A survey of middleware for mobile ad hoc networks

Un état de l'art des intergiciels pour les réseaux mobiles ad hoc

---

Guilhem Paroux
Isabelle Demeure
Deborah Baruch

2007D004

2007

Département Informatique et Réseaux
Groupe Systèmes, Logiciels, Services

Ecole Nationale Supérieure des Télécommunications

A survey of middleware for mobile ad hoc networks

Un état de l'art des intergiciels pour les réseaux mobiles ad hoc

Authors:

Guilhem Paroux (ENST – France Télécom R&D)
Isabelle Demeure (ENST)
Deborah Baruch (France Télécom R&D)

RESUME
Ce document présente un état de l'art des intergiciels (middleware) pour les réseaux mobile ad hoc. Son objectif est de répertorier les principales solutions existantes afin d'identifier les fonctionnalités usuelles des intergiciels pour réseaux mobiles ad hoc, mais aussi pour en répertorier les manques.
L'étude couvre les principaux intergiciels identifiés en s'attachant à les étudier suivant un même plan, établit après l'étude de JXTA, un framework de référence en environnement fixe.
L'étude montre qu'il existe un groupe de fonctionnalités communes aux différents intergiciels, même si leurs réalisations varient d'un intergiciel à l'autre. Ces fonctionnalités de communication et d'organisation du réseau constituent le noyau dur de tout intergiciel. L'étude montre également que les fonctionnalités de gestion d'énergie et de sécurité sont manquantes pour l'ensemble des intergiciels.
Suite à l'étude de chaque intergiciel, nous dressons une rapide comparaison des solutions apportées pour gérer différentes caractéristiques des réseaux mobiles ad hoc, avant de conclure.

ABSTRACT
This document surveys middleware for MANETs. Its goal is to study the existing solutions in order to identify the functionalities that are usually found in middleware for MANETs and the ones that are obviously missing.
All identified middleware are studied following the same outline derived from a study of JXTA, a reference framework for peer-to-peer systems in fixed environments.
The study concludes that all middleware for MANETs implement a set of functions such as group management and communication facilities. However power management facilities and security that are two key issues in MANETs are not addressed.
The document concludes on a comparative study of the identifies middleware for MANETs.

1 Introduction

Mobile Ad hoc NETworks (MANETs) are infrastructure-less networks composed of mobile devices with limited resources [Chlamtac, 2004]. In a MANET, the devices must therefore organize themselves in order to create the network. They cannot rely on a pre-configured infrastructure. This is both an advantage and a drawback. The main advantage is the possibility to deploy the network everywhere. Moreover, the network can be created spontaneously when devices are able to communicate together. However, since the network is organized "on the fly" by the participating devices, the organization is slow and can change during the network life. The applications have to take into account the dynamic changes in the network topology and the limited resources of participating devices. The most studied issue in MANETs is routing (see, for example, [Abolhasan, 2004] for a survey of routing protocols in MANETs). In this paper, we do not focus on routing but rather on higher layers, and more specifically on the middleware.

The middleware is a software layer that typically stands between the operating system and the network on the one hand and the distributed applications on the other hand. The middleware aims to provide applications designers with an abstraction of the complexity introduced by distribution. In fixed environment, middleware technologies are well known and used successfully. They provide functionalities such as object or component distribution, communication, and resource discovery. Examples of middleware are OpenCORBA [Ledoux, 1999] and Globe [Van Steen, 1999]. Another example is JXTA, an open-sourced middleware initiated by Sun Microsystems to support peer-to-peer (P2P) applications [Jxta].

In this paper, we survey mobile ad hoc network middleware. All middleware are studied following the same plan. Although it is not designed for MANETs, in Section 2, we first present JXTA, as a reference middleware for peer-to-peer applications in fixed environment. In section 3, we survey middleware for MANETS. We start with JXME, an adaptation of JXTA for mobile devices [Jxme]. We then review various middleware for mobile ad hoc networks: Selma, Proem, Steam, JMobiPeer and Expeerience, Infoware, MeshMdl, Emma, MPP, MIN. In Section 4, we propose a comparison of the previously introduced middleware for MANETs; we put forward a list of desired functionalities that have not been addressed or have been insufficiently addressed by existing middleware. Section 5 concludes this survey paper.

2 JXTA

In this section, we present JXTA, a middleware for Peer-to-peer applications in fixed environments [Jxta], [Jxta, 2003], [Gong, 2001a], [Gong, 2001b]. Although JXTA was not designed for MANETs, we choose to study it as a reference system for decentralized peer-to-peer applications.

2.1 Context

JXTA is a Peer-to-peer (P2P) middleware initiated by Sun Microsystems in 2001 and now supported by a large community of developers. New versions are regularly produced.
As of today, there are more than 90 ongoing projects in the JXTA community. Projects aim to improve JXTA portability or to provide new additional services.

