Applying Machine Learning to Reduce Overhead in DTN Vehicular Networks

Lourdes P. Portugal-Poma, Cesar A. C. Marcondes, Hermes Senger, Luciana Arantes

1Departamento de Computação – Universidade Federal de São Carlos (UFSCAR)
Caixa Postal 676 13565–905 São Carlos–SP
2LIP6 – Université de Paris 6 - CNRS - INRIA
4, Place Jussieu 75005, Paris - France
{lourdes.poma,marcondes,hermes}@dc.ufscar.br, luciana.arantes@lip6.fr

Abstract. VANETs benefit from Delay Tolerant Networks (DTNs) routing algorithms when connectivity is intermittent because of the fast movement of vehicles. Multi-copy DTN algorithms spread message copies to increase the delivery probability but increasing network overhead. In this work we apply machine learning algorithms to reduce network overhead by discriminating the worst intermediate nodes for the transmission of copies. The scenario is a VANET of public buses that follow specific routes and schedules. This repetitive behavior creates an opportunity for applying trained classifiers to predict the occurrence of performance-related events. As the main contribution, our method decreases overhead without degrading delivery probability.

1. Introduction

Vehicular Ad Hoc Networks (VANETs) are Mobile Ad Hoc networks (MANETs) of vehicles establishing wireless communications. Like other MANETs, VANETs are networks with dynamic topology where nodes can be servers or clients by executing routing algorithms in distributed environments. In addition, VANETs are a special type of MANETs because of the fast movement of its nodes, which creates intermittent end-to-end connectivity paths. Thus, the maintenance and the construction of routing tables in topology based routing algorithms, becomes harder to realize, specially in proactive algorithms that reconstruct periodically the whole routing table [Abolhasan et al. 2004].

To resolve intermittent end-to-end connectivity, VANETs benefit from Store Carry and Forward (SCF) paradigm. This paradigm is the foundation of the Delay Tolerant Networks (DTNs) [Vasilakos et al. 2011], being an approach to deliver messages in computer networks without establishing connectivity paths between source and target nodes. In the SCF, messages are forwarded hop-by-hop in opportunistic contacts between nodes, i.e., when pairs of nodes establish a wireless communication between them. When there is no other neighbor node to receive a message, it remains stored and waiting for a next opportunity to transmit. For this reason, there is a trade-off between: (i) storage resources versus packet loss ratio, and (ii) delay versus delivery probability. However, in challenging scenarios (such as sparse networks), nodes could deliver more messages using the SCF paradigm than using topology based routing algorithms, but...
at cost of a considerable increase in delay due to the additional message storage delay [Franck and Gil-Castineira 2007].

In the SCF paradigm, an important issue is the selection of the next intermediate node to whom transmit a message (next hop). This decision is made in every node and based on local information. A multi-copy solution consists in creating and spreading message copies to increase the delivery probability in applications where the target node’s location is unknown. However, the flooding of copies could result in network overhead and, depending on the network density and the spatial node distribution, it can induce strong performance degradation.

In this work we apply machine learning algorithms to reduce network overhead by discriminating the worst intermediate nodes for the transmission of message copies in multi-copy DTN routing. The experiment scenario is composed of one VANET of public buses that follow specific routes and schedules. This repetitive behavior allows trained classifiers to predict the occurrence of performance-related events. In summary, our work makes the following contributions:

- We proposed a method that uses a classifier in order to support the decisions of a routing algorithm and to adapt its behavior according to local conditions and the quality of the intermediate nodes. Our method decreases network overhead without degrading the probability of message delivery;
- We evaluated our method with realistic scale traces (254 vehicles in average). The latter revealed zones of highly varied density of vehicles. This behavior is not usually captured by mobility synthetic models;
- We compared the connectivity characteristics between synthetic mobility models and the real traces to highlight the influence of the choice of a model in the overhead evaluation. The result shows that synthetic mobility models can render some bias in the overhead evaluation;
- We conducted performance evaluation experiments of our method with two algorithms whose design has different purposes: Spray and Wait (SaW) [Spyropoulos et al. 2005] is an algorithm with low overhead and good performance in scenarios where nodes move freely, and Epidemic [Vahdat and Becker 2000] is a good algorithm to be used in sparse and low density networks.

