An Energy-Aware Middleware for collaboration on small scale MANets

Isabelle Demeure, Guilhem Paroux, Javier Hernando-Ureta, Amir R. Khakpour, Julien Nowalczyk.

***Abstract*— This paper presents an energy aware middleware to support collaborative applications on small scale Mobile Ad-hoc Networks (MANets) made of handheld terminals. This middleware provides a set of communication facilities including a publish-subscribe event system robust to transient disconnections, security features and energy management. The paper gives an overview of the middleware architecture, presents its main functionalities and explains how the middleware is made energy-aware. It presents measurements and actual experiments that were conducted to validate the middleware. The middleware is distributed under LGPL licence on SourceForge.**

***Index Terms*— MANets, middleware, energy management, security , event system.**

I. Introduction

This paper presents Transhumance, an energy-aware middleware to support collaborative applications on small scale Mobile Ad-hoc Networks made of handheld terminals. A Mobile Ad hoc Network (MANet) is a self-configuring network of mobile nodes connected by wireless links [1]. In a MANet, the nodes may act both as terminals and routers. In Transhumance, we target small networks of up to 20 handheld nodes, which correspond to manageable human size groups. We assume that nodes move at pedestrian speed. The handheld terminals are characterized by their limited capacities in terms of energy (since the mobility implies battery-operated devices), memory and CPU.

Transhumance targets spontaneous collaborative services such as the “Team Exploration” treasure hunting game that was used for the final project experiments [2]. Other examples are services allowing users to discover their neighbourhood and to share their profiles in order to meet other people.

MANets bring new constraints: in particular, since nodes act both as end-user terminals and as routers and since they are mobile, they may become out of reach (for a short or long period of time, or even permanently). This must be addressed by the middleware in order to ease the development and the deployment of applications. The middleware must also provide applications with common features such as communication mechanisms, resources discovery and security management. Energy management is also a crucial issue for battery operated mobile devices [3].

We conducted a bibliographical study of existing middleware systems [4] in which we analyzed systems such as Proem [5], JMobiPeer [6] and Steam [7]. The study showed the existence of a common set of functionalities in the studied systems: communication protocols, group structure and service discovery. It also showed that security and energy management were often identified as key issues, but were never completely addressed in a fully integrated and operational solution.

Transhumance is designed to meet this requirement. It provides the applications with full functionalities for communication (transport protocol, event service), group management, service discovery and management and additional high-level services such as vote and chat. Transhumance also provides security mechanisms based on groups. Concerning the energy awareness, Transhumance middleware is designed to be adaptable to the energy level.

As we shall see later in this article, the experiments conducted with groups of users running a game application showed the interest of using a middleware such as Transhumance when the ad-hoc communications are not stable. The measurements performed also showed the ability of the middleware to significantly reduce energy consumption thanks to the energy management. The middleware is distributed under LGPL licence on SourceForge [8].

The remainder of this paper is organized as follows. Section 2 gives an overview of Transhumance and describes its architecture. Section 3 focuses on the communications in Transhumance. Section 4 presents the security mechanisms. Section 5 is dedicated to the power management in Transhumance. Section 6 presents the first experiments performed. We conclude in Section 7.

Manuscript received September 12, 2008. This work was partly supported by the French National Research Agency (ANR) Transhumance project and by a contract with Orange Labs.

I. Demeure is with TELECOM ParisTech, Institut TELECOM, 46, rue Barrault, 75634 Paris Cedex 13, France. (phone: +33.1.45.81.72.86; fax: +33.1.45.81.31.19; e-mail: isabelle.demeure@telecom-paristech.fr).

G. Paroux is with TELECOM ParisTech, Institut TELECOM, 46, rue Barrault, 75634 Paris Cedex 13, France. (e-mail: guilhem.paroux@telecom-paristech.fr)..

J. Hernando-Ureta was with TELECOM ParisTech, Institut TELECOM, 46, rue Barrault, 75634 Paris Cedex 13, France. His current e-mail: is javier.hernando@gmail.com.

A. R. Khakpour was with TELECOM ParisTech, Institut TELECOM, 46, rue Barrault, 75634 Paris Cedex 13, France. He is now a PhD student at Michigan State University (e-mail: khakpour@msu.edu).

