A TACTICAL ACTIVE INFORMATION SHARING SYSTEM FOR MILITARY MANETS

Lionel BARRÈRE
lionel.barrere@labri.fr
LaBRI, Université Bordeaux 1
351 Cours de la Libération
33405 Talence Cedex, France

and

Serge CHAUMETTE
serge.chaumette@labri.fr
LaBRI, Université Bordeaux 1
351 Cours de la Libération
33405 Talence Cedex, France

and

Jacques TURBERT
jacques.turbert@dga.defense.gouv.fr
CELAR
Route de Laille, BP7
35998 RENNES ARMEES, France

ABSTRACT

Operation fields often give rise to situations where communication has to be carried out without any infrastructure. One of the solutions that can be used to support this context is based on mobile ad hoc networks (MANets). We believe that when this kind of versatile architecture is considered it is necessary to have dedicated services that take their context into account and still work in case the network is slow or even broken, instead of relying on a routing layer. Situation management and more precisely battlefield situation management can thus still be supported in spite of poor communication conditions.

The application that we present in this paper illustrates this approach. It makes it possible to share a map of the battlefield between soldiers. Its major originalities are that it is totally decentralized and that it handles what we call active information. Active information are information that are updated by the different entities of the network. Applications that use active information have never been experimented in a military context.

Two major difficulties when designing such applications are to clearly define their context of use and the associated limitations, and to provide a significant evaluation. We address these problems in this paper, based on our prototype implementation.

The work presented in this paper is carried out at LaBRI (Laboratoire Bordelais de Recherche en Informatique) and more precisely in the SOD (Systèmes et Objets Distribués) research team. It is partly achieved within the framework of the Sarah (Services Asynchrones pour Réseaux Ad Hoc) project supported by the ANR[1]. It is also partly supported by the DGA[2] by means of a PhD grant.

1 INTRODUCTION

Military operations always reflect societies, and in the digital age, the military effectiveness depends on the information quality, availability, and on reactive information sharing. The characteristics of information are of great importance for the military. Information, produced by an imaging satellite, an airborne radar, or a soldier in operation, have to be available at the right moment to the decision maker. This decision produces new information to be delivered to highly mobile battlegroups. In a rapidly changing environment, the soldier needs constantly updated information and constantly creates new information. If we consider the warfighter as a "sensor", the group of soldiers needs to collect, to merge and to exchange a sum of information units. In the today's flattened command organization, the soldier interaction with the battlegroup provides tactical superiority based on a common understanding of the situational awareness. This new operational concept supposes a digital environment: digitization will provide the warfighter a constantly updated vision of the battlespace environment.

These information flows suppose an effective and seamless network, able to support the information demanding military operations. Networks offer new capabilities to support information exchange between soldiers working in small groups in a low intensity peace keeping operation or high intensity warfare. Ad Hoc networking, an active field of research, is one of the solutions envisioned to provide flexibility to the military tactical deployment. The context, constraints and expectations in the considered fields of operation are as follows. The network is organized as a multilevel communication architecture. A few nodes in a small geographical area compose a MANet with high speed interconnexion (100k to 10Mbps). At a higher level, some of these nodes can be linked with a global infrastructure such as a telecommunication network which provides wide area communication, the counterpart being a low transmission rate (1k to 100kbps). This is the basic architecture that can be assumed to be available in the near future for military on field equipments and thus applications. These technologies authorize new ways to produce, and to share information. The concept of GroupWare, familiar to the engineer, will have its counterpart in the soldier's environment. The distributed tactical map applicative software presented in this paper is an example of the benefits of the digitization to the warfighter.

This work is a contribution to the design of new applications for mobile ad hoc networks, or MANets. MANets are versatile networks where nodes can move, appear, or disappear at anytime. Two approaches make it possible to implement applications for MANets. The first is to use routing