2.2 Objectives

JXTA is designed after three objectives: interoperability, platform independence and ubiquity. A more detailed description can be found on the official website. [Jxta]

2.2.1 Interoperability

Traditionally, a peer-to-peer application provides only one type of service and uses specific protocols. This is the case for Gnutella [Clip2, 2001] and Skype [Skype] which define their private protocols. Gnutella is designed to support p2P file exchange whilst Skype supports Internet telephony. Users from the Gnutella and the Skype communities cannot collaborate in order to increase the connectivity and the efficiency of the two networks. JXTA aims to provide a common layer to all P2P applications in order to allow them to share resources and to improve the reliability of the network.

2.2.2 Platform independence

Heterogeneity is an important issue in large decentralized systems. JXTA aims to provide platform-independent protocols to avoid this kind of problem. Any application complying with the JXTA protocols specification can be implemented on a variety of hardware and operating systems.

2.2.3 Ubiquity

JXTA is said to be working on "every device with a digital heartbeat". The performance heterogeneity implies that JXTA protocols have to be designed in order to work on the smallest devices without overloading them.

2.3 Architecture

JXTA is organized in three layers. JXTA core regroups the basic functionalities offered by JXTA in order to develop P2P applications. The other layers are not necessary, but often useful. JXTA creates a common layering structure at the conceptual level as depicted in Figure 1.

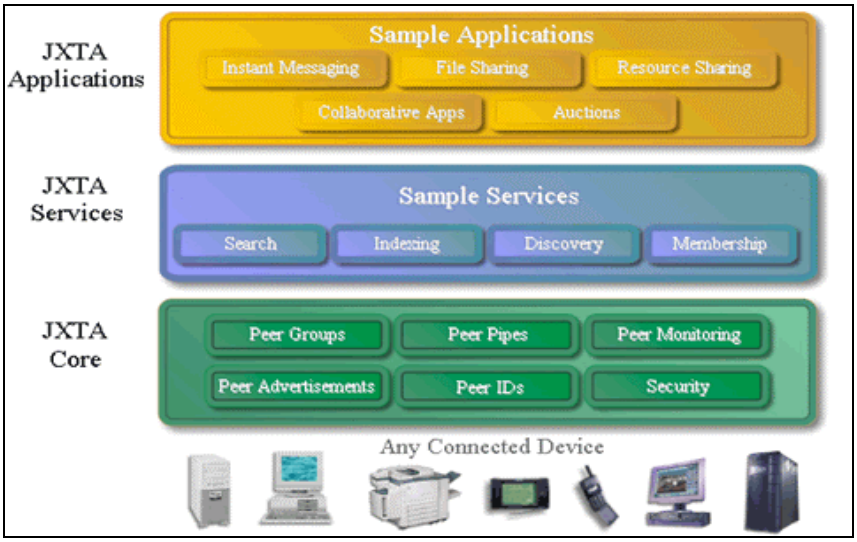

Figure 1: JXTA architecture (extracted from SUN's website)

2.3.1 Core layer

The core layer regroups the basics of P2P applications and allows developing higher level applications. Communication mechanisms, security management and resource identification are examples of functionalities available in the core layer.

2.3.2 Services Layer

Built on the JXTA core protocols' foundation, peer services are the building blocks of full-blown P2P applications. The services are not necessary in order to build P2P applications but they can improve application performances.

2.3.3 Applications Layer

Here live P2P applications. The possibilities offered by JXTA are numerous as the variety of projects we can found on the JXTA web pages shows it. These projects are independent and are not necessary in JXTA.

2.4 Device management

JXTA is designed to work on workstations which do not have particular constraints for resources. Energy is not limited, memory quantity is sufficient for many applications and CPU power is important.

2.5 Communication

2.5.1 Communication model

At the physical level, JXTA uses classical protocols to allow peers to communicate. On local area network, JXTA uses UDP and TCP depending on the type of the message that has to be sent. In larger networks, JXTA uses HTTP in order to allow communication through firewalls and NATs.

JXTA offers mechanisms for communication between peers, namely pipes. Pipes are virtual communication channels used to send and receive messages between services and applications. They provide an abstraction over the peer endpoints and can connect two or more peer endpoints. They offer two modes of communication:

- *Point-to-point pipes* connect exactly two pipe ends with a unidirectional and asynchronous channel.
- *Propagate pipes* connect one output pipe to multiple input pipes. Messages are sent to all listening input pipe ends in the current peer group context.

Basically, communication is performed thanks to advertisements. When a peer creates a new service, it publishes an advertisement for it. The advertisement is sent to the rendezvous peer and to every peer directly connected to the sender. The connections are done thanks to pipes.

2.5.2 Resource identification

JXTA uses UUID, a 128-bit datum, to refer uniquely to a resource, such as a peer, an advertisement or a service. The identifiers are expected to be "statistically" unique, i.e. the probability to generate randomly two similar identifiers is almost null.

