BODYF – A Parameterless Broadcasting Protocol Over Dynamic Forest

P. Ruiz, B. Dorronsoro, D. Khadraoui, P. Bouvry, L. Tardón.

***Abstract*— Mobile Ad hoc Networks (MANETs) are composed of mobile devices which spontaneously communicate each other without any previous existing infrastructure. Thus, the resulting network is highly fluctuating and has a dynamically changing topology. Dealing with ad hoc networks, broadcasting is one of the main operations, since many other applications use it very often. In order to avoid the broadcast storm problem in wireless networks, many researchers work on the design of efficient broadcasting algorithms. In this contribution, we present BODYF, a broadcasting protocol that relys on a tree-based topology. The behavior of this algorithm does not depend on any parameter, what makes it challenging and suitable for dealing with MANETs. We compare here our proposal to DFCN, an efficient broadcasting neighbor topology based protocol, and also to Simple Flooding, the most simple broadcasting technique which does not require any knowledge.**

I. Introduction

Broadcasting constitutes one of the fundamental low-level network operations which serves as the basis of higher level applications, such as routing, in mobile ad hoc networks (MANETs). In MANETs, the limited radio range of the nodes, as well as node mobility, cause the unreachability of some nodes at a given time and a highly fluctuating topology. This is the reason why some researchers are focusing on optimizing the behavior of these algorithms, e.g., maximizing the number of nodes reached, and minimizing both the time required and the network overload [1].

Recently, there exists also a tendency in the ad hoc networks field focused on the development of new mobile networks composed of vehicles. In Vehicular Ad-hoc Networks (VANETs), vehicles can communicate each other (Car to Car communication) or with road-side units that allow access to backend systems which provide warnings, traffic information, etc. Vehicle communication is a major research topic, covered by many national and international research projects as CARLINK [2]. Applications promise to make our driving safer, more efficient, and funnier. This includes warning applications, e.g., cars are able to send warning messages to other cars alerting them of a danger ahead, weather and traffic conditions, etc. In VANETs, due to the high speed of the devices (car PCs), the topology of the network is even more dynamic than in MANETs, what difficulties the communication between them.

In MANETs, we can typically differentiate broadcasting protocols into *heuristic-* and *topology-based protocol*. We are interested in *topology-based protocols* which are subcategorized into *neighbor topology based protocol*, *source-tree based protocol* and *cluster-based protocol* [3]. In this paper, we propose a new source-tree based broadcasting algorithm, called Broadcasting Over Dynamic Forest (BODYF). We will compare BODYF to a neighbor topology based protocol, Delayed Flooding with Cumulative Neighborhood (DFCN). In contrast to BODYF, which requires a tree-based topology established in the network, DFCN does not exchange any message for stablishing its topology; it just needs the one hop neighborhood knowledge obtained using the beacons (*hello messages* that devices send for notifying their presence). We will also compare BODYF to Simple Flooding, a well known broadcasting technique which does not require any knowledge about the neighborhood or the topology, and either makes any attempt to reduce the number of forwarded messages.

In addition to the comparison (in terms of coverage, bandwidth and elapsed time) of the behavior of these so different broadcasting protocols, which is already interesting by itself, with this study we can also check, whether the overload caused by the tree-based topology is worthy or not.

The rest of this paper is organized as follows: Section II describes DAGRS, the model for creating topologies used for the creation of our tree, and presents BODYF, the broadcast algorithm over this tree. Section III introduces DFCN the broadcast protocol with one hop neighborhood knowledge. The mobility model and the simulator used are presented in Section IV. After that, the results are shown in Section V and finally, Section VI concludes the paper.

II. Broadcasting Over Dynamic Forest, BODYF

In this section we present BODYF, a broadcasting protocol over a dynamic forest. Although broadcasting using a tree structure is a well known and widely used technique [4], it is typically claimed to be inappropriate for ad hoc networks, being the maintenance of the tree very sensitive to network changes. In this work, we built the tree using DAGRS (dynamicity aware graph relabeling systems) [5], a general model for creating dynamic topologies, and developed the broadcast algorithm on top of this topology.

D. Khadraoui and P. Ruiz work at C.R.P. Henri Tudor, Luxembourg. B. Dorronsoro and P. Bouvry work at the Faculty of Sciences, Technology and Communications, University of Luxembourg. L. Tardón is with the Department of Communication Engineering, University of Málaga, Spain. (Email: {djamel.khadraoui, patricia.ruiz}@tudor.lu {bernabe.dorronsoro, pascal.bouvry}@uni.lu, lorenzo@ic.uma.es).