[1] Agence Nationale de la Recherche
[2] Délégation Générale pour l'Armement

protocols (such as OLSR [1]). These protocols hide mobility and make it almost possible to run classical applications on a MANet. Nevertheless, these routing layers are often inoperative when a node becomes unreachable. The second approach consists in designing new applications that are dedicated to MANets. These applications can, by design, ensure a certain level of reliability by taking into account the versatility of this kind of network. The system described in this paper is representative of this second approach. It makes it possible to generate and modify a global document shared by the nodes of a mobile ad hoc network, in a totally distributed manner. The shared document that we are considering in this paper is a military tactical information map. At the beginning, the map is owned by one single officer. He can split it in different pieces that can be distributed to different groups or individual troopers. Each group or individual is then in charge of updating the map with information that it can collect on the battlefield. The chief of a group can decide on further splitting the map between some of his soldiers. Soldiers or groups meeting on the battlefield can then communicate in a peer to peer manner to rebuilt partly updated versions of the whole map. We thus combine partial maps in order to understand an overall situation, what contributes to the battlefield situation management (and can also be used, for instance, in disaster management).

The rest of this paper is organized as follows. In section 3 we describe the internal structure of the system and the lock management and synchronization mechanisms that we have setup. In section 4 we then present our implementation and its evaluation. We describe the context of use and the associated limitations of our approach in section 5 and we sketch out possible improvements. We eventually conclude in section 6 and give additional directions for future work.

2 RELATED WORK

The domain of MANets is very active. A lot of projects are being developed, ranging from new routing protocols to high level applications. In this section we focus on applications that deal with sharing/disseminating information.

One of the major categories of applications on MANets are file systems. Several approaches can be found in the literature. AdHocFS [2] is such a system developed at INRIA[3]; this system is based on the use of groups of mobile devices. All groups can synchronize their versions of the files on a central server. Within each group only one node can modify a particular block of a file at a time, the exclusion mechanism being based on a token management system. Another example of a distributed file system is MFS [3]. It seems that file systems are hard to deal with for MANets because they need stability to ensure data consistency. For example AdHocFS uses a central server to ensure consistency between distinct ad hoc user groups. Another important class of MANet applications that address information sharing are those dealing with the dissemination of information (see for instance [4–7]). The major difficulty of these systems is to ensure that the maximum number of mobile terminals will eventually get the files (or any kind of data) they want. There is no issue of consistency here because the files to disseminate cannot usually be modified.

MANets can also be a way to improve human safety, and this has been widely studied for road traffic. In [8] A. Bejan and R. Lawrence present an application that uses GPS and communicating devices. Each vehicle continuously informs its neighborhood about its position, and the vehicles can then have a global vision of the traffic. Convoy driving is another domain that can benefit from MANets; M.A. Khan and L. Blni present a system in [9] that can be used by truck drivers on highways to form and stay in convoy. This system makes it possible to keep security distances and helps the driver in giving acceleration or deceleration instructions.

Another important security domain targeted by MANets is that of support tools for rescue teams. Cenwits [10] is such a tool. This application makes it possible to disseminate location of hikers to evaluate their position when they need rescue. This system uses a totally decentralized view and opportunistic connectivity between hikers or between a hicker and an access point that transmits information to a central server. Workpad [11] is another system that helps to organize collaborative work of rescue teams. Its is supported by a two levels architecture, a back-end peer-to-peer network and a front-end composed of groups of mobile devices. Each group contains a coordinator that is the relay between the front-end and the back-end. At the front-end level, the system does not really propose any connectivity management solution within groups: nodes are asked to move to maintain connectivity. When a node is going to be disconnected another one is elected to follow it, becoming a bridge and the system then uses a multi-hop approach. There is nothing to ensure data consistency in case of connectivity loss, the assumption being that this does not happen in the target context.

The domain of rescue systems is the topic of a large number of research projects [12–14]. Issues that are present in a rescue context are really close to the issues that can be found in a military context. In fact warfare architecture management is very similar to collaboration management in rescue teams where severals groups have to achieve a different, still collaborative mission. We believe that the system that we have designed and that we describe in this paper contributes to solve some of these issues.

3 DESCRIPTION OF THE SYSTEM

Our system makes it possible to modify a battlefield map, and more generally speaking a document, in a collaborative manner between the *mobile units* (MUs for short) of a MANet. This application is totally distributed and decentralized and all the operations in the network are pairwise. The map to share is cut into pieces, and some or all of the pieces

[3] Institut National de Recherche en Informatique et en Automatique

are distributed over the MUs, so that they can modify it, so as to reflect the significant information that the soldiers can collect on the battlefield. Since multiple copies of some or all of the parts of the map can be duplicated among the MUs of the network, we must ensure that at any time only one MU can modify a given area of the map.

