Projet d'Etude de Recherche
Sujet : "Best-effort group service in dynamic networks" [DKP10]

Proposé et encadré par :
CASTEIGTS Arnaud
CHAUMETTE Serge
JOHNEN Colette

Étudié par :
AMROUCHE Bilal
NEGGAZ Mohammed Yessin
PROUILLAC Adrien

Master Informatique Specialité Réseaux, Systèmes, Mobilités Promotion :
2012/2013
## Contents

### Introduction

1. **Context Presentation**  
   1.1 Technology Context ................................................. 3  
   1.2 Protocol explanation and definition .............................. 6  

2. **Study of the algorithm** ............................................ 10  
   2.1 The algorithm GRP .................................................. 10  
   2.2 Doubts .............................................................. 15  

3. **Problem determination** ............................................. 19
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.2</td>
<td>Deduction</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Implementation and evaluation</td>
<td>21</td>
</tr>
<tr>
<td>3.1</td>
<td>Implementation</td>
<td>21</td>
</tr>
<tr>
<td>3.1.1</td>
<td>Development environment</td>
<td>21</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Experimental context</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Scenarios of tests</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Random Waypoint</td>
<td>22</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Group model</td>
<td>24</td>
</tr>
<tr>
<td>3.2.3</td>
<td>VANET model</td>
<td>25</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Student model</td>
<td>26</td>
</tr>
</tbody>
</table>

Conclusion 29

Bibliography 30

Sitography 31
List of Figures

1.1 MANET Network example. From: http://www.ece.iupui.edu/\protect\ unhbox\voidb@x\penalty\@M\{}dskim/manet/images/adhocnet.gif . . . . . . . 3
1.2 VANET Network example. From: http://www.cs.sjtu.edu.cn/\protect\ unhbox\voidb@x\penalty\@M\{}yzhu/nrl/images/VANET_ILL.jpg . . . . . . . . . . . 4
1.3 Military usage example. From: http://www.onera.fr/conferences/ drones/images/guerre-drones-militaires.jpg . . . . . . . . . . . . . . . . . . . 4
1.4 Group example. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
1.5 Self-stabilisation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.1 Progress of the algorithm on a node . . . . . . . . . . . . . . . . . . . . . 11
2.2 Symmetry verification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
2.3 Lists and views of nodes after the convergence . . . . . . . . . . . . . . . . 12
2.4 Rejection to ensure the safety . . . . . . . . . . . . . . . . . . . . . . . . . 14
2.5 Concurrent acceptance by a group . . . . . . . . . . . . . . . . . . . . . . 15
2.6 The effect of CompatibleList on the maximality . . . . . . . . . . . . . . . 16
2.7 The new CompatibleList test . . . . . . . . . . . . . . . . . . . . . . . . . . 16
2.8 Example of a distributed request . . . . . . . . . . . . . . . . . . . . . . . 17
2.9 Separation and propagation problem . . . . . . . . . . . . . . . . . . . . . 18
2.10 Separation of groups and violation of continuity . . . . . . . . . . . . . . . 19

3.1 Experimental result: Random waypoint model . . . . . . . . . . . . . . . . 23
3.2 Experimental result: Group model . . . . . . . . . . . . . . . . . . . . . . . 25
3.3 Experimental result: Vanet model . . . . . . . . . . . . . . . . . . . . . . . 26
3.4 Experimental result: Student model . . . . . . . . . . . . . . . . . . . . . . 27
Introduction

The article we have studied during the PER project is 'Best-effort Group Service Dynamic Networks : Bertrand Ducourthial, Sofiane Khalfallah, Franck Petit, New York, NY, USA, 2010'. It presents a protocol called GRP, dedicated to mobile ad-hoc networks.

It offers properties such as the self-stabilization and it build stable groups of mobile devices that satisfies a Best-effort requirement to ensure the continuity of service. One constraint is that the diameter of mobile groups is limited by a parameter $D_{max}$.

Our objectives in this project were to understand the protocol properties, to implement it and evaluate its behavior. We report these activities in the present document.

First, we present the context of the article. In this part, we present the authors, the scope of GRP, the state of the art and objectives.

Secondly, the algorithm is detailed. We present the algorithm and explain its expected behavior. Then we discuss some problems we have found in it, followed by suggestions of corrections/improvements.

The third chapter presents the tests we have realized to evaluate the protocol in specific situations, where each situation corresponds to a particular model of mobility. One of the mobility models, the student model, is an original contribution of this work.

Finally, we present the result of our work and conclude on the performances of the GRP algorithm. We also provide some perspective of future work, which we believe are interesting to explore.
Chapter 1

Context Presentation

Contents

1.1 Technology Context ..................................................... 3
   1.1.1 Ad-hoc networks presentation ................................. 3
   1.1.2 Ad-hoc network difficulties ................................. 5
1.2 Protocol explanation and definition ................................. 6
   1.2.1 Dynamic group approach .................................... 6
   1.2.2 Presentation of the GRP problematic ....................... 6
   1.2.3 Property definition ........................................... 7
   1.2.4 Existing solution ............................................. 8

In this part, we study the paper’s subject. First we present briefly the authors. After that, we study the Ad-hoc networks, their problematic, their scope and the open problem of research on such networks. Finally, we present the article context. We present the problematic "best-effort group service in dynamic networks". Then we explain how the protocol works, and we present the characteristics and theoretical limits. We give definition of the protocols properties self-stabilization and best-effort solution.
CHAPTER 1. CONTEXT PRESENTATION

Authors  Title of the paper: best-effort group service in dynamic networks

This paper was published in the proceedings of the 22nd ACM symposium on Parallelism in algorithms and architectures. This conference took place in 2010 on Santorini (Greece). The authors are Bertrand Ducourthial, Sofiane Khalfallah and Frank Petit.

