



A Study of Token Traversal Strategies on Tree-Based Backbones for Mobile Ad Hoc - Delay Tolerant Networks

Apivadee Piyatumrong, Pascal Bouvry
Université du Luxembourg,
The Faculty of Science, Technology and
Communication (FSTC),
CSC, Luxembourg
Email: apivadee.piyatumrong@uni.lu,
pascal.bouvry@uni.lu

Frédéric Guinand
Le Havre University,
LITIS, Le Havre, France
Email: frederic.guinand@univ-lehavre.fr

Kittichai Lavangnananda
School of Information Technology,
King Mongkut's University of
Technology Thonburi,
Bangkok, Thailand
Email: kitt@sit.kmutt.ac.th

Abstract—Tree-based backbone establishment and maintenance in Mobile Ad hoc - Delay Tolerant Networks is often operated through the use of traversing tokens. A study and framework are proposed here for various token traversal strategies on tree-based backbones. The proposed strategies execute in distributed and purely decentralized manner, and require only 1-hop knowledge. Aiming at providing the highest robust and quality of services, these token-traversal strategies are studied in particular with an algorithm for merging and maintaining the different trees based on the quality of the nodes. For the robustness aspect, the use of a trust-based evaluation framework is assumed and weights the different nodes based on their quality of cooperation. Three cost functions are implemented in order to evaluate the trust based framework proposed, including another function for evaluating tree-convergence time. Results and comparison charts are provided to illustrate the trade-off between the various strategies in terms of performance, cost (memory and communication) and robustness.

Keywords-Purely Decentralized and Distributed Algorithms; Wireless and Mobile Techniques for Delay Tolerant; Token-Based Distributed Algorithm; Cooperative Networks; Spanning Forest

I. INTRODUCTION

Mobile Ad Hoc - Delay Tolerant Networks (mobile ad hoc DTNs) constitute an emerging subclass of mobile ad hoc networks (MANETs) that feature frequent and long duration partitions. Furthermore, this kind of network evolves quickly and unpredictably. Hence, the most vital features, which applications working on mobile ad hoc DTNs need, are the flexibility and survivability of the whole system. These characteristics render the centralized approaches, such as a dedicated node, inefficient to work in such an environment. Therefore, mobile ad hoc DTNs need self-configuring, decentralized and robust algorithms to cope with the dynamic and the partitioned nature of the environment. This implies that the solution provided to the network must be done locally but yet effective globally.

Operations within ad hoc networks rely on cooperation between nodes. A number of research are dedicated to the

cooperative enforcement approaches. These research [1], [2] aim at enhancing the robustness, the availability and/or the overall throughput in a purely ad hoc network. This cooperation scheme tries to cope with 'selfish nodes'. Such nodes can deteriorate the robustness (efficient communication and survivability) of the network since they do not give collaborative efforts. In these approaches, the terms of *trust* and *reputation* are used to represent the *cooperative level* toward other stations in the community. This implies that the node having high trust level is more likely to cooperate and accomplish the task or the request which has been placed on it.

Recently, the paper [3] exploits the existence of such trust / reputation level of each station, and proposes 'Greedy Trusted Spanning Tree' (G-TRUST) algorithm for creating of robust trusted spanning trees (tree-based backbone) in mobile ad hoc DTNs. In this system, it tries to push the lower trust level nodes to the leaves of the tree. Since the nodes staying at leaves position do not relay any information for their neighbors, the higher communication success rate can be expected. However, this is true only if a spanning tree of the network, or a spanning tree of a connected component, is available. This means there must be a spanning tree covering entirely a connected component.

Since the spanning tree establishment in distributed system is often operated through the used of tokens, and, the paper of [4] suggests that token behavior may have an impact on the time required to build spanning forests (many spanning trees in the same network). Moreover, stating in [5] that techniques for traversing the token which perform well in static networks are not necessarily well suited in networks with high mobility. Furthermore, with closely studying the mechanism of G-TRUST, we foresee that the token behavior has also an impact on the robustness of the spanning forest. These motivate us (1) to investigate a new token traversal in high mobility network using G-TRUST algorithm, and, (2) to determine a suitable strategy, such that the trusted spanning trees (created by G-TRUST) are robust to the dynamic

features of the underlying mobile ad hoc DTNs. Hence, the objectives of this work are to enhance the robustness metrics, the performance ratio and convergence speed rate (described in details later in Section IV and Section V) on the whole networks by focusing on improving the token traversal behavior in trusted spanning forest approach.