2.5.3 Shared resource management

All network resources are represented by advertisements. Basically, advertisements are metadata structures resource descriptors represented as XML documents. The use of XML and the advertisement standardization ensure JXTA language and platform independence.
Project JXTA has defined the six following protocols over transport protocols in order to improve the efficiency of the peer network organization.

- The Peer Discovery Protocol enables a peer to find advertisements on other peers.
- The Peer Resolver Protocol provides P2P applications with a generic request and response format to use when communicating with other peers.
- The Peer Information Protocol allows a peer to learn about other peers capabilities and status.
- The Rendezvous Protocol allows peers to connect to the rendezvous.
- The Pipe Binding Protocol allows a peer to bind a pipe advertisement to a pipe endpoint.
- The Endpoint Routing Protocol allows a peer to ask a peer router for available routes for sending a message to a destination peer.

2.5.4 Resource discovery

In JXTA v2.0, a Shared Resource Distributed Index (SRDI) has been implemented in order to improve search efficiency. The SRDI works as follows. When an edge-peer sends a query, the rendezvous peers to which it is connected search in its local cache to determine if its indices contain the requested information. If not, a default limited-range walker algorithm is used to walk the set of rendezvous looking for a rendezvous which contains the index. When a rendezvous contains the index, it notifies the edge-peer which publishes and owns the requested information and that peer will answer directly to the requesting peer.

2.6 Logical network organization

2.6.1 Degree of distribution

Basically, there is no hierarchy between peers in the network. However, for technical reasons like NATs (Network Address Translation) or firewalls, it is possible to organize the peer network around nodes providing specific services. Three types of peers are distinguished:

- *Edge peers* constitute the basic type of peer. They can perform everything in the network but have no specific function for network management.
- *Relay peers* are used in order to traverse firewalls and NATs. They maintain information about other peers and route the messages. They can also be used as buffers.

Rendezvous peers are used to forward the discovery requests to other peers. They are points of meeting for other peers. They constitute a sub-network which aims to increase the speed of discovery in the network.

2.6.2 Peer groups

A peer group is a collection of peers that have agreed upon a common set of services. Peers self-organize into peer groups, each identified by a unique peer group ID. Each peer group can establish its own membership policy from open to highly secure and protected. Peers may belong to more than one peer group simultaneously. Peer groups are very useful to constitute secured domains in the network or to share information about a common interest.

2.6.3 Localization

JXTA does not provide mechanisms for localization. Moreover, the localization in fixed environment does not present the same interest than in wireless networks.

2.6.4 Proximity/Neighborhood

As JXTA works in fixed environment, each peer is aware of its physical neighborhood composed of other peers on the same local network. When two peers on the same network want to communicate, they use TCP protocol. When they are distant, they use HTTP. However, this information is not more exploited by JXTA which considers that every peer has the same position in the network.

2.7 Security

JXTA does not implement a complete security solution but what is called a *trust model*. The JXTA trust model allows peers to be their own certification authorities. JXTA has implemented a virtual transport based on TLS (Transport Layer Security) to provide secure communications between peers. When a JXTA secure pipe is created, a virtual TLS transport is instantiated. All data moved through secure pipes is then multiplexed over this single instance of a virtual TLS transport. The transport is bi-directionally secured end-to-end with TLS, independently of JXTA relays and the underlying physical transport.

2.8 Additional functionalities

As we have seen, JXTA includes in its architecture a service layer [Services]. This layer aims to provide additional services to the core of the middleware. For instance of available services, we can notice JXTA-RMI which aims to provide to JXTA a set of Remote Method Invocation (RMI) functionalities to JXTA. Another example is JxtaSpaces which provides JXTA with a distributed shared memory service.

2.9 Discussion

JXTA seems to be very complete by providing numerous functionalities for peer-to-peer application designers. The functionalities are numerous and allow the developers to deal with different aspect of peer-to-peer application design in fixed environment. The different projects of the community let us hope that JXTA will be improved regularly in the future.
However, JXTA present two main drawbacks. First, it is not well documented. Even if it is not hard to understand its global working, several functionalities need to be better explained. Second, the scalability is not ensured. There is no real experiment to test the scalability. The experiments that we conducted are not encouraging.

3 Survey of middleware for Mobile Ad hoc networks

3.1 JXME

3.1.1 Main bibliographical references

[Jxme], [Jxme, 2002], [J2ME]

3.1.2 Context

Since JXTA aims to work on "every device with a digital heartbeat", it must be compatible with small devices. JXME (JXTA for J2ME) is a specific project of the JXTA community meant to provide JXTA on small constrained devices.
As of today, there is only one version of JXME developed for use with Java as programming language. JXME is regularly updated; the last release was announced in December 2005.

3.1.3 Objectives

JXME follows JXTA's objectives of interoperability, platform independence, and ubiquity. However, JXME has additional objectives to reach:

- It must be compliant with devices with limited resources such as cellular phones and PDAs.
- It must be compatible with JXTA in order to allow mobile devices to communicate with devices in fixed environment.
- It must be compliant with the J2ME programming environment.