Bertrand Ducourthial, [1], is in charge of the computer systems and networks teaching speciality at the University of Technology of Compiègne (UTC). He received the « Habilitation à Diriger des Recherches» in 2005. He has been full Professor since November 2010. His research work deals with highly dynamic ad hoc networks (such as inter-vehicle networks): networking, distributed algorithms, security, software architecture.

Sofiane Khalfallah, [2], obtained his engineer degree in computer network at INSAT with honours. He did his research master at INSA in Lyon. Finally he did a PhD with Bertrand Dcourthial.

Frank Petit, [3], is currently a professor at University Pierre et Marie Curie Paris 6, is member of Laboratory and he participated LIP6 (UMR 7606) to the INRIA team REGAL. He works on Algorithmic aspects of synchronization, Stabilization and fault-tolerance, Cohort of robots, Dynamic networks

1.1 Technology Context

The main objective of best-effort group service in dynamic networks is to provide a new algorithm for the group management problem in Mobile Ad-hoc networks.

1.1.1 Ad-hoc networks presentation

Device  Most of ad-hoc networks is formed of mobile devices equipped with wireless systems to communicate. If a great number of mobile devices are close and communicate, then disturbances in communication can appear. Some mobile devices have low broadband and low processing power.

Figure 1.1: MANET Network example. From: http://www.ece.iupui.edu/~dskim/manet/images/adhocnet.gif
Ad-hoc networks Nowadays, ad-hoc networks are interesting for research in communication between mobile devices. Such networks are named MANET 1.1 for Mobile Ad-hoc Networks. VANET 1.2 for Vehicle Ad-hoc Networks is a kind of MANET, which deals with a specific problematic of vehicular networks.

Ad-hoc networks can be formed spontaneously. So, if some nodes are sufficiently close to communicate and to share useful information. Then, they can create an ad-hoc network to ensure their application goal. For example, in a VANET network 1.2, if two vehicles are on the same road, they can communicate to keep their distance, or avoid a collision.

Usage area Ad-hoc networks can be used in different situations:

Military scenarios Ad-hoc network can be formed spontaneously between an air-plane drone and soldiers to transmit information about the battlefield and enemy positions.1.3

Rescue operations between an UAV drone and firemen to inform the firemen of forest fire status.

VANET it allows collision avoidance, safety distance maintenance and the determination of alternative route if traffic jam.
Sensor networks to create a synergy between active devices, and passive sensors.

1.1.2 Ad-hoc network difficulties

Ad-hoc network does not have a predefined infrastructure. This means all nodes are independent. So, in a group application, all nodes cannot necessarily communicate with all group members. Generally nodes are used also to forwards information. In some cases a member must use its neighbors to communicate with an other member and reach the destination. In a predefined infrastructure, some nodes are designated to transmit data to the right member, named router.

In a dynamic network, nodes are mobile. A node cannot ensure a central application. If the central node leaves the network, all the data is lost and the group enters a unstable phase. In addition, if a node changes its position, it can change its status. So a node can leave the group easily without warning. The other members must update their data, and the node leaving the group must detect its new status.

Mobile devices use low broadband and low processing power, it requires a slight protocol using limited memory and minimal operations. Wireless communication links between two nodes must be symmetric. If a node sends a message to an other node, this node should reply. Finally, wireless link is vulnerable to jamming. The protocol should regulate the number of nodes which communicate.
1.2 Protocol explanation and definition

1.2.1 Dynamic group approach

The protocol organizes various mobile devices in entities named *Group*. A *Group* is a set of mobile elements which communicate to accomplish a task or a mission 1.4.

The protocol named *GRP* for *GRouP* offers a dynamic group approach. A mobile device can communicate with its neighbors and move on the field. The mobility of nodes in the network causes appearance at disappearance of links. Those topological changes represent the dynamics. The protocol should deal with this dynamics and ensure the stability and the maintenance of the groups as far as possible.

1.2.2 Presentation of the GRP problematic

Each group must obey some obligation. A node have to be apart of a group. A node can belong only to one group. Lonely node form a group of one element.

In such groups, one node have an entire view of its group. The view ensures the knowledge of the group and its members.

A group cannot have a diameter higher than a limit defined in the protocol setting. This limit is named Dmax.

Groups should merge to form new larger group if this resulting group has diameter < Dmax. This property ensure the growing of groups.

Protocol organization The protocol organizes various mobile devices in entities named *Group*. A *Group* is a set of mobile elements which communicate to accomplish a task or a mission 1.4.

