Token Traversal Strategies of a Distributed Spanning Forest Algorithm in Mobile Ad hoc - Delay Tolerant Networks

A. Piyatumrong[1], P. Ruiz[1], Pascal Bouvry[1], F. Guinand[2], and K. Lavangnananda[3]

[1] Faculty of Science, Technology & Communication, University of Luxembourg
{apivadee.piyatumrong,patricia.ruiz,pascal.bouvry}@uni.lu
[2] Le Havre University, France
frederic.guinand@univ-lehavre.fr
[3] School of Information Technology, Bangkok, Thailand
kitt@sit.kmutt.ac.th

Abstract. This paper presents three distributed and decentralized strategies used for token traversal in spanning forest over Mobile Ad Hoc Delay Tolerant Networks. Such networks are characterized by behaviors like disappearance of mobile devices, connection disruptions, network partitioning, etc. Techniques based on tree topologies are well known for increasing the efficiency of network protocols and/or applications, such as Dynamicity Aware - Graph Relabeling System (DA-GRS). One of the main features of these tree based topologies is the existence of a token traversing in every tree. The use of tokens enables the creation and maintenance of spanning trees in dynamic environments. Subsequently, managing tree-based backbones relies heavily on the token behavior. An efficient and optimal token traversal can highly impact the design of the tree and its usage. In this article, we present a comparison of three distributed and decentralized techniques available for token management, which are Randomness, TABU and Depth First Search.

Keywords: Token traversal, spanning tree, distributed system, delay tolerant networks, Depth First Search.

1 Introduction

Networks spontaneously and automatically created between neighboring mobile devices are commonly called ad hoc networks. The main advantage of this kind of networks is that no infrastructure or administration system is required. The signal strength can be weakened due to the appearance and disappearance of the devices, the mobility of nodes, and obstacles in the environment. These phenomenons lead to frequent and long duration partitions of the network. An emerging subclass of ad hoc networks, Mobile Ad Hoc Delay Tolerant networks (henceforth called *mobile ad hoc DTNs* for brevity), is characterized by including these undesirable behaviors. This unpredictable and highly fluctuating topology makes challenging many aspects like efficient communication, routing, etc.

Previous researchers demonstrated the validity of spanning trees in networking area [1], [2], etc. Establishing a spanning tree in the network is a well known prerequisite strategy for providing efficient communication and routing algorithms in wired networks. Furthermore, recently it is also a tendency to use them in mobile ad hoc DTNs [3–5]. One common mechanism used in spanning tree algorithms is the utilization of tokens. In [6], the authors state that techniques for traversing the token that perform well in static networks are not necessarily well suited in networks with high mobility. Thus, a new study of token traversal in high mobility network must be undertaken. Also in [7], it is concluded that the token movement strategies impact on the tree construction, and, therefore, on the topology management. This motivated us to study, implement and compare different token traversal techniques in order to determine which strategy performs better in different environments in mobile ad hoc DTNs.

Dynamicity Aware - Graph Relabeling System (DA-GRS) [8] is a model for creating and analyzing decentralized topologies and algorithms targeting dynamically distributed environments like mobile ad hoc DTNs. Up to now, the token traversal strategies used in DA-GRS are based on the assumption that no memory is used in the mobile nodes. These techniques are random and Tabu [7].

In this study, we applied Depth First Search (DFS) for the first time to DA-GRS. We considered to include DFS in our comparison since it is a very well known strategy for static tree traversal and the idea of DFS has been used for mobile ad hoc networks in recent works [9]. However, due to the highly fluctuant topology, having an ordering strategy might not be a good idea. Thus, a deep study and also a comparison between DFS and other techniques are needed.

This study assumes that spanning trees provide a reliable path way for efficient communications and services. Thus, having a spanning tree covering as many nodes as possible in the shortest time is desired. In the context of this study, the spanning tree must span the entire connected communication graph. Therefore, we implemented and compared different strategies for traversing the token in the tree topology in terms of the performance ratio and the convergence speed rate. The performance ratio is measured as the number of different partitions (or connected components) of the underlying network divided by the number of existing trees. The convergence speed rate shows how fast multiple trees belonging to the same partition merge into one tree. We compare three different distributed strategies: Randomness, TABU, and Depth First Search (DFS), described later in Section 4.