The remainder of this paper is organized as follows: section 2 presents different approaches for reducing the overhead in multi-copy routing. The section also presents a discussion on how classifiers are applied in MANETs. Our proposed method is described in section 3 while, in section 4, we highlight the main characteristics observed in scenarios with real traces. In section 5, we discuss some experimental results related to the evaluation of the proposed method with both SaW and Epidemic routing algorithms. Finally, in section 6 we conclude the paper and present suggestions for future work.

2. Related Work

Our Machine Learning Classification (+C) approach is related to works of different areas. We start by describing from naive DTN routing algorithms, like Epidemic, up to decision tree based classifiers, which we use in our proposed method.
**Naive multi-copy routing algorithms.** Epidemic is the fundamental multi-copy, high overhead, routing algorithm [Vahdat and Becker 2000]. If applied without any control, nodes transmit in every opportunity to others that do not have a copy, like an epidemic disease. Despite of that, the overhead can be small and delay / delivery performance better in low density scenarios and sparse networks.

**Naive algorithms with flood control.** In order to reduce network overhead, basic flooding control measures can be applied. Some approaches aim at limiting the time to live of messages (TTL), the number of hops, or tuning the message space storage. However, implementing these approaches require intense parameter tuning, not always trivial to implement due to DTNs’ dynamic behavior. Some other approaches are inspired in the analogy of vaccines, where once an “anti-packet” is received it induces the node to reject or delete a copy of an specific message [Haas and Small 2006]. Finally, Spray And Wait (SaW) [Spyropoulos et al. 2005] is another solution which sets a maximum number of copies per original message that can exist in the entire network. This value is decreased at every copy transmission. When it equals to one, nodes only transmit directly to the target node. In summary, the goal of all the above solutions is to reduce network overhead without any other consideration about other performance issues. Consequently, the performance of these algorithms is quite dependent on the mobility model and nodes’ spatial distribution.

**Statistical and context information algorithms.** Other solutions exploit statistical and context information to select the best intermediate nodes. Nodes use utility functions to qualify intermediate nodes and to decide if they should transmit a copy or not. Commonly, mobility statistics and connectivity information is used since the latter has an influence on performance. The knowledge of network connectivity properties and the qualification of nodes allows the design of more effective routing protocols for specific environments [Vendramin et al. 2012]. For example, the Lobby index ($L_{\text{indx}}$) [Korn et al. 2009] is a two-hops connectivity metric suggested by [Pallis et al. 2009] to qualify nodes. It detects dense zones where highly connected nodes are located. To compute the $L_{\text{indx}}$ in a network, every neighbor from a node $x$ reports to $x$ its connectivity degree. Then, a node $x$ calculates the maximum integer $l$ such that $x$ has at least $l$ neighbors with a value of degree greater than or equal to $l$. For $L_{\text{indx}}$ to be applied, an additional and exclusive channel for beaconing transmission is needed.

**Node encounter probabilities algorithms.** ProPhet [Lindgren et al. 2003] and MaxProp [Burgess et al. 2006] are algorithms that employ utility functions to take advantage of information about node encounter probabilities. However, besides mobility information, nodes can collect other kind of information to find network behavior patterns. For instance, the time interval of the day could be related with the network load. Thus, machine learning algorithms were used with the purpose of detecting convenient opportunities for message transmission according to the learned transmission patterns. Basically, machine learning based approaches consist in collecting data of transmissions, processing the data conveniently and use a trained classifier to support future transmission decisions.