The protocol named *GRP* for *GRouP* offers a dynamic group approach. A mobile device can communicate with its neighbors and move on the field. The mobility of nodes in the network causes appearance at disappearance of links. Those topological changes represent the dynamics. The protocol should deal with this dynamics and ensure the stability and the maintenance of the groups as far as possible.

Protocol constraints To avoid overloading communication links, the diameter should be adapted to the situation. It limits the number of nodes which communicate, if to much node send to much information, the broadband can be full. The algorithm must be light
to limit the processor load.
The protocol ensures robustness against:

- Network communication fault between two devices,
- Change of network topology.

1.2.3 Property definition

1.2.3.1 Self-stabilization

We consider a stable state the state where the elements of the network reach the expected result and this result remains in successive configurations. An unstable state is a that where nodes have an incomplete and incoherent information. Then the algorithm should ensure the convergence to a stable state from an unstable state.

We call a self-stabilizing algorithm, the algorithm that, from any state, reaches the expected result and a stable state in a finite time. If network fall in an unstable phase, the algorithm enter in a convergence phase to recover a stable status. If an event occur on a convergence phase, the convergence phase is extended.

In this paper 3 property ensure the stability recovery:

1. Agreement property: Each node must agree to add new node in group
2. Safety property: Resulting group diameter should not exceed Dmax
3. Maximality property: Each group must merge with its neighbor groups to reach the maximum diameter

Additional information Super-stabilizing protocols for dynamic distributed systems

This paper [DH95] deals with self-stabilization. It describes the self-stabilizing property and why it is needed to ensure a reliable ad-hoc communication protocol. It also present research on self-stabilization and the problematic related. The article provides a formal description of the context of Ad-hoc networks.

We also see some information to classify self-stabilizing protocols. So for this article three standards are used to evaluate a self-stabilizing protocol: Citation [DH95]
• stabilization time: the maximum amount of time it takes for the protocol to reach a legitimate state

• super-stabilization time: the maximum amount of time it takes for a protocol starting from a legitimate state, followed by a single topology change, to reach a legitimate state

• adjustment measure: the maximum number of processors that must change their local states, upon a topology change from a legitimate state, so that the protocol is in a legitimate state.

1.2.3.2 Best-effort

Definition  Way to ensure « the best we can » a property.

Best-effort on GRP  On an important topology change, GRP use a best effort solution to preserve the continuity property which guaranty, each node cannot be force to leave his group.

1.2.4 Existing solution

The ad-hoc network is a new subject in research. Some articles describe the different problems and offer information to suggest a solution. Some articles propose some models to try to solve those problems.

1.2.4.1 Max-Min D-Cluster Formation in Wireless Ad-Hoc Networks

This paper [APVH00] offers a similar model. We think it was a base to best-effort group service in dynamic networks paper. This paper tries to create a model limiting the diameter of ad-hoc networks which communicate. The diameter of the network is specified on input ([APVH00]d-hops parameter) and all clusters try to fuse into a network with a [APVH00]d hops max.

They represent an ad-hoc network as [APVH00]a set of clusters. A cluster-head is elected by all clusters. This cluster-head forms [APVH00]a virtual backbone. In this case the cluster is used to control the distance between all nodes. So, if a node needs too many hops between it and the cluster-head it is ejected from the network and it forms its own network. If a topology change occurs, the cluster-head is re-elected by all clusters.

This model offers a balanced and stable algorithm.

1.2.4.2 Random Walk for Self-Stabilizing Group Communication in Ad-Hoc Networks

The article [DSW02] presents an other approach also using dynamic groups, and mobile agents. This technique [DSW02]supports group membership and multicast, and also supports resource allocation. The approach sets nodes as [DSW02]processors which can process mobile code. Each processor executes a program that is a sequence of step. A step is a period of time before an execution of code. In each step, the processor [DSW02]may receive and send a special message called an agent. An agent can transport code to be
executed by the processor. This agent \[DSW02\] changes the algorithm on fly without re-programming and update source.
So each node gives instructions to its neighbours with the mobile agent. Each node which receives the agent retrieves the information and re-sends randomly the agent to the next agent with its data at each step. After some time agents inform all nodes of the topology of the network.
Chapter 2

Study of the algorithm

Contents

2.1 The algorithm GRP ................................................. 10
  2.1.1 Progress of the algorithm ............................... 10
  2.1.2 Self-stabilisation ........................................... 11
  2.1.3 Safety in GRP ................................................ 13
  2.1.4 Agreement in GRP .......................................... 14
  2.1.5 Maximality in GRP .......................................... 14
  2.1.6 Continuity .................................................... 15
  2.2 Doubts ............................................................ 15
    2.2.1 Maximality .................................................. 15
    2.2.2 Priorities ................................................... 17
    2.2.3 Propagation (separation problem) ...................... 17
  2.3 Problem determination ........................................ 19
    2.3.1 Continuity .................................................. 19
    2.3.2 Deduction ................................................... 19

2.1 The algorithm GRP

The system uses the distributed algorithm GRP to achieve the expectations discussed above. Each node has a list and a view. The list of a node contains its ancestors in the group ordered by distance. The view is the set containing the nodes belonging to its group. Neighboring nodes exchange their lists so that the algorithm can progress towards the expected result.