J. Nowalczyk is with Thales Communications, 160, bd de Valmy - BP 82, 92704 Colombes Cedex, France. (e-mail : julien.nowalczyk@fr.thalesgroup.com).

II. TRANSHUMANCE ARCHITECTURE

This section gives an overview of the Transhumance middleware architecture and functionalities. Note that an earlier description of Transhumance can be found in [20]. It was done when we were in the process of developing the middleware. Since then, we finalized the implementation, tested validated and integrated the Transhumance functionalities. In this process some of the functionalities were reviewed, completed and refined.

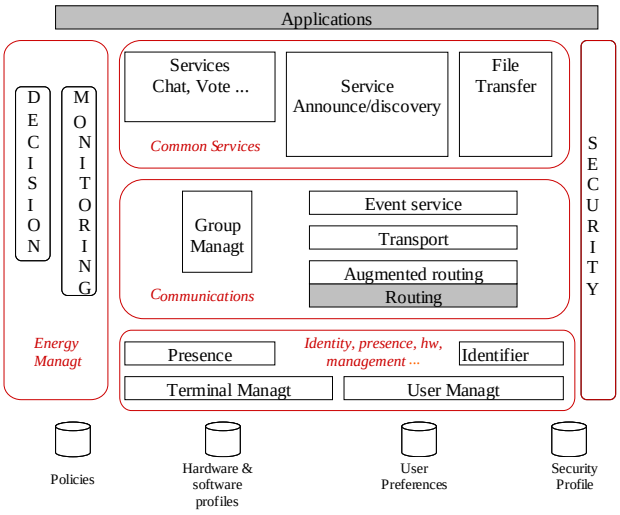

Fig. 1 - Transhumance Node Architecture

Figure 1 presents the architecture of Transhumance. It is organized into five functionality blocks: energy management, communications, "identity, presence & hardware management", common services and security.

The *energy management* block involves a monitoring module and a decision module. The decision module decides, based on a policy and on the information about the energy collected by the monitor, of the adaptation actions to be executed. The possible actions are implemented in middleware modules and consist in adaptations of their behaviour that reduces the energy consumptions (e.g. stopping messages acknowledgements in the transport protocol). Energy management is detailed in Section 5.

The *communication* block relies on the OLSR routing protocol [9]. This choice will be discussed in Section 3. Transhumance "augments" the routing protocol with additional functions provided by the augmented routing module. The Transport module is a UDP-based transport protocol that supports message fragmentation, acknowledgment (optional) and message encryption (optional). Transhumance also supports a publish-subscribe event-based system that enforces message persistency ("guaranteed" message delivery). In addition, this event system provides basic point-to-point and group message passing in push-pull mode. Finally the communication block includes a group management module that is in charge of managing communities of users sharing common interest. Services and security are offered within groups.

The "identity, presence & hardware management" block includes 4 modules: user management, terminal management, identifier and presence. Users of Transhumance do not need any preliminary knowledge such as security certificates or a list of resources; they must, however, define their profile (name, address, points of interest, etc) and their preferences (privacy, Transhumance and service configuration parameters, etc). This is supported by the user management module. Terminal management abstracts some useful functionalities of the underlying operating system and hardware, such as file system calls and interface to battery. It therefore acts as an adaptor. The identifier module manages 3 types of identifiers: terminal hardware identifier, communication identifier (in practice an IP address chosen within an interval) and user identifier.. The presence module indicates who is present in the MANet and within how many hops.

The *advanced services* block regroups high-level services such as chat service, file transfer service and voting service. It also provides an announcement / discovery service to advertise existing services.

T*he security component* looks after security of the node resources, groups and communications. It is managed in a fully distributed way. It comprises ciphering and a certificate manager. It will be discussed in more details in Section 4.

In the following sections we provide details on the communication block, the security block and the energy management block.

III. COMMUNICATIONS

In this Section, we come back on the Transhumance communication stack introduced before.

A. Routing Protocol and Augmented Routing

Transhumance relies on a multihop ad-hoc routing protocol. We evaluated different solutions and chose OLSR (Optimized Link State Routing) [9]. More specifically we chose to use the "UniK" implementation [10] in particular because it is operational and supports plug-ins, which make it easy to enhance the protocol functionalities. OLSR is often said to be limited in terms of scaling but it turns out to perform well in Transhumance that targets small scale networks of up to 20 nodes. The knowledge concerning the network topology acquired at the routing layer is forwarded through a plug-in to the upper layers in a cross-layer spirit, to improve the middleware efficiency. The topology information is used in particular by the event system.