This work studies a number of possible heuristics to be used for token traversing in purely distributed and decentralized manner of G-TRUST algorithm, and in a dynamic network like mobile ad hoc DTNs. The aim is to study the trade-off among these heuristics in order to choose the most suitable heuristic to be used for different setting of networks using G-TRUST. The following section describes related works and notions of mobile ad hoc DTNs, and G-TRUST in more details. Section III gives the details of all heuristics used in this study. The definitions of all cost functions, the measurement mathematical models, are described in Section IV. The experimentation methodology, results and analysis are included in Section V. Finally, the study is concluded in Section VI and the future work is suggested in Section VII.

II. RELATED WORKS

A. Mobile Ad Hoc - Delay Tolerant Networks (mobile ad hoc DTNs)

Mobile Ad Hoc - Delay Tolerant Networks (mobile ad hoc DTNs) are fluctuating networks populated by a set of moving materials equipped with wireless communicating devices. They are mobile, ad hoc configuring, and frequently partitioned. At a given moment, two stations belonging to distinct partitions can neither communicate directly nor indirectly (using multi-hop communications) [6], [7].

In this kind of network, each station can reach a subset of the other stations using wireless communication abilities, if possible. Such communication ability is typically defined by a communication range and may be constrained by natural obstacles (e.g. walls, building, etc.). The network can be represented by its dynamic communication graph. At a given moment t , the communication graph, $G(t)$, of such a network is a pair $(V(G(t)), E(G(t)))$, where $V(G(t))$ is a finite set of elements, called vertices, $E(G(t))$ is a binary relation on $V(G(t))$ - a subset of pairs of elements of $V(G(t))$. The elements of $E(G(t))$ are called edges and constitute the edge set of $G(t)$. An edge between node x_i and x_j indicates that, at time t , it is possible for x_i and x_j to exchange information. At moment t , $G(t)$ may be partitioned into a set of m disjoint connected subgraphs.

B. Original algorithm of G-TRUST: Dynamicity Aware-Graph Relabeling System (DA-GRS)

DA-GRS [8] is a model introduced for the conception and the analysis of decentralized applications and algorithms targeting dynamically distributed environments like MANETs or mobile ad hoc DTNs. DA-GRS proposed and

used an algorithm, ‘spanning forest’, for constructing and maintaining a spanning forest in DTNs, by relying on careful rules-based token management. Henceforth this concept will be referred to as ‘original algorithms’ for brevity. The work in [8] described rules to handle four different scenarios, (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 (this original algorithm refers to this event as the synchronization method), and, (d) tokens traversal in general case (This original algorithm use randomness as a token traversal strategy). These rules can be viewed in DA-GRS’s visual representation as illustrated in Figure 1.

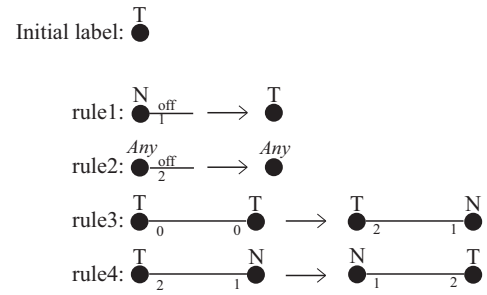


Figure 1. The four rules of the original algorithm [8]

The re-labeling idea of DA-GRS can be explained as follows. The circle represents a node. ‘T’ refers to token. When this ‘T’ sign appears above a node, it means that node is holding a token. The number ‘0’ on an edge means the communication edge is not a part of spanning tree yet. The number ‘1’ and ‘2’, which is attached to each ‘spanning tree edge’, gives the pointer to the direction of token location. The number ‘2’ means the token is behind this node or this node is parent (parent-child tree), in contrast, number ‘1’ means the token is not behind this node or this node is a child. The word ‘off’ represents the breakage situation of the communication edge.

According to the rules of DA-GRS, at one moment in time, only two tokens can meet and be merged at any instance. This synchronization method is known as ‘rendezvous assumption’ [9]. One remark about this synchronization method in a dynamic communication graph is that, if we consider the communication space, several synchronizations may happen simultaneously at different locations.

C. Greedy Trusted Spanning Tree (G-TRUST)

‘Greedy Trusted Spanning Tree’ or G-TRUST [3] is a distributed algorithm constructing trusted spanning tree in a dynamic network like mobile ad hoc DTNs. This algorithm attempts to create robust spanning forest by relying on the cooperative or trust level of each neighbor. The higher trusted nodes tend to give a higher quality of service,

therefore they tend to be located inside tree so that information can be assured the successful communication. Excepting the fact that G-TRUST utilizes trust/cooperative level of nodes, it is an extension of the original algorithm. In G-TRUST the rendez-vous assumption of the original algorithm is relaxed. Thus, in G-TRUST, several tokens can meet simultaneously. Furthermore, by relying on ‘Greedy’ heuristics, G-TRUST can select the highest trusted node to merge when applicable. The merging operation in G-TRUST is described in algorithm 1.