Figure 1 shows a sample hypothetical battlefield map and the way it has been splitted into pieces, or *battlefield areas*. Splitting a map makes it possible to distribute some areas to different groups on the battlefield, allowing each group to work on its own copies or *instances* of the areas it has been given. The overall map will be referred to as a *shared map*.

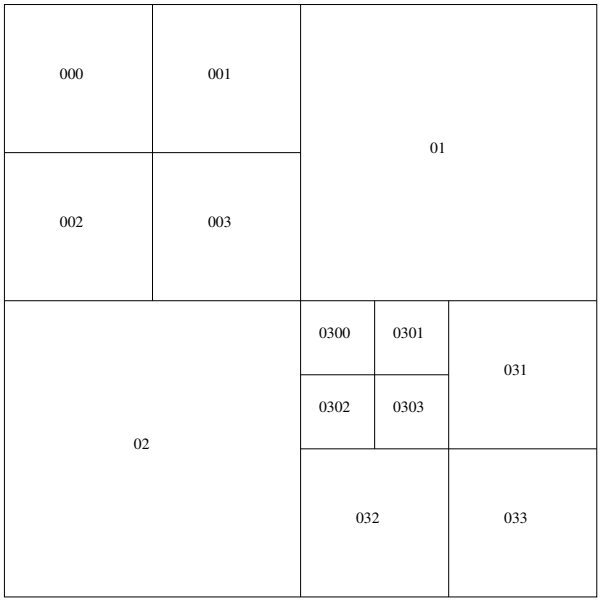


Figure 1: A sample shared battlefield map

3.1 INTERNAL DATA STRUCTURE

The map shown figure 1 is internally described by the tree of figure 2. The leaves store the effective information (the basic areas of the map) and the internal nodes are used to maintain the structure of the map (the organization of the areas). We have chosen to use a quaternary tree because it perfectly corresponds to the way the map is organized and can be splitted (each area can be splitted easily into four sub-areas). This structure makes it possible to identify each block according to its geographic location: its name is composed of the name of its father and its position given as matrix like indices.

Two operations are used to manage the way the map is splitted into pieces and have an influence over the structure of the tree. We refer to these operations as *splitting* and *merging*. They are described bellow.

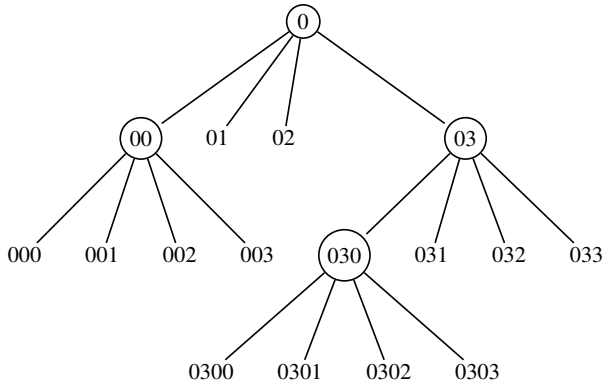


Figure 2: The tree representation of the map shown figure 1

Splitting operation

Splitting makes it possible to divide a region of the map into four sub-areas. By doing so, it will be possible to distribute different sub-areas to different MUs so that they can work in an independent manner on the shared map.

The effect on the tree is to modify the target leaf into a node having four children leaves, one per newly created sub-area of the map.

Merging operation

Merging has the exact opposite effect to splitting. It consists in combining four neighbor areas of the map into one single area. The effect on the tree is to compact the corresponding node and its four children into one single leaf.

3.2 STATES MANAGEMENT

Multiple instances and possibly multiple versions of a block can be present on the battlefield. We must ensure that only one MU at a time can modify a block: we use locks to ensure this form of consistency. Each block is provided with a lock that is shared between all its instances. A mobile node that wants to modify a block must acquire the associated lock.

STATES

The state of each internal node and leaf is defined in this section. This will be used to work out where the lock is, and how it can be exchanged between mobiles units. Here are the different possible states of the nodes of the tree of a given MU:

- for internal nodes:
  - NODE: An internal node does not contain any information to modify. It is used to maintain the structure of the map.

