Token Traversal Strategies of a Distributed Spanning Forest Algorithm in Delay Tolerant MANETs

Apivadee Piyatumrong, Patricia Ruiz, Pascal Bouvry, Frédéric Guinand, Kittichai Lavagnananda

Token Traversal Strategies of a Distributed Spanning Forest Algorithm in Delay Tolerant MANETs

Apivadee Piyatumrong, Patricia Ruiz, Pascal Bouvry*, Frédéric Guinand†, and Kittichai Lavangnananda‡,
*Université du Luxembourg, The Faculty of Science, Technology and Communication (FSTC)
Computer Science and Communications Research Unit (CSC), Luxembourg
Email: apivadee.piyatumrong@uni.lu, patricia.ruiz@uni.lu, pascal.bouvry@uni.lu
†Le Havre University, LITIS, Le Havre, France
Email: frederic.guinand@univ-lehavre.fr
‡School of Information Technology,
King Mongkut's University of Technology Thonburi, Bangkok, Thailand
Email: kitt@sit.kmutt.ac.th

***Abstract*—This paper presents several distributed and decentralized strategies used for token traversal in spanning forest over DTMs. DTMs or Delay Tolerant Mobile ad hoc networks are characterized by some undesirable behaviors like the disappearance of mobile devices, connection disruptions, network partitioning, etc. Providing efficient communications in DTMs is a very challenging issue. Techniques based on tree topology are well known for increasing the efficiency of network protocols and/or applications. One of the main features of the tree based topologies is the existence of a token traversing in every tree. The traversal of this token through a tree has a high impact on its characteristics, such as the topology of the constructed tree and the performance (quality of service) it can provide. The efficiency of communications relies on the availability of the spanning tree topology. In this sense, a complete tree structure over a network is desired. Furthermore, the convergence speed rate used in constructing this tree is important.**

I. Introduction

Networks spontaneously and automatically created among neighboring mobile devices are commonly called mobile ad hoc networks (MANETs). The main positive advantage of this kind of networks is that no infrastructure or administration system is requiered, as well as the flexibility they have. Due to the appearance and disappearance of the nodes, the mobility, and the obstacles, the signal strength can be weakened and frequent and long duration partitions may occur. An emerging subclass of MANETs called Delay Tolerant Mobile ad hoc networks (DTMs) are characterized by including these undesirable behaviors. This unpredictable and highly fluctuating topology make challenging many aspects like efficient communication, routing problems, etc.

DTM 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 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. Therefore, the graph at a given time t, $G(t)$, is composed of $(V_t(G), E_t(G))$.

Establishing spanning tree in the network is a well known strategy for efficient communication and routing algorithms in wired networks, but recently it is also a tendency to use them in MANETs [1], [2], [3]. In [4] the authors stated that techniques for traversing the token that perform well in static networks are not necessarily well suited in network with mobility. Also in [5], it concluded that the token movement strategies impacts on the tree construction. Therefore, in this work some different strategies for traversing the token in the tree topology are implemented and compared in terms of the performance ratio and the convergence speed rate. The performance ratio is measured as the number of existing trees divided by the number of different partitions (or connected components), and the convergence speed rate shows how fast multiple trees belonging to the same partition merge into one tree. We compare four different distributed strategies: Randomness, TABU-like, Depth-First Search (DFS) and finally Depth-First Search-like (DFS-like), described later.

The rest of the paper is organized as follows, next section introduces the model used for creating the tree topology over existing DTMs in pure distributed and decentralized manner. After that, in Section III all the compared strategies for circulating the token are presented. The experiments are explained in Section IV and the results obtained are shown in Section V. Finally Section VI concludes the work.

II. The Spanning Forest Algorithm of DA-GRS

DA-GRS [6] is a model for creating and analyzing decentralized applications and algorithms targeting dynamically distributed environments like DTMs. Normally, such applications and algorithms are often very difficult to set up, describe and validate. Using DA-GRS is a convenient way to design algorithms for DTMs, since its outstanding properties are localized in a dynamic working manner.