Algorithm 1 Look for other tokens around token τ_i

```

1:  $\tau^{best}$  is token owned by the most trusted neighborhood
2: if  $\tau^{best} \neq \emptyset$  then
3:   Merge_With( $\tau_i, \tau^{best}$ ) //merge the two tokens
4: else
5:   Move-Token( $\tau_i$ ) //continue to move the token randomly
6: end if

```

Indeed, the advantage of G-TRUST occurs when several tokens meet. It allows each node owning a token to choose the most trusted neighbor to merge.

III. CONSTRUCTING AND TRAVERSING TRUSTED SPANNING TREE IN DTMS

Aiming at constructing spanning trees (forest), both original and G-TRUST algorithm utilize a number of rules for token management. A token, which is initially possessed by each node, is unique within each spanning tree. In order to merge two spanning trees, token of each spanning tree must meet and agree to operate merging process. After this merging process, a bigger spanning tree is created and one token becomes obsolete in order to remain a unique token in a new bigger tree. Later, the token moves to another node within the same tree in order to find other tokens belonging to other spanning trees whose merging process reoccurs.

These activities, occurring during spanning tree construction and maintenance, may be categorized into three different scenarios, (a) handling the partitioning of the communication graph, (b) merging operation when tokens meet (synchronization process), and (c) token traversal/circulating in a spanning tree. G-TRUST has been proposed to improve quality of the created tree by dealing with the merging operation approach (using greedy method). The objective of this paper is to improve the quality, the robustness, and the performance of the created spanning trees (spanning forest of a communication graph) via enhancing *token traversing approach*.

The walk of the token impacts 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 [10]. In the context of this study, we want the token to traverse less but has more chance to meet another token (where the direct effect is a bigger tree and higher

performance ratio). In other words, we want the fastest convergence 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 (the ideal performance ratio equals to one). At the same time, high robustness of the created spanning tree (in terms of high availability based on cooperative/trust level) is also desired.

In this section, we describe all heuristics used for studying the traversing trusted spanning tree. These heuristics work in a purely decentralized and distributed manner. They are Randomness, DFS and TABU-like, and are described below.

A. Randomness

The Randomness here follows the uniform distribution law. Randomness is the heuristic used in the original algorithm by selecting randomly among the list of neighbors. Hence, this heuristic can be used as the lower bound to compare with other proposed heuristics. The moving token operation using ‘Randomness’ is described below.

Algorithm 2 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  $\rho$ 

```

B. Depth-First Search (DFS)

In order to reach every node in a spanning tree efficiently, Depth-First Search concept is proposed here for token movement. The applied DFS can be found used in vary purposes [11], [12], [13] in the context of MANETs. Depth-First Search is an algorithm for traversing or searching a tree or graph. One starts at one node and explores as deep as possible along each branch before backtracking at the leaves.

In this work, DFS imitates the traversal of the classical centralized Depth First Search algorithm from literature at the global point of view. However, since we cannot achieve global knowledge in mobile ad hoc DTNs, the DFS itself utilizes a number of memory in each node, and working in purely distributed and decentralized manner. In this work, DFS is applied in every node of the communication graph, and utilizes the neighbor list information provided by the beaconing process. In this implementation, it is necessary to keep information 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 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 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

```

C. TABU-like

TABU-like is an adaptation from Tabu search [14], a metaheuristic algorithm. Tabu search uses memory structures to enhance the performance of a local search method, once a potential solution has been determined, it is marked so that the algorithm does not visit that possibility repeatedly.

TABU-like list, belongs to each token, is a list of visited nodes. This idea was originally mentioned in [15] and exposed in [4], and can be seen in algorithm 4. In contrast to the DFS, the TABU-like utilizes a limited memory size in each token. In this work, the $memory_size$ of one is considered according to the result in the previous work. For brevity, henceforth we will use ‘TABU-like{1}’ to represent the usage of TABU-like at $memory_size$ equals to one. The longer the list, the higher number of different nodes that token visits already and has been recorded. It is remarkable that a size of information in TABU-like list affects directly to the communication bandwidth usage in the network.