D. Khadraoui and P. Ruiz acknowledge that this work was partially supported by CELTIC CARLINK project financed by the Ministry of Luxembourg (CARLINK project website: `http://carlink.lcc.uma.es`)

Proceedings of the 2008 High Performance
Computing & Simulation Conference ©ECMS
Waleed W. Smari (Ed.)
ISBN: 978-0-9553018-7-2 / ISBN: 978-0-9553018-6-5 (CD)

A. Dynamicity Aware Graph Relabeling Systems, DAGRS

DAGRS is an extension of Graph Relabeling Systems, GRS [6]. It is a high level abstraction model that can be used to drastically improve the development of self-organized systems. The model supports the design of algorithms in which a computation step only involves direct neighbors and where a device can react to the appearance/disappearance of its neighbors [7]. The main advantage of DAGRS is that it makes the description of algorithms easier to understand and validate.

The network is represented as a graph, where the mobile devices are the set of vertices *(V)*, and the links between them are the edges of the graph, *(E)*. The dynamicity of the network is represented by the fact that both *V* and *E* can change at any time.

It is important to emphasize that DAGRS model does not itself model services or applications, it just models the mechanisms to handle with topology changes and interaction between devices.

A spanning tree of a graph is a connected cycled-free subgraph. In fact, in dynamic networks we should talk about spanning forest, since the network is typically partitioned. In this model we only use one-hop neighbor information, so it is a localized algorithm.

Initially, all devices are labelled *T*, what means they are tree themselves. The algorithm performs on the basis of four rules described after and represented in Figure 1; where *T* represents a node with token, *N* is a device without token, and *Any* means it can be both of them. The numbers on the edges are labels representing the route to the token.

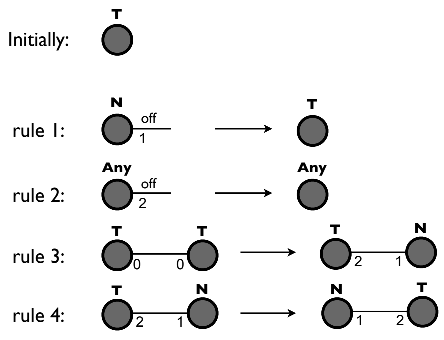

Fig. 1. DAGRS rules for creating spanning forest topologies.

Every tree has only one token, and the possible operations of this token are: circulation, merging, and regeneration. As explained before, every device is initially a tree itself, with only one element. Only two nodes with token can merge. When two nodes with token meet each other, rule 3 allows merging the two trees, deleting one of the tokens (there only can be one token in each tree). The circulation of the token is represented by rule 4. In our implementation, the token explores the tree in a depth-first manner. Both rules 1 and 2 deal with a broken link. In rule 1, the token must be regenerated (label 1 represents the route to the token, so if this link breaks the token is lost), meanwhile in rule 2, the node has nothing to do with the maintainance of the token (label 2 represents a different link from the route to the token). Applying these easy rules to every device, we can build a dynamic tree topology in the network, in a decentralized way.

It is very important to remark that there is only one token per tree in the forest, since it is the way for avoiding cycles. Only two nodes with token can merge, so since there is only one token in the tree, two nodes belonging to the same tree are not able to merge never (it is impossible both of them have the token at the same time).

As we explained above, we are using these simple rules of DAGRS for creating and maintaining the tree, but as we are dealing with high speed mobile environments and also with distributed systems, we exchange some messages between nodes for merging trees and also for circulating the token. DAGRS do not specify how to implement a token. In our protocol, the token is a message. For circulating the token, a device sends this message to one of its neighbors. When a device receives the token, it should check if there is any neighbor with token in its neighborhood just to merge their trees, but this behaviour leads in a high level number of messages interchanged. So we take advantage of the high mobility and use it: only when a change in the one hop neighborhood is detected, the device with token checks if there is any other device nearby with token.

B. The proposed broadcasting protocol, BODYF

BODYF is a broadcasting protocol specifically designed for communication dynamic networks based on spanning tree topologies. We suppose the tree-based topology is already established in the network, maybe for routing or any other necessity [8]. Once we have the tree (or the forest due to the network partitioning), we use it for broadcasting. Therefore, our main goal is not the tree itself, but the design of a new broadcasting protocol that can achieve the best possible coverage and broadcast time at a minimum cost, using the information of the network provided by the tree-based topology.

