

# Transhumance: A power-sensitive middleware for data sharing on mobile ad hoc networks

Guilhem Paroux<sup>\*\*</sup>, Ludovic Martin<sup>\*\*</sup>, Julien Nowalczyk<sup>†</sup>, Isabelle Demeure<sup>‡</sup>

**Abstract**—This paper presents Transhumance a middleware supporting peer-to-peer applications on pocket PC-based mobile ad hoc networks (MANETs). Transhumance provides full communication and deployment facilities as well as complete and extended data sharing facilities. It also addresses two missing points in the existing middleware: security and power management. It targets small networks of up to 20 mobile nodes.

**Index Terms**—Mobile ad hoc networks, middleware, data sharing, power-sensitive, security

## I. INTRODUCTION

A Mobile Ad hoc NETWORK (MANET) [8] is a self-configuring network of mobile nodes connected by wireless links. The nodes are free to move therefore the network topology may change. MANETs may involve laptops or lower capability nodes such as pocket PCs. MANETs are therefore also characterized by the limited capacities of their nodes in terms of energy (since the mobility implies battery-operated devices), memory and CPU. In spite of this limitation, because they do not require a pre-existing network infrastructure, MANETs should enable a whole set of new spontaneous services. An example of such service, that we shall use all along this paper, is a treasure hunting game. The game consists in finding a treasure by solving enigmas. Two or more teams are opposed and a game master holds and distributes the enigmas to the teams. The goal for a team is to be the first to solve the enigmas and to give the answers to the game master. Each team member holds a Pocket PC with a WiFi card. It is used to receive the enigmas from the game master, to communicate among team members when they are apart, to collect the answers and to return them to the game master. Other examples of services are collaborative document edition, and services allowing the users to discover their neighbourhood and to share their profile in order to meet other people.

So far, research on mobile ad hoc networks has mainly focused on routing protocols [9]. Some attempts such as

JMobiPeer [13] have been made to develop middleware for MANETs. However, none of the proposed solutions seem to adequately address major issues such as security and power management. We aim to develop a middleware for pocket PC-based mobile ad hoc networks that provides applications with full communication and deployment facilities as well as security mechanisms, data-sharing facilities and power management. The middleware is expected to support peer-to-peer applications.

The work presented in this paper is part of the French government funded RNRT-Transhumance project. The objective of this project is to develop a middleware to support peer-to-peer applications on mobile ad hoc networks. We decided to focus on peer-to-peer applications in which, contrary to client-server applications, all nodes play symmetric roles. They are more suitable to the dynamic nature of MANETs in which nodes may disappear at any time. In a client-server application, if the server disappears, the application is down.

In Transhumance, we chose to target small networks of up to 20 nodes which correspond to manageable human size groups. We assume that nodes move at pedestrian speed (up to 5 kilometres per hour) and we therefore exclude vehicular mobile ad hoc networks [15]. Transhumance is designed to support various degrees of mobility: very dynamic networks (e.g. emergency operation after an earthquake), static networks (e.g. a meeting around a table) and intermediate situations.

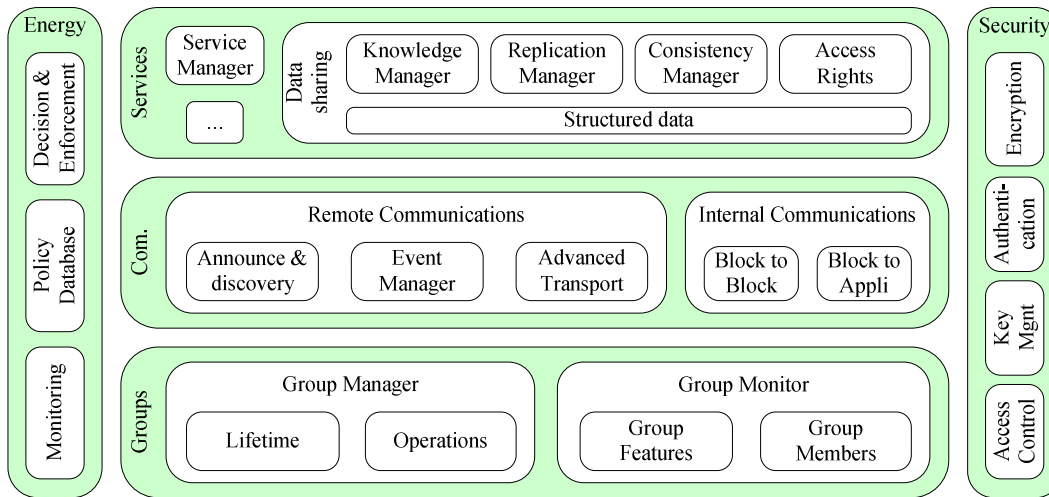
In this paper we describe the middleware that we have designed within Transhumance. The remainder of the paper is organized as follows. Section II gives an overview of Transhumance and describes its architecture. Section III describes the notion of group, as it is used in Transhumance. Section IV presents the communication mechanisms. Section V describes the data sharing. Section VI presents the security mechanisms. Section VII gives an overview of the power management in Transhumance. We conclude in Section VIII.