D. Memory Usage

A remarkable different point among those three heuristics is the usage of memory. While Randomness has no memory usage, both DFS and TABU-like utilize either memory on node or in the token. Employing some memory in DFS and TABU-like should enhance the decision making on the next

Algorithm 4 Using TABU-like 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

```

move. This should achieve higher coverage area of the token traversal, and, as a consequence, bring faster convergence of spanning trees over a communication subgraph.

However, DFS and TABU-like utilize different kinds and different sizes of memory. In DFS, the list is stored inside a mobile node. Although, the memory of each mobile node is limited, the current trend of memory in mobile node is either increasing its built-in memory size or offering a memory card slot for the liberty of extending the memory. However, in a very limited resource such as sensor device, this might not be manageable. For TABU-like, each token carries a number of information depending on its $memory_size$ and send over the network. Thus, the algorithm costly utilizes the communication bandwidth. However, the previous work [15] shows the benefit of TABU-like with $memory_size$ equal to one. This means the bandwidth usage is very small (in this case, the $memory_size$ is one meaning that only one MAC address is recorded). Hence, the usage of TABU-like{1} is still convincing to the current study.

Although, the Depth-First Search and Tabu list are not a new novelty, the usage of both heuristics in a dynamic communication graph like mobile ad hoc DTNs and the implementation in purely distributed and decentralized environment in this work had never been done in this context.

IV. ROBUST TRUSTED SPANNING TREE AND FOREST IN MOBILE AD HOC DTNS

Nodes with higher trust level are more likely to be able to complete their tasks than lower ones. Furthermore, having nodes with low trust levels localized on leaves is advantageous since they would not be responsible for forwarding information to others. Moreover, losing them at these positions has little effect on the overall structure. Hence, robust trusted spanning tree is a tree which has

low trusted nodes as leaves or as close as possible. The number of tree(s) over the connected subgraph is equally important. An ideal situation is when there exist a single trusted spanning tree in a connected subgraph. This allows us to assess how robust the algorithm does and how fast the algorithm can construct and/or recover the tree from the dynamic communication graph at any moment t .

Trust level of a node n , denoted by $trust(n)$, where $trust(n) \in Z^+$, defines the levels of quality of service it can provide. Whether a node n can be trusted is determined by a given threshold. Let $\Theta_t = \{n' \in V_t(G) | trust(n') \leq threshold\}$ be the set of all low-trustable nodes at moment t . A node in a cooperative network can have low level of trust for various reasons such as low battery, poor communication signal, moving out of communication range, etc, but finally resulting in communication failure.

In order to determine robust trusted spanning trees, the previous work [3] introduced quality measurement for a trusted spanning tree by means of two cost functions. In order to assess the quality of overall network or the trusted spanning forest, this work adopts the idea of those two cost functions and introduces three new cost functions for measuring the spanning forest. These cost functions are $performanceRatio()$, $weight()$, and $isolateLowTrustedNode()$

In order to assess the quality of the created trust spanning forest, the value of functions from different studied heuristics will be compared, where a higher value for all these three cost functions indicate a superior quality. All the example scenarios used in this work applied the labeling idea from DA-GRS. The letter and number appeared in each node present the node's id and trust value, respectively.

A. $performanceRatio()$ 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 connected subgraph divided by number of trees. This quality is determined by the following ratio.

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