The contribution of this paper is thus three-fold: (1) the design and implementation of DFS for DA-GRS for the first time and the re-implementation of other two token traversal techniques, (2) implementation of a framework which allows different token movements strategies based on realistic mobility models (e.g. highway and shopping mall scenarios), and (3) a comprehensive study and analysis of the three proposed strategies which is currently missing in the literature.

The paper is organized as follows; Section 2 introduces the conceptual rules for constructing and maintaining a spanning forest in mobile ad hoc DTNs in a purely distributed and decentralized manner. Later, in Section 3, the realistic communication models used for creating the tree topology over an existing mobile ad hoc DTNs are explained. After that, in Section 4 all the compared strategies for circulating the token are presented. The experiments realized in this paper are explained in Section 5 and the results obtained are shown in Section 6. Finally, Section 7 concludes the work.

2 The Spanning Forest Algorithm using DA-GRS model

Dynamicity Aware - Graph Relabeling System (DA-GRS) [8] is an extension of Graph Relabeling System, GRS [10]. It is a high level abstraction model that can improve the development of self-organized systems. All the mechanisms underlying it are for managing mobile ad hoc DTNs efficiently. DA-GRS just models how to handle with topology changes and interaction between devices, but it does not itself create services or applications.

A tree is defined as a free cycle graph. A graph composed of several trees is called, hereinafter, a forest. Assuming this, DA-GRS proposes some rules for constructing and maintaining a spanning forest in mobile ad hoc DTNs represented in Figure 1 and described after. In this figure, the circle represents a node. Letters on top of the nodes mean: (1) 'T' if the node possesses the token, (2) 'N' if the node does not possess the token, and (3) 'Any' when the node can possess or not the token. The labels '0', '1' and '2' on the edge represent the route to the token. And finally, label 'off' describes a broken link.

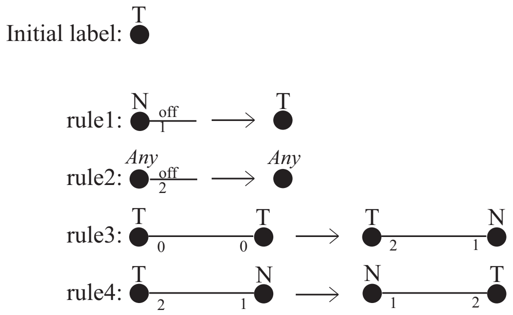

Figure 1. DA-GRS rules for creating and maintaining spanning forest topologies

Dynamic networks are characterized by mobility and possible connection disruptions, hence, devices need to handle with this changes when creating and maintaining the spanning tree. DA-GRS proposed four rules (as shown in Figure 1) to handle with four different situations. In the initial state every device has the token (what means it is a tree itself), and these 4 rules are:

- rule 1: A tree link breaks, and the node belongs to the sub-tree which does not possess the token. In this case the node must regenerate the token, otherwise there will exist a tree without a token (which is an undesirable situation).

– rule 2: A tree link breaks, and the broken link occurs at a node which currently belongs to the sub-tree which possesses the token. In this case, the node does nothing regarding the maintenance of the token.
– rule 3: When a node with token meets another device possessing a token; both nodes will try to merge their trees in order to obtain a bigger tree from the two existing ones. The trees merging process starts. The result of this rule remains a bigger tree and only one token (the merging process discards one token automatically in order to remain one and only one token within a tree).
– rule 4: Token traversal in general case: the token visits the nodes of the tree following a given strategy.

An important feature of this model is that in each tree one and only one token exists. Furthermore, only two nodes possessing token (thus belonging to different trees) can start the trees merging process. As we are dealing with trees, cycles are not allowed. DA-GRS manages to avoid them since it is not possible to have two nodes belonging to the same tree and possessing a token at the same time.

3 Utilizing and Applying DA-GRS for Creating the Spanning Forest

For creating a spanning forest over a mobile ad hoc DTN using DA-GRS in a decentralized way, nodes must exchange some messages between them in order to have knowledge of who else is possessing a token in the neighborhood and also to merge trees. Since no global knowledge is considered, a more detailed communication syntax needs to be specified. Therefore, the proposed message sequence that devices must exchange in a decentralized system is explained in the following:

3.1 Beaconing

In order to have knowledge of the one-hop neighborhood most decentralized systems utilize beacons (also called 'hello messages') [11]. For that purpose, every node sends periodically a message alerting about its presence. For considering a node as a neighbor, one must receive a beacon of the node regularly. A node will not be a neighbor anymore when its beacon is not received within a predefined time.