Manuscript received January 15, 2007. This work was supported in part by the French Government ANR-RNRT Transhumance project.

♣ France Telecom R&D, 38-40 avenue du Général Leclerc, 92 130 Issy-Les-Moulineaux, France (e-mail: guilhem.paroux@orange-ftgroup.com)

♠ Ecole Nationale Supérieure des Télécommunications (GET-ENST), 46, rue Barrault, 75634 Paris Cedex 13, France. (e-mail: firstname.name@enst.fr)

♦ Thales 160, bd de Valmy - BP 82, 92704 Colombes Cedex, France (e-mail: firstname.name@fr.thalesgroup.com).



**Figure 1: Transhulance architecture**

## II. TRANSHULANCE OVERVIEW

This section gives a brief overview of Transhulance. More precisely, we describe services provided by Transhulance to end-users and its global architecture.

### A. End-user services

To start Transhulance users require no preliminary knowledge such as security certificates or a list of resources; they must, however, define their profile (name, address, points of interest...) and their preferences (privacy, Transhulance and service configuration parameters...). This information is used to customize Transhulance and may be exchanged between users.

In Transhulance, users gather into communities sharing a common interest called *groups* (see section III for further details). Users can view existing groups with their properties (mainly the common interest and the attached services). They may join one or more group(s). They may also create new groups.

Members of a group are able to communicate with each other. They may also use services attached to the group such as data sharing, multi-user chat, vote and e-payment. Users may decide to start a new service in the group. Conditions and parameters for starting a service depend on this service.

### B. Global architecture

Transhulance middleware lies above the operating system and below the end-user applications. The figure 1 presents the architecture of Transhulance. It can be split in five major components:

- The *group management component* defines and maintains communities of users who share a common interest. Most of the Transhulance mechanisms are based on these groups.
- The *communication management component* handles both communications inside a single node and

communications between remote nodes. In the latter case, communications can be either synchronous or asynchronous.

- The *service management component* provides advanced services to applications. New services may be added in the future but currently the main service supported by Transhulance is the data sharing service.
- Transverse to the 3 above presented components, *the security management component* looks after security of the node resources, groups and communications.
- Finally the *energy management component* adapts mechanisms of all previous components to the participating nodes energy levels.

## III. GROUPS

In Transhulance, a *group* is a community of users who share a common interest. This "interest" may be of various natures: a hobby (e.g. trains, flowers...), a profession (e.g. engineer, teacher...), a membership (e.g. Thales, ENST...), etc. It may also be a specific resource to share or a service (e.g. a chat). Any Transhulance user is automatically registered in the *Transhulance* group which is the base group gathering all Transhulance users<sup>1</sup>. A user may belong to as many groups as he or she wishes; however, as we shall see in the section on "Security", group access may be controlled. Any user may create, enter or leave a group anywhere at any time. We assume that within a group members trust each other and may freely share their resources.

Figure 3 describes the treasure hunting game scenario. It shows 2 teams of 4 players and a game master who broadcasts questions and checks answers. All users belong to the **G** group which corresponds to game members. The **M** Group contains game masters (in this simple scenario, there is only one game master). Team *A* (resp. *B*) players belong to group **A** (resp. **B**).

<sup>1</sup> The group *Transhulance* (**T**) is not represented on figures. It must be implicitly added.

Finally, group **A+** (resp. **B+**) is the union of group **A** (resp. **B**) and group **M**. The notion of group is important because all Transhulance services, including security, are linked to groups. For instance, a data  $d$  is shared amongst the members of a group. Two users belonging to different groups cannot work together over  $d$ . In addition, by accepting to be part of a group users accept to share some of their resources as required by the services attached to the group even if they don't actively use the service. For example, a group member may have to host a shared data replica even if he or she does not use the shared data.