JXME is proposed in two versions: a "proxy" version and a "proxyless" version. In the "proxy" version, mobile peers must be connected to a proxy peer to communicate. A proxy peer is a wired environment peer that implements specific services. In the

"proxyless" version, mobile peers can communicate like JXTA peers in wired environment, using adapted mechanisms.

3.1.4 Architecture

In the proxied version, mobile peers only implement several core functionalities. The major part of services necessary to participate to a JXTA network is available from the proxy peers.
In the proxyless version, mobile peers implement more functionality similar to JXTA. The architecture of the middleware does not include the service layer. Since these services are not necessary, they are not included in the middleware.

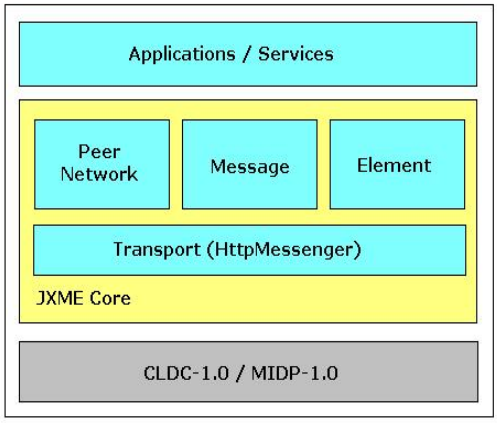

Figure 2: Proxied JXME architecture

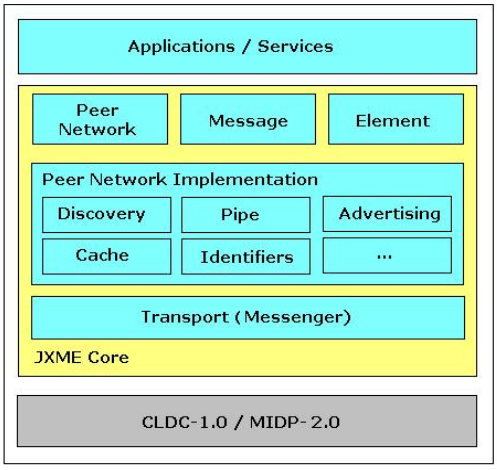

Figure 3: Proxyless JXME architecture

3.1.5 Device management

JXME functionalities are adapted to mobile devices. Full services available on JXTA peers are not implemented on mobile peers in order to reduce memory and energy consumption. Functionalities available on mobile devices are light versions of functionalities offered by JXTA.

3.1.6 Communication

3.1.6.1 Communication model

In the proxied version, JXME mobile peers use HTTP to communicate with proxy peers in transmission range. Proxies work as relays to propagate the messages in the fixed network. Proxies can be seen as points of centralization in the network, which makes this version not suitable for MANETs.
In order to decrease resources' consumption, JXME manipulates binary messages in place of the XML messages supported by JXTA. Proxies translate the message format between wired and wireless environment.

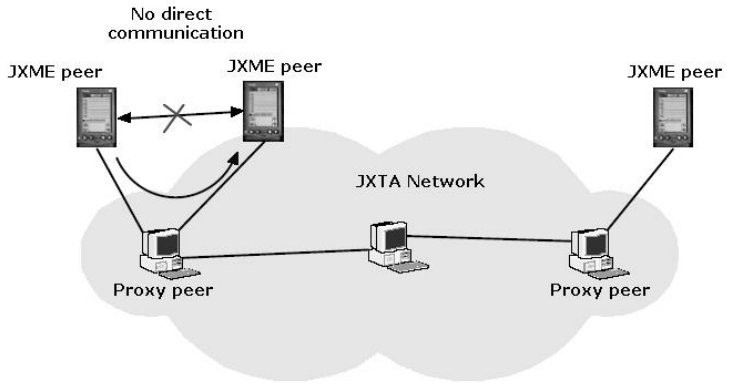

Figure 4: JXME proxied network (extracted from JXME website)

In the "proxyless" version, mobile peers implement more functionalities and can communicate directly. TCP communication is supported. Advertisements are formatted in XML and mobile peers can create pipes.

3.1.6.2 Resource identification

Resources are identified by unique identifiers and are represented by advertisements like JXTA.

3.1.6.3 Shared resource management

JXME manages shared resources in the same way as JXTA. However, the protocols are adapted to limited capabilities environments. In the proxied version, mobile peers rely on a proxy peer to participate to a JXTA network. In the proxyless version, mobile peers implement light versions of JXTA protocols.

3.1.6.4 Resource discovery

Resource discovery is performed in the same way as in a JXTA network. Mobile peers can publish advertisements and initiate discoveries.
In the proxied JXME, the proxy peer plays a central role in resource discovery by translating and transmitting the queries.
In the proxyless version, mobile peers can only act as edge peers (and cannot be rendezvous peers). They support a light version of the Shared Resource Distributed Index (SRDI).

3.1.7 Logical network organization

3.1.7.1 Degree of distribution

The main limitation with respect to JXTA is that in JXME mobile peers cannot be rendezvous or relay peers.