**Classification algorithms.** Classification consists in attributing objects to one of many categories or classes. Formally, according to [Tan et al. 2005], classification is the task of learning an objective function $f$ that maps each set of attributes to one of the prede-
fined classes. Thus, a classifier constructs a classification model from a dataset (training
data set) using a machine learning algorithm. Given a set of \( n \) attributes, \( \{a, b, \ldots, n\} \),
and a set of \( m \) classes, \( \{C_1, C_2, \ldots, C_m\} \), a training dataset is a set of tuples where each
tuple corresponds to:

\[
\begin{align*}
\text{Attribute values} & \quad \rightarrow \quad C_i \\
\text{1} \leq i \leq m
\end{align*}
\]

Using the classification model, the trained classifier assigns a class to an unan-
notated tuple which has only a set of attribute values, \( \{a_{\text{value}}, b_{\text{value}}, \ldots, n_{\text{value}}\} \). Also,
the classifier must have the capacity of generalization, this mean that the classifier must
assign a class even for unannotated tuples that were not seen in the training dataset.

**Routing and broadcasting using classifiers.** There are still few approaches
that suggest the use of classifiers to support routing and dissemination decisions.
[Colagrosso 2005] use the Naive Bayes classifier to be trained with information of past
broadcast retransmissions for message dissemination. Afterward, the classifiers pre-
dict if a new broadcast attempt will be successful or not. Classifiers were also used to
find mobility patterns from people movement to construct predictive models to forecast:
“where”, “how long will stay” and “who people will meet” people in different locations
[Vu et al. 2011]. [Ahmed and Kanhere 2010] exploit the repetitive behavior of public
buses movement to develop an extensible Framework and a single-copy DTN solution.
As a matter of fact, this is the first work in DTN routing that use classifiers to improve
routing decision, and it is the most related work to our proposal. It consist in keep a
resident Naive Bayes classifier that uses information of end-to-end past deliveries (thus,
acknowledgment propagation is required) to determine the most likely nodes that will de-
liver a message to a specific target node. This scheme outperforms a single-copy version
of MaxProp. However, when compared with Epidemic, its performance is lower in de-
livery probability and delay. The reason for that is that in scenarios of low to moderate
density Epidemic flooding has usually the best performance in this metrics, but with high
network overhead.

**Decision tree classification algorithms.** The C4.5 decision tree [Quinlan 1993]
is a machine learning algorithm used for classification. The C4.5 training phase generates
a decision tree using training data. The decision tree is composed by: (i) internal nodes
that represent conditional operations on the attributes and (ii) terminal nodes associated
to class attributes. Initially, all the tuples are attributed to the root node of the decision
tree. The next step consists in finding a conditional operation based on one of the tuples’
attributes. This operation divides the root node into two or more internal nodes. The
new internal nodes contain a subset of the root node’s tuples. This division operation is
repeated recursively until the generated nodes are pure. The purity of a node is a measure
of the homogeneity of its tuples’ class distribution. In terminal nodes, the most frequent
class determines the class attribute. To choose the attribute and the conditional operation
applied to it, all the possible options are evaluated and the one that maximizes the pu-
riety of the resulted subsets is used. With all in mind, the merit of the C4.5 comes from
[Tan et al. 2005]: (i) it makes a automatic selection of the best attributes to create the
structure of the decision tree and (ii) the decision tree is a comprehensible form to visu-
alize the attributes’s influence in the classification of tuples. J4.8 is the latest and slightly
improved version, called C4.5 revision 8, implemented in Weka [Witten et al. 2011]. J4.8 was the last public version of this family of algorithms before the commercial C5.0 was released. The full complexity of decision tree induction is \( O(\alpha N (\lg N)) \), where \( \alpha \) is the number of attributes and \( N \) is the number of instances [Witten et al. 2011].

3. Applying Data Classification for Improving Routing Decisions

Our proposal is to apply data classification in order to predict which nodes are the most suitable to behave as intermediate forwarders according to temporal and local connectivity conditions. Our method is composed of three phases, as illustrated in Figure 1. During the first phase, data about message forwarding is collected by all the network nodes. In the second phase, all data collected from the nodes is combined into a single training dataset at some machine with enough capacity (such as a server in a datacenter or cloud provider). This could be done in the end of the workday. Thus, the collected data could be stored and processed incrementally day after day to provide richer information to retrain the classifier. As a result of this, the trained classifier produced by J4.8, i.e., the decision tree built during the classifier training, can be represented using a few dozens of bytes which are then transmitted for every mobile routing node. In the last phase of our method, each mobile node uses the received classifier to predict if neighboring nodes are good forwarders and decide whether to transmit a message copy or not. The classification of each message is very fast, being computed in time \( O(h) \), where \( h \) is the height of the decision tree.