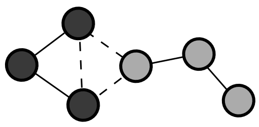

Fig. 2. Possible types of neighbors in a spanning forest algorithm.

Assuming we have the forest, we will distinguish between logical neighbors, which are the ones belonging to the same tree, and potential neighbors, which are those that do not belong to my tree yet but are in communication range, as it is shown in Figure 2. In this picture, devices in same color belong to the same tree. The links between devices are represented as a continuous line if the neighbors belong to the same tree and are connected (logical neighbors), or as a discontinuous line for neighbors either in communication range but which do not belong to the tree (potential neighbors) or belonging to the same tree but not connected in order to avoid cycles.

A pseudocode of our proposed broadcasting algorithm is given in Algorithm 1. We suppose that all the messages have an unique identifier, and when a device receives the same message more than once, it will directly discard it, no processing is done.

When a device wants to spread a message, instead of doing a multicast only to its logical neighbors (the neighbors in the same tree), it will broadcast the information, what supposes the same load for the network, allowing the potential neighbors to also receive the message. When a device receives the message from a logical neighbor, it will forward it if it was not received before, otherwise it is dropped (lines 2-4). But if it was not received from a logical neighbor (i.e., from its own tree), the device will wait until it receives the token (line 5). Once the device receives the token, it will forward the message if and only if, during the time it was waiting for the token, it did not receive the same message again from a neighbor belonging to its tree (lines 6-10). That is the way the message can spread through different trees, trying to avoid the dissemination of the same message more than once in the same tree.

Algorithm 1 Pseudocode of BODYF.

```
Data: m: the incoming broadcast message.
Data: d: the node receiving broadcast message.
Data: s: the node which sent m.
 1: if m is received for the first time then
 2:    if s and d belong to the same tree then
 3:       d → forward m;
 4:    else
 5:       wait until the token is received → d is token;
 6:       if d received m also from its tree then
 7:          d → discard m;
 8:       else
 9:          d → forward m;
10:       end if
11:    end if
12: else
13:    drop m
14: end if
```

III. Delayed Flooding with Cumulative Neighborhood, DFCN

DFCN [9], [10] is an efficient broadcasting protocol specially designed for MANETs. It needs the information about the one hop neighbors provided by the beacons. DFCN requires to embed into the broadcast message a list of all the neighbors of the sender nodes. This broadcasting algorithm focuses on different goals:

1) Minimize network overhead.
2) Provide a protocol dealing with the network density. More precisely, as illustrated in Figure 3, broadcasting in a low density network is difficult because the protocol needs to make a very good use of the mobility to achieve good coverage. When the density increases, the connectivity gets better, although the network may be partitioned. When the network is highly connected, the broadcasting protocol must be bandwidth-efficient in order to minimize the risk of packet collisions.
3) Provide a localized protocol which operates with 1-hop neighborhood information. One challenge is to achieve better performance by using less network information.

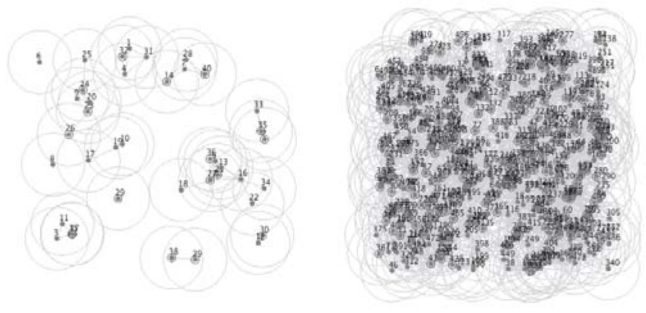

Fig. 3. Node density in MANETs. Picture on the left is a very sparse and partitioned network. The one on the right shows a very dense network.

In this section we explain how DFCN works [9]. First of all, we give a general idea of the problem and, after that, we explain the problem in detail.