3.1.7.2 Peer groups

Groups are supported by JXME. In the proxied version, group mechanisms are performed by proxy peers. In the proxyless version, group mechanisms are managed by mobile peers. The notion of group is the same as in JXTA. Every peer can create a group and apply the membership policy that it wishes.

3.1.7.3 Localization

JXME does not provide mechanisms to localize mobile peers.

3.1.7.4 Proximity/Neighborhood

JXME does not provide features for proximity or neighborhood management.

3.1.8 Security

In the proxied version of JXME, security is performed by proxy peers in order to keep only light functionalities on mobile peers.
In the proxyless version JXME does not provide security features in order to be light enough to work on constrained devices.

3.1.9 Additional functionalities

In the proxied version, additional functionalities are located on proxy peers. It is exactly the same approach as in JXTA: the services are situated in the service layer and are under development.
In the proxyless version, mobile peers do not offer additional functionalities, but as long as they can communicate with peers on fixed environment, they can use services available on them.

3.1.10 Discussion

JXME is a significant attempt at designing middleware for mobile devices. However, in view of the available functionalities, JXME is not suitable for MANETs in its proxied version.
In the proxyless version, the lack of documentation and the impossibility to test it for the moment does not allow us to conclude. However, the network architecture seems to be more suitable for MANETs.

3.2 Selma

3.2.1 Main bibliographical references

[Görgen, 2004], [SOUL]

3.2.2 Context of the project

SELMA stands for Self-organized Marketplace-based Middleware for Mobile Ad-hoc Networks. It was developed by the members of the SOUL project at the University of Trier, Germany in 2003-2004. SELMA is an open-source project, implemented in Java.

3.2.3 Objectives

SELMA is a middleware designed for MANETs. It provides functionalities such as positioning, neighborhood discovery, wireless communication, routing protocols.

SELMA services are based on a communication pattern resembling traditional marketplaces. Marketplaces are geographical regions in the network where peer density is the most important. SELMA aims to locate the major part of peers' activity in marketplaces. To this purpose, SELMA uses mobile agents in order to help peers reach distant marketplaces.

3.2.4 Architecture

The middleware architecture is divided in three layers. The communication abstraction is the lowest layer and provides generic methods for positioning, wireless communication and device discovery. The agent platform layer represents the major part of the middleware functionalities. It includes routing protocols, localization and marketplaces management. Finally, the highest layer regroups the specification of two types of agent: application agents and service agents.

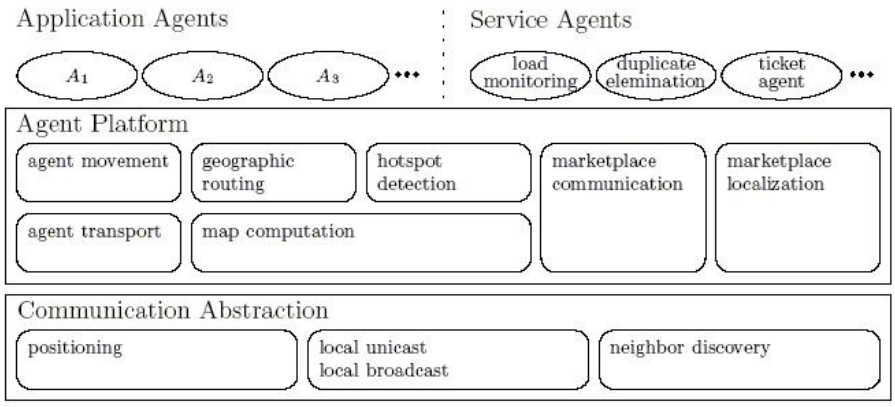

Figure 4: Architecture of SELMA [Görgen, 2004]

3.2.5 Device management

One of SELMA's goals is to save energy of mobile devices. The authors argue that the marketplace pattern provides for reducing communication overhead. As a consequence, the energy is saved thanks to the limitation of the number of messages on the network. Positioning and geographic routing also contribute to the middleware efficiency with respect to energy consumption. Nevertheless, the use of a positioning system on mobile device is an additional source of energy consumption.

3.2.6 Communication

3.2.6.1 Communication model

SELMA provides two kinds of communications: message broadcasting and mobile agent communication. Message broadcasting is used in order to discover the

neighborhood. The authors do not give details about the messages. However, they say that two basic wireless primitives are used: local unicast and local broadcast. At a higher level, mobile agents, coupled to marketplaces, constitute the basic way for applications to communicate. In order to communicate, the mobile agents use broadcast and unicast. Marketplaces are locations in the network where application activity is concentrated. They are created thanks to mobile agents which monitor the network and determine the good location to deploy a marketplace. The coordinates are then widespread in the network thanks to mobile agents. In marketplaces, mobile agents use limited range broadcast or unicast addressing to communicate.
Mobile agents support logical mobility in SELMA. They transport information and represent their host in remote places. In order to reach marketplaces, mobile agents use geographic routing to avoid useless movements. Mobile agents also have a dedicated transport protocol to manage their movements.