2.1.1 Progress of the algorithm

Figure 2.1 shows the progress of the algorithm. A node has two timers, timer Ts and timer Tc. The timer Tc is renewed with duration t1 and Ts with t2, such as t1>t2.
At the expiration of $T_s$, the node broadcasts its list computed previously (initial list = (current node)), upon the reception of a message (neighbor list) it stores the received lists in a message Set $\text{msgSet}$, the $\text{msgSet}$ is updated at each reception, new sender list is added, the lists of old senders replace this sender’s existing list. At the expiration of $T_c$ the node computes its list and its view (output) from the received lists ($\text{msgSet}$) and resets its $\text{msgSet}$.

In wireless networks, links may not be symmetric. To deal with this problem, the algorithm uses marked nodes to check the symmetry of the link. The figure 2.2 show how grp resolve this problem. For asymmetric links the process does not continue. At every reception of a new node list, the receiving node replaces the received list by its sender marked. A node which receives a list in which it is in the second set (at distance 1 from the sender), whether marked or not, it accepts the list.

### 2.1.2 Self-stabilisation

To ensure the self-stabilization of the system, this uses a parametric message passing algorithm. This kind of algorithm is self-stabilizing when the operator used is idempotent $r$-operator. Self-stabilization with $r$-operators is detailed in [DDT06]. GRP algorithm uses the operator $\text{ant}(\text{list1}, \text{list2})$ which is based on the s-operator $\text{list1} + \text{list2}$ and
the surjective mapping (r-mapping) $r(list)$.

The operation $+$ gives as result a list containing sets resulting from the union of each two sets of the same position in the two input lists. Than redundancy of the nodes in the different sets is removed by keeping the node appearing earliest in the list. $r(list)$ puts an empty set at the beginning of the list. $ant(list1, list2) = list1 + r(list2)$.

Example:

\[ ant((\{1\}, \{2\}, \{3,4\}), (\{5\}, \{6,7\}, \{8\})) = (\{1\}, \{2\}, \{3,4\}) + (\emptyset, \{5\}, \{6,7\}, \{8\}) = (\{1\}, \{2,5\}, \{3,4,6,7\}, \{8\}) \]

\[ ant((\{1\}, \{2,4\}), (\{2\}, \{1,3\}, \{5,6,4\})) = (\{1\}, \{2,4\}) + (\emptyset, \{2\}, \{1,3,4\}, \{5,6\}) = (\{1\}, \{2,4\}, \{1,3,4\}, \{5,6\}) = (\{1\}, \{2,4\}, \{3\}, \{5,6\}) \]

At $T_c$ expiration, after all received lists in $msgSet$ have been checked, the node computes its list using the ant operator described previously and takes $msgSet$ as input. The current node list is cleared and reconstructed at every computation. The list of the current node is obtained by applying, ant on the received lists in the $msgSet$, beginning by the current Node’s list. For example if the current node is "1" and its mgset contains the lists $(\{2\}, \{1,4\}, \{3\})$ the list received from "2" and $(\{3\}, \{1,4\}, \{2\})$ the list received from "3" then:

\[
list_1 = ((list_1 ant list_2) ant list_3) \\
= (((\{1\} ant (\{2\}, \{1,4\}, \{3\})) ant (\{3\}, \{1,4\}, \{2\})) \\
= (((\{1\}, \{2\}, \{4\}, \{3\}) ant (\{3\}, \{1,4\}, \{2\})) \\
= (\{1\}, \{2,3\}, \{4\})
\]

The ant operator allows each node to compute its list of ancestors sets ordered by distance, which gives the node enough information about its group to make decisions.

\[ ant((\{1\}, \{2\}, \{3,4\}), (\{5\}, \{6,7\}, \{8\})) = (\{1\}, \{2\}, \{3,4\}) + (\emptyset, \{5\}, \{6,7\}, \{8\}) = (\{1\}, \{2,5\}, \{3,4,6,7\}, \{8\}) \]

\[ ant((\{1\}, \{2,4\}), (\{2\}, \{1,3\}, \{5,6,4\})) = (\{1\}, \{2,4\}) + (\emptyset, \{2\}, \{1,3,4\}, \{5,6\}) = (\{1\}, \{2,4\}, \{1,3,4\}, \{5,6\}) = (\{1\}, \{2,4\}, \{3\}, \{5,6\}) \]

After the convergence , as shown in Figure 2.3 , each node has all the nodes of its
group ordered by distance in its list and a view containing the nodes of its group (calculating view is further detailed in section).

The self-stabilisation is guaranteed for static tasks, to deal with the problem of dynamic and bounded group diameter ($D_{\text{max}}$), the algorithm uses tests and conditions to behave well in the case of topology changes and to avoid the merging of groups large enough to form a group whose diameter $> D_{\text{max}}$ (diameter constraint).

### 2.1.3 Safety in GRP

To ensure the diameter constraint, the algorithm uses conditions to check if the candidate node can be added to the group or not. Accepting a node means accepting its group. If a group is accepted, the current node’s list is merged with the candidate node’s list. Such a verification is symmetric, in other words, if a node "1" rejects a node "2" because its group is too large to be accepted, then the group of "1" is too large to be merged with that of "2".