When the network is sparse, it is quite difficult to spread a message, hence a node should forward the message as soon as another device is met. This would lead to good results since every re-emission proves to be useful because of the reduced number of meeting points between stations. However, in a dense environment, this strategy would lead to catastrophic results. As a consequence, DFCN sets a random delay (*RAD*) when a node receives a message for the first time (before forwarding it). If the density is low it will forward the message inmediately after a new neighbor is met, but if the density is high, it will wait before resending it (in order to avoid collisions). This perception of the density corresponds directly to its neighborhood and, in DFCN, it is managed with a threshold called *densityThreshold*.

DFCN attaches in the message a list with the neighbors of the sender, *T(m)*. The list *T(m)* is managed as follows: when a node *s* sends a message *m* to its neighbors, it knows that all of them will receive *m* (unless some collisions occur). *T(m)* is set with the neighbors of *s*, *N(s)*. DFCN uses this list to decide whether the received message will be forwarded or not, in terms of the neighbors that already received the message which are in the one hop neighborhood. For that, we set a threshold called *minBenefit* which is formally defined on the *benefit*, computed as the ratio between the neighbors of *s* which do not belong to *T(m)*, and the total neighbors of *s*, *N(s)*, see Equation 1. The higher the benefit, the higher the probability of re-emission.

$$benefit = \frac{|N(s)\text{-}T(m)|}{|N(s)|} \qquad (1)$$

The behavior of DFCN is driven by the following three events:

- the reception of a message, referred to as reactive behavior;
- the expiration of the *RAD* of a message;
- and the detection of a new neighbor, referred to as proactive behavior;

When one of these three events occurs, DFCN reacts by behaving in a specific manner, as described below.

1) Message reception event: if a message *m* is received for the first time, a *RAD* is then assigned to *m*. Otherwise the message is dropped. This behavior corresponds to Algorithm 2. All messages are univocally identified. If a device already received the message, it will be discarded. No processing is done.

Algorithm 2 Behavior of DFCN upon message reception.

```
Data: m: the incoming broadcast message
Data: s: the node which recives m
1: if m is received for the first time then
2:     rad(m) $\leftarrow$ random $\in$ [0, maxRAD];
3: else
4:     s drops m
5: end if
```

2) RAD *expiration event:* when the *RAD* of a message expires, its hosting node computes the ratio of neighbors that did not receive it yet. If the ratio is greater than the threshold *minBenefit*, the message is forwarded, otherwise it is dropped. If the message is emitted, then *T(m)* is set to *N(s)*. Algorithm 3 shows this behavior.

Algorithm 3 Behavior of DFCN when *RAD* expires.

```
Data: m: the message candidate to immediate emission
Data: s: the node that received m
1: benefit $\leftarrow \frac{|N(s)-T(m)|}{|N(s)|}$
2: if benefit$\geq$minBenefit then
3:     T(m) $\leftarrow$ N(s);
4:     s $\rightarrow$ forward m
5: end if
```

3) New neighbor event: each time a node *s* detects a new neighbor, the *RAD* for all messages is set to zero. Messages are hence immediately candidate to re-emission. If *N(s)* is higher than the threshold *densityThreshold* (our network is dense), this behavior is disabled, see Algorithm 4.

DFCN behaves depending on several parameters (e.g., *densityThreshold*, *minBenefit* and *RAD*) which need to be tuned for every different network. As an example, DFCN was optimized in [11] for three particular scenarios: mall, highway and metropolitan area, reporting very different parameter values. This optimization was made according to the coverage, the usage of the network and the broadcasting time, depending on the necessities of each application.

Algorithm 4 Behavior of DFCN when a new neighbor is detected.

```
Data: M(s): the set of messages received
Data: s: node with a new neighbor
1: if $|N(s)|$ < densityThreshold then
2:     for m $\in$M(s) do
3:         rad(m) $\leftarrow$ 0;
4:     end for
5: end if
```

IV. Mobility Model and JANE Simulator

In this work, we evaluate BODYF and compare it to DFCN in a particular environment, Vehicular Ad hoc Networks (VANETs). So, we consider every device is a car, with communication capabilities. Our scenario for the simulations is the city center of Luxembourg, see Figure 4. The simulation area is 1323m x 1863m, what means 2.465 Km^2. The speed of cars oscillates between 30 and 75 km/h.

Many researchers are focusing on the realism of simulations, and are hence, creating their own mobility models to emulate how the protocols would behave in a real scenario [12]. We did the same and developed a mobility model where devices move along streets, stop at crossroads and also overtake each other.