3.2.6.2 Resource identification

Resources are not explicitly addressed in SELMA. There is no information on resource identification or agent identification. An identifier is assigned to every device but no details are given on how to generate the identifiers.

3.2.6.3 Shared resource management

The shared resources and services are not clearly described in Selma. Mobile agents seem to constitute the major part of the resources shared in the network. They provide the peers with the ability to move shared resources to marketplaces and to be represented into them. The agents constitute a shared resource because they can be hosted by every peer in the network in order to perform their movements. We can also notice that service agents are a resource shared by peers in the network. They are used in order to manage marketplaces and contribute to the good work of the network.

3.2.6.4 Resource discovery

As we have seen, the most important resources are mobile agents and marketplaces location. These two kinds of resources are advertised in the network thanks to the positioning service. The positioning service allows to locate resources in the network and to send mobile agents. The documentation on SELMA gives no specific details on resource discovery.

3.2.7 Logical network organization

3.2.7.1 Degree of distribution

In SELMA all peers are equal. This is probably due to the fact that the most important entity in SELMA is not the peer but the mobile agent. The network is completely decentralized.

3.2.7.2 Peer groups

SELMA does not explicitly use group mechanisms. However, peers belonging to a marketplace can be seen as a group of peers. They have common interest and they host mobile agents participating to this marketplace.

3.2.7.3 Localization

In order to allow peers to locate marketplaces and to send their mobile agents, SELMA provides a localization service. Localization uses coordinates provided by the positioning system (eg. GPS). SELMA computes a map of the network with Cartesian coordinates. When a mobile device cannot use GPS, it calculates its current position thanks to neighborhood coordinates.
Marketplace positions and applications available in these marketplaces are disseminated over the network. Redundant information is avoided thanks to a specific hash function which allows identifying uniquely a same resource available many times in the network. There is no information about the hashing algorithm.

3.2.7.4 Proximity/Neighborhood

SELMA provides a neighbor discovery protocol. If the device is equipped with a technology such as Bluetooth providing a device discovery protocol, SELMA uses it. Otherwise, a periodic broadcast is sent in order to discover devices.

3.2.8 Security

As mentioned by the authors, there is no provision for security in SELMA.

3.2.9 Additional functionalities

SELMA is essentially based on the mobile agent paradigm. Additional services can be found in service agents. The most interesting service is a load balancing mechanism. When a marketplace is no longer used, service agents can decide to close it. Conversely, when a marketplace is overloaded, service agents can divide it into two or more marketplaces.

3.2.10 Discussion

The article does not give any detail on basic functionalities such as resource identification and resource management. Everything seems to be done thanks to mobile agents. SELMA seems to provide a good framework for the applications.

3.3 Proem

3.3.1 Main bibliographical references

[Kortuem, 2001a], [Kortuem, 2001b], [Proem]

3.3.2 Context of the project

Proem is a project developed in the early 2000s by a team of the Wearable Computing Laboratory at the University of Oregon, USA.
It is developed in Java. The project is over and no information on the support is given.

3.3.3 Objectives

Proem aims to provide a complete solution to develop and deploy collaborative peer-to-peer applications on MANETs. The main objective is to provide functionalities commonly used by mobile peer-to-peer applications. The authors give the following list of objectives:

- Adaptability to the operating environment.
- Universality of applications supported by Proem.
- Interoperability between heterogeneous systems.
- Platform independence.
- Extensibility of components.
- High-level development support for application developers.

In order to reach these objectives, Proem provides applications with a complete execution environment. Proem defines the notion of peerlets which are simple structured applications that follow an event-based programming model. The services and protocols available in Proem are used by peerlets during their execution. An application wanting to run on Proem must therefore use peerlets.

3.3.4 Architecture

As depicted in Figure 2 the architecture of Proem is divided into four parts. The lower layer is the protocol stack which contains the four Proem protocols. Over this layer, we find two components. First, the peerlet engine which controls the execution of peerlets. Second, a set of services which provide peerlets with commonly used functionalities. The last part is constituted by the service API.

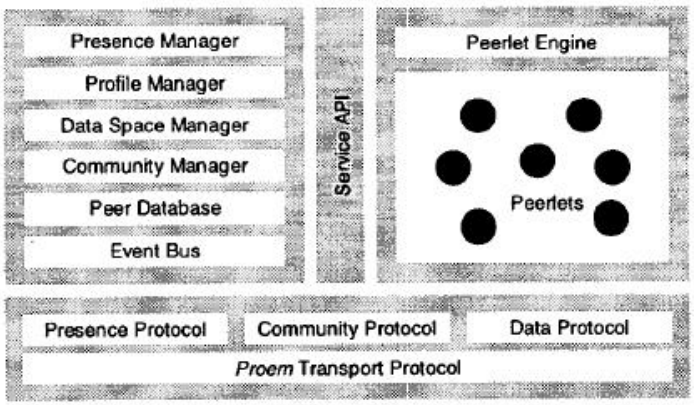

Figure 2: Proem architecture [Kortuem, 2001b]

3.3.5 Device resources management

The authors identify mobile device limitations as a characteristic of MANETs. However, Proem does not provide functionalities to reduce energy consumption or to manage memory. Protocols do not seem to take into account the limited capabilities in their behavior.