B. Transport Protocol

In MANETs, it is difficult to ensure stable connections, due to the changes in the network topology. We therefore chose a non-connected UDP based solution. We extended UDP with fragmentation, acknowledgement and encryption facilities. In order to be adaptable to the energy level, we chose to enforce three modes of communication: simple, acknowledged and secured.

The *simple* mode is UDP in which the packets are

transmitted to the destination without any other communication control and thus packet delivery is not guaranteed. In the *acknowledged* mode, the receiver must acknowledge the packets received. The retransmission and acknowledgement model are designed in a way to resist transient nodes disconnections (of less than 10 seconds according to the measurements conducted, see [12]).

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 provides a socket interface that is available to the other middleware components. It supports point-to-point as well as group message-passing.

Our solution is most similar to TPA (Transport Protocol for Ad hoc Networks) an efficient transport protocol for MANets [12]. Both are based on UDP and provide fragmentation and acknowledgement. However, our transport protocol was designed to be adaptable to the energy level; besides, at the time we were developing Transhumance no TPA source code was available.

C. Event-based Communication

The event service provides functionalities to create and filter events. An event can be seen as a structured message, composed of the following fields:

- Type of the event: advertisement, data, query, answer, undefined.
- Identifier is a unique ID to identify the event in the network.
- Subject of the event is a free character string.
- Content represents the data contained in the event.
- SenderID represents the sender of the event.
- Lifetime in minutes.
- Persistence indicates if the event is persistent (with delivery guarantee) or not.
- 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. In order to address transient disconnections and network partitioning the event system supports event persistency (an event may be kept for a given time or until it is delivered to all its subscribers).

Events are also supported in other middleware for MANETs such as Steam [7] and Emma [13]. 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. Transhumance employs Chapar, a novel event system that uses the underlying routing protocol for event dissemination [16]. Contrary to other event systems that rely on a single broker to handle event publications and subscriptions [14, 15], Chapar replicates the event brokers on the Multipoint Relays (MPRs) defined in OLSR. This choice is made to avoid having a single point of failure and a performance bottleneck. The event brokers are responsible for subscription and for leading the published events to their corresponding subscribers. Thus, we may consider Chapar as a self-configured overlay network using cross-layer information to store and forward the events from publisher(s) to subscriber(s). Using the underlying routing layer enables us to constitute the multicast trees to deliver events instead of using expensive unicast communication and flooding which is not scalable. Moreover, the OLSR routing information empowers us to handle node mobility in the network and cope with network transient topology changes.

The event system supports both real-time event dissemination and storing events in the brokers until their lifetime elapses. This functionality helps the event system to deal with network partitioning which is likely to happen in actual mobile ad-hoc networks. For instance, if the network is portioned into two partitions, the events published in one partition cannot be notified to subscribers connected to the other partition. However, using memorized events, the published events are stored in the broker nodes and as network topology changes over time, any subscriber that joins the network within the event lifetime period is notified of the event.

In Chapar, the subscription and the notification are one-hop communications since each node has at least one broker in its neighborhood. Thus, for real-time event forwarding to subscribers, the intermediate brokers that constitute the overlay network lead the published event to the right broker nodes, so it will be delivered to the subscriber. For memorized events, a copy of the published event is stored in every single broker in the network. The subscribers who are present at the publication time are notified of the event, and those who are disconnected from the network will be notified when they connect to the network.

In [16], we show that the event system yields good performance in particular because it causes less network overhead than other solutions. All Chapar calculations and algorithms are based on simple hash functions and logical operations which makes it very light in terms of computation and resource consumption.

D. Groups

The notion of group is very often used in existing middleware for MANETs. Two approaches are distinguished: 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 where they play a central role. Members of a

group of interest share a common interest. There is no location condition. This is the basic approach in JMobiPeer and Proem. The groups provide various functionalities: membership management, communication mechanisms and resource sharing are the most widespread.