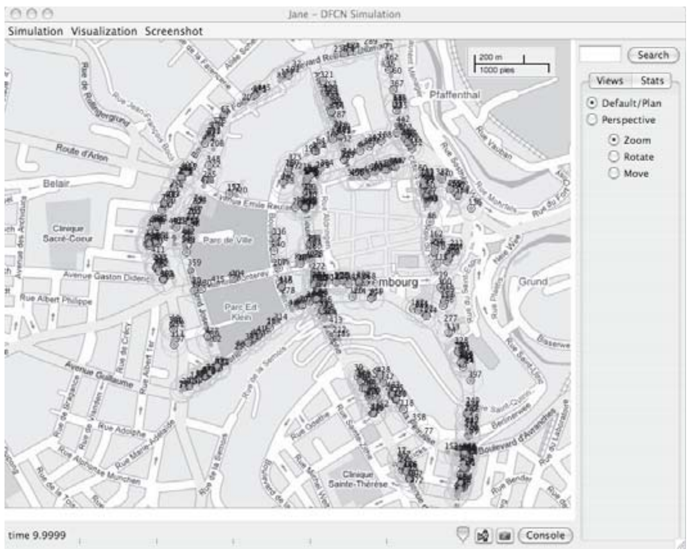

Fig. 4. Luxembourgian mobility model.

The devices move on a straight line with a constant speed from one position to another. The mobility model uses a directed graph given as XML file for device movement. Vertices are crossroads and contain routing probabilities, all possible destinies have the same probability of being taken; the next route is chosen at random. Edges can have an arbitrary width so that devices move on a lane and not only on a strict line (allowing overtaking). To make a real simulation we also have sense in the lanes, so the most congested areas in Luxembourg city center are reflected in our model. There are much more dense areas than others.

In this environment the diffusion process is more difficult than in the case of using random waypoint, since movement is more restricted, but for sure, the simulation is much more realistic. Our main goal is to get results as close as possible to a test on real devices moving along roads in a city center.

For simulating these scenarios we are using JANE simulator [13], [14]. The main feature of JANE is its three steps development; it facilitates the evaluation and minimizes the effort needed for software development in MANETs, so that the evaluated code in the simulated environment can be directly executed in real scenarios without modifications.

V. Simulation Results

To compare these protocols we are simulating them with the mobility model explained in section IV. We want to compare several aspects of their behavior in a VANET. These issues are:

- the broadcast coverage: that means, the number of devices that finally receive the message;
- the message complexity: the number of broadcast and unicast needed to spread the message in the network;
- the broadcast time: the time needed for the diffusion of the message;
- the bandwidth used: the use of the network;

We are comparing the protocols in two scenarios: (1) the first one is a dense network with 1000 devices, and (2) the second one is not so dense, 500 devices (hereinafter sparse network in our context).

As we explained before, the speed of the devices varies between 30 and 75 km/h. As we are dealing with car PCs, the coverage range of the devices is around 100 m.

In such scenarios we needed a training process for tuning and adjusting DFCN parameters. We tried to maximize the number of the devices reached when we tuned the parameters. As a result, we are using *minBenefit*=0.4 and *RAD*∈[0 6] seconds, for both scenarios, and the threshold regarding the density is the only one changing between the two cases: we set *densityThreshold*=15 for the dense network and *densityThreshold*=12 for the sparse one.

We also include in the comparison the well known Simple Flooding (SF) broadcasting protocol [15], [16]. It is the most intuitive idea for disseminating a message in a network. It does not try to reduce the number of re-emissions, so it does not need any knowledge about the neighborhood. The strategy of this algorithm is quite simple, when a device receives a message, it will send it only once. This algorithm was included in our comparison in order to show the validity of the other compared protocols, since Simple Flooding reaches in really short time all devices in a partition, but it is not efficient at all regarding the bandwidth. Additionally, Simple Flooding is not working properly in partitioned networks, since it is not able to spread the message outside the partition where the source node is.

In our experiments, we disseminated a message every 30 seconds during a period of 10 minutes (that means we made 20 broadcasting processes starting from the same device, but from different positions since it was moving). This was simulated in 30 different topologies to make sure our results were realiable.

Table I presents the average of the results obtained regarding the number of devices receiving the message, the number of forwarded messages needed in each dissemination and the ratio between them for the 30 different topologies.

TABLE I
Results obtained for both dense and sparse networks.