DA-GRS proposed some rules for constructing and maintaining a spanning forest (given that a forest is a graph whose components are trees [7]) in DTMs. Figure 1 illustrates these simple rules of DA-GRS. These rules handle four different scenarios, which are: (a) partition occurs at a node which belongs to the spanning tree that possesses the token, (b) partition occurs at a node which belongs to the spanning tree which does not possess the token, (c) when a token meets another token (rendez-vous assumption), and, (d) token traversal in general case (randomly).

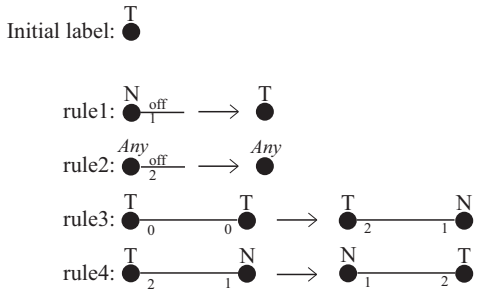

Fig. 1. DAGRS rules for creating and maintaining spanning forest topologies

An important feature of this model is that in each tree exists one and only one token. Furthermore, only two nodes possessing token can do the merging tree operation. Since we are constructing, a tree cycles are not allowed, and DAGRS manages to avoid them because it is impossible to have two nodes belonging to the same tree possessing token at the same time.

A. Simulating DA-GRS as a Network Protocol

For creating spanning forest over a DTM using DA-GRS, a more detailed communication syntax needs to be specified. Following those four scenarios defined in DA-GRS, three different communication syntax or three message sequence types are proposed as follows.

1) Beaconing: In order to have knowledge of the one-hop neighborhood most decentralized systems utilize beacons (also called 'hello messages') [8]. For that purpose, every node sends periodically a message alerting about its presence. In order to consider a node as a neighbor, one must receive a beacon of the node regularly. A node will not be a neighbor anymore when one does not receive any beacon from that particular node 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, 'rules1' and 'rule2' in Figure 1 can be applied. Based on *Beaconing Rate* of IEEE802.11 [9], the time interval used for periodically sending the beacon is 100 millisecond.

2) Synchronization method and messages for Tree Merging Process: 'Rule3' in Figure 1 represents the spanning tree construction scenario (merging trees process). DA-GRS uses rendez-vous assumption as their synchronization method at this merging process. This rendez-vous assumption states that at one moment in time, only two tokens can meet and be merged. The previous work [10] proposed 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 its neighbors also possess token. If any neighbors of this broadcasting node possesses token and received *'findingTk'* will reply using *'ACK_finding'* message. *'ACK_finding'* is an expression of agreement to do merging of tokens. 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, the token is circulated. The message sequence of this process is illustrated in Figure 2.

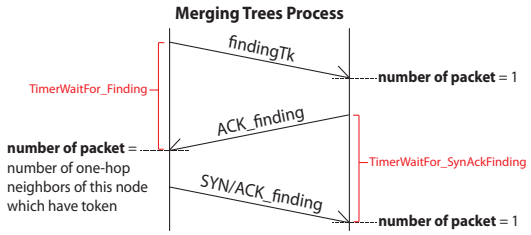

Fig. 2. Message sequence diagram of merging trees

3) Message for Token Traversal: 'Rule4' in Figure 1 stands for token traversal in general case (randomly). When a node sends a broadcast message for finding a neighbor possessing token, it also establishes a timer as addressed in previous section. If the timer finishes and there is no answer from any neighbor, the token movement takes place. If there is no non-tree member

neighbor, the node will directly move the token, see Figure 3.