The value of approaches to one means higher quality of the constructed forest. Having a trusted 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 dynamic of the network because it can construct a tree covering the connected subgraph.

Figures 2(a) and (b) illustrate the measurement of all cost functions proposed here. In the figure 2(a), the communication graph $I(t)$ has two connected subgraph, and each connected subgraph has one spanning tree. On the

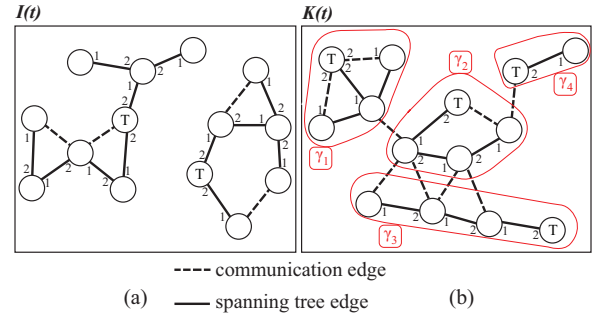


Figure 2. An example scenario for Illustrating the proposed cost functions for Spanning Forest

contrary, the communication graph $K(t)$ depicted in figure 2(b) has only one connected subgraph but four spanning trees ($\gamma_1, \dots, \gamma_4$). Thus, the $performanceRatio(I(t))$ and $performanceRatio(K(t))$ equal to 1 and 0.25, respectively.

B. $weight()$ function

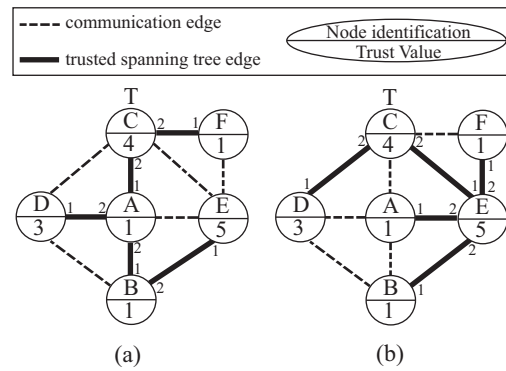


Figure 3. An example scenario for Illustrating the details of $weight()$ and $isolateLowTrustedNode()$ cost functions

Nodes with higher trust level are more likely to be able to complete their tasks than lower ones. The $weight()$ function introduced in previous work [3], can be used to assess trust spanning trees with respect to this objective. Having $V(\gamma)$ as the set of all nodes in a spanning tree γ , the $weight()$ function of a trusted spanning tree can be determined by the following equation:

$$weight(\gamma) = \sum_{x \in V(\gamma)} trust(x) * tree_degree(x) \quad (2)$$

The function $tree_degree(x)$ represents the number of one hop neighbors of node x in the tree. Figures 3(a) and (b) are examples to illustrate how the $weight()$ function can assess this quality where the threshold used in this example is equal to one. In Figure 3(a), the node with lowest trust level gets the highest $tree_degree$, while the node with highest level gets the lowest $tree_degree$ (i.e the node A has a trust level of 1 and $tree_degree$ of 3, while

the node E has a trust level of 5 and *tree_degree* of 1), hence the $weight(\gamma_a)$ function for this trusted spanning tree is 22. Figure 3(b) depicts the opposite (i.e. the node with the highest trust level possesses the highest *tree_degree* (node E), while the node with the lowest level possesses the lowest *tree_degree* (node E)). The $weight(\gamma_b)$ function for this trusted spanning tree is 34. In order to measure this function on the whole graph (the spanning forest), equation 3 is introduced below.

$$weight(G(t)) = \sum_{\gamma \in G(t)} weight(\gamma) \quad (3)$$

Based on the Figure 2, $weight(I(t))$ gives 83, while $weight(K(t))$ gives 66.

C. *isolateLowTrustedNode()* function

Since low trust level nodes have tendency to break away from the network, allowing them to have high degrees in the tree will increase the likelihood of disconnection from the spanning tree. Therefore, in order to minimize the re-connecting task, nodes with lowest trust levels should be assigned the lowest *tree_degree* position in the tree. The functions *isolateLowTrustedNode()* is introduced as a mean to assess trusted spanning forest with respect to this objective.

This function evaluates the efficiency of a trusted spanning forest by noting how well it can isolate non-trustable nodes n' where $n' \in V(G(t))$ and $trust(n') \leq threshold$. The function measures the percentile of n' nodes at leaf. The higher value of *isolateLowTrustedNode()* function signifies better quality trusted spanning tree. Let $\Theta^*(\gamma) = \{n' \in \Theta(\gamma) | tree_degree(n') = 1\}$ be the set of low trustable nodes being leaves in the tree γ . The *isolateLowTrustedNode()* function is defined by $isolateLowTrustedNode(\gamma) =$

$$\left(\frac{|\Theta^*(\gamma)|}{|\Theta(\gamma)|} \right) * 100 \quad (4)$$

Hence, the *isolateLowTrustedNode()* value for Figure 3(a) is 33.33% while this value is 100% for Figure 3(b). Equation 5 is introduced below as a mean to measure the same objective in a spanning forest. The result given by this equation for Figure 2(a) and (b) equal to 0% and 100% respectively.

$$isolateLowTrustedNode(G(t)) = \left(\frac{\sum_{\gamma \in G(t)} |\Theta^*(\gamma)|}{\sum_{\gamma \in G(t)} |\Theta(\gamma)|} \right) * 100 \quad (5)$$