		Devices	Broadcasts sent	Ratio
Dense	**BODYF**	$\mathbf{608.9 \pm 30.31}$	312.13 ± 9.78	0.51
	DFCN	395.23 ± 166.92	$\mathbf{194.185 \pm 86.29}$	**0.49**
	SF	292.54 ± 115.86	292.59 ± 115.86	1
Sparse	**BODYF**	$\mathbf{134.81 \pm 29.11}$	82.84 ± 15.90	0.62
	DFCN	123.71 ± 55.58	$\mathbf{68.29 \pm 32.25}$	**0.55**
	SF	72.44 ± 31.60	72.49 ± 31.60	1

As Table I shows, the area covered by BODYF (in terms of devices receiving the message) is considerably higher than the one achieved by DFCN or SF, in dense and sparse networks. The number of forwarded messages in BODYF is always higher, but also the devices reached, so we calculated the ratio between forwarded messages and covered area to make a fair comparison of the algorithms. The ratio is calculated as the number of broadcast per device reached. Simple Flooding is the one with less coverage and higher ratio. We can see that the number of messages sent by DFCN per device reached is slightly lower than in the case of BODYF.

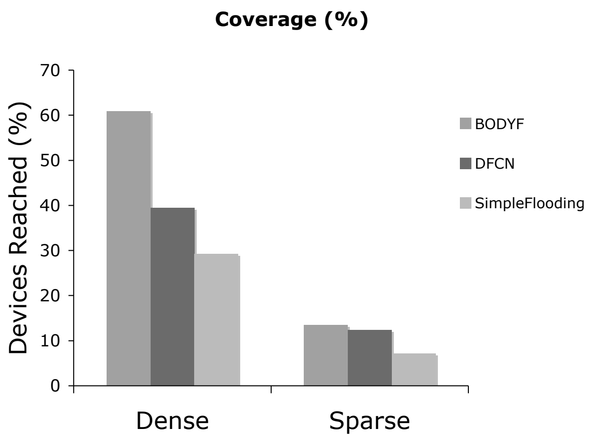

Fig. 5. Coverage percentage in both dense and sparse networks.

In Figure 5, we show the percentage of devices receiving the message, what gives an idea of the total area covered. As it can be seen, BODYF clearly outperforms DFCN and Simple Flooding in the number of devices reached by the broadcasting process.

This is an important feature for VANETs, since there are many applications demanding large coverage of devices at a reasonable cost, like emergency or traffic jam notifications. One of the main reasons for minimizing the number of forwarded messages in MANETs is the battery consumption in processing and re-sending a message. In our work, we are dealing with VANETs, where the optimization of the battery use is not necessary so that all devices have power supply. Additionally, the difference between DFCN and BODYF in terms of this ratio is quite low, therefore we can say that BODYF makes a reasonable use of the network resources.

We should also take into account the network overload in our comparison. DFCN inserts all the neighbors of the device which is forwarding the message. This list is composed by addresses, each address is an IPv6 address, so it is 16 bytes. In our simulations, we obtained that the average of the number of neighbors of each forwarding node was 28.70 and 15.92 in dense and sparse networks respectively. Therefore, we were able to calculate the average of the overload related to this list in all the broadcast process, for both dense and also sparse networks:

1) Dense network
   - The extra bandwith needed for sending the list of neighbors in the message is **89.18 Kbytes** (approx. 713.46 Kbits).
2) Sparse network
   - The extra bandwith needed for sending the list of neighbors in the message is **17.39 Kbytes** (approx. 139.15 Kbits).

We can compare the total bandwidth the broadcast process uses. For BODYF and Simple Flooding, as they do not add any load to the message, we just need to calculate the number of bytes of the broadcast messages sent (the header size is 40 bytes). DFCN includes the list of the neighbors the sender node has, so, apart from the number of messages sent (with headers), we need to sum the load this list adds. The results are shown in Table II. As we can see, even needing more forwarded messages to disseminate the information, BODYF use less bandwidth than DFCN regarding the data of the message. SF is the one with less bandwidth usage, but it is because of the low rate of coverage. The difference between Simple Flooding and BODYF in terms of overload is very low, even when the coverage achieved of the latter is much higher. Between the two protocols with higher coverage, DFCN sends less forwarding messages than BODYF, but due to the list of neighbors that DFCN includes in the message, the total bandwidth used by BODYF is lower.

TABLE II
BANDWIDTH USED BY THE BROADCAST PROCESS.