Using this beaconing both a broken communication link and the appearance of a new one-hop neighbor are detected, and thus, 'rule 1' and 'rule 2' in Figure 1 can be applied. Based on *Beaconing Rate* of IEEE802.11 [12], the time interval used for periodically sending the beacon is 100 millisecond.

3.2 Trees Merging Process

'rule 3' in Figure 1 represents the spanning tree construction scenario (trees merging process). DA-GRS uses rendez-vous assumption [13] as synchronization method at this merging process. This rendez-vous assumption states that at one moment in time, only two nodes possessing token can meet and be merged. We consider that this assumption is too rigid in real world communications. Thus, this work proposes to relax this assumption by allowing a node to choose one token among the tokens owned by its neighbors.

In a distributed system a node has no ability to know if there exists any node with token in its neighborhood. Thus, nodes holding a token will broadcast a packet, *'findingTk'*, to verify whether any of its neighbors also possesses a token. If any neighbor of this broadcasting node possesses a token and receives *'findingTk'* will reply using a *'ACK_finding'* message. *'ACK_finding'* is an expression of agreement to merge their trees.

Moreover, this particular neighbor will set its status to wait for *'SYN/ACK_finding'* to confirm the merging process within a predefined period, *'TimerWaitFor_SynAckFinding'*. As we are working with a discrete simulator, the time duration of the timers is one simulation step.

After broadcasting *'findingTk'*, the broadcasting node will wait within a predefined duration, *'TimerWaitFor_Finding'*. At the end of this waiting time, the broadcasting node selects one of its neighbor and a *'SYN/ACK_finding'* message will be sent using unicast to this selected neighbor. In case, there is no node with token in the neighborhood, at the end of this timer the token is circulated. The message sequence of this process is illustrated in Figure 2.

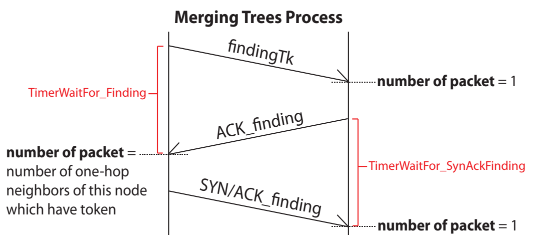

Figure 2. Message sequence diagram for merging trees

3.3 Token Traversal

'rule 4' in Figure 1 stands for token traversal in general case. When a node sends a broadcast message for finding a neighbor possessing a token, it also establishes a timer as addressed in previous section, *'TimerWaitFor_Finding'*. If the timer finishes and there is no answer from any neighbor, the token movement takes place. If there is no neighbor belonging to a different tree, the node will directly move the token, see Figure 3.

4 Token Traversal Strategies in a Decentralized System

As explained in previous sections, in DA-GRS and usually when dealing with spanning trees, the system need a token for creating and maintaining a tree. Every node, at some

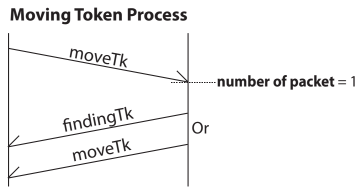

Figure 3. Message sequence diagram for traversing the token

moment, must possess the token, since it allows looking for neighbors with token to merge trees. The way this token moves along the tree impacts on the spanning tree construction. In literature, tree traversal refers to the process of visiting each node in a tree data structure in a particular manner [14]. In the context of this study, we want the token to traverse less but has more chance to meet another token. In other words, we want the fastest rate of the tree construction to cover a connected subgraph, which means less number of trees or in the best case, remaining only one tree over a connected subgraph. In previous work, the token traversal in DA-GRS has only been implemented using random or Tabu techniques. This is the first time that DFS is applied to DA-GRS.

This section gives a detailed explanation of the three strategies used in this study. It is worth noting that all strategies are working in distributed and decentralized manner suiting to work in mobile ad hoc DTNs.