The more detailed explanation is as follows. Both graphs $I(t)$ and $K(t)$ have the same number of node and the same proportion of trust value of nodes, but different forest topologies. Within each graph, there exists one low trusted node according to the given threshold value. The location of this low trusted node (y) in $K(t)$ is good at this time t

since it is at leaf of tree γ_4 , while the low trusted node (x) in $I(t)$ is in middle of a tree, having two trusted spanning tree edges.

However, the *performanceRatio()* of $K(t)$ is 0.25, which means the current good value of *isolateLowTrustedNode()* is not valid to the objective. If there are only few changes in the forest topology and node y and z of $K(t)$ can merge in several time steps later, *isolateLowTrustedNode(K(t))* will become 0%. This case shows the impact of function *performanceRatio()* has on other cost functions.

V. EXPERIMENTATION AND RESULTS

A. Experiment methodology

Suitable networks for simulation of any mobile ad hoc DTNs ought to comprise lay-out of nodes (e.g. citizens), environmental properties and radio propagation (communication link) which reflect real-world situations. The networks used in this work were generated by Madhoc [16], 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', are 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	2.20
Node Density (per km^2)	1100	70.4
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	

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 $t \in \{t_1, t_2, t_3, \dots, t_{400}\}$. 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.

Using 'GraphStream' [17] to replay the simulation traces and relying on *Beaconing Rate* of IEEE802.11 [18], 100 milliseconds is used as time-interval to send a beacon. This beacon also carries the movement of the token in this work. Hence, the movement of token has no cost in this respect. However, the cost occurs from the merge operation in the spanning tree construction. Since the work uses UDP packets, in order to do uni-casting, the protocol needs to ensure the delivery in merge operation. This can be done using ACK and SYN/ACK message as shown in Figure 4.

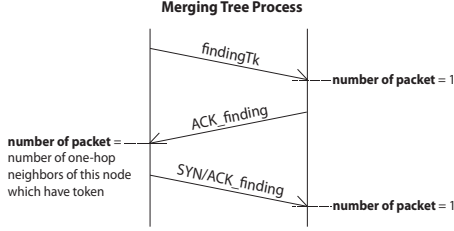


Figure 4. Message Sequence Diagram of merging trees process used by G-TRUST algorithm

This limited Beaconing Rate drives the launching of new static graphs $G(t_{next})$. As a consequence, the number of merging trees process is limited too. This fact surely gives impact to the simulation results. The $convergenceSpeedRate()$ is measured based on the number of iterations spending in simulation and present the time used in converging a spanning tree over an existing connected subgraph. Let Δ is the number of iterations the algorithm required in trying to achieve the least $performanceRatio()$ and Δ^* is the number of iterations it required for a particular $G(t_i)$. Having $performanceRatio()$ equal to one within $G(t_i)$ is an ideal situation. However, having limited merging process provides no guarantee that $performanceRatio()$ will be one, in other words, it is always possible to have multiple trees per connected component at any time t_i of graph G . In such case, the number of iterations used within that $G(t_i)$ will be counted into Δ . The lower the value of $convergenceSpeedRate()$ is, the faster the algorithm can converge a connected component into a tree. Thus, the trusted spanning trees are prompt faster to provide efficient communication and management; information can be disseminated systematically via the created tree. The $convergenceSpeedRate()$ can be written as below.

$$convergenceSpeedRate(G(t)) = \left(\frac{\Delta(G(t))}{\Delta^*(G(t))} \right) * 100 \quad (6)$$

B. Results and Analysis

This work assumes 5 different levels of $trust(n)$ where $trust(n)$ equal to 1 is the lowest trust level. The $threshold$ value, which is used to determine the acceptable level of cooperativeness of any node, is equal to one. This means any node having trust level equal to one is a low-trustable node, and is undesirable to be place in the middle of tree. Please note that the methodology in collecting and computation of trust value is out of scope of this study. The results presented in Figures 5 and 6 provided the comparison for the 3 main strategies studied in the two real-world mobility models, 'Shopping Mall' and 'Highway' mobility model respectively. The strategies are Randomness (as a lower bound), DFS and TABU-like{1}. The simulations were done using 100 runs per heuristic per mobility model. Since the