- for external nodes (leaves):
    1. LOCKED: this MU is currently locking the block and can thus modify it.
    2. UNLOCKED: this MU has the lock, but it is not locking the block. The lock is thus available for an other MU.
    3. UNAVAILABLE: the MU does not have the lock on this block.
    4. DESIRED: the MU does not have the lock on this block, but it is willing to acquire it.

TRANSITIONS

We have to consider two different kinds of transitions (see table 1), i.e. internal transitions and synchronization transitions. Internal transitions are those that happen locally at a single MU, based on a user action. In table 1, T_1, T_2, T_4 and T_6 are internal transitions. Synchronization transitions are transitions that involve two MUs. They take place when a node synchronizes its version of the map with a neighbor, so as to update its own version, and/or to get or give one or several locks. This process will be described in details in section 3.3. T_3 and T_5 are synchronization transitions.

T_1:	LOCK.	→	UNLO.	Releasing the lock
	LOCK.	→	UNAV.	**Impossible**
	LOCK.	→	DESI.	**Impossible**
T_2:	UNLO.	→	LOCK.	Locking the block
T_3:	UNLO.	→	UNAV.	Another MU in the network gets the lock associated to the block
	UNLO.	→	DESI.	**Impossible**
	UNAV.	→	LOCK.	**Impossible**
	UNAV.	→	UNLO.	**Impossible**
T_4:	UNAV.	→	DESI.	The MU wants to lock the block
T_5:	DESI.	→	LOCK.	The MU gets the lock from another MU
	DESI.	→	UNLO.	**Impossible**
T_6:	DESI.	→	UNAV.	The MU does not want to lock the block anymore

Table 1: Transition table of the system

3.3 SYNCHRONIZATION BETWEEN MUS

One of our major goals is to ensure that every user has the most recent version of the battlefield map as possible. This requires synchronization between MUs when they meet on the battlefield. To achieve this goal, we must be able to compare different instances of a block or map area, and to effectively synchronize two MUs.

Version comparison

Many instances and thus versions of a block can be present in the network at the same time. To be able to work out which is the most recent, we provide each block instance with a version counter. At the beginning each (*de facto* unique instance of a) block has version number 0. This counter is then incremented as follows.

1. When the lock on a block instance is released its version counter is incremented.
2. When a merge operation is applied the version counter of the father node is incremented before it is compacted into a leaf.

When two nodes synchronize, each node needs to know which blocks it has to update. The comparison procedure is as follows:

1. Version counter comparison:
   If the version counters differ than the block with the smaller one needs to be updated.
2. States comparison:
   If the version counters are the same then the states of the block instances are compared. States can be used to order the versions as follows:
    1 NODE. The instance of a block carried by a node is necessarily more recent than an instance carried by a leaf.
    2 LOCKED or UNLOCKED. An instance that has the lock (locked or not) is necessarily more recent, since exchanging a lock happens when synchronizing two nodes.
    3 UNAVAILABLE or DESIRED

Synchronization protocol

When two nodes want to synchronize their respective versions of the map, they proceed as follows:

1. The two MUs exchange a description of their trees. This description is composed of the name, the state and the version number of each block instance. For the whole tree, this is assembled in the prefix order of a depth first search.
2. Each MU can work out which blocks it has to update and sends a request to the peer MU containing the list of the blocks to update, i.e. those it wants to receive.
3. The MUs then update their respective versions of each block.

4. Once the copies of the map have been updated, the two nodes can, if it is both possible and required, simultaneously apply transitions T_3 and T_5. By doing so, the MU that was trying to get the lock can acquire it from the MU that was not currently using it.

Scenario

Figure 3 shows a scenario where five nodes synchronize with each other. First, A synchronizes with E and B synchronizes with C (step [1] of figure 3). Of course the involved nodes do not modify their block instances during this process. At the end of the fourth step ([4], figure 3) each node has synchronized with all the other nodes of the network.