![Figure 1. Phases of the method: (1) Data collection; (2) The classifier is trained with the dataset; (3) Routing is executed based on the classifier’s knowledge](image)

3.1. Data collection

The goal of the data collection phase is to store information about message transmissions. This data will be used to train the classifier. Our method organizes the collected data as tuples. Each tuple corresponds to data from a single message replication, where message replication is defined as follows:

**Definition 1** A message replication is an event composed by the reception of a copy of a message \( m \) by an intermediate node \( n \) and the successful transmission (by \( n \)) of a new copy of \( m \) to another intermediate node \( n' \).

Each tuple \( t = \{\eta, \theta, \gamma, l, \tau, \delta\} \) is composed by the following message replication attributes: the **Node ID** (\( \eta \)) contains the identifier of the node that received the message copy; **Region code** (\( \theta \)) contains the location of the node at the instant when it received...
the message copy; the **Reception time** \((\gamma)\) contains a timestamp of the instant when the message copy was received; **Lobby index** \((l)\) measures the neighborhood density at two-hops in the moment when the message copy was received; **Time passed** \((\tau)\) represents the time interval between the reception time and the successful transmission of the message copy the next node; **Distance** \((\delta)\) refers to the distance between the location when the message copy was received and the location when the message copy was transmitted to the next node.

The attributes collected for each message replication are illustrated in Figure 2. To compute the region code attribute we divided the total movement area in squares of \(1 km \times 1 km\). For the reception time attribute, time was discretized into 600s intervals. Our data collection strategy does not require end-to-end acknowledgment. Instead, our method is able to predict suitable intermediate nodes based only on local conditions, without considering the relation between intermediate and target nodes. This is important because end-to-end acknowledgement is a costly procedure in VANETs.

![Figure 2. Data collection. It starts when \(B\) receive a copy of message \(m\) and it ends when \(B\) retransmits a copy of \(m\) to another node.](image)

This set of attributes does not include the class attribute. So, all the set of tuples must be processed to give each tuple its correspond class attribute.

### 3.1.1. Generating the class attribute

In order to train the classifier, our method annotates the tuples generated in the data collection phase with class labels. The class labels \(C_i\) are computed from the time passed \(\tau\) and distance \(\delta\) attributes. First, for each tuple, we computed the fraction:

\[
r = \frac{\delta}{\tau}
\]

Then, our method uses the value \(r\) to group tuples with close values in the same class. For this, we defined threshold values so that each group contains approximately the same number of tuples. Thus, we can obtain \(m\) groups that determine a set of \(m\) class attributes, \(C = \{C_1, \cdots, C_m\}\).

The correspondence between the tuple of collected data and an annotated tuple generated after processing the entire dataset can be described as follows:
Once the annotation process is complete, the set of resulting tuples constitutes the training dataset. Then, for any given two tuples \( t \) and \( t' \) with different class labels, the following relation holds:

\[
\text{If } \quad t \rightarrow C_i \quad \text{and} \quad t' \rightarrow C_{i+k} \text{ then } \frac{\delta}{\tau} < \frac{\delta'}{\tau'} \quad \text{where} \quad k \in \mathbb{Z}^+
\]  

### 3.2. Training the classifier

The training phase consists in generating a decision tree using the J4.8 classifier. The classifier is trained and the resulting decision tree is sent for each node. A segment of the resulting decision tree is illustrated in Figure 3. The J4.8 chose the node identifier to be placed at the root level of the decision tree, and the region code attribute for the second level of the decision tree. Thus, a trained classifier can predict the class of a specific node, according to the region where the node will transmit. The other levels of the decision tree are quite different depending of the information collected by every node.