	BODYF	DFCN	Simple Flooding
Dense	$12.49 KBytes$	$96.95 KBytes$	$\mathbf{11.70KBytes}$
Sparse	$3.31 KBytes$	$20.12 KBytes$	$\mathbf{2.90KBytes}$

In BODYF, we do not need to add any extra load to the broadcast message. Even though it is out of the scope of this paper, since we suppose the tree-based topology is already needed by the network, and not only for our broadcasting; we here also calculate how expensive is, in terms of messages sent, the creation and maintenance of the tree, and the circulation of the token. The messages exchanged for building and maintaining the tree-based topology are similar to ACKs, while we use a 40 bytes header for each exchanged message in our estimations. We have measured the number of broadcast and unicast needed for creating and maintaining the network and also for circulating the token. We made this simulation in both networks, and the results obtained were:

1) Dense network
   - bandwidth used for the creation and maintenance of the tree, and also for the circulation of the token: **48.72KBps** (approx. 389.82 Kbits/second).
2) Sparse network
   - bandwidth used for the creation and maintenance of the tree, and also for the circulation of the token: **23.93 KBps** (approx. 191.45 Kbits/second).

The amount of data used by the tree-based topology for a high dense network of 1000 devices is 389.82 Kbits/second. We can make the average per device also, and it would be around 389.82 bits/second per device in dense networks or 382.9 bits/second for every device in sparse ones, what is not a high overload, since the tree-based topology provides many advantages.

Finally, we also compared the algorithms in terms of broadcast time. For that, we did 30 simulations with different topologies where only one broadcast was sent. In each topology we let the network evolve for 60 seconds, and after that spread the message, each time from a different point of the network (different devices). For each topology we start the broadcast from 10 different devices. We consider the broadcast is finished if no more messages are fowarded after 7 seconds. We chose 7 seconds since the delay set for each device (in DFCN) oscilates between 0 and 6 seconds. We did that for the three protocols with both network densities. The results are shown in Table III, where we specify the average time, the number of devices reached in each broadcast process and the ratio between these two values. Notice that, in this table we are testing 30 topologies and starting the dissemination process from 10 different devices for each topology, so the results in Table III are the average values after 300 simulations. Regarding the time needed for spreading a message, we must take into account that BODYF takes longer, but the difference in devices reached is important. We calculated the ratio between the number of devices reached and the time the dissemination took (devices/s), and in both networks, SF had a higher rate, since it is extremely fast. DFCN has the worst ratio in both densities. Simple Flooding is much faster than BODYF but the coverage reached is much lower. BODYF nearly achieved double size of devices than SF in dense networks.

TABLE III
TIME NEEDED FOR DISSEMINATING A MESSAGE .

		Time (s)	Devices	Ratio
Dense	**BODYF**	90.43 ± 18.85	$\mathbf{479.90 \pm 107.78}$	5.30
	DFCN	27.97 ± 6.45	93.83 ± 31.02	3.35
	SF	$\mathbf{8 \pm 0}$	245.83 ± 125.98	$\mathbf{30.73}$
Sparse	**BODYF**	23.56 ± 5.68	$\mathbf{37.01 \pm 13.06}$	1.57
	DFCN	17.19 ± 2.83	24.13 ± 6.56	1.40
	SF	$\mathbf{8 \pm 0}$	34.31 ± 9.39	$\mathbf{4.28}$

One of the main problems of DFCN is the difficulty to tune all its parameters. In our case, once we tuned the parameters, if we changed something in the simulation, the number of devices reached was highly reduced. This behavior is shown in the average of devices in both tests, since there is a really big difference between them (coverage and time tests) being the scenario, topologies, speed, number of devices, and thresholds the same in both experiments. So we can conclude DFCN is highly dependent on the topology, while this is clearly not the case of BODYF or SF.

VI. Conclusions

In this paper we present a broadcasting protocol over a tree-based topology, BODYF, and we compare its performance versus DFCN and Simple Flooding. DFCN is a neighbor based topology protocol which is fast, designed to minimize the resources required, and generally accepted by the community [1], [10], [17]. Simple Flooding is one of the bases of broadcasting protocols [15], [16].