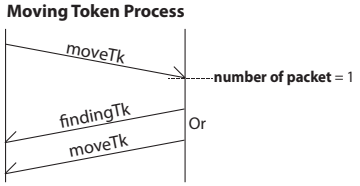

Fig. 3. Message sequence diagram of traversing the token

III. Token Traversal in a Decentralized System

According to DAGRS's rules, initially every node possesses a token which is unique in each spanning tree. In order to merging two spanning trees, the token of each spanning tree must meet and agree to operate in merging process. After this merging process, a bigger spanning tree is created and one token becomes obsolete.

The walk of the token 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 [7]. 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 remain only one tree over a connected subgraph. This section gives a detailed explanation of four strategies used in this study. It is worth noting that all strategies are working in distributed and decentralized manner suiting to work in DTMs.

A. Randomness

The Randomness here follows the uniform distribution law. Randomness is the heuristic used by DA-GRS. The process is done by selecting a node randomly among the list of neighbors. The moving token operation using 'Randomness' is described in Algorithm 1.

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

1: α is the set of neighbors of node ν
2: node ρ is a node selected randomly from set α
3: move token τ_i from node ν to node ρ

B. TABU-like

TABU-like [11] includes a list (called Tabu-like list) in the token of forbidden movements in which the most recent nodes possessing the token are stored, so that the algorithm does not visit that possibilities repeatedly.

Each token has a list of visited nodes, the tabu-like list. This idea was proposed in the previous work [5], and it can be seen in Algorithm 2. The TABU-like utilizes a limited memory size in each token.

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

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

In this work, the *memory_size* of one is considered according to the result in [5]. For brevity, henceforth we will use 'TABU-like{1}' to represent the usage of TABU-like at *'memory_size'* equals to one. This is equivalent to not allowing sending the token to the node from which the current one received it. It is remarkable that the size of information in TABU-like list affects directly to the bandwidth usage in the network.

C. Depth-First Search (DFS)

DFS is commonly used as token movement technique [1], [12], [13] when dealing with tree based topologies. It imitates the traversal of the classical Depth First Search algorithm and, thus, is an ordering traversal. This implementation was done according to the idea used in [1]. In order to traverse systematically like classical algorithm in distributed and dynamic systems, DFS utilizes the neighbor list information provided by the beaconing process.

In this implementation, it is necessary to keep information of the first node that sends the token to the current device (henceforth, we refer to this first node as 'upper neighbor'), and to keep also information of neighbors receiving token from this current device. In this way, the node will definitely sends the token to all its neighbors. Whenever the current node receives token back from its neighbors (and this is not the first time this node receiving token), the current node will send 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 send token to the first neighbor of the

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

D. Depth-First Search–Like (DFS-like)

DFS-like is introduced by imitating the traversal of the classical Depth First Search algorithm (explained above) but also includes some randomness. This is done in the sense that every branch is explored in depth, but it is not necessary to visit all the neighborhood before backtracking the node. Each node records the choice of node which token was sent to. If all neighbors are in the DFS-like list, then all information in DFS-like list is cleared and one select any node randomly. Then, records this node to the end of the DFS-like list. The moving token operation using 'DFS-like' heuristic is described in Algorithm 4.

E. Memory Usage

A remarkable different point among those four heuristics is the usage of memory. While in Randomness, there is no memory usage, DFS, DFS-like and TABU-like utilize either memory on node or in the token.

However, DFS, DFS-like and TABU-like utilize different kind and different size of memory. In DFS and DFS-like, the list is stored inside the node. The difference between DFS and DFS-like regarding the memory usage is that in DFS the mac address of the 'upper neighbor' must be also stored, that means 6 extra bytes is used. For TABU-like, each token carries a number