In Transhumance, the group of interest approach is preferred. Since we target small networks, proximity is a less important factor than in a bigger network. When a user enters the MANet, he or she is automatically registered in the *Transhumance* base group. A user may belong to as many groups as he or she wishes provided he or she is granted access (see Section 5). We assume that group members trust each other. By accepting to be part of a group users accept to share their resources as required by the services attached to the group.

Group management is implemented in a decentralized way. It provides the functionalities for creating a new group, discovering an existing group, joining a group, leaving a group and listing members of groups we belong to.

Groups may be created as *time-limited, persistent* or *undefined (default).* If *undefined*, the group exists as long as at least one user belongs to it. 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. Finally, a *persistent* group never disappears completely, even when it has no members. Each former member of the group keeps minimal information (security, services, etc) to rebuild it quickly.

On each node, a group monitor component manages the information related to groups. In particular, it knows:

- groups features (description, persistency, and list of available services) and
- groups members (only for groups that the user belongs to).

Note that each node may not have the most up-to-date information. A notification mechanism is provided for warning Transhumance components when groups are modified.

IV. Security

One of Transhumance challenges is to provide a decentralized security model for services and applications in mobile ad hoc networks. The proposed security mechanisms must therefore be able to work in a fully autonomous way and should not rely on a server that could become out of reach during the ad-hoc network life. This precludes the use of traditional solutions that rely on servers either to distribute certificates or to authenticate users.

Most of the existing middleware such as Steam and Emma do not integrate any security features. Others, such as JMobiPeer, adopt a common certificate-based approach that implies the use of a centralized infrastructure during an initialization phase. Proem 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).

Existing systems such as [17] answer authentication and confidentiality requirements thanks to an initialization phase allowing to recover a certificate from a fixed architecture. The Transhumance security model is based on a group approach. This approach fits most of the ad-hoc applications architectures [18] and facilitates the integration of security. Existing group security studies concentrate on contributory key agreement protocols [19], but these protocols cause a lot of computational and bandwidth costs which can not be afforded in pocket-PC based MANETs.

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

The *Authentication* block deals with the admission of new members in a security group. A security group is a set of peers with no 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. 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 group.

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. We use asymmetric key pairs. Our group approach reduces cryptographic costs.

V. ENERGY MANAGEMENT

A. Motivations

Transhumance is designed to run on handheld terminals that have limited power resources. Energy management is usually well addressed in hardware design (such as low-power CPU or wireless network card) and in the operating system (switch to idle mode when unused, adjust screen brightness) [3, 21]. It is also well addressed in the design of MANet specific energy-

aware routing protocols [22, 23]. However, little is done regarding energy management at the middleware layer in spite of the fact the middleware services (communication or discovery mechanism, for instance) constitute an important source of energy consumption. SANDMAN proposes a solution for an energy-aware middleware for MANET [28]. SANDMAN periodically turns the wireless card of the nodes to sleep mode in order to save energy. Service discovery and message transport relies on network clustering: cluster heads are responsible for gathering the messages and for waiting for destination nodes to wake up. This approach is very efficient thanks to the low power consumption of nodes in sleep mode. However, it implies to constitute and to maintain cluster in the network. Moreover, the cluster heads support heavier trafic than the other nodes and in case the users adopt selfish behaviours, the system becomes inefficient. Our approach does not distinguish particular roles and is designed to adapt its behaviour to the users needs. It does not suffer from potential user selfish behaviour that may block the power management.

B. Adaptability principles

Our approach is therefore complementary to the energy management performed by the hardware and the operating system. In order to adapt the middleware behaviour to the available energy, each module of the middleware is designed to be adaptable. When the energy level is high, the middleware provides all functionalities with the best quality of service. When the energy level decreases, the functionalities are degraded step-by-step (several intermediate energy levels are specified) in order to preserve the battery. The adaptations result in a compromise between extended battery lifetime and quality of service.

The adaptation actions may consist in the adjustment of a parameter or the use of an alternative algorithm. Examples of possible adaptations in Transhumance are:

- the use of alternative ciphering algorithm, less secured but lighter,
- The use of a non-acknowledged transport protocol (resulting in possible loss of data but reduced network activity),

Let us consider the transport protocol as an example. When the energy level is high, the transport protocol may be secured and acknowledged. When the energy level decreases (e.g. less than 75% of the full battery), a first adaptation consists in providing a less secured ciphering algorithm at a lower cost. A second adaptation (e.g. energy level lower than 50% of the full battery) consists in providing a non-secured communication. When the energy is lower (e.g. less than 25% of the full battery), the messages are not acknowledged anymore. The adaptations progressively reduce the middleware energy consumption by limiting the CPU activity and by reducing the amount of communication.

