Spray and Wait algorithm and the Epidemic with Oracle.

B. Notification delivery rate analysis

1) Trace Mobility: Figure 4 shows that in the baseline scenario the average delivery rate for monitoring notification messages was lower than 66% when the Spray and Wait algorithm was used. The difference of the delivery rate between the Spray and Wait and the Epidemic with Oracle is less than 4%. The delivery rate in the Hierarchical solution was constantly 100%, regardless of the employed routing algorithm.

Regarding the remote report messages, the average delivery rate observed in the Simple VDTN Monitoring solution was always inferior to 37% when the Spray and Wait was used, and always inferior to 40% with the Epidemic with Oracle algorithm. With the Hierarchical solution the average delivery rate remained inferior to 30% when the Spray and Wait algorithm was used, and inferior to 32% when the Epidemic with Oracle approach was employed.

2) SPMBM: Figure 5 shows that the delivery rate for monitoring notification messages are always higher than 95% in the Simple VDTN Monitoring Solution and always superior to 99% when our Hierarchical solution is employed, both cases considering the usage of the Spray and Wait algorithm. If the Epidemic with Oracle algorithm is used, the performance is higher than 99.8% in both approaches. The high density of the network explains the high delivery rate.

C. Discussion of the results

Based on these results, we conclude that the proposed hierarchical organization is a better fit for VDTNs, since it significantly reduces the delivery delay and increases the delivery rate for sparse and dense networks. Regarding near real-time applications, the hierarchy allows the effective monitoring of a group of nodes, making it possible to implement those types of applications. At the same time, for non real-time applications, the use of hierarchy reduces the amount of messages in the network due to the aggregation process in the MLMs, at the cost of possibly increasing the communication delay and reducing the delivery rate.

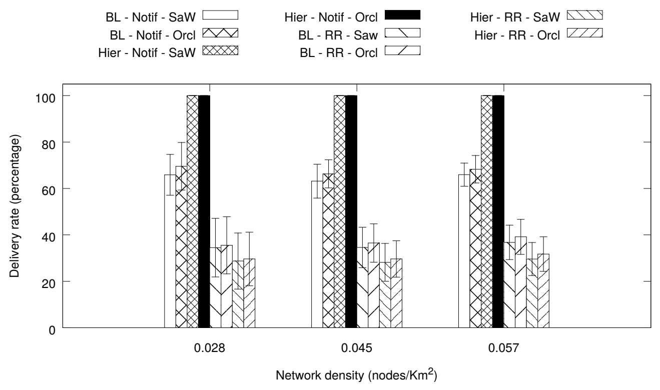

Fig. 4. Delivery rate comparison between the evaluated scenarios using trace mobility

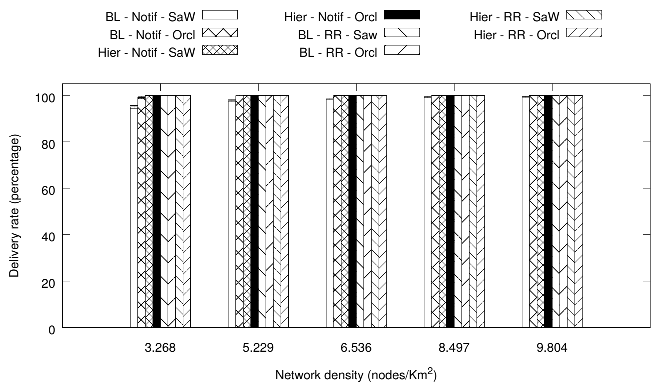

Fig. 5. Delivery rate comparison between the evaluated scenarios using the SPMBM model

VII. Conclusions and Future Work

This paper investigated the monitoring of Vehicular Delay-Tolerant Networks (VDTNs). We use local communication to avoid the high cost of remote communication, so that near real-time applications can be deployed on the network. In order to implement management by delegation, we proposed algorithms to identify stable node groups and to elect managers.

Simulations showed that the proposed architecture enhances the reliability and response time of the management tasks, due to the smaller number of message forwards to reach a manager. This allowed the proposed architecture to cope with near real-time and non real-time monitoring.

As future work, more refined algorithms for the election of leaders and group detection will be proposed. Finally, we will investigate how to perform node configuration in VDTNs.

References

[1] G. Karagiannis, O. Altintas, E. Ekici, G. J. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[2] A. Bujari, C. Palazzi, D. Maggiorini, C. Quadri, and G. Rossi, "A solution for mobile DTN in a real urban scenario," in *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, April 2012, pp. 344–349.

[3] B. F. Ferreira, J. J. Rodrigues, J. a. a. Dias, and J. a. N. Isento, "Man4VDTN A network management solution for vehicular delay-tolerant networks," *Computer Communications*, vol. 39, pp. 3–10, 2014.

[4] E. Birrane and R. G. Cole, "Management of Disruption-Tolerant Networks: A Systems Engineering Approach," in *SpaceOps Conference*, 2010.

[5] J. L. Torgerson, "Network Monitor and Control of Disruption-Tolerant Networks," in *SpaceOps Conference*, 2014.

[6] E. Birrane and V. Ramachandran, "Delay Tolerant Network Management Protocol," draft-irtf-dtnrg-dtnmp-01, Internet Engineering Task Force, 2014.

[7] J. Nobre, P. Duarte, L. Granville, L. Tarouco, and F. Bertinatto, "On using P2P technology to enable opportunistic management in DTNs through statistical estimation," in *IEEE International Conference on Communications (ICC)*, 2014, pp. 3124–3129.

[8] G. Goldszmidt and Y. Yemini, "Distributed management by delegation," in *International Conference on Distributed Computing Systems (ICDCS)*, 1995, pp. 333–340.

[9] A. Papalambrou, A. Voyiatzis, D. Serpanos, and P. Soufrilas, "Monitoring of a DTN2 network," in *Baltic Congress on Future Internet Communications (BCFIC Riga)*, 2011, pp. 116–119.

[10] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *2nd International Conference on Simulation Tools and Techniques*, 2009.

[11] F. Wang, Y. Xu, H. Zhang, Y. Zhang, and L. Zhu, "2FLIP: A Two-Factor Lightweight Privacy Preserving Authentication Scheme for VANET," *Vehicular Technology, IEEE Transactions on*, vol. PP, no. 99, 2015.

[12] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith, "Byzantine Fault Detectors for Solving Consensus," *The Computer Journal*, vol. 46, no. 1, pp. 16–35, Jan. 2003.

[13] V. Soares, J. Rodrigues, J. Dias, and J. Isento, "Performance analysis of routing protocols for vehicular delay-tolerant networks," in *Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on*, Sept 2012, pp. 1–5.

[14] J. G. Jetcheva, Y.-C. Hu, S. PalChaudhuri, A. K. Saha, and D. B. Johnson, "CRAWDAD data set rice/ad_hoc_city (v. 2003-09-11)," Downloaded from http://crawdad.org/rice/ad_hoc_city/, Sep. 2003.