#### A. Features of groups

In the group management component, the Group Manager sub-component provides the functionalities for:

- creating a new group;
- discovering existing group;
- joining a group;
- leaving a group;
- listing members of groups we belong to.

Groups may be created as *time-limited*, *persistent* or *undefined*. By default it is *undefined*: the group exists as long as at least one user belongs to it and disappears when no user remains. In the *time-limited* mode, a timer is associated with the group. When there are no more users in the group or when the timer runs out, the group disappears automatically (even if users still belong to it). However, it is possible to extend its lifetime if a user requests it. This option frees the user of having to explicitly request to leave the group in situations where a maximum duration is known in advance (ex: meeting). It also avoids the support of no longer used groups that may unnecessarily consume resources. Finally, *persistent* group never disappears completely, even when they have no members. Each former member of the group keeps minimal information (security, services...) to rebuild it quickly.

The Group Monitor manages the knowledge about groups. In particular, it knows 1) groups features (description, persistency, and list of available services) and 2) groups members. Note that the group manager may not have the most up-to-date information but timers limit the lifetime of out-of-date data. The Group Monitor also provides a notification mechanism for warning Transhulance components when groups are modified.

#### B. Related Work

The notion of group is very often used in existing middleware for MANETs. We can differentiate two approaches: proximity groups and groups of interest. Proximity groups imply that the group members are located in the same area. They are mainly used in Steam [7] where they play a central role. The groups of interest are constituted of members sharing a common interest. There is no location condition. This is the approach of JMobiPeer [13] or Proem [11]. The groups provide different functionalities: membership management, communication mechanisms and resource sharing are the most widespread.

In Transhulance, we chose the approach of groups of interest. Since we target small networks, proximity is a less important factor than in a bigger network.

## IV. COMMUNICATION

In this Section, we present the communication stack used in Transhulance. It includes a transport protocol designed to address MANET specificities and an event-based facility that turns out to be well adapted to MANET environments. Transhulance relies on an ad hoc routing protocol, which is not included in the middleware.

#### A. Routing Protocol

The goal of the ad hoc routing protocol is to provide higher layers with routes. There are many routing protocols for MANETs. Since routing is a low-level functionality, we chose to keep it outside of the middleware and to use an existing protocol. We evaluated different solutions and chose OLSR (Optimized Link State Routing) [18]. More specifically we chose to use the "UniK" implementation [19] in particular because it supports plug-ins, which make it easy to enhance the protocol functionalities. The knowledge concerning the network topology acquired at the routing layer will be transmitted to the upper layers to improve the middleware efficiency.

#### B. Transport Protocol

The transport protocol is included in the Transhulance middleware. In MANETs, it is difficult to ensure stable connections, due to the changes in the network topology. The use of a connected protocol such as TCP is therefore not suitable. UDP constitutes a better solution although it provides limited functionalities. Hence we chose to develop a connection-less transport protocol, enhancing UDP functionalities and supporting fragmentation. The protocol enforces three modes of operation: simple, acknowledged and secured.

In the *simple* mode, the protocol behaves like UDP: the packets are formatted and transmitted to the destination without any other control; packet delivery is not guaranteed.

In the *acknowledged* mode, the receiver must acknowledge the packets received. However it is not a connected protocol, because the sender and the receiver did not previously establish the communication. This mode is typically used during file transfer, in order to make sure that the file has been completely received.

The *secured* mode is an acknowledged mode with encrypted messages. The keys used to cipher the messages are generated by the security block. The secured mode is mainly used to send confidential data.

The transport protocol supports a socket interface that is available to the other middleware components. It supports point-to-point as well as group message-passing. The transport protocol is mainly used by the event service and the file transfer service. Note that the applications and high-level services are more likely to use the event services in order to

communicate.

### C. Event-based communication

The transport protocol provides basic communication facilities through sockets. Transhulance also provides the applications with an event-based communication service. The event service provides functionalities to create and filter events. An event can be seen as a structured message, composed of the following fields:

- The *Type* of the event: advertisement, data, query, answer, undefined.
- The *Identifier* is a unique ID to identify the event in the network.
- The *Subject* of the event is a free character string.
- The *Content* represents the data contained in the event.
- The *SenderID* represents the sender of the event.
- The *Lifetime* in minutes.
- The *Persistence* indicates if the event is persistent (with delivery guarantee) or not.
- The *Range* indicates if the event is internal (for the local system) or external (for the network)

The event service follows the publish/subscribe model. A device interested in receiving particular events must subscribe. The event service proposes filtering facilities. An application can create filters on the event subject, the sender, the content, etc. When an event is received, it goes through the different filters and the event is notified to the corresponding subscribers. Otherwise, the event is dropped.