In this work, we have compared these protocols in a realistic scenario (a vehicular ad hoc network) with two different number of devices. Our results show that, the coverage achieved by BODYF is much higher than DFCN or Simple Flooding in both scenarios, sparse and dense. Although the number of messages used for achieving this coverage is higher, the difference between DFCN and BODYF in terms of the ratio was very small, hence, we concluded that BODYF makes reasonable use of the network resources. The low level of coverage achieved by Simple Flooding, is due to the network partitions. This protocol is not able to disseminate the message outside the partition where the source node is, while DFCN and BODYF are. Regarding the broadcast time, we checked that Simple Flooding is the fastest protocol, but the coverage is lower than BODYF. BODYF also outperformed DFCN, achieving higher rate of devices per second.

Finally, we can conclude that if the main goal of the application is to maximize the covered area we should use BODYF, but if the network resources are very limited or having a high coverage is not really necessary (e.g., using a hybrid network, we just want to spread a message up to the closest hotspot), DFCN could be slightly more suitable. However, in the VANET scenario considered here, network resources are not as limited as in the case of MANETs, in which devices have, for instance, a limited battery life.

As future work, we plan to compare BODYF to other source-tree based and cluster-based protocols to really know how good its behavior is versus other kind of approaches. We would also like to run other algorithms on top of our tree-based topology in such highly mobile networks, as it could be to propagate a large document, even downloading parts of this information from different devices if it is needed.

References

[1] E. Alba, B. Dorronsoro, F. Luna, A. J. Nebro, P. Bouvry, and L. Hogie, "A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan manets," *Computer Communications*, vol. 30, no. 4, pp. 685–697, 2007.

[2] "Carlink project website. http://carlink.lcc.uma.es."

[3] Y. Yi, M. Gerla, and T. J. Kwon, "Efficient flooding in ad hoc networks: a comparative performance study," in *IEEE International Conference on Communications (ICC'03)*, 2003, pp. 1059–1063.

[4] A. Jüttner and A. Magi, "Tree based broadcast in ad hoc networks," *Mobile Networks and Applications*, vol. 10, no. 5, pp. 753–762, 2005.

[5] A. Casteis and S. Chamuette, "Dynamicity aware graph relabeling system - a local computation model to describe manet algorithms," in *Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing and System*, 2005, pp. 198–204.

[6] E. Sopena and Y. Metivier, "Graph relabeling systems: a general overview," *Computers and Artificial Intelligence*, vol. 16, no. 2, pp. 167–185, 1997.

[7] A. Casteis, "Model driven capabilities of the da-grs model," in *Intl. Conf. on Autonomic and Autonomous Systems (ICAS'06). San Francisco. USA*, 2006, p. 24.

[8] S. J. Yang, H. K. Oh, and S. H. Park, "Efficient multicast routing protocol for mobile hosts in IPv6 based networks," *Electronic Letters*, vol. 38, no. 16, pp. 936–938, 2002.

[9] L. Hogie, P. Bouvry, F. Guinand, G. Danoy, and E. Alba, "A Bandwidth-Efficient Broadcasting Protocol for Mobile Multi-hop Ad hoc Networks," in *Demo proceeding of the 5th International Conference on Networking (ICN'06)*. IEEE, October 2006, p. 71.

[10] L. Hogie, F. Guinand, and P. Bouvry, "A heuristic for efficient broadcasting in the metropolitan ad hoc network," in *8th Int. Conf. on Knowledge-Based Intelligent Information and Engineering Systems*, 2004, pp. 727–733.

[11] E. Alba and B. Dorronsoro, *Cellular Genetic Algorithms*, ser. OR/CS Interfaces. Springer-Verlag, 2008.

[12] A. Andronache and S. Rothkugel, "Hytrace-backbone-assisted path discovery in hybrid networks," in *The Seventh International Conference on Networking ICN*, to appear.

[13] D. Gorgen, H. Frey, and C. Hiedels, "JANE–the Java Ad-hoc Network Environment," in *Proceedings of the 40th Annual Simulation Symposium*, 2007, pp. 163–176.

[14] "The Java Ad-hoc Network Environment (JANE). http://syssoft.uni-trier.de/jane/index.php/JANE_Basics."

[15] B. Williams and T. Camp, "Comparison of broadcasting techniques for mobile ad hoc networks," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, 2002, pp. 194–205.

[16] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, 1999, p. 151162.

[17] L. Hogie, "Mobile ad hoc networks, modelling, simulation and broadcast-based applications," Ph.D. dissertation, University of Luxembourg, Luxembourg, 2007.