Algorithm 4 Using DFS-like heuristic in $Move\_Token$ (τ_i) process of a node ν

1: α is the set of neighborhood of node ν
2: β is the DFS-like list in node ν
3: δ is node that sends τ_i to ν
4: Set $availableNode$ = α - β - δ
5: **if** $availableNode \neq \emptyset$ **then**
6: node ρ is a node selected randomly from set $availableNode$
7: move token τ_i from node ν to node ρ
8: add ρ to the end of list β
9: **else**
10: clear list β
11: node ρ is a node selected randomly from set α
12: move token τ_i from node ν to node ρ
13: add ρ to the list β
14: **end if**

of information depending on its *memory_size* and sends over the network. Thus, the algorithm costly utilizes the communication bandwidth. However, the previous work [5] shows the benefit of TABU-like with *memory_size* equal to one. This means the bandwidth usage is small and equal to 48 bits as overhead. Hence, the usage of TABU-like{1} is still convincing to the current study.

IV. Experiment Methodology and Measurements

A. Experiment methodology

The networks used in this work were generated by Madhoc [14], 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 I.

TABLE I
Parameters used in the experiments

	Shopping Mall	High way
Surface (km^2)	0.32	1.0
Node Density (per km^2)	1000	80
Number of Nodes	100	80
Avg. Number of Partitions	2.68	1.7
Number of Connections	389	405
Average Degrees	7.82	10.17
Velocity of Nodes (m/s)	0.3-3	20-40
Radio Transmission Range	40-80 m	

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 $t \in \{t_1, t_2, t_3, ..., t_{40}\}$. 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.

B. performanceRatio() function

At 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 number of trees per connected subgraph. This quality is determined by the following ratio.

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

The value of approaching to one means higher quality of the constructed tree. Having a spanning tree per a connected subgraph enables more efficient communication and management, since at least, information can be disseminated systematically via the created spanning tree. This means the algorithm is robust with respect to the dynamism of the network because it can construct a tree covering the connected subgraph.

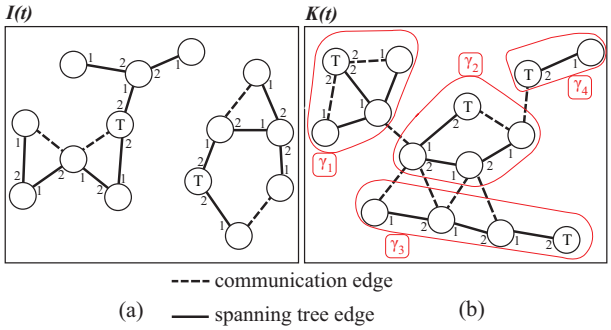

Fig. 4. An example scenario for illustrating the proposed cost functions for spanning forest

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 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 4, respectively.

C. convergenceSpeedRate() function

The $convergenceSpeedRate()$ is measured based on the number of iterations in simulation. Let Δ be the number of iterations the algorithm required trying to achieve the least $performanceRatio()$ and Δ^* 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 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 Δ. The lower the value of $convergenceSpeedRate()$ is, the faster the algorithm can converge a connected component into a tree. The $convergenceSpeedRate()$ can be written as below.

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

V. Results

In this section we present the comparison results of the four strategies for circulating the token in a decentralized tree based algorithm. These four strategies are: randomness, TABU-like$\{1\}$, DFS and DFS-like. 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 100 runs.

In the shopping mall environment, we can see that DFS and DFS-like have the best behavior among these four strategies, see Figure 5. Between them, DFS is converging a little bit faster than the other, and also has less number of trees per connected component, but the difference is insignificant.

Figure 6 shows results obtained from the highway scenario. In this environment DFS and DFS-like also outperforms the other strategies without much difference between themselves.