3.3.6 Communication

3.3.6.1 Communication model

Proem uses an event-based communication model. It defines four communication protocols, one basic transport protocol and three higher-level protocols.
The transport protocol is connectionless and asynchronous and can be implemented on top of various protocols like HTTP, TCP or UDP. Messages are the basic unit of communication between peers. They are XML-formatted in order to guaranty platform independence and interoperability.
Other protocols are:

- The presence protocol, which allows peers to announce their presence in the network.
- The data protocol, which allows peers to share and synchronize data.
- The community protocol, which manages messages for community membership.

At a higher level, the event bus service provides a publish-and-subscribe model. Components and peerlets can announce the availability of data item by publishing an event or express their interest in data by subscribing to update events.
The presence manager service also uses the event-based model. When a peer enters or leaves the network, it sends a broadcast message. The presence manager notifies the other peers when receiving the message.

3.3.6.2 Resource identification

Entities (e.g. peers, individuals, data spaces or communities) are identified by names. The names are Uniform Resource Identifier (URI). Each name is unique and refers to only one entity. Proem allows entities to have multiple names in order to guarantee a certain form of anonymity. Proem also defines profiles which are XML-based data structure to describe entities.

3.3.6.3 Shared resource management

Data spaces are collections of resources that are cooperatively owned and managed by a set of peers. The data protocol allows peers to share and synchronize data. Proem also provides the data spaces manager as a service.
Proem provides mechanisms for forming communities of entities as we will see it in the next part.

3.3.6.4 Resource discovery

The presence protocol is the first way to discover resources in the network. Using this protocol, peers announce their presence in the network and can be identified by other ones.
The data spaces are another structure where peers may discover resources. A specific protocol and the corresponding service are available to manage the data contained in the spaces.
There are no other details on resource discovery mechanisms available in the literature on Proem. In particular, the protocols for data space construction or exploration are not presented.

3.3.7 Logical network organization

3.3.7.1 Degree of distribution

In Proem networks, every peer plays the same role. There is no central entity and no special type of peers. It is a pure, or flat, peer-to-peer architecture.

3.3.7.2 Peer groups

Proem introduces the notion of communities. A community is a set of entities, such as peers, individuals, data spaces or other communities. The purpose of communities is to regroup entities sharing a common interest. The authors argue that the notion of community is different from the notion of group developed in other middleware. Community membership is not owned by a particular entity. The membership is conferred upon a membership token, circulating on the network. In order to enter a community, an entity has to produce the token signed by a minimal number of community members.

3.3.7.3 Localization

Proem provides no facilities for localization of resources in the network.

3.3.7.4 Proximity/Neighborhood

Proem does not consider relation of proximity in the network. The protocols work independently from the distance between the peers.

3.3.8 Security

Proem does not address the security issue, even if the authors recognize that it is an important issue in wireless networks.

3.3.9 Additional functionalities

Proem offers a peerlet development kit to developers. It allows them to easily develop applications by providing a framework and a set of APIs in order to make applications compatible with the peerlet execution environment.

3.3.10 Discussion

Proem provides mobile applications with a very complete set of functionalities for shared resource management. The identifiers, names and profiles are good mechanisms to clearly identify the resources.
With the peerlet framework, the peerlet mechanism is a more integrated approach than JXTA which just provides applications with a set of protocols. However, constrained resources are never taken into account by Proem. It constitutes a major drawback of Proem.

3.4 Steam

3.4.1 Main bibliographical references

[Meier, 2002], [Meier, 2003], [Meier, 2004], [Meier, 2005]

3.4.2 Context of the project

Steam (Scalable Timed Events And Mobility) is a middleware for mobile ad hoc networks developed by a team of the Trinity College Dublin, Ireland, essentially during the years 2002 to 2004. The middleware implementation is not available.

3.4.3 Objectives

Steam is an event-based middleware for mobile ad hoc networks. It aims to be used by collaborative applications including small indoor and large outdoor environments. It was designed for IEEE 802.11b-based wireless local area networks (WLANs). Steam provides support for location awareness, proximity detection and distributed events filtering. It addresses the very high dynamicity of the mobile ad hoc networks and the need for completely distributed software in such networks.

3.4.4 Architecture

Steam was designed in order to exploit the proximity of the devices during the event filtering. This implies the presence of a location service in Steam.
The location service works thanks to a positioning device like GPS and provides the event service with location information.
The event service is the core of the middleware. It manages the event by publishing local events and gathering remote events. The filter engine, which determines the events relevance for the applications, is a part of the event service. The proximity discovery service provides several protocols to allow the discovery of the mobile devices in the host neighborhood.
The group communication service provides protocols to deal with group membership and message delivery in the proximity-based groups.