4.1 Randomness

The Randomness here follows the uniform distribution law. Randomness is the heuristic used by DA-GRS by default. The process is done by selecting a node randomly among the list of neighbors. The description of the 'Randomness' traversal technique is described in Algorithm 1.

Algorithm 1 Using Randomness heuristic in $Move\_Token$ (τ_i) process of a node ν

```
1: $\alpha$ is the set of neighbors of node $\nu$
2: node $\rho$ is a node selected randomly from set $\alpha$
3: move token $\tau_i$ from node $\nu$ to node $\rho$
```

4.2 TABU

TABU creates a list of forbidden movements in which the most recent nodes possessing the token are stored. This list is called as tabu list. The algorithm consults the tabu list before sending the token to a neighbor in order to avoid visiting the same node repeatedly. Tabu list uses a fix size of memory, *memory_size*, to set the number of stored nodes in the list. This list is sent within the token, no node memory is used. In Algorithm 2 a detailed description of this strategy is given.

Algorithm 2 Using TABU heuristic in $Move\_Token$ (τ_i) using a defined value of *memory_size* processing at a node ν

```
1:  $\alpha$ is the set of neighbors of node $\nu$
2:  $\beta$ is the TABU-like list which has size equal to memory_size
3:  Set $availableNode$ = $\alpha$ - $\beta$
4:  if $availableNode \neq \emptyset$ then
5:      node $\rho$ is a node selected randomly from set $availableNode$
6:      token $\tau_i$ move from node $\nu$ to node $\rho$
7:      if the number of item of $\beta$ reach the memory_size then
8:          remove the first item from list $\beta$
9:          add $\rho$ to the end of list $\beta$
10:     else
11:         add $\rho$ to the end of list $\beta$
12:     end if
13: else
14:     node $\rho$ is a node selected randomly from set $\alpha$
15:     remove item $\rho$ from list $\beta$
16:     token $\tau_i$ move from node $\nu$ to node $\rho$
17:     add $\rho$ to the end of list $\beta$
18: end if
```

Applying this technique to DA-GRS was proposed in previous work [7]. The *memory_size* of the list (its length) was also studied in [7], and it was demonstrated that a tabu list longer than 1 entry of device did not provide much better results than using a tabu list with size 1. Therefore, we use in our study TABU with size list equal to 1. For brevity, henceforth we will use 'TABU$\{1\}$' to represent the usage of TABU at *'memory_size'* equals to one. This is equivalent to prohibiting sending the token to the node from which the current one received it.

4.3 Depth First Search (DFS)

DFS is commonly used as token movement technique [5, 15, 16] when dealing with tree based topologies. It imitates the traversal of the classical Depth First Search algorithm and, thus, it is an ordering traversal strategy.

In order to traverse systematically like the classical algorithm in distributed and dynamic systems, DFS utilizes the neighbor list information provided by the beaconing process. Thus, the neighbor list is always up to date. Furthermore, in this implementation, it is necessary to keep information inside each node. To be more specific, these information are: (a) about the node that sends the token to the current device for the first time (henceforth, we refer to this first node as 'upper neighbor'), and (b) information of neighbors receiving the token from this current device. In this way, the node will definitely sends the token to all its neighbors using the neighbor list and the information stored (a) and (b). It will not send the token back to the upper neighbor meanwhile all the list of neighbors is not visited.

The mechanism is as follows: whenever the current node receives the token back from its neighbors (and this is not the first time this node receives token), the current node will send the token to the next neighbor in the neighbor list. Once the list is finished, the token is sent back to the 'upper neighbor' if it has not gone from the neighborhood. Otherwise, this current node will become its own 'upper neighbor' and will send again the token to the first neighbor of the its neighbor list. This implementation is described in Algorithm 3.