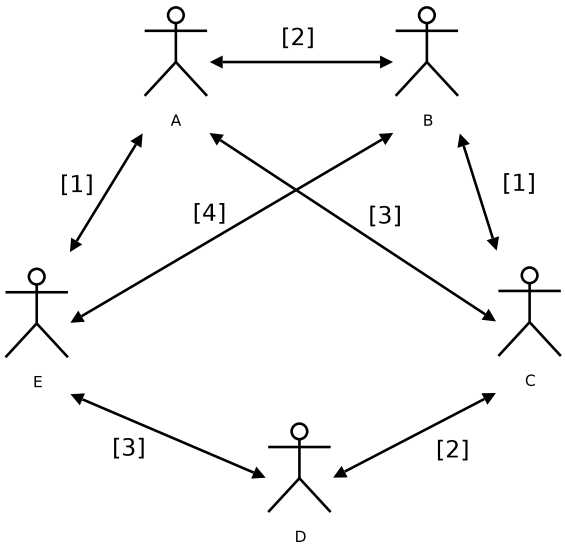


Figure 3: A synchronization scenario

In the worst case $n(n-1)/2$ steps are required to provide all the nodes with an up to date version of the map. The proof of this is straightforward: the worst case, which should rarely happen, is when it is needed to synchronize each node with all the other nodes of the network (i.e. no indirect synchronization occurs) and when only one synchronization takes place at a time.

4 IMPLEMENTATION AND EVALUATION

A prototype implementation of this strategic map sharing application has been built using the Java™ [4] technology and a WiFi TCP/IP network support. This prototype makes it possible to share strategic information such as the location of soldiers. Each MU can lock a block and add soldiers on the map using the supplied GUI. This implementation uses the OpenMap™ [15] open source library to draw maps.

[4]Java and all Java-based marks are trademarks or registered trademarks of Sun microsystems, Inc. in the United States and other countries. The authors are independent of Sun microsystems, Inc.

Implementation choices

Our implementation is a prototype therefore, as of writing, the GUI (see fig 4) is not very sophisticated. Internally, the data structure describing each block contains a vector of strategic pieces of information. Each piece of information is identified by its location on the map. Thus, when a block is splitted, its associated vector can easily be divided and its parts associated to the newly created sub-areas of the map.

When a user wants to synchronize its version with its neighborhood, a discovery process is initiated and the user can choose a neighbor to synchronize with in the resulting list.

A sample scenario

We now describe the scenario illustrated figure 4, as seen by the same mobile terminal *N*. This scenario is played by two mobile units that synchronize three times, exchanging information and locks.

Fig 4 (a): The mobile unit *N* locks the whole map.

Fig 4 (b): *N* has splitted the map and it has released the lock on the bottom right area, so as to share it with other nodes.

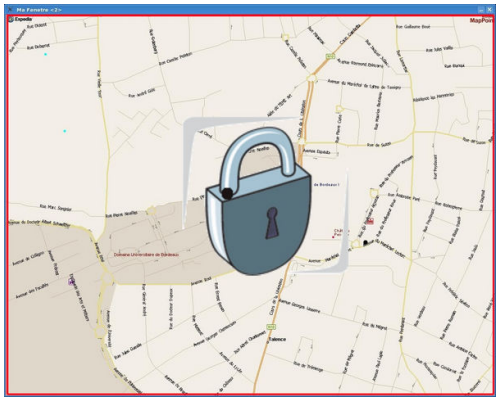
Fig 4 (c):
1. After a synchronization between *N* and another node *M*, *M* has the same version of the document as *N*.
2. *M* puts the bottom right block in the state *DESIRED*, meaning it is looking for the lock on that area. On the following synchronization with *N*, *M* then gets the lock on the bottom right block.
3. *M* modifies it and on the following synchronization *N* gets the updated block.

Fig 4 (d): *N* has released the locks on the top right and the bottom left areas of the map and at the same time it has modified the top left block by splitting it several times.

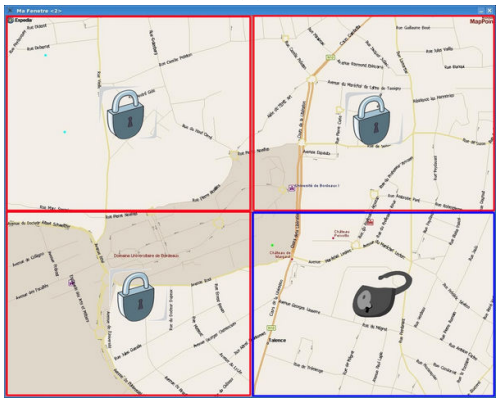
Evaluation

Evaluating such a system is not an easy task because there is no real performance issue since the quantity of data that is exchanged is very limited. The only criterion that we can measure is the synchronization time between two nodes, which depends on the quantity of information to be exchanged.