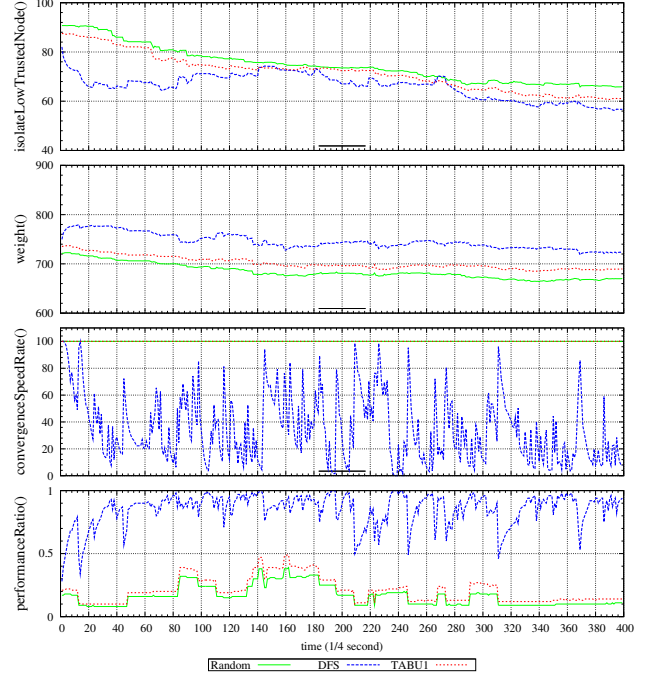


Figure 5. Comparison of performanceRatio(), convergenceSpeedRate(), weight() and isolatingLowTrustNode() measuring among all studied algorithms in 'Shopping Mall' mobility model

duration of every simulation is 100 seconds or 400 time steps, the result of each run was the average result occurred within the simulation time. Hence, the final result is the average result value from those 100 runs.

Since, the ideal $performanceRatio()$ is at one, which means there exists only one tree within one connected component, both Figures 5 and 6 show the best performance of DFS strategy. Indeed, among the three strategies, DFS achieved significantly higher quality in all aspects except for isolatingLowTrustNode() in both mobility models.

From the results, DFS can gives less number of spanning trees than the other heuristics. This means DFS create a bigger trusted spanning tree and thus $weight()$ value can be expected to be higher than several smaller trusted spanning trees. While the other two cannot merge or collect spanning trees efficiently, there is more chance for low-trusted nodes on each small spanning tree to be at leaves because of the G-TRUST algorithm itself. Hence, the cost function on isolatingLowTrustNode() can give a good result when there are more spanning trees in a forest, which is contrast to the objective of $performanceRatio()$ and this study.

To summarize, DFS achieved the best $performanceRatio()$ among all three algorithms in both mobility models. This means using DFS heuristic as a way to conduct the movement of token is the most efficient way for merging all nodes in the same connected component to be in the same tree. $convergenceSpeedRate()$ shows

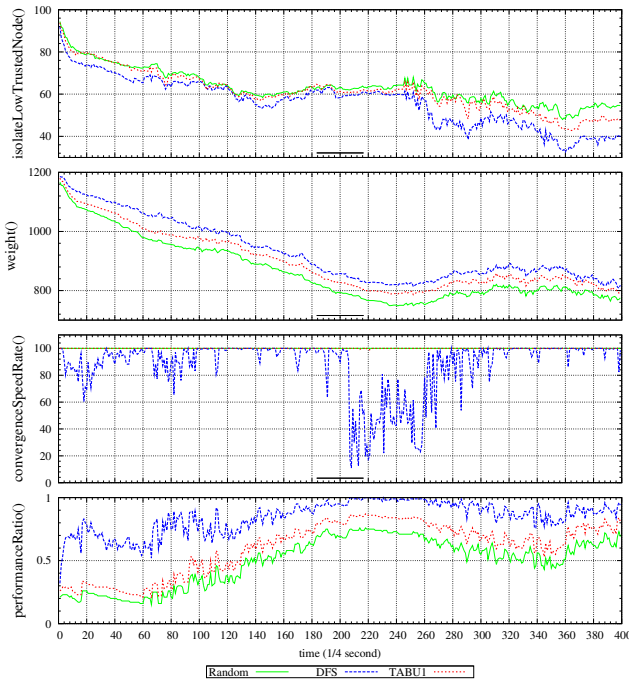


Figure 6. Comparison of $performanceRatio()$, $convergenceSpeedRate()$, $weight()$ and $isolatingLowTrustNode()$ measuring among all studied algorithms in 'Highway' mobility model

how fast all three strategies can influence the convergence of a tree on a connected component of a network in both mobility models. From both figures, it is apparent that DFS uses lower rate compare to other heuristics. Hence, it can be concluded that DFS can create trusted spanning forest faster than other strategies.

VI. CONCLUSIONS

For Mobile Ad hoc - Delay Tolerant Networks, maintaining a high quality of service is a real challenge due to its highly dynamic nature and appearance of partitions. By using a cooperative enforcement paradigm to represent the cooperative level of nodes in network, trusted spanning forests are created and used as backbones, composed of the most trusted nodes, to provide high availability of tree-base backbone at any time. In order to truly operate on mobile ad hoc DTNs, the operations need to be decentralized, self-configuring and robust. A heuristics, G-TRUST, has been proposed in order to construct global robust spanning forests. This algorithm relies on a token traversal mechanism. The behavior of such traversals influences the choice and quality of the constructed forest.

The contributions of the present paper consist in (1) a design and implementation of a token-traversal, DFS, for G-TRUST, (2) implementation of TABU-like and a framework for studying the various token-traversal strategies used in trust-based backbone establishment and (3) assessing their

quality and giving analysis of the results. In particular, two different approaches on information storage (i.e. considering token and/or node memory) emerged. They were represented by the Depth First Search algorithm, named DFS, and a Tabu search based algorithm, named TABU-like, for token traversal mechanism in G-TRUST. These DFS and TABU-like heuristics are based on classical concepts but are, in this version, totally decentralized heuristics in order to adapt to mobile ad hoc DTNs requirements. This work studied the effect of these heuristics towards the robustness of the trusted spanning forest versus their costs (convergence speed, performance ratio and memory).