The events are also used in other middleware for MANETs such as Steam [7] and Emma [12]. Contrary to Steam, we do not take into consideration the distance between the sender and the receiver. Our approach is more similar to that of Emma (e.g. every device can communicate with all other devices). However, the events do not use a dissemination (epidemic) algorithm to reach their destination. They rely on the transport protocol. If the event has multiple recipients, it is sent to each recipient.

## V. DATA SHARING

Data sharing provides a virtual common space or "*sharing space*" where the members of a group share data as illustrated by Figure 2. Data put in the sharing space can be read and modified in a collaborative way. Once shared, a data cannot be removed from the sharing space.

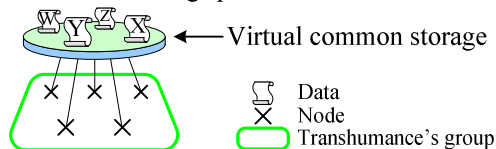


Figure 2: Sharing space

Because of node mobility and their potential disappearance, we avoid the use of a central server for hosting the sharing space. Therefore, in practice, the sharing space is distributed over the local memories of the sharing space members. For instance, in Figure 3, the sharing space of the treasure hunting game is hosted by the group **G** members. Shared data are

accessible by any player. In Figure 3, shared data are: the questions (**Q**), the responses of team *A* and *B* ( $R_A$  and  $R_B$ ) and the solution (**S**).

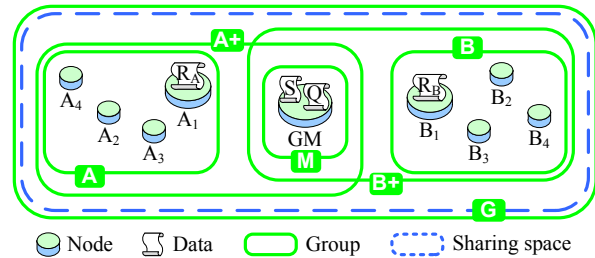


Figure 3: Treasure hunting game scenario

Because of potential network partitions, data must be replicated. The goal is to have at least a replica of each data in each network partition in case of partition. In addition the replicas must be kept consistent when a modification (write) is performed. Another consequence of the sharing space distribution is that the knowledge of its content (i.e. shared data) may be partial and/or out-of-date. Finally, all shared data should not be accessible by everybody. For instance, in Figure 3, the responses of team *A* should not be readable by team *B*.

The following sections address in more details the aspects relative to knowledge, replication, consistency and access rights.

### A. Knowledge Manager

The knowledge manager component maintains a view of the sharing space as up to date as possible. It provides research functions. The research can be based either on the data name or on the data description. In fact, in Transhulance metadata are associated to data as illustrated by Figure 4. These metadata correspond to data attributes such as author, keyword list and abstract. Metadata also define the structure of data that may be split into semantically independent parts. These parts, called *segments*, are defined manually by the user who creates it. The splitting allows for small sharing units. It simplifies consistency management by avoiding conflicts and saves energy and bandwidth by transmitting only relevant data parts. For instance,  $R_A$  may be split in segments in such a way that each segment corresponds to the response to a given question (see Figure 5).

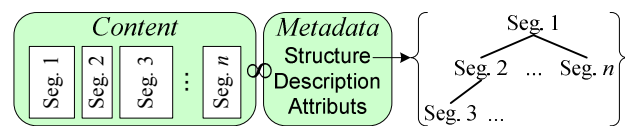


Figure 4: Structures data of Transhulance

The metadata is small compared with the data content. Therefore, the Knowledge Manager caches all metadata corresponding to data previously discovered and favours local researches over network wide researches in order to answer faster and to consume less energy.

### B. Replication Manager

The replication manager component maximizes data accessibility by replicating data. It prevents data unavailability in case of network partition and data disappearance when the

storing user leaves. However, all data do not have the same importance. Thus users must specify the lifetime of data they share. A data can be *temporary*, *group-linked* or *persistent*. *Temporary* data are not managed by the Replication Manager and are the first to be removed when free memory is needed. *Group-linked* data disappear when the corresponding group disappears. And *persistent* data are stored on some users' devices after the group disappearance. In addition, a user may request to permanently host a replica of a given persistent data.