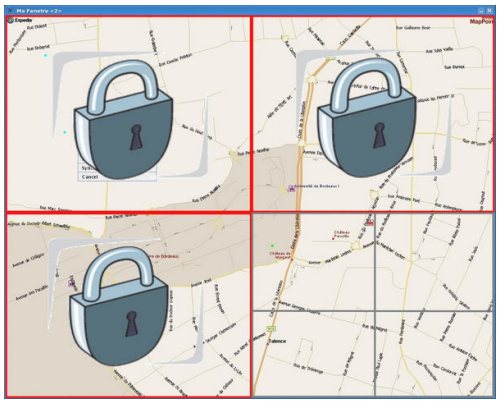
To be efficient, our system must guarantee that two users that want to synchronize their versions of the map remain reachable by each other during this process. If the synchronization time between two nodes were too long, the synchronization could fail and as a consequence the oldest version would not be updated.



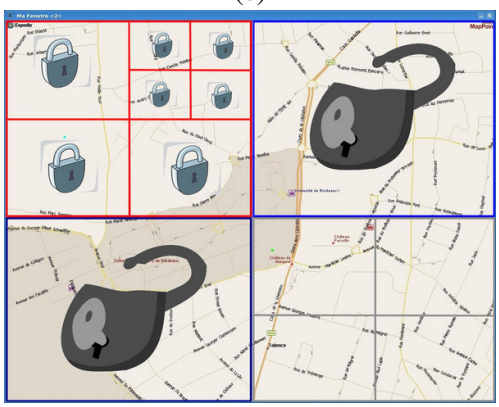
(a)



(b)



(c)



(d)

Figure 4: A scenario played with our implementation

We have tried synchronizing two nodes, with different tree structures and contents. But when the tree is modified, the quantity of data does not grow very quickly. When splitting, we add around 10 Kbytes to the internal data structure and the strategic data are at most 1 Kbyte. Thus, the tree size parameter is not very important and we have not considered that as a significant parameter for the evaluation.

The figures that we present here have been measured between two laptops running Linux Debian with Ethernet and wireless 802.11. These figures are the worst that we obtained out of thirty measures. The tree used for the synchronization was composed of twenty five leaves.

1. We first took a number of measures on an Ethernet 100Mbit/s network and the delays we obtained were comprised between 150 ms and 200 ms.

2. We then did the same measures on a wireless network 802.11g at 54 Mbit/s and the resulting delay did not exceed 300 ms.

Results of the evaluation put in context

Considering the walking speed of a soldier, which can be at most around 6 mph, and the above figures, a soldier can move by 31.5 inches during a synchronization operation. We consider this distance small enough to ensure that the MUs of the system stay in the communication range of each other during the process. In the worst case, i.e. when the connectivity is not maintained long enough, each MU stays in its previous state, i.e. its map is not updated. This ensures consistency in our system.

5 CONTEXT OF USE AND ASSOCIATED LIMITATIONS

We have identified what could be considered limitations of our system. In fact, these are due to the nature of MANets that reflect the way communication between people work in the real world. Thus, they should rather be considered as constraints due to the context of use, which we have to live with. We discuss two of them in this section.

Nature of information

Our system has not been designed to share real time information. For instance, if a unit as something critical to report we cannot guarantee that this piece of information will be made available to the other units in a reasonable amount of time: the corresponding portion of the map can be locked by another unit, unreachable or even “lost” - see bellow -.

“Lost” block, slow or broken network

Our system relies on a totally decentralized approach where the knowledge of each node is limited to its direct neighborhood. This is robust and makes it possible to support large

and versatile networks: the global system keeps on working even though the network is slow, broken, or nodes become unreachable. This is due to the fact that each node remains in a consistent state even when it becomes isolated. Nevertheless, it also has some drawbacks. In particular, if a node that has the lock on a given area of the map disappears or even crashes, the lock also disappears, and consequently it will be impossible to modify the corresponding block anymore. This problem could be solved, especially in this military context, by adding a notion of hierarchy. It should be easy to map the tree data structure corresponding to the map onto a hierarchy tree, each node (officer) of the hierarchy tree being in charge of the corresponding area of the strategic map. Officers could then be allowed to generate new locks on the sub-tree they are in charge of, modifying the associated version number (in a specific manner) at the same time.