Algorithm 3 Using DFS heuristic in $Move\_Token$ (τ_i) process of a node ν

```
1: $\alpha$ is the set of neighborhood of node $\nu$
2: $\beta$ is the DFS list in node $\nu$
3: $\varpi$ is '$upper\ neighbor'$
4: $\delta$ is the latest node that send $\tau_i$ to $\nu$
5: if $\varpi$ is empty then
6:     $\varpi = \delta$
7: end if
8: Set $availableNode = \alpha - \beta - \varpi$
9: if $availableNode \neq \emptyset$ then
10:     node $\rho$ is the first node from set $availableNode$
11:     move token $\tau_i$ from node $\nu$ to node $\rho$
12:     add $\rho$ to the end of list $\beta$
13: else
14:     clear list $\beta$
15:     if $\varpi$ is in the set $\alpha$ then
16:         move token $\tau_i$ from node $\nu$ to node $\varpi$
17:         set $\varpi$ to empty
18:     else
19:         $\varpi = \nu$
20:         Set $availableNode = \alpha - \delta$
21:         node $\rho$ is the first node from set $availableNode$
22:         move token $\tau_i$ from node $\nu$ to node $\rho$
23:         add $\rho$ to the end of list $\beta$
24:     end if
25: end if
```

5 Experiment Methodology and Measurements

5.1 Experiment methodology

The networks used in this work were generated using a discrete network simulator, Madhoc [17]. An ad hoc networks simulator that provides mobility models allowing realistic motion of citizens in variety of environments. Two real-world mobility models, 'shopping mall' and 'highway', were selected in the simulations using the parameters summarized in Table 1.

Mobile ad hoc DTNs can be represented as a dynamic communication graph (G), where the mobile devices are the set of vertices (V), and the links between them are the

Table 1. Parameters used in the experiments

	Shopping Mall	High way
Surface (km^2)	0.32	2.24
Node density (per km^2)	347.85	72.55
Number of nodes	110	160
Avg. Number of partitions	1.95	15.9
Number of connections	446	498
Average degrees	8.13	6.23
Velocity of nodes (m/s)	0.3-3	20-40
Radio transmission range	40-80 m	
Network technology	IEEE 802.11b	

edges of the graph, (E). The dynamism of the network is represented by the fact that both V and E can change at any time. Therefore, the graph at a given time t, $G(t)$, is composed of $(V(G(t)), E(G(t)))$.

We derived communication graphs from Madhoc which performs simulation in discrete-time. So the communication network corresponds to a series of static graphs: $G(t)$ for $t \in \{t_1, t_2, t_3, ..., t_{400}\}$. Between two consecutive times t_i and t_{i+1} the communication graph remains the same. However, using such a short timing-snapshot, 1/4 seconds between two consecutive times is considered sufficient to reflect the reality. We made 100 runs for each experiment in order to have reliable results. That means we are simulating the behavior of a mobile ad hoc DTN for 100 seconds over 100 different topologies.

5.2 performanceRatio() function

At a given moment t, $G(t)$ may be partitioned into a set of m connected subgraphs. Having Γ as the set of all spanning trees at moment t of $G(t)$. The quality of the algorithms can be assessed by the number of connected subgraphs (m) over number of trees created (Γ). This quality is determined by the following ratio.

$$performanceRatio(G(t)) = \left(\frac{m}{\mid \Gamma \mid}\right) \tag{1}$$

The value of the performance ratio approaching to one means higher quality of the algorithm (less number of trees in a connected subgraph). Having a spanning tree per connected subgraph enables more efficient communication and topology management, since at least, the information can be disseminated systematically via the created spanning tree. This means the algorithm is robust regarding the dynamism of the network because it can construct a tree covering all the nodes conforming the connected subgraph.

Figures 4(a) and (b) illustrate the measurement of all cost functions proposed here. In the figure 4(a), the communication graph $I(t)$ has two connected subgraphs, and each connected subgraph has one spanning tree. On the contrary, the communication

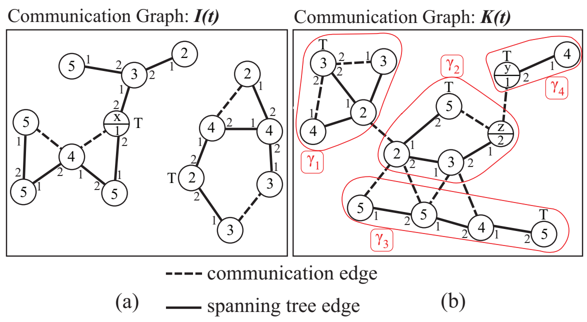

Figure 4. An example scenario for illustrating the proposed cost functions for spanning forest

graph $K(t)$ depicted in figure 4(b) has only one connected subgraph but four spanning trees ($\gamma_1, ..., \gamma_4$). Thus, the $performanceRatio(I(t))$ and $(K(t))$ equal to 1 and 0.25, respectively.