The Replication Manager enforces a collaborative replication protocol inspired by [1] and [2]. It assumes that several nodes that collaborate may achieve a higher accessibility level than if they act independently. The protocol replicates data according to the needs of the local user, the needs of its neighbours and the number of replicas already existing in the neighbourhood. Currently the metric used to evaluate user's needs is the data access frequency.

The mechanism limits the replication of the same data which wastes the global storing space. It also restrains the risk of disappearance for non-frequently used data.

Finally, Transhumance adapts its replication strategy to the energy level: little energy implies little collaboration.

### C. Consistency Manager

Two types of consistency models can be distinguished: the *pessimistic* ones prevent replicas from diverging whereas the *optimistic* ones allow replicas to diverge and to be resynchronized. Each model has pros and cons. The optimistic consistency does not restrain data availability but the reconciliation of diverging data may be expensive. The pessimistic consistency is simple and efficient but is not usable with mobile nodes. Some systems, like AdHocFS [3], provide a *hybrid* consistency that is locally pessimistic and globally optimistic for combining the best of the two models.

Transhumance also provides a hybrid protocol that prevents replica conflicts in one-hop neighbourhood by announcing write operations. However, distant replica may diverge. Thus, this protocol is able to detect and resolve existing conflicts. In addition, it works in all situations. In a small stable network, it behaves as a pessimistic protocol. In a very dynamic network, it behaves as an optimistic protocol. And in an intermediate network with small stable groups, it behaves pessimistically in these groups and optimistically otherwise. Thus, hybrid consistency is the default mode of Transhumance. However, it is not the most efficient regarding energy consumption. Thus, users may choose to share a data with a "true" optimistic or pessimistic consistency. Dynamic switch between consistency models is not allowed.

### D. Access Rights

A user must specify access rights when it shares a data. Access rights are based on the Transhumance groups that this user belongs to (when it shares the data). For instance, on Figure 5, when the game master shares **Q** (the questions), he or she specify access rights for groups **M**, **G** and **T**. He may also assign specific rights to groups **A+** and **B+**. But he may not

modify rights for groups **A** and **B**.

In addition, rights may be different on different segments. For instance, in the treasure hunting game, initially **S** is not accessible to players of team **A** or **B**. But if team **A** is too late, the game master may decide to give it the response *i* as shown in Figure 5 (in exchange of a penalty).

Since segments are hierarchically structured, access rights are inherited from parents to children. Thus it is not required to define access rights for all segments. Users must only specify rights for the data root (see **Q**, **S**, **R<sub>A</sub>** and **R<sub>B</sub>** in Figure 5).

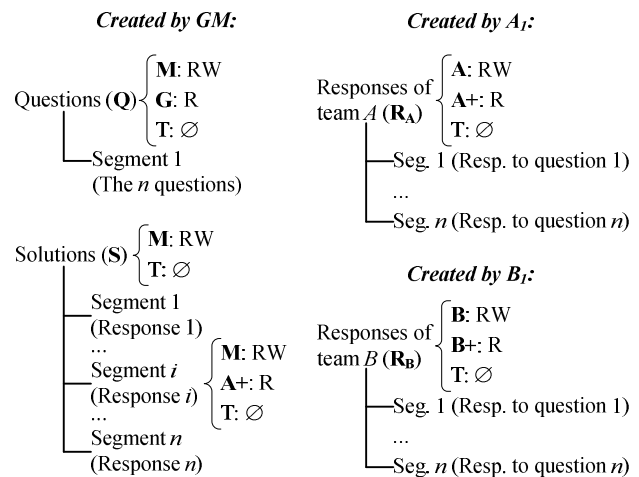


Figure 5: Examples of access rights

### E. Related work

The data sharing service of Transhumance is designed for any kind of MANETs (static, very dynamic or intermediate) involving limited capability devices. The middleware that are most similar to Transhumance are XMIDDLE [4] and AdHocFS [3].

XMIDDLE, developed at University College of London presents the user with a tree data structure based on XML data. In addition, it uses semantic information to improve automatic reconciliation. Unfortunately, this structure proved to be unsuitable for limited capability devices and difficult to keep consistent because of its dynamic sharing unit. The advantage of Transhumance is to separate metadata from the content of data. Only metadata must be kept in active memory while content is accessed directly on storage memory (with system primitives). Moreover, segments (defined statically during data creation and contained into files) are easier to reconcile than XML branches. And tools of traditional wired file system can be reused.