For this verification, the algorithm uses the function `CompatibleList` which uses as a test the comparison of the sum of the current node list size and the received list size with $D_{\text{max}}+1$. The result of this sum should not exceed $D_{\text{max}}+1$, otherwise, the request is rejected by ignoring the candidate node. This verification is done only for new accepted nodes. Lists of nodes which are in the same group are not checked. Figure 2.4 represents an example of this operation.

When a node "1" accepts a node "2", "2" is added to the list of "1" at distance 1 (second set in list$_1$). "2" then accepts "1" at the next reception of list$_1$. For "2", in this case, the diameter constraint should be fulfilled for list of "1" (symmetric verification).

As noted, each node checks the diameter constraint independently and checks the received lists of candidates separately. In other words, when a node receives requests from several neighbor nodes, it checks if the list of each node is compatible apart. The result of the verification can be positive for multiple nodes. In addition, a node does not know, what other nodes in its group are accepting or rejecting at this moment. So, several groups can be accepted concurrently by distant nodes of a group but even by the same node. This leads to the creation of a group whose diameter $> D_{\text{max}}$. The problem can be resolved for the same node but distant nodes cannot be agreed to one decision at the same time. Such a distributed algorithm cannot prevent this kind of conflict.

To resolve this problem, the algorithm corrects after acceptance (Figure 2.5). After the acceptance of many groups, which leads to having a group whose diameter $> D_{\text{max}}$, at least two nodes (as many nodes as groups accepted) will have lists with size $D_{\text{max}}+2$ (group diameter $= D_{\text{max}}+1$). The question is, why do these nodes accept merging that gives a group with diameter=$D_{\text{max}}+1$ if it checks at every reception with `CompatibleList`? The answer is, at the moment when such a node receives a list with size $D_{\text{max}}+1$, the sender of this list belongs to its group (view), then the `CompatibleList` test is not checked. In this case, only one of the accepted groups can stay. The other
groups should leave. The algorithm uses the concept of priority to deal with such a choice. Nodes of a group have priorities of all the nodes belonging to the same group. Each node having list with size $= D_{\text{max}} + 2$ compares its priority with that of other too far nodes. If it has not the priority, it leaves the group by ignoring the senders of the received lists with size $D_{\text{max}} + 1$. Those lists contain the too far nodes. Eventually, only one node (one accepted group) stays. The priority is calculated according to the oldness in the group. If a node does not belong to a group, its priority counter $P$ increases (priority decreases) every $T_c$ expiration. By comparing $P_u$ and $P_v$, if $P_v > P_u$, then $P_u$ has priority.

An example is provided on Figure 2.5

### 2.1.4 Agreement in GRP

To be sure that the accepted nodes are approved by all the nodes of the group, each node waits for a moment to add accepted nodes to its view. This time corresponds to $D_{\text{max}} * t_1$. At $T_c$ expiration after computing the list, the node affects a quarantine period $= D_{\text{max}}$ to each new added nodes. Quarantine period of nodes already accepted decreases by 1. Accepted node is added to view when its quarantine period is null. This gives a certain stability to the view and allows the node to be sure that all nodes of the group agreed and that their views converge to the same result.

### 2.1.5 Maximality in GRP

Maximality property depends on the test $\text{CompatibleList}$. While the estimated diameter of the group resulting from the merger future is correct, the maximality is guaranteed for the decision of the node doing the test.
2.1.6 Continuity
Mechanisms discussed previously and mainly those used to ensure the safety property, contribute to ensure the continuity property. Thanks to CompatibleList, nodes cannot leave their group when CompatibleList test result is false. The only case when a node can leave its group is when several groups are accepted at the same time, as shown in Figure 2.5, when the safety property is violated. The only way to correct is that some nodes must leave the group. In this case the algorithm cannot ensure both safety and continuity (more details in the next section).

2.2 Doubts
After studying the algorithm, it seems that the algorithm does not provide all the desired properties for various reasons. Some tests and conditions do not satisfy what we really expect and do not correspond to actual conditions. As well, lack of received information and treatment of these makes impossible to totally verify some properties.

2.2.1 Maximality
2.2.1.1 CompatibleList
Problem The algorithm mainly uses the function CompatibleList to prevent the formation of new groups with a diameter higher than Dmax. The test used to check is to compare the sum of the size of the list received and the size of the list of the current node with Dmax. Knowing that at the moment the test is done the two lists can contain marked or shared nodes, the sum of the two lists in the test may, in some cases, give a superior size to the size of the group resulting from the future merging. This verification exceeds the field of safety to "overflow" on maximality, which prevents the algorithm to
ensure the property of maximality. Figure 2.6 shows an example of this case.

\[ \text{Dmax} = 1 \]

Figure 2.6: The effect of CompatibleList on the maximality

**Proposition** For this problem, instead of comparing the sum of the current list and the received list with \( \text{Dmax} + 1 \), we propose to use a union function that takes two lists as parameters, and returns the union of the two lists without marked or redundant nodes, and guards the right distances of nodes. This allows to preview the diameter of the group resulting from the future merging. Figure 2.7 shows the behavior of this test in the latter case.