In any of the previous cases, the Randomness strategy is the worst one always. This behavior was expected, since using this random technique many nodes in the tree can hardly possess the token, so the merging process in those area is rarely happened. TABU-like$\{1\}$ improves Randomness because it ensures that one neighbor will

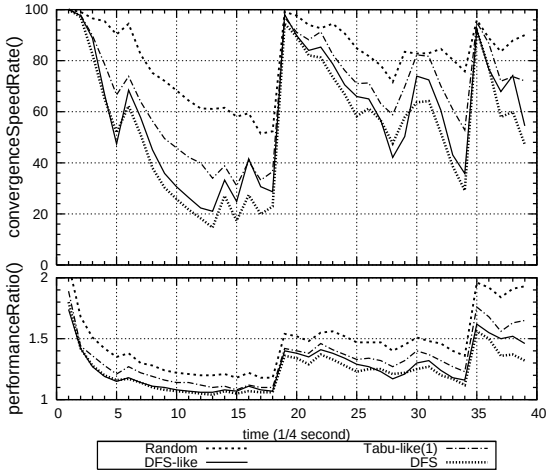

Fig. 5. Comparison of convergenceSpeedRate() measuring among all studied algorithms in 'Shopping Mall' mobility model

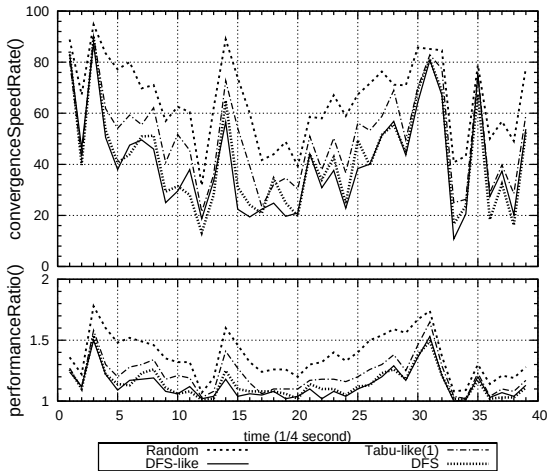

Fig. 6. Comparison of convergenceSpeedRate() measuring among all studied algorithms in 'highway' mobility models

not possess the token twice consecutively. This affirms the intuition that a technique which traverses all the nodes in the tree usually provides better results.

As the values do not 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 there is no significative difference between DFS and DFS-like either in any of the environments (mall and highway) or the two parameters measured (speed of convergence and performance ratio), but there are many cases in which both are significative better than TABU-like$\{1\}$ or Randomness. Randomness is statistically the worst strategy.

VI. Conclusion and Future Work

Providing efficient communication in delay tolerant mobile ad hoc networks is a difficult task which presents a real challenge. In this work, four different strategies for token movement through the tree topology: Randomness, TABU-like$\{1\}$, Depth-First Search (DFS) and Depth-First Search-like (DFS-like) were systematically studied and compared in terms of the perfomance ratio and the speed of the convergence. The performance ratio measures the number of spanning trees per connected component, the closer to one the better performance. The speed of convergence 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 in the partition.

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 both DFS and DFS-like outperform the no ordering ones like randomness and less ordering such as TABU-like$\{1\}$. We can suspect that including some randomness in an ordering strategy may help to reduce the convergence speed in some cases (This is the difference between DFS and DFS-like). Although, the differences between DFS and DFS-like are not significant in any case.

As future work we plan to study the impact of these techniques to the tree as such is used by any high level application, 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.

References

[1] 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.

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

[3] H. Lim and C. Kim, "Multicast tree construction and flooding in wireless ad hoc networks," in *3rd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2000, pp. 61–68.

[4] 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.

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

[6] 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.

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

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

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

[10] A. Piyatumrong, P. Bouvry, F. Guinand, and K. Lavangnananda, "Trusted spanning tree for delay tolerant MANETs," in *2008 IEEE/IFIP International Symposium on Trust, Security and Privacy for Pervasive Applications (TSP-08)*, vol. 2, December 2008, pp. 293–299.

[11] F. Glover and M. Laguna, "Tabu search," in *Modern Heuristic Techniques for Combinatorial Problems*, C. Reeves, Ed. Blackwell Scientific Publishing, 1993, pp. 70–150.

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

[13] 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.

[14] 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.