AdHocFS, developed at INRIA of Rocquencourt France, provides collaborative replication and hybrid consistency. Unfortunately, these protocols are based on network clustering which simplifies algorithms but requires minimal network stability. We do not make any assumption about network stability. Thus our protocols do not rely on clustering.

AdHocFS is concerned with energy saving and its protocols were designed for minimizing communications. Transhumance goes further. It adapts the features of its protocols according to

the energy level. For instance, the collaborative replication becomes more and more individualist when the energy decreases.

We found no literature about access rights specifically designed for MANETs. In particular, in XMIDDLE, all data are public. And, AdHocFS provides "full or no" control and is unable to define read only accesses.

## VI. SECURITY

One of Transhulance challenges is to provide a decentralized security model for services and applications in mobile ad hoc networks. It means that the proposed security mechanisms must be able to work in a fully autonomous way.

Most of the existing middleware do not integrate any security features ([7], [10], [12]). Others, such as [13], adopt a common certificate-based approach that implies the use of a centralized infrastructure during an initialization phase. [11] introduces a decentralized reputation model, but trust model are not adapted to our context because they introduce a strong dependency to the network environment (obtaining a consistent trust value is related to the number of successful relationships with the other networks members).

In addition, Transhulance provides access control mechanisms to the data sharing system. Existing systems such as [3] or [14] answer authentication and confidentiality requirements thanks to an initialization phase allowing to recover a certificate from a fixed architecture. But none of them proposes an access control system; it is therefore an innovative component for mobile ad hoc data sharing systems.

The Transhulance security model is based on a group approach. This approach fits most of the ad-hoc applications architectures [5] and facilitates the integration of security, especially for access control mechanisms. Existing group security studies concentrate on contributory key agreement protocols [6], but these protocols cause a lot of computational and bandwidth costs which can not be afforded in pocket-PC based MANETs. Moreover, they are not designed to be easily integrated in an access control system.

Our security model is composed of the following functional blocks: authentication, key management, encryption and access control.

The *Authentication* block deals with the admission of new members in a security group. A security group is a set of peers with hierarchical structure. It is identified thanks to a shared secret: the group public key. This group public key is distributed in the shared space and can be attached to different levels of trust. For example, the public key of group **A** is considered as more trustful if it is delivered by group **A** than by group **A+** (Figure 3). The admission process consists in the transmission of the group key from a group member to a requester. The requester may ask any member of a given group to let him enter the group. By co-opting the requester, the group member acts as a trust authority. Once in a group, a member is considered as trustful as the other members and may in turn act as an admission authority. Finally, different

admission modes are available: Diffie-Hellman keys exchange, proximity channels (i.e. set up a private auxiliary transmission channel for example using Bluetooth), or unencrypted keys exchange in clear text. These various admission modes aim at providing flexibility to the user by using contextual authentication during the bootstrapping phase.

The *Key Management* block is a central point. It maintains the overall secrets necessary to apply the proposed security model and it deals with the distribution of the keys within the shared space.

The *Encryption* functional block offers a set of security functions to encrypt, decrypt and sign applicative data, ensuring confidentiality and integrity. This block relies on the key management block which provides the right cryptographic keys.

The *Access Control* block enforces advanced access right patterns maintained by the Data Sharing service (see section V.D). These patterns are defined by the data creator for all the groups he belongs to. The access rights for a piece of data published by  $A_1$  belonging to the group **A**, **A+** and **T** are given in Figure 5. Data and access rights are then signed with the private keys of all the groups previously defined in order to ensure data integrity over the whole shared space. In Figure 3, since group **A** is the only one that has the right private group key, if another group modifies the data, all the nodes of the shared space will detect this modification by verifying the signature with the public key of group **A**.

Finally, the access control model implies the use of asymmetric key pairs. Our group approach reduces cryptographic costs. But the challenge still remains in dealing with the cryptographic computations, not only during the user data exchanges but also during the underlying data reconciliation phases.

## VII. POWER MANAGEMENT

The mobile devices involved in mobile ad hoc networks, have limited energy capacities. We aim to propose a solution to manage the available power, considering the users and the applications needs. Some solutions regarding power management in MANETs have been proposed [16], [17]. However, none has been integrated into a middleware.