C. Energy management operation

Figure 2 describes the architecture of the energy management mechanisms integrated in every mobile node supporting Transhumance. The two main elements are the *monitoring* module and the *resource manager*.

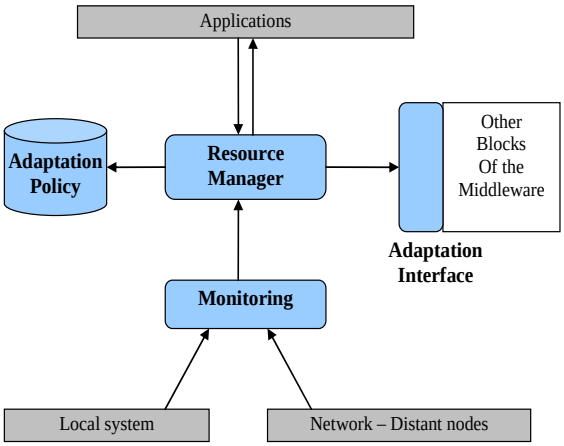

Fig. 2 – Node energy management architecture

The resource manager of each node uses an adaptation policy to decide which adaptations must be taken. The decisions are guided by information gathered from all participating nodes. The local battery level is regularly monitored to evaluate the local energy consumption. This information is sent to the local resource manager and to remote nodes. The different resource managers gather energy levels received from other nodes in order to compute the average energy level in the network that we term “global energy level”. Consequently, the different nodes in the network share the same information. Since the network size is small, the computation of the global energy level does not induce an important latency. In case a node does not receive one or more of the messages, the event manager sends them again. The average energy level in the network is used by the resource manager of the different nodes in order to determine if global adaptations have to be performed.

When the decision to adapt the middleware behaviour is taken, the resource manager sends adaptation orders to all concerned modules via the *adaptation interface* implemented by every module. The middleware modules then modify their behaviour according to the resource manager’s request. The resource manager also communicates with the applications in order to warn them about changes in the middleware behaviour. We see this issue in more details in the section devoted to the interaction with the applications.

The adaptations may consist in the adjustment of a parameter or the use of an alternative algorithm. For example, as mentioned before, in the security module, an adaptation may consist in using an alternative ciphering algorithm, less secured but lighter. In the transport module an adaptation would be to use a non-acknowledged protocol (possible loss of data but reduced network activity).

D. Adaptation Policy

The policy defines the middleware behaviour for different

thresholds of energy level. Each threshold maps to a set of adaptations to modify the middleware behaviour. The policy is used by the resource manager to send adaptation orders to the middleware modules when the energy thresholds are reached.

We distinguish between two types of adaptations: *local* and *global*. Local adaptations only impact the local system: for example, the decrease in the frequency of system monitoring. Local adaptations are decided separately by each node, based on their local energy level. Since there is no interaction between the local adaptations of the different nodes, each node is responsible for its local policy. Global adaptations have an impact on the whole network and must therefore be shared by all the nodes (in the communication range). In order to do so, the global policy is common to all the nodes in the network. In a same way, all the nodes must agree on the decision to take a global adaptation. The decision not to encrypt messages (or no longer acknowledge them) is an example of global adaptation that must be taken by all nodes. If it was not the case, communication could be disrupted, some nodes continuing to encrypt while the corresponding receivers do not decrypt anymore.