5.3 convergenceSpeedRate() function

The $convergenceSpeedRate()$ is measured based on the number of iterations in simulation. Let $\varDelta$ be the number of iterations the algorithm required trying to achieve the least $performanceRatio()$ and $\varDelta^*$ be the number of iterations required per $G(t)$. Having $performanceRatio()$ equal to one within $G(t)$ is an ideal situation. However, having limited merging process (explained in Section 3) causes no guarante that $performanceRatio()$ will be one, in other words, it is always possible to have multiple trees per connected component at any time t of graph G. In such case, the number of iterations used within that $G(t)$ will be counted into $\varDelta$. The lower the value of $convergenceSpeedRate()$ is, the faster the algorithm converges a connected component into a tree. The $convergenceSpeedRate()$ can be written as below.

$$convergenceSpeedRate(G(t)) = \left(\frac{\varDelta(G(t))}{\varDelta^*(G(t))}\right) * 100 \qquad (2)$$

6 Results

In this section we present the comparison results obtained for the three different strategies studied for circulating the token in a decentralized tree based algorithm. These three strategies are: Randomness, TABU$\{1\}$, and DFS. The comparison was made in terms of the speed of the convergence of the tree and the performance ratio explained both in the previous section. The results shown are the average of simulating 100 topologies for 100 seconds each topology.

Figure 5 and 6 show the simulation results for the shopping mall and highway environment, respectively. From both figures, DFS clearly gives the best behavior among these three strategies for both environments. Furthermore, both figures show the impact

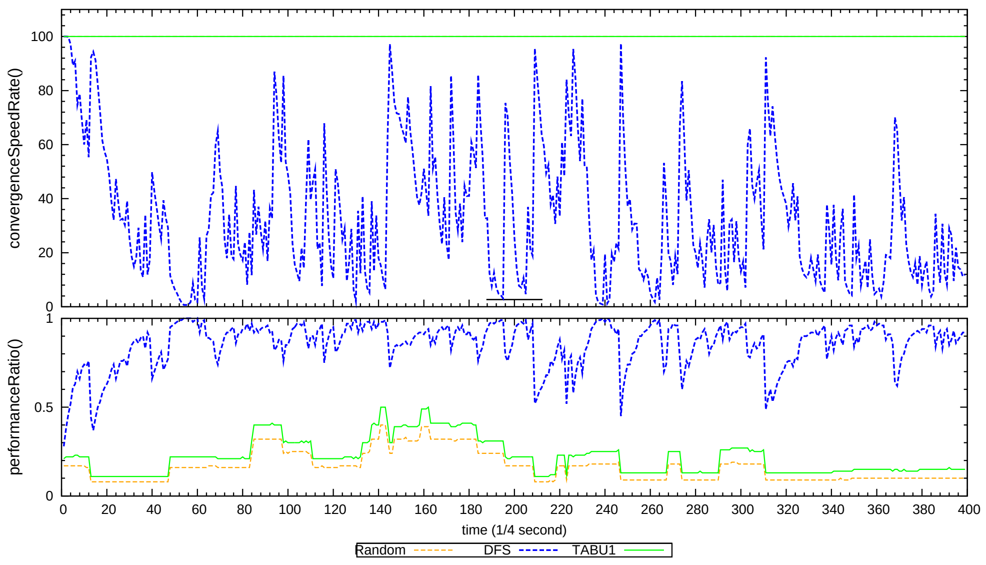

Figure 5. Comparison of convergenceSpeedRate() measuring among all studied algorithms in 'Shopping Mall' mobility model

of the mobility model toward the resulting tree. Easily observing from Figure 5 (shopping mall scenario), the difference between DFS and the other strategies is very large. Contrary, the resulting gap from DFS to other strategies in the highway scenario, Figure 6, is not as big as what we can see in shopping mall environment. This is because of the speed of the devices and hence, the highly fluctuant topology. Thus, we measure the differences between DFS and the other two strategies (averaging results over the simulation time). The results of this measurement are shown in Table 2 to demonstrate the difference in terms of the percentile of the distance. Furthermore, as the result values do not follow a normal distribution in any case, we apply the Kruskal-Wallis test in order to obtain statistical significance with 95% probability in our comparisons. The results show that DFS are significantly better than TABU$\{1\}$ and Randomness.