### A. Overview

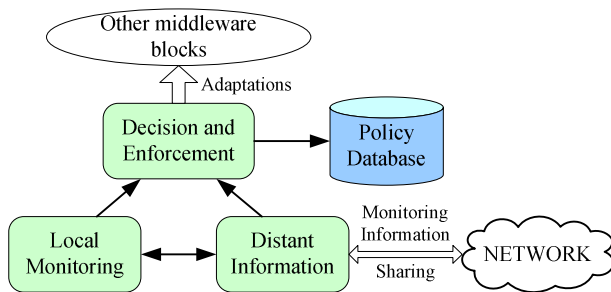
Transhulance is a power-sensitive middleware: it can adapt its behaviour to lower or raise power consumption. In order to do so, each block of the middleware is designed with the objective of being adaptable. Several kinds of adaptations can be performed such as: change argument values, use alternative algorithms and give up some services if they are too costly. For instance, in the transport layer, it is possible to use two different encryption algorithms when using secured communication. The first has a high cost and is much secured whereas the second has a low cost but is less secured.

Policy-based adaptation is enforced. A policy is a set of "if ... condition ... then ... adaptation action ..." rules. When the conditions are reached, the corresponding actions are taken.

The policies are stored in the policy database and they are managed by the decision and enforcement block.

We finally introduce two notions to refine the power management: the *behaviour* and the *activity*. They are set by the user. The behaviour represents the user's participation in the network. It can be *selfish* (low participation) or *generous* (collaborating actively). The activity represents the working operating mode. It can take the following values: *meeting* (motionless participants), *networking* (interacting mobile group of devices), or *seeking* (intermittent connection). The activity is used to determine the network profile. Advantage can be taken from the knowledge of the environment. For example, when the activity is set to "meeting", because the participants are in a same room, it is possible to reduce the power consumption by limiting the transmission range of the messages (if we consider that in a same room, each couple of devices can communicate directly).

### B. Operation



**Figure 6: Power management architecture**

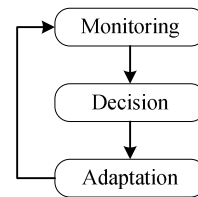
Figure 6 shows the architecture of the power management in Transhumance.

The Monitoring block collects information such as battery level, estimated lifetime, network activity and CPU charge. It is exploited in order to determine the evolution of the local system. Alerts can be set up in order to warn other blocks of changing conditions.

The Neighbourhood block manages information about the remote devices in the network. It periodically sends the local monitoring information to the other devices and collects the remote information. The collected information is grouped to establish statistics about the remote devices working conditions. The information also contains the activity and the behaviour that have been chosen by the other devices.

The Decision and Enforcement block is the central point of the power management. It aggregates the local and global information in order to take decisions. It checks the various rules conditions and sends the adaptation actions to be performed to the corresponding middleware blocks.

Figure 7 shows the power management cycle described previously.



**Figure 7: Power management cycle**

## VIII. CONCLUSION AND FUTURE WORK

In this paper we presented the design of a middleware for mobile ad hoc networks named Transhumance. It aims to support peer-to-peer applications on MANETs with a particular interest in data sharing. It provides the applications with communication mechanisms, group facilities and resource sharing and discovery facilities. It also addresses two missing points in the existing middleware: security and power management.

In Transhumance, the notion of group of users sharing a common interest is central. Most Transhumance mechanisms are based on them. In particular, this is the case for communication, data sharing and security services.

Transhumance provides an advanced data sharing service. This service uses a new data structure suitable for limited capability devices and for making consistency management easier. Users can choose between three consistency models according to the network stability: optimistic, pessimistic and hybrid consistency. A high accessibility level is achieved thanks to a collaborative replication protocol that takes into account the needs of local user and neighbours. Finally, Transhumance fills a gap in MANET data sharing: it proposes a way to define access rights based on groups of interest.

Enforcing security at the group level facilitates both access control, and the definition of group security policies in a totally decentralized ad hoc context. A challenge however lies in the efficient integration of the access control mechanisms with the data sharing system; a difficulty resides in the expensive signature checking phase that is involved both when a user requests data access and during the data reconciliation process.

Transhumance provides an integrated solution for power management. The proposed solution aims to adapt the middleware behaviour to the user's needs as well as to the local and global working conditions. The two main challenges are to provide good solutions of adaptation for a majority of working conditions and to ensure the coherence of the behaviours in the network. We expect to show that the energy consumption is adapted to the user's needs: low consumption to increase the battery lifetime, higher consumption to have full functionalities.