IF (100>LOCAL-ENERGY>75) THEN
- No_adaptation

IF (75> LOCAL-ENERGY >45) THEN
- Neighborhood :: maximum_discovery_range = 4 hops
- Advertisement :: maximum_response_per_query = 25

IF (45> LOCAL-ENERGY >20) THEN
- Advertisement :: maximum_response_per_query = 10
- Groups :: create_group_allowed = FALSE

IF (20> LOCAL-ENERGY >0) THEN
- Monitoring :: monitoring_frequency = 30s

IF (100>GLOBAL-ENERGY>75) THEN
- No_adaptation

IF (75> GLOBAL-ENERGY >45) THEN
- Transport :: ciphering_enabled = FALSE
- Data :: lazy_propagation = TRUE

IF (45> GLOBAL-ENERGY >20) THEN
- Transport :: no_ack = TRUE
- File_Transfer :: max_file_size = 512 kB

IF (20> GLOBAL-ENERGY >0) THEN
- Data :: last_modification_propagation = FALSE

Fig. 3 - Adaptation Policy

Fig. 3 shows an example of policy. The first part of the policy concerns local adaptations (based on the energy level of the node). The second part concerns the global adaptations (based on the average level of energy in the network). We may notice the progressive adaptation of the advertisement module. At first, the module provides full functionality. It is then degraded and the maximum response per query is set to 25 (for a local energy level lower than 75%). Then this parameter is set to 10 when the energy level is under 45%. Similarly, we can notice the progressive degradation of the transport protocol. The encryption is not maintained for an energy level lower than 75% and then the acknowledgments are not used for an energy level lower than 45%.

E. Application adaptation

When the middleware adapts its behaviour, it degrades certain features, and services expected by applications may no longer be available. Applications should adapt their functioning in harmony with the services provided by the middleware. For example, when the middleware no longer provides the acknowledged transport, an application waiting for an acknowledgment is blocked. To avoid this situation, the applications adapt their behaviour to the middleware.

The adaptations of the applications must be coordinated with those of the middleware. The application adaptation operates in two steps. The first takes place at the start of the application. The application provides the middleware with an *adaptation profile* specifying which middleware adaptations are allowed by the application. The second step occurs during the execution. When an energy threshold is reached, the middleware adapts its behaviour and warns the application that modifications were made. The application then knows the features that are no longer available (as specified in its profile), and can adapt its behaviour.

IF (HIGH-ENERGY) THEN
- *No_adaptation*

IF (MEDIUM) THEN
- *File_Transfer :: max_file_size = 2048 kB*
- *Transport :: ciphering_enabled = FALSE*

IF (LOW) THEN
- *File_Transfer :: max_file_size = 512 kB*

IF (VERY-LOW) THEN
- *Transport :: no_ack = TRUE*

Fig. 4-Application profile

Fig 4 represents an adaptation profile of a file sharing application. At the highest energy level, the application does not tolerate any adaptations. The application then agrees to limit the maximum size of shared files, and does not support message encryption. When the energy level is low, the maximum size of shared data is decreased again. At the lowest energy level the messages do not need to be acknowledged anymore.

In a first approach, we consider that the nodes run only one application at a time. As a consequence, there is no risk of conflict between two applications requiring incompatible adaptations. It is still possible to satisfy the adaptation requirements of multiple applications. In such a case, the adaptation policy would consider the less restrictive adaptations in order to satisfy all the applications. However, the adaptation policy will be less efficient.

VI. EXPERIMENTS

In this chapter, we present experiments that we conducted to

test and validate Transhumance middleware. In Section A, we describe an experiment conducted with eight users playing the "Team Exploration" treasure hunting game [2]. Before this "real life" experiment, we had conducted a number of tests on various parts of the system using an emulator developed at TELECOM ParisTech. The game experiment showed us that Transhumance is a well integrated and operational middleware exhibiting a set of functionalities that is of real help in ad-hoc environments.

In Section B, we introduce measurements that were performed to test and validate the energy management in Transhumance.

Fig. 5 - Screenshot of the Team Exploration Game

A. *Functional experiment*

A full version of the middleware including the augmented routing layer, the transport protocol, the event system, the group management module, the identification module, the security management as well as a multi-user chat service was tested during real experiments conducted within the Transhumance project. These experiments consisted in sessions of the "Team Exploration" treasure hunting game. The game involved 2 teams of 4 players each, who played for half an hour. Each player was given a handheld terminal equipped with a wifi card. The Transhumance platform and the game ran on the handheld terminal. One of the sessions was run in the historical Parisian area of "la Butte-aux-Cailles". Figure 5 shows a screen shot of this session. The interface of the game is provided by a map of the area partitioned in 20 rectangles. On the left of this map, 5 pictures are displayed. The top one appears "blurred"; it is the place of final meeting. The 4 pictures below correspond to photos that were taken in the area. Players must find in which area (a rectangle on the map) each of the 4 pictures was taken. There is a limited time to localize the pictures and when a proposal is made (using the event system) it must be approved by the other members of the team through the game interface.