Table 2. The percentile of the distance from DFS to the other strategies

		TABU$\{1\}$	Randomness
Highway	performanceRatio	23.71%	34.71%
	convergenceSpeed	12.38%	12.40%
Mall	performanceRatio	63.70%	69.19%
	convergenceSpeed	66.42%	66.42%

According to Table 2, the differences are up to 60-70% when compare the result of DFS and the other strategies in shopping mall model. On the other hand, for highway

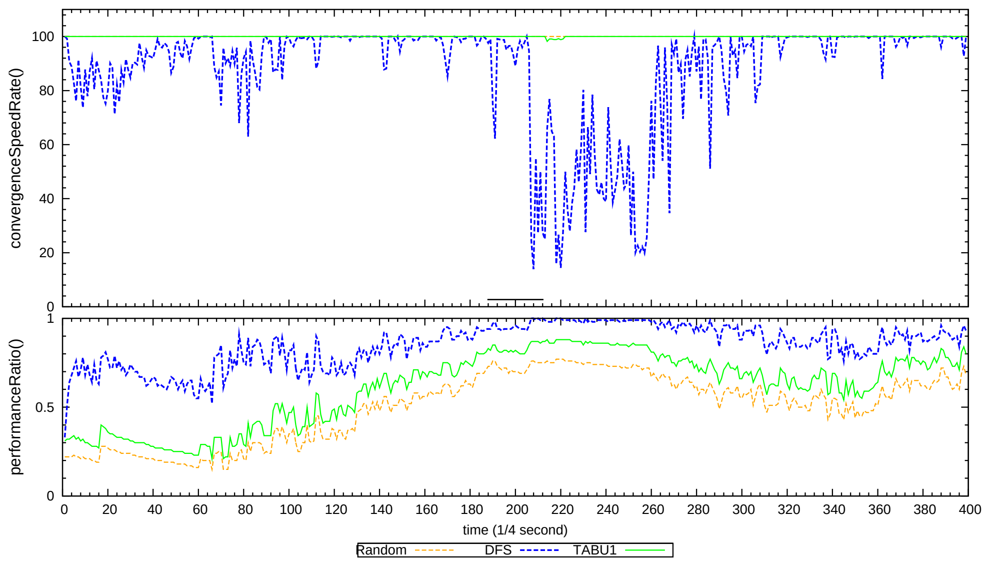

Figure 6. Comparison of convergenceSpeedRate() measuring among all studied algorithms in 'highway' mobility models

model, the differences of results between all the strategies are distinguishable and also statistically significant, but not so huge different (12-35% of differences) as found in shopping mall model. This comes from the fact that the highway model has a high fluctuating mobility. Thus, the topology is more likely to change than in the shopping mall.

The overall results show that the Randomness strategy is the worst one in both environments. This behavior was expected, since when using the random technique many nodes in the tree can hardly possess the token, so the merging trees in those areas rarely happened. TABU{1} ensures that one neighbor will not possess the token twice consecutively. Thus, TABU{1} achieves better distribution of the token than Randomness.

As stated at the beginning of this paper, our intuition suggest that a strict ordering strategy may not be a promising technique for a high changeable topology. The reason to this intuition is that the topology is changing a lot in a short time while the token is moving mostly in the same area during such small period of time. However, the experimentation results deny our intuition. Our results show that DFS behaves better than Randomness and TABU{1}. Thus, it can be confirmed that an ordering strategy like DFS can work well under highly changing topology.

In Table 3 we resume the behavior of each technique since we can consider this is a multi-objective multi-constraint problem, and depending on the necessities of each situation a technique or another can be used. As it can be seen in Table 3 the only strategy that uses no memory at all is Randomness. For those using memory it is possible to distinguish between using the memory in the node like DFS, or using memory in the token as TABU.