\[ \text{Dmax} = 1 \]

Figure 2.7: The new CompatibleList test

### 2.2.1.2 Distributed request

Each node receives information from its neighbours, each node makes decisions based on what it has received, regardless of the decisions of the other nodes. Two nodes belonging to the same group cannot make a decision for collaborative acceptance of a group whose two nodes are candidates (pair: candidate/node). This will limit the extension of groups. In some cases, for example the case shown in Figure 2.8, groups refuse the request of other groups whose merging gives a group with diameter \( \leq \text{Dmax} \).
CHAPTER 2. STUDY OF THE ALGORITHM

2.2.2 Priorities

The computation of the priorities is based on the incrementation of their counter. Isolated nodes continue incrementing their counter where nodes belonging to a group have a constant priority. This can give to different nodes the same priority, which leads to the creation of a too large group (diameter $> \text{Dmax}$). In other words, if two different nodes have the same priority and if those nodes are involved in the correction (have lists with size=$\text{Dmax}+2$) after getting a too large group, the conflict cannot be resolved.

2.2.3 Propagation (separation problem)

**Problem** The algorithm uses as structure (input / output) lists ordered by the distance of the nodes relative to the current node. Without additional information a node cannot know to which nodes, a node is directly linked. For example, if "1" has neighbours "2" and "3", "3" and "4" are related to "2", "1" does not know if "4" is linked to "3" or "2". Even if "4" is directly related to "3", "3" receives lists telling that "4" is at distance 2 from "1", "3" cannot know if "4" is linked to "2" or not. In the case of disconnection of "4", "3" stops receiving lists from "4" but continues to receive lists containing "4" from its neighbours. "3" believes that "4" is connected to "2" or "1" because it receives a list from "1" telling that "4" is at distance 2, "2" does the same, so it accepts. The same thing happens with "1" and "2".

The disconnected node is no longer in the group but the group is maintained with a false view and bad lists are circulating. All nodes in the group ("1", "2", "3") believe that 4 is in the group. In the case of a separation of two parts of the group whose diameters $> 0$, the same thing happens. The views remain unchanged. For the nodes of the two separated groups, the same group is maintained. Figure 2.9 shows an example of this case.

This can happen in the case of a physical separation or even in the case of a rejection after merging, when several nodes in the same group accept groups simultaneously.
A simple detection of disconnection by the nodes the neighboring disconnected node cannot deal with this problem of information propagation.

**We propose this solution** Each node must have a counter `existenceCounter`, initialized to 0. A node sends its list with its identity and `existenceCounter = 0`. A node receives several lists containing, possibly, some identical nodes with different `existenceCounter` (it depends on the length of the path taken). For all nodes in received lists, it saves the minimum `existenceCounter` for nodes having the same identity (name). The node calculates its after processing all the lists received, at that time the node has minimum values of all the received nodes. At the building of his list, the node assigns to each node added an `existenceCounter` whose value corresponds to the lowest value received for this node incremented by 1. If the `existenceCounter` value > `(Dmax * 2) - 1` the node is not added to the list.

Taking the minimum value of the `existenceCounter` guarantees that at every reception, counters have taken the shortest path from the received node to the current node. This value, therefore, must not exceed `Dmax - 1` if the node, whose identity is received, is really in the group. But in the case of simultaneous acceptance by distant nodes of the same group, we have to accept the constitution of a group of up to a diameter `Dmax * 2`, in order to correct this simultaneous acceptance. The groups are then fused to make one group, then, one of the added groups leave. If we consider that `Dmax - 1` is the maximum value of `existenceCounter` for which a node can be added to the list, it prevents the formation of the group after the simultaneous acceptance which does not allow the resolution of that problem. This generates incorrect information in the views and lists.
Putting \((D_{\text{max}} \times 2) - 1\) as a maximum value for the counter of `existenceCounter`, this solution will delay the elimination of nonexistent nodes in the case of a normal groups construction, but ensures the resolution of the problem of the simultaneous acceptance.

### 2.3 Problem determination

#### 2.3.1 Continuity

After different groups have been accepted concurrently by distant nodes, if the merging of one or many of those groups gives a group with diameter \(< D_{\text{max}}\), the nodes of other accepted groups, which have lists with size \(D_{\text{max}} + 2\) are not the candidates for which their groups are accepted. In other words, those nodes are not directly attached to the main group (group that has accepted concurrently). If one of the groups whose merging with the main group gives a group with diameter \(< D_{\text{max}}\) has the priority, the resulting group is extended and, other groups are divided when quitting, because the nodes that do not have priority (too far node which have lists with size \(= D_{\text{max}} + 2\)) in the other groups will ignore nodes that belong to their groups. The ignoring node rejects a part of their groups. This part will be merged with the main group to form a group with diameter \(= D_{\text{max}}\.

Figure 2.10 represents an example of this problem.

![Figure 2.10: Separation of groups and violation of continuity](image)

#### 2.3.2 Deduction

High dynamic in the network leads to produce this latter event with high frequency. Moreover, the connection and disconnection of nodes, when the duration of convergence is elevated, prevents the nodes to have a good view of their groups. The period of
convergence depends on $D_{\text{max}}$ and $t_1$ (compute frequency). In these conditions, the algorithm will not be able to provide the expected results.