After a delay of 10 minutes, the set of photos changes. The count down stops only if the player has validated an image with the other members of the team. The "blurred" photo is revealed when 4 other photos have been localized. Once a team has all the photos, its members must rush and reunite on the premises. The list of players displayed on the right of the screen shows the connections to other players (and the number of hops to a player). Player "G" is within one hop, players B, C, D within two hops, players A, E, F within 3 hops. This was done using the presence service.

Through the game, the players were able to experience the capacity of the middleware to operate properly in presence of disconnections.

B. *Energy management evaluation*

We also conducted a set of measurements in order to evaluate the gain in term of energy consumption thanks to middleware adaptability. We measured the energy consumption of the middleware activity when running a photo-sharing application. The objective was to evaluate the reduction of energy consumption thanks to the middleware adaptation.

In these experiments we adapted the transport protocol, by suppressing the acknowledgements. In a first test, the messages were not acknowledged by the receiver. In a second test, each message was acknowledged by the receiver (the size of an acknowledgment is 69 Bytes).

Table 1 shows the average current consumed before and after adapting the transport protocol (i.e. with and without acknowledgments). The current intensity is measured thanks to an API available on Windows Mobile. Our results represent the average current consumed on a period of 20 minutes of the same activity.

TABLE I. EXPERIMENT RESULTS

Transport	**Average Current consumed**
Acknowledged	730 mA
Not acknowledged	580 mA

We notice an important difference in power consumption between the acknowledged and the non-acknowledged versions. Despite the fact that acknowledgments only represent 6% of the traffic, the reduction of energy consumption is of approximately 20% when acknowledgments are suppressed [24]. Our analysis shows that data sending is an important source of energy consumption. These results are in line with the results relative to the power consumption of a wireless card, studied in [25]. We also studied the cost of ciphering and we measured a reduction of energy consumption of approximately 20% when stopping this service.

The energy consumption due to adaptation mechanisms is really negligible compared to the gain brought by adaptations. The measurements show that the energy consumption variation due to the system monitoring is almost null. The number of monitoring messages sent between the nodes is also negligible compared to the application-dependent traffic. Their size is less than 100 bytes and they are sent only every minute. For a network of 10 nodes, running the monitoring service during 2 hours, only 10*120*0,1 = 120 kB are sent for

energy monitoring of the network. Compared to this, sending a photo (which is frequent in the collaborative game scenario) costs approximately 300 kB and the acknowledgement of every packet costs 20 kB.

VII. Conclusion and Future Work

We presented the Transhumance energy-aware middleware for small scale MANets of up to 20 handheld nodes. Transhumance is an integrated middleware solution including a set of functionalities that perform well in an ad-hoc environment where disconnections happen frequently and there is no node that can be used as a central server. It includes security features and is energy-aware, two aspects that, to the best of our knowledge, are not found in publicly available middleware. Both the middleware[1] and the Team Exploration game are available on SourceForge [8].

The experiments performed showed the usability of the middleware in real conditions. They also showed that the energy management significantly reduces energy consumption.

We are currently pursuing the work on energy awareness and more thoroughly evaluating the approach. The objective is to measure the global decrease of energy consumption in the network. We will also compare various adaptation policies. Finally, we will more precisely measure the additional cost of the energy management in order to conclude on the effective gain of our approach.

We are also pursuing work on data sharing on MANets that should later be integrated in Transhumance [26, 27].

VIII. References

[1] I. Chlamtac, M. Conti, J. Liu: “Mobile ad hoc networking: imperatives and challenges”. Ad Hoc Networks, Elsevier, Vol. 1, Issue 1, p. 13-64 (2003).

[2] I. Demeure, A. Gentès, J. Stuyck, A. Guyot-Mbodji, L. Martin: “Transhumance: a Platform on a Mobile Ad hoc NETwork Challenging Collaborative Gaming”. 1st International Workshop on Collaborative Games (CoGames 2008), May 19-23, 2008. Irvine, California, USA

[3] R-N.Mayo, P. Ranganathan: “Energy consumption in mobile devices: why future systems need requirements-aware energy scale-down”, HP Laboratories Palo Alto, Technical Report, 2003.