Table 3. Multi-objective multi-constraint study

	DFS	TABU	Randomness
No memory	No	No	Yes
Token memory	No	Yes	No
Node Memory	Yes	No	No

7 Conclusions and Future Work

Providing efficient communication and topology management in delay tolerant mobile ad hoc networks is a difficult task which presents a real challenge. We found out that token traversal techniques generally used in tree-based algorithms has a significant impact to the resulting tree. In this work, three different strategies for token movement through the tree topology: Randomness, TABU$\{1\}$, and Depth First Search (DFS) were systematically studied and compared in terms of the performance ratio and the speed of convergence. The former measures the number of spanning trees per connected component at a given moment, the closer to one the better performance. The latter gives an idea of how fast different trees belonging to the same connected component merge and form a solely tree composed of all the nodes within the partition.

To the best of our knowledge, it is the first time DFS is applied to DA-GRS, a mobile ad hoc DTN system, and also it is the first time a comparison between token traversal techniques is done in the literature.

For doing the comparison, two different scenarios were selected: (1) a shopping mall where the movement of the device is slow, and (2) a highway where the nodes move at high speeds. We found out that ordering strategies for token traversal helps to merge trees faster. This can be confirmed since DFS outperform the no ordering techniques like Randomness and less ordering such as TABU$\{1\}$.

As future work, we plan to study the impact of these techniques to any high level application when using the tree, i.e., when disseminating a message through the whole network using this tree based topology, routing, etc. Since the token movement affects the creation of the tree, therefore we also want to study how these strategies impact on the robustness of application using tree-based topology.

8. References

[1] E. C. R. John M. McQuillan, Ira Richer, "The new routing algorithm for the arpanet," *IEEE Transactions on Communications*, vol. COM-28, no. 5, pp. 711–719, May 1980.

[2] "IEEE standard 802.1d-2004: IEEE standard for local and metropolitan area networks: Media access control (mac) brideges," June 2004.

[3] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM, 2007, pp. 32–40.

[4] J. Su, A. Goel, and E. de Lara, "An empirical evaluation of the student-net delay tolerant network," *Mobile and Ubiquitous Systems, Annual International Conference on*, pp. 1–10, 2006.

[5] P. Ruiz, B. Dorronsoro, D. Khadraoui, and P. Bouvry, "BODYF–A Parameterless Broadcasting Protocol Over Dynamic Forest," in *Workshop on Optimization Issues in Grid and Parallel Computing Environments, part of the High Performance Computing and Simulation Conference (HPCS)*, 2008, pp. 297–303.

[6] N. Malpani, Y. Chen, and J. L. Welch, "Distributed token circulation in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 154–165, 2005.

[7] Y. Pigné, "Modélisation et traitement décentralisé des graphes dynamiques - application aux réseaux mobiles ad hoc," Ph.D. dissertation, L'Université du Harve, December 2008.

[8] A. Casteigts, "Model driven capabilities of the DA-GRS model," *ICAS '06: Proceedings of the International Conference on Autonomic and Autonomous Systems*, p. 24, 2006.

[9] G. R., M. A., and P. S., "Minimizing broadcast latency and redundancy in ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 16, pp. 840–851, 2008.

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

[11] M. Gast, *802.11 Wireless Networks: The Definitive Guide*, 2nd ed. O'REILLY, April 2005.

[12] "IEEE standard 802.11: Wireless lan medium access control and physical layer specifications," IEEE Computer Society, August 1999.

[13] L. Cardelli, "An implementation model of rendezvous communication," Lecture notes in Computer Science 197, 1985.

[14] E. G. Goodaire and M. M. Parmenter, *Discrete mathematics with graph theory*, 2nd ed. Printice-Hall, Inc., 2002.

[15] N. Bauer, M. Colagrosso, and T. Camp, "An agile approach to distributed information dissemination in mobile ad hoc networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Washington, DC, USA, 2005, pp. 131–141.

[16] I. Stojmenovic, M. Russell, and B. Vukojevic, "Depth first search and location based localized routing and qos routing in wireless networks," in *Intl. Conf. on Parallel Processing (ICPP'00)*, 2000.

[17] L. Hogie, P. Bouvry, F. Guinand, G. Danoy, and E. Alba, "Simulating Realistic Mobility Models for Large Heterogeneous MANETS," in *Demo proceeding of the 9th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM'06)*. IEEE, October 2006, pp. 129–141.