The middleware specifications have been completed and the development has begun. The implementation is done in C++ on Nokia 770 devices [20] under the Linux operating system. We intend to test the middleware both on a MANET emulator and in real conditions in order to prove the efficiency of our solutions.

## REFERENCES

- [1] M. Shinohara, H. Hayashi, T. Hara, S. Nishio, "Replica Allocation Considering Power Consumption in Mobile Ad Hoc Networks". In Proc. of PerCom Workshops 2006, p. 463-467
- [2] L. Yin and G. Cao, "Balancing the Tradeoffs between Data Accessibility and Query Delay in Ad Hoc Networks". In Proc. of 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS'04), pp. 289-298, 2004.
- [3] Malika Boulkenafed, Valérie Issarny, "AdHocFS: Sharing Files in WLANs". In proc. of the Second IEEE International Symposium on Network Computing and Applications, p.156, April 16-18, 2003
- [4] C. Mascolo, L. Capra, S. Zachariadis and W. Emmerich. "XMIDDLE: A Data-Sharing Middleware for Mobile Computing". In Personal and Wireless Communications Journal 21(1). Kluwer. April 2002.
- [5] Y. Kim, D. Mazzocchi, G. Tsudik, "Admission Control in Peer Groups". Second IEEE International Symposium on Network Computing and Applications, NCA 2003, April 2003, p. 131-139.
- [6] R. Bashkar , "Group Key Agreement in Ad-Hoc Networks". Technical report RR-4832, INRIA-Rocquencourt, May 2003. <http://www.inria.fr/rrrt/rr-4832.html>
- [7] Meier R., Cahill V.: "STEAM: Event-Based Middleware for Wireless Ad Hoc Network". Proceedings of the International Workshop on Distributed Event-Based Systems, Vienna, Austria, p. 639-644 (2002).
- [8] Chlamtac I., Conti M., Liu J. J.-N.: "Mobile ad hoc networking: imperatives and challenges". Ad Hoc Networks, Elsevier, Vol. 1, Issue 1, p. 13-64 (2003).
- [9] Abolhasan M., Wysocki T., Dutkiewicz E.: "A review of routing protocols for mobile ad hoc networks". Ad Hoc Networks, Elsevier, Vol. 2, Issue 1, p. 1-22 (2004).
- [10] D. Gorgen, H. Frey, J. K. Lehnert, P. Sturm : « SELMA: A middleware platform for self-organizing distributed applications in mobile multihop ad-hoc networks » (2003).
- [11] G. Kortuem, "Proem: a middleware platform for mobile peer-to-peer computing". ACM SIGMOBILE Mobile Computing and Communications Review, vol. 6, no. 4, p. 62-64, 2002.
- [12] M. Musolesi, C. Mascolo, S. Hailes, « EMMA: Epidemic Messaging Middleware for Ad hoc networks ». Journal of Personal and Ubiquitous Computing, Springer, Vol 10, No 1, February, 2006, p. 28-36. .
- [13] M. Bisignano, G. Di Modica, O. Tomarchio, « JMobiPeer: a middleware for mobile peer-to-peer computing in MANETs ». First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW'05) pp. 785-791.
- [14] Plagemann T., Andersson J., Drugan O., Goebel V., Griwodz C., Halvorsen P., Munthe-Kaas E., Puzar M., Sanderson N., Skjelsvik K.: "Middleware Services for Information Sharing in Mobile Ad-Hoc Networks: Challenges and Approaches". Proceedings of IFIP Workshop Challenges of Mobility, Kluwer (2004).
- [15] M.M. Artimy, W. Robertson, W.J. Phillips: "Connectivity in inter-vehicle ad hoc networks", Canadian Conference on Electrical and Computer Engineering, 2004. Volume 1, Page(s):293-298
- [16] J.-R Lorch, A.-J. Smith: "Software strategies for portable computer energy management", IEEE Personal Communications Magazine, vol. 5, no. 3, p. 60-73, 1998.
- [17] L.M Feeney: "Energy-efficient communication in ad hoc wireless networks", in Mobile Ad hoc Networking, IEEE Press, Pages 301-327, ISBN 0-471-37313-3.
- [18] T. Clausen, P. Jacquet. "Optimized Link State Routing Protocol (OLSR)" Technical Report RFC 3626, IETF, 2003.
- [19] UniK OLSR implementation: <http://www.olsr.org>
- [20] Nokia 770 presentation: <http://europe.nokia.com/770>