6 CONCLUSION AND FUTURE WORK

In this paper we have presented a system that makes it possible to share a strategic map and to work on it in a collaborative and totally distributed manner. The map can be splitted into small areas that can be duplicated, distributed and updated in a decentralized manner by several mobile units at the same time. The consistency of the shared information is guaranteed whatever the evolution of the connectivity is within the MANet at hand. The context of use of the resulting system, the associated limitations and possible improvements have been discussed.

In the future, the current application will serve as a study prototype that will make it possible to elaborate additional features and solutions in collaboration with the DGA[5], so as to improve information sharing on the battlefield.

REFERENCES

[1] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). Request for Comments: 3626, October 2003.

[2] Malika Boulkenafed, Valérie Issarny, and David Menté. AdHocFS : A serverless file system for mobile users. Technical Report RR-4303, INRIA, 2001.

[3] Benjamin Atkin and Kenneth P.Birman. MFS: an adaptive distributed file system for mobile hosts. Technical report, Departement of Computer Science Cornell University, Ithaca, 2003.

[4] Hervé Roussain and Frédéric Guidec. Dissémination asynchrone d'information en mode peer-to-peer dans les réseaux ad hoc. In *Proceedings of the first french-speaking conference on mobility and ubiquity*. Université de Bretagne Sud, 2004.

[5] Siddhartha K.Goel, Manish Singh, Dongyan Xu, and Baochun Li. Efficient peer-to-peer data dissemination in mobile ad-hoc networks. In *Proceedings of International Workshop on Ad Hoc Networking*. Departement of Computer Sciences Purdue University, August 2002.

[6] Maria Papadopouli and Henning Schulzrinne. Design and implementation of a peer-to-peer data dissemination and prefetching tool for mobile users. In *Proceedings of the 2nd ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems*, 1999.

[7] Maria Papadopouli and Henning Schulzrinne. Seven degrees of separation in mobile ad hoc networks. In *Proceedings of the IEEE Globecom '00'*, 2000.

[8] Alina Bejan and Ramon Lawrence. Peer-to-peer cooperative driving. In *Proceedings of ISCIS2002 International Symposium on Computer and Information Sciences*, pages 259–264. Computer Science Departement, University of Iowa, CRC Press, October 2002.

[9] Majid Ali Khan and Ladislau Blni. Convoy driving through ad-hoc coalition formation. In *Proceedings of the 11th IEEE Real-Time and Embedded Computing Systems and Applications*, 2005.

[10] Jyh-How Huang, Saqib Amjad, and Shivakant Mishra. Cenwits: A sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of SenSys'05*. ACM, 2005.

[11] Massimo Mecella, Tiziana Catarci, Michele Angelaccio, Berta Buttazzi, Alenka Krek, Schahram Dustdar, and Guido Vetere. Workpad: an adaptive peer-to-peer software infrastructure for supporting collaborative work of human operatoes in emergency/disaster scenarios. In *Proceedings of the 2006 International Symposium on Collaborative Technologies and Systems*. IEEE Computer Society, 2006.

[12] Takaaki Umedu, Hiroaki Urabe, Jun Tsukamoto, Kazuki Sato, and Teruo Higashino Teruo Higashino. Manet protocol for information gathering from disaster victims. In *PERCOMW '06: Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, page 442, Washington, DC, USA, 2006. IEEE Computer Society.

[13] Roberto Aldunate, Sergio F. Ochoa, Feniosky Pea-Mora, , and Miguel Nussbaum. Robust Mobile Ad Hoc Space for Collaboration to Support Disaster Relief Efforts Involving Critical Physical Infrastructure. *Journal of Computing in Civil Engineering*, 20:13–27, January/February 2006.

[14] R.and Yuu-Heng Cheng Chadha, J. Chiang, G. Levin, Shih-Wei Li, and A. Polylisher. Policy-based mobile ad hoc network management for drama. In *Proceedings of MILCOM 2004 : the Military Communications Conference.*, volume 3, pages 1317 – 1323. IEEE Computer Society, 2004.

[15] BBN Technologies. OpenMap™: Open Systems Mapping Technology. http://openmap.bbn.com/.

[5] Délégation Générale pour l'Armement