The best-effort requirement says that outside of these conditions the algorithm should ensure the continuity and self-stabilizing properties.
Chapter 3

Implementation and evaluation

Contents

3.1 Implementation ................................................................. 21
  3.1.1 Development environment ........................................... 21
  3.1.2 Experimental context .................................................. 21
3.2 Scenarios of tests ............................................................... 22
  3.2.1 Random Waypoint ....................................................... 22
  3.2.2 Group model ............................................................... 24
  3.2.3 VANET model ............................................................. 25
  3.2.4 Student model ............................................................ 26

3.1 Implementation

3.1.1 Development environment

The project was developed on GNU/Linux. We used the IDE Eclipse and Simulation library *JBotSim* [4] to simulate the interaction between mobile devices.

*JBotSim* is a simulator based on Java technology. It offers the possibility to simulate distributed algorithm in mobile ad-hoc network. This simulator uses an event-driven paradigm, where every nodes execute some specified action when particular events occur. *JBotSim* is published under the LGPL licence.

3.1.2 Experimental context

Simulation parameters:

- Experimental field: the area in which the nodes can move
- Number of nodes: number of node deployed on the experimental field
- Dmax: Max diameter of a group
• Node speed: Distance travelled on time period

Metric of interest:

• Stability: it is the percentage of time when the views are stable. Consider the evolution of the network as a sequence of topological configuration. Given an execution \( e = c_1, c_2, c_3, \ldots \) a stablePeriod is a time period between two unstable phase. We consider the system as unstable, if we detect a difference between the view of two successive configuration \( c_i.views \neq c_{i+1}.views \). Then the stability is defined as 

\[
\text{Stability} = \left( \frac{\text{stableTime}}{\text{totalTime}} \right) \times 100, \text{ where } \text{stableTime} = \sum \text{stablePeriod}.
\]

A model of mobility is a mathematical model that can be used to simulate one or more aspects of the movement of nodes on a surface. Several studies show that the mobility model affects the performance of wireless networks (see for example [BSH03] [CBD02] in the case of routing protocols).

There are several types of mobility models, we can quote:

**The individual models.** For the individual models, nodes movement is completely independent from the other nodes movement. The individual models which exist are the following ones:

• Random Walk
• Random Waypoint
• Random Direction

**The group models.** In a model of group, nodes move together as a group to achieve together a mission. The models of group which exist are the following ones:

• Exponential Correlated Random
• Column
• Nomadic Community.

**The Realistic models.**

• Gauss-Markov: Model base on a probability matrix. Realistic, if right parameters are selected.
• VANET model: Model base vehicular move.

### 3.2 Scenarios of tests

#### 3.2.1 Random Waypoint

**Context** In our test we use Random Waypoint because this model makes no hypothesis on the plan of mobility (Random). This model offer a huge number of event. It helps to evaluate the robustness against topology changes depending to variable number and variable speed and compare with other models.
CHAPTER 3. IMPLEMENTATION AND EVALUATION

Algorithm  This model is described by the following algorithm:

Parameters:
F - Field = 800*800 distance unit
V - Nodes speed = 5 distance unit per x units of time
T - Time = 3000 units of time
X - Speed factor -> variable
N - Number of node -> variable

Algorithm:
1. Select the destination uniformly in the space of simulation
2. Move in a straight line towards the destination with a v speed
3. Back to 1

A test runs for a time T. In each test we vary X and N, we measure the stability of the system.

The evolution of the stability according to the speed and the number of nodes allows us to define the limits of the algorithm in such model of mobility.

![Figure 3.1: Experimental result: Random waypoint model](image)

**Experimental result**  The figure 3.1 represents the test results in the form of a 3D curve. The x axis represents the number of nodes N, y axis represents stability and axis z is the speed factor X of nodes. We note that the stability collapses when moving speeds exceed 35 and the number of nodes exceeds 13.

**Conclusion**  The collapse of the stability is due to the high mobility of nodes which prevents the creation of the groups. Indeed nodes appear and disappear quickly, even before convergence is reached. This prevents the algorithm to reach the stable state.
3.2.2 Group model

Context  We use a group model to evaluate the algorithm stability in a drones network. This model simulates a movement in network of autonomous elements and tests the fusion capacity of groups. Groups are physically maintained (nodes of a group have the same movement) so nodes cannot move out of their groups if they are not rejected. We use Dmax as a parameter because the convergence period, the acceptance and rejections are related to Dmax.

Algorithm  This model is described by the following algorithm:

Parameters:
F - Field = 800*800 distance unit
V - Nodes speed = 5 distance unit per 5 units of time
T - Time = 3000 units of time
D - Dmax -> variable
N - Number of node -> variable

Algorithm:
1. Each node inform a controller class of its view
2. The controller determines for each group (view) a destination point
3. Each node requests a destination point
4. The controller gives to the requesting node the destination (target) that correspond to its view (group)
5. Node moves in a straight line towards the determined destination with a speed V
6. Node can leave its group with a probability of 1/40

A test runs for a time T. In each test we vary the Dmax and N of all node and we vary the number of nodes deploy on the experimental field. For each measurement, we determine the stability.