![Figure 3. Example of a part of a decision tree obtained after the training phase.](image)

**The leaf nodes correspond to class attributes \( C_i \).**

### 3.3. Routing using the classifier

The trained classifier is used to decide if a message copy should be forwarded to a neighbor node. Before transmitting a message copy, the transmitting node requests to its neighbor nodes their lobby index \( l \) and region code \( \theta \) values. These values are combined to the node ID \( \eta \) and reception time \( \gamma \) to generate a tuple. The classifier takes this tuple as input to predict a class \( C_i \) for the intermediate node. Once the class is predicted, the node is able to decide if it will transmit a copy or not.

According to Equation 4, the maximum values of \( i \) in \( C_i \) reflect nodes that are able to retransmit a copy at greater distance or in lower times, i.e., nodes more capable of spreading copies when compared with nodes of classes with lower values of \( i \). However, given that we applied this solution to different algorithms, namely Epidemic and Spray and Wait (SaW) in its binary mode, the values of \( C_i \) qualify nodes differently depending on the algorithm. Figure 4 shows the relation of quality \( (Q_i) \), i.e., convenient nodes to be intermediates, and the value of class \( (C_i) \) for each algorithm.
Our approach to reduce overhead for the two routing algorithms can be described as follows.

**Epidemic.** We set a retransmission probability vector $V_i$, where $V_i$ is the probability to transmit copies to a node of class $C_i$. Thus, the overhead is reduced by transmitting copies with low probability to nodes with a high spreading capacity (high classes values), and with great probability to low class nodes, i.e., those nodes in conditions of difficult spreading of copies, such as nodes in region of low density, that require Epidemic to deliver a message.

**SaW.** Since a maximum value of copies ($L$) for a message is defined, the only way to decrease network overhead is to deliver messages using fewer copies than $L$. To this end, those high class nodes are chosen as intermediate nodes while low class nodes should not take part in the spreading process. The rationale for this choice is that high class nodes have higher capacity to retransmit copies than the low class nodes. The reason for such a difference is both the vehicle’s route characteristics and the regions’ density where vehicles transit. Hence, if high class nodes can find intermediate nodes easily, they will probably also find target nodes easily due to vehicles concentration in determinate zones. If this assumption holds, it is possible to use fewer copies than $L$ and, therefore, deliver a message before applying direct transmission.

### 4. Scenario and Simulation Environment

Evaluation experiments were conducted using the version 1.5.1 RC1 of The Opportunistic Networking Environment (ONE) Simulator [Keränen et al. 2010]. The ONE is a movement and networking simulator for evaluation of DTN routing and applications. In order to include the classification functionality, we used the API of the machine learning software WEKA [Frank et al. 2009]. For evaluating the proposed method, we used the ONE simulator to reproduce real traces of buses. The latter include location information in short time intervals of buses in the King County, Seattle, Washington [Jetcheva et al. 2003], and were obtained from CRAWDAD\(^1\). However, we considered only a subset of these traces which correspond to 144 km\(^2\) of area and movements from 7a.m to 12p.m. (five hours of simulation). Additionally, for the comparison of real traces versus synthetic mobility models, we applied two mobility models included in the ONE: (i) Random Waypoint (RWP) and (ii) Shortest path map based movement (SPMB). We, thus, performed two group of simulations: (i) simulations for the comparison and characterization of scenarios with real traces and synthetic mobility models, and (ii) simulations for the evaluation of

\(^1\)http://crawdad.cs.dartmouth.edu/
the proposed method, using only scenarios with real traces. The simulation parameters are summarized in Table 1 of the Section 4.2.

4.1. Comparison of scenarios with real traces and mobility models

Aiming at highlighting the movement characteristics of the real traces scenario, we have compared its connectivity properties and performance with those scenarios with synthetic mobility models. To this end, we have chosen the (i) Random Waypoint (RWP), where nodes move randomly in the area, and (ii) Shortest path map based movement (SPMB), where nodes move randomly from a selected point to another in the area using the shortest path and according to a map topology. The results of such comparison allow us to conclude that:

Density and Connectivity. Differently from synthetic models, the real traces scenario presents varied number of nodes which depends on the time of the day (see Figure 5). Thereby, the scenario of real traces has two types of nodes: (i) vehicles that always stay inside the area, and (ii) vehicles that enter and leave the area. Therefore, the former corresponds to vehicles that can be selected as source and target nodes when messages are created while the latter includes vehicles that should only participate in copy spreading. In average, in the traces scenario, the area has 254 vehicles and this value was used to set the total number of nodes in synthetic model scenarios. Due to the movement restrictions, the traces scenario presents particular connectivity properties. It presents groups of nodes with higher connectivity than the other scenarios. Thus, the lengths of the diameter and the average shortest path are greater than the obtained values from synthetic model scenarios (see Figure 8). Also, these measures reveal that there are regions in the city where groups of nodes are highly connected even in hours of less transit activity. Figure 7 shows that nodes are highly concentrated in specific zones.

Contact Time. Contact time seems to characterize how much each model restricts node movements. Higher restricted movement models present higher values of contact time (see Figure 6). Real traces scenario has higher values because vehicular movement is ruled by traffic lights, roads and points of interest that lead to vehicular congestion. On the one hand, high contact time may help to the spreading of copies. On the other hand, in zones of high concentration of nodes, it might increase network overhead.

Performance. We evaluated the performance by using three measures: (i) the delivery probability, (ii) the average delay to deliver a message and (iii) the network
overhead, i.e., \( \frac{\text{# of copies successfully transmitted}}{\text{# of messages delivered}} \). Given that vehicles follow specific routes, movements in the real traces scenario are more restricted than in the SPMB scenario. Thus, in the former, the message delivery probability between two randomly chosen nodes is lower than in the latter. On the other hand, since real traces scenario has highly connected groups of nodes, the delivered messages could be delivered with lower delays (see Figure 9). However, such a fast propagation of copies not only reduces delivery time but also increases network overhead (see Figure 10). Consequently, the scenario with real traces is more challenging for message delivery and prone to cause much more network overhead.

### 4.2. Simulation parameters

The basic parameters used in our experiments are shown in Table 1. The time for message creation is the interval of time after one message is created in the whole network. In synthetic model scenarios, nodes were distributed in groups with different speed ranges.
to approximate the speed distribution to the one that we find in real traces. The proposed method was evaluated using the parameters of the real traces scenario (see Table 1).

Furthermore, additional parameters were setting in order to perform the first and the third phases of our method, i.e, data collection and routing using the classifier phases. It is described as follows:

### Table 1. Simulation parameters.

<table>
<thead>
<tr>
<th></th>
<th>Real Traces</th>
<th>Synthetic models(RWP / SPMB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission rate</td>
<td>5Mbps</td>
<td>5Mbps</td>
</tr>
<tr>
<td>Transmission range</td>
<td>250m</td>
<td>250m</td>
</tr>
<tr>
<td>Storage capacity</td>
<td>unlimited</td>
<td>unlimited</td>
</tr>
<tr>
<td>Message size</td>
<td>128KB</td>
<td>128KB</td>
</tr>
<tr>
<td>Time for message creation</td>
<td>≈ 60s</td>
<td>≈ 60s</td>
</tr>
</tbody>
</table>
| Number of nodes          | variable    | Group 1: 71 nodes
                          |             | Group 2: 99 nodes
                          |             | Group 3: 61 nodes
                          |             | Group 4: 23 nodes
                          |             | Total: 254 nodes
| Node speed (m/s)         | 0 to 22     | Group 1: 0 to 2
                          |             | Group 2: 2 to 5
                          |             | Group 3: 5 to 10
                          |             | Group 4: 10 to 22

**Data collection phase.** We used SaW and Epidemic algorithms with their default behavior. For SaW routing, the maximum number of copies per message ($L$) was set to 300. Based on the results of the simulations, we obtained seven classes and three classes for Epidemic and SaW training classifiers respectively. We have chosen this two algorithms because is interesting to reveal how our method performs in scenarios with strict overhead control measures (SaW) and in scenarios of free spreading of copies (Epidemic). Also, in our experiments, SaW had a better perform that more complex algorithms like Prophet, with a network overhead of more than 260 units lower, a average delay of more than 100 seconds below and with similar delivery probability of $≈84\%$.