[4] G. Paroux, I. Demeure, D. Baruch: “A survey of middleware for mobile ad hoc networks”, Technical Report 2007D004, Ecole Nationale Supérieure des Télécommunications, France, January 2007.

[5] 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.

[6] 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.

[7] R. Meier, V. Cahill: “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] Transhumance project on SourceForge http://sourceforge.net/projects/transhumance.

[9] T. Clausen, P. Jacquet: “Optimized Link State Routing Protocol (OLSR)" Technical Report RFC 3626, IETF, 2003.

[10] UniK OLSR implementation: http://www.olsr.org

[11] J. Hernando-Ureta: “Design of a transport service for Transhumance”, University Rovira i Virgili (Tarragona), Security and Computer Engineering Master’s Degree thesis, July 2008.

[12] G. Anastasi, E. Ancillotti, M. Conti, A. Passarella: “Experimental Analysis of a Transport Protocol for Ad hoc Networks (TPA)”, in Proc. of ACM international workshop on Performance evaluation of wireless ad hoc, sensor and ubiquitous networks, Terromolinos, Spain, 2006, pp. 9-16.

[13] M. Musolesi, C. Mascolo, C. Hailes: “EMMA: Epidemic Messaging Middleware for Ad hoc networks”. Jour. of Personal & Ubiquitous Computing, Springer, Vol 10, No 1, Feb., 2006, p. 28-36.

[14] M. Hapner, R. Burridge, R. Sharma, J. Fiall, K. Stout: “ Java Message Service,” Sun Microsystems Inc., Santa Clara, CA 2002.

[15] OMG. CORBA Notification Service Specification. Needham, MA. Aug. 2002.

[16] A. R. Khakpour, I. Demeure: “ Designing and Prototyping an Event-based Communication System on Mobile Ad Hoc Network”, Technical Report 2008D009, Ecole Nationale Supérieure des Télécommunications, France, July 2008.

[17] M. Boulkenafed, V. 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

[18] 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.

[19] 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

[20] G. Paroux, L. Martin, J. Nowalczyk, I. Demeure. “Transhumance: A power sensitive middleware for data sharing on mobile ad hoc networks”. ASWN 2007 - Seventh international Workshop on Applications and Services in Wireless Networks. Santander, Spain, May 2007.

[21] 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.

[22] M. Tarique, K.E. Tepe, M. Naserian M.: “Energy Saving Dynamic Source Routing for Ad Hoc Wireless Networks”, Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05), 2005.

[23] J-E. Garcia, A. Kallel, K. Kyamakya, K. Jobmann, J-C. Cano, P. Manzoni: “A novel DSR-based energy-efficient routing algorithm for mobile ad-hoc networks”, IEEE Vehicular Technology Conference Fall, 2003.

[24] G. Paroux, I. Demeure, L. Reynaud : “Un Intergiciel Adaptable à l'Energie”, 8ème Conférence Internationale sur les NOuvelles TEchnologies de la REpartition (NOTERE’08), Lyon, France, June 2008.

[25] L.M. Feeney, M. Nilsson: “Investigating the energy consumption of a wireless network interface in an ad hoc networking environment”, Infocom 2001, pp. 1548-1557.

[26] J. Botia, H. Ha Duong, I. Demeure, A. Gómez-Skarmeta. “A Context-aware Data Sharing Service over MANet to Enable Spontaneous Collaboration”, 6th International Workshop on Distributed and Mobile Collaboration (DMC 2008). WETICE", Rome, Italy, June 2008.

[27] H. Ha Duong, I. Demeure. “Partage de données sur réseau mobile ad hoc”, CDUR 2008, Lyon, France, June 2008.

[28] Schiele, G., Handte, M., and Becker, C. 2008. Experiences in Designing an Energy-Aware Middleware for Pervasive Computing. In Proceedings of the 2008 Sixth Annual IEEE international Conference on Pervasive Computing and Communications - Volume 00 (March 17 - 21, 2008). IEEE Computer Society, Washington, DC, 504-508.

[1] The version available on SourceForge does include the monitoring and adaptation mechanisms for energy awareness but does not yet include the latest developments on energy-management.