The experimentation results showed that DFS yields better results both in terms of robustness and convergence speed when applying those decentralized heuristics with G-TRUST. Also, there is no additional cost of communication, since the algorithm relies on the existing token exchange. However, as stated earlier in Section III, DFS does utilize memory within mobile nodes where the size of this memory usage depends on the size of $tree_edge$ of a node. Thus, this might not be applicable with very small memory nodes in a very dense network: in that case the Tabu search is more suitable.

VII. PERSPECTIVES

The experimentation results raises the trade-off between having a high $performanceRatio()$ and having many strong robustness trees but formed into smaller size. It must be emphasized here that the real objective of this work is to enhance the quality of service by means of managing robust trusted spanning forest (to have high availability of backbone at any time). This suggests the necessity for the robustness measurements based on the whole communication graph. However, according to the result, it reveals that the trusted spanning forest management in different levels might give a promising solution to the problem. By having many small size, but strong robust spanning trees, the node selected among the strongest trusted nodes of each robust spanning tree may act as connecting points and collaborate with other nodes of the neighbor trusted spanning trees. These merit further investigation on the dynamic mobility pattern and the sub-structure volatility in the future work. We also believe that the choice and the tuning of local heuristics in mobile ad hoc DTNs shall rely on global information consisting of global network environment parameters (e.g. hostile network or not) and on the existing mobility models/patterns in such networks.

REFERENCES

- [1] G. F. Marias, P. Georgiadis, D. Flitzanis, and K. Mandalas, "Cooperation enforcement schemes for manets: a survey: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 6, no. 3, pp. 319–332, 2006.

- [2] M. Seredynski, P. Bouvry, and M. A. Klopotek, "Preventing selfish behavior in ad hoc networks," in *Congress on Evolutionary Computation (CEC 2007)*, pp. 3554 – 3560, IEEE Computer Society, September 2007.
- [3] A. Piyatumrong, P. Bouvry, F. Guinand, and K. Lavangananda, "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, pp. 293–299, December 2008.
- [4] A. Casteigts, S. Chaumette, F. Guinand, and Y. Pigné, "Distributed maintenance of anytime available spanning trees in dynamic networks." unpublished manuscript.
- [5] 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.
- [6] K. Fall, "A delay tolerant network architecture for challenged internets," *Proceedings of ACM SIGCOMM 2003, Computer Communications Review*, vol. Vol 33, August 2003.
- [7] L. Hogue, *Mobile Ad Hoc Networks: Modelling, Simulation and Broadcast-based Applications*. PhD thesis, University of Le Havre, University of Luxembourg, April 2007.
- [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] L. Cardelli, "An implementation model of rendezvous communication." Lecture notes in Computer Science 197, 1985.
- [10] E. G. Goodaire and M. M. Parmenter, *Discrete mathematics with graph theory*. Prentice-Hall, Inc., 2nd ed., 2002.
- [11] 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)*, pp. 297–303, 2008.
- [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), pp. 131–141, IEEE Computer Society, 2005.
- [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] F. Glover and M. Laguna, "Tabu search," in *Modern Heuristic Techniques for Combinatorial Problems* (C. Reeves, ed.), Blackwell Scientific Publishing, 1993.
- [15] Y. Pigné, *Modélisation et Traitement Décentralisé des Graphes Dynamiques - Application Aux Réseaux Mobiles Ad Hoc*. PhD thesis, L'Université du Havre, December 2008.
- [16] L. Hogue, F. Guinand, and P. Bouvry, "The madhoc metropolitan adhoc network simulator." <http://litis.univ-lehavre.fr/~hogie/madhoc/>.
- [17] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné, "Graphstream: A tool for bridging the gap between complex systems and dynamic graphs," in *EPNACS: Emergent Properties in Natural and Artificial Complex Systems*, 2007.
- [18] "Ieee standard 802.11: Wireless lan medium access control and physical layer specifications." IEEE Computer Society, August 1999.