**Routing phase.** Since the classifiers has been previously trained, the routing algorithms were modified to make routing decisions according to the decision trees. By notation, we distinguish the algorithms with classifier using the standard nomination of the algorithms plus the string “+C”. For “Epidemic+C” we use a retransmission probability vector $V$ for the seven classes obtained, $V = \{1, 1, 0.80, 0.50, 0.20, 0.10, 0\}$ (values were set accordingly to the rationale of Section 3.3), where the element $i$ of $V$ corresponds to the probability of retransmission to nodes of class $C_i$. In the case of “SaW+C”, we exclude the worst intermediate nodes from the copy spreading process, i.e., the nodes of the most inferior class. Hence, “SaW+C” transmits a copy to any node predicted with a different class than $C_1$. 

203
Finally, for the evaluation of the method, we experiment with “Epidemic+C” without any additional overhead control measure and, for “SaW+C” evaluation, we defined six scenarios with different values of \( L = \{5, 10, 30, 50, 100, 150\} \). All of them exploit the same trained classifier.

5. Evaluation on Real Traces

We evaluate our proposed method (+C) using movement traces of buses of Seattle (Washington) reproduced on The ONE simulator. For this evaluation, we performed two groups of simulations as specified in the Section 4.2: (i) one group of two simulations to collect data using the algorithms SaW and Epidemic in their normal behavior, and (ii) another group of simulations using a different subset of messages and the classifier support to make routing decisions. The results in Figure 11 show a performance comparison of original SaW and SaW with classifier support (“SaW+C”).

Figure 11. SaW and “SaW+C” performance comparison. (a) Delivery probability. (b) Network overhead. (c) Avg. delivery delay.

Figure 12. Epidemic and “Epidemic+C” performance comparison. (a) Number of transmissions. (b) Delivery probability. (c) Network overhead. (d) Avg. delivery delay.
As desired, our method improve the performance of the algorithm SaW in delivery probability as well as network overhead. Although the quantitative values are still modest, they are consistent in all scenarios when the number of copies of SaW varies. The delivery delay has also improve since, for each number of copies configured (see Figure 11(c)), the average delay decreases by 231.3, 25.2, 23.5, 24.2, 3.0 and 4.1 seconds respectively.

We also compared Epidemic and “Epidemic+C” as shown in Figure 12. The results were encouraging, “Epidemic+C” reduced significantly the amount of transmissions, and consequently the network overhead, without decreasing the delivery probability. Despite of that, there was a small increment of 29.04s in message delivery delay. This can explained by the way Epidemic floods the networks and thus delay is minimal. But, as we compare horizontally SaW and Epidemic delay (Figure 11(c) vs Figure 12(d)) it is easy to see that replicating less copies has implications in delivery delay.

6. Conclusion and Future Work
To reduce network overhead in multi-copy DTN routing, we applied a machine learning algorithm to exclude nodes predicted as bad intermediates from the copy spreading process. We have applied our classification mechanism (+C) on top of two regular DTN routing algorithms in order to verify the performance improvement that our approach provides under real trace scenarios (Seattle buses). The results confirmed that performance of SaW can be improved with the classifier support (SaW+C). Such gain is due to a limited number of copies distributed more conveniently by transmitting them to more suitable spreading nodes, on regions of moderate concentration of vehicles. We also applied our method on the Epidemic routing algorithm (Epidemic+C) and we have obtained a reduction in network overhead without degrading message delivery probability. Despite of such a promising result, we have observed a slight increase in delivery delay due to the use of fewer copies.

We have also conducted evaluation experiments about the impact of mobility on multi-copy routing. We observed that the network overhead is greater in real traces scenarios than in synthetic movement models. This can be attributed to the fact that real scenarios present non-uniform distribution of nodes, movement restrictions, and high variability density in different regions of the city. The experiments also helped us to model the decision tree parameters.

For future work, we intend to focus on mobility information and use machine learning algorithms to predict where and for how long the buses will stay in certain part of the map. Also, given that the spreading capacity of nodes can be quantified, we can explore machine learning algorithm for regression, such as CART, ANNs or SVMs.

References