Experimental result  The figure 3.2 represents test result shown as a curve 3D. x axis represents the number of nodes, the z axis represents Dmax and the y axis represents the stability.

We note that the surface of the curve is composed of two parts. Stable part (dmax in [2,6]) in this part of the curve the percentage of stability close to 100% and that even for a large number of nodes. In the second part (Dmax = 2) we note that the percentage of stability decreases when the number of nodes exceeds the value 16.

Conclusion  This is due to the high density of the network which offers the possibility to create large group and the low value of Dmax (2) which limits the size of group training, this causes attempt to mergers.
CHAPTER 3. IMPLEMENTATION AND EVALUATION

3.2.3 VANET model

Context  We use a VANET model to evaluate the stability of the algorithm in a VANET application. This model simulate a vehicle application and analyse the robustness for applications.

Algorithm  This model is describe by the following algorithm:

Parameters:
F - Field = 800*800 distance unit
V - Nodes speed = 5 per distance unit x units of time
I - Intersections of roads -> 9 crossroads
D - Distance between intersections 400
T - Time = 3000 units of time
X - Speed factor -> variable
N - Number of node -> variable

Algorithm:
1. Each node choose a next reachable crossroads
2. Move in a straight line towards the determined destination with a v speed
3. Back to 1

Experimental result  The figure 3.3 represents the changes of stability depending on the speed and the number of nodes. Overall, the stability decreases almost constantly.

Conclusion  This is due to the fact of the formation and physical separation of the groups in the roads. This behavior is close to that of random movement test
but the limitation of movement of nodes on the roads limits the frequency of events (candidature and separations). In addition, this model of mobility forces physically the maintenance of groups because of the defined directions that nodes following (roads), which explains the higher stability for a large number of nodes compared to random wayPoint test. Instability is explained by the crossing of nodes in the road causing a convergence time during the connection and disconnection, this disturbs the stability without interest, because the crossing nodes in the road never leads to an useful formation of a group, we are sure that the group splits just after its formation in this case. GRP protocol remains adapted to this model of mobility because stability is higher in the case closer to reality. In extreme cases the stability is not null either.

3.2.4 Student model

**Context**  We use a special realistic model named Student model to evaluate the stability of the algorithm in a school application. This model simulate a scholar application and analyse the efficiency of the algorithm for a school application as sharing document or video during a teaching.

**Algorithm**  This model is describe by the following algorithm:

**Parameters:**
F - Field = 800*800 distance unit
V - Nodes speed = 5 distance unit per 5 units of time
T - Time = 3000 units of time
D - Dmax -> variable
N - Number of node -> variable
CHAPTER 3. IMPLEMENTATION AND EVALUATION

Algorithm:
1. Each node choose a next teaching
2. Each node join its teaching room
3. Each node wait the end of its teaching
4. Each make a break time: they choose a destination randomly
5. Each node wait the end of its break time
6. Back to 1

A node spends 90% of class time and 10% pause

Figure 3.4: Experimental result: Student model

**Experimental result**  On this graph 3.4 we can see the stability in terms of nodes number and Dmax value. We can see that more we have nodes and more Dmax is high, it becomes slightly less stable. But the stability does not fall. It remains between 100% and 80%. The sudden falls can be explained by the fact that nodes when joining classes form groups with concurrent acceptance which extends the convergence period and generally causes others concurrent acceptances.

**Conclusion**  With this result we can note that GRP is adapted to the Student model. With a great number of students (nodes) we have a good stability, and more Dmax is important more the system is unstable. The algorithm is overall stable because node move in a little period of time, so the algorithm have a great period to recover it stable status.
Conclusion

GRP provide a self-stabilizing system to ensure communication in an Ad-hoc network. The algorithm offer a solution without a central node such as [APVH00] Max-Min D-Cluster Formation in Wireless Ad-Hoc Networks.

After having studied the paper, we found some design mistake, for which we suggested corrections and improvements. The resulting algorithm was tested under a variety of mobility models, including the student model (an original contribution of our work), the random waypoint model, and a VANET model (vehicular networks). Our simulation results showed that the algorithm GRP is not efficient in every context. In fact, it seems to be more relevant to those ad-hoc networks with a little number of events and relative stable periods, such as in the student model.

What we have learn This project was the opportunity to learn a lot about mobile ad-hoc networking and related problems. It was interesting for us to study this article because ad-hoc network is a recent topic of research. This subject was complementary to our RSM classes (in particular, it offers an other vision of the SAC lessons). It was also the opportunity to learn how research is done in general and we had the satisfaction to correct and improve the studied algorithm. On a practical side, the simulations were for us a good practice of Java development.

Future works The study of the algorithm allowed us to determinate the main problems and understand what limits the performances of the algorithm. After this limits and its causes have been well defined, some future works can contribute to improve the performance of the algorithm. In order to evaluate the GRP algorithm in an more absolute sense, it could be interesting to design and implement a centralized optimal algorithm for this problem and compare its stability to that of GRP.
Bibliography


Sitography