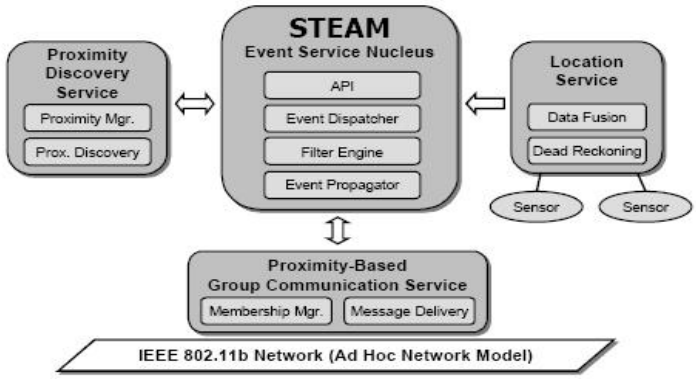

Figure 3: Steam architecture [Meier, 2003]

3.4.5 Device management

Steam does not deal explicitly with Device management. However, thanks to the use of the proximity notion, we can argue that Steam reduces the consumption of energy. Indeed, if the event are located and have a small range of dissemination, the resources spent in forwarding messages are reduced.

3.4.6 Communication

3.4.6.1 Communication model

Steam is an event-based middleware and uses publish-subscribe communication. The peers play two roles: event producers and event consumers. The consumers have to subscribe to event types in order to be notified when an event of the good type is received. Steam provides the consumers with an event filter which allows the application to filter the events regarding their subject, content type, proximity. The producers define their event types.
The proximity group is an important part in the communication model. Communication between mobile devices is mainly performed in a limited proximity group. We will give some details about these groups in a next part.

3.4.6.2 Resource identification

The services are identified thanks to the Proximity Discovery Service (PDS). The PDS uses a hashing algorithm that generates 24 bit identifiers.
The events include fields like subject, content type and attribute list. The fields allow the devices to identify the event types available in the network.
The proximity groups are uniquely identified thanks to their location, the interests of their producers and consumers.

3.4.6.3 Shared resource management

Steam essentially focuses on events. The applications use events to advertise their services and the data are exchanged in the attribute list of the events. There is no detail about the nature of the services or data available in the network. It can be easily understood if we consider that the events are data independent. The structure of the events does not depend on what they carry.

3.4.6.4 Resource discovery

The proximity discovery service and the events are the two ways to discover resources. The proximity discovery service allows the hosts to discover the mobile services available in the associated proximity group.

3.4.7 Logical network organization

3.4.7.1 Degree of distribution

Steam uses a totally distributed architecture. There is no need for an infrastructure and every participant plays the same role in the network.

3.4.7.2 Peer groups

Steam does not provide group functionalities. The notion of proximity groups introduced by Steam is more complete. We present it in next part.

3.4.7.3 Localization

Steam uses positioning systems on mobile devices in order to provide the application components with location information. The location information is computed in the location service. The localization is used in different ways.
The events are location-aware, that is they are linked to a geographical area and are assumed to be more relevant in this area.
The proximity groups, which we introduce in the next part, are geographical groups and use the location service.

3.4.7.4 Proximity/Neighborhood

Steam introduces the notion of proximity groups. The idea is to provide the applications with a local one-to-many communication model. A group is a local set of application components hosted by mobile devices. It is identified both by the functionalities it offers, i.e. the types of events, and its geographical position. To apply for membership, a device must be in the geographical area of the group and must be interested in the group topics.
The Proximity-based Group Communication Service (PGCS) allows the mobile devices to create and join groups by managing the membership. A discovery service allows the peers to find groups, thanks to the proximity group identification. The authors argue it is more interesting to find local groups of interest than to look for a particular device.

3.4.8 Security

Steam does not provide mechanisms for security.

3.4.9 Additional functionalities

No additional functionality is proposed in Steam.

3.4.10 Discussion

The notion of proximity presented in Steam is very interesting. The authors argue that the relevance of the events is higher in a small range around the event producers. The proximity groups allow the peers to receive only local events which are supposed to be more relevant. This is probably due to the fact that the services generating the events are geographically closer and so, easier to use.
The resources of mobile devices are not taken into consideration. The GPS consumes a lot of energy and the resources are limited. What is the life time of a device running Steam?

3.5 JMobiPeer & Expeerience

3.5.1 Main bibliographical references

[Bisignano, 2003], [Bisignano, 2004a], [Bisignano, 2004b], [Bisignano, 2005]

3.5.2 Context of the project

We present the two projects at the same time because JMobiPeer is an improvement of Expeerience. The two projects are leaded by a team of the University of Catania, Italy, during the years 2002 to 2005. The objective is to adapt JXTA to mobile ad hoc networks. The development is done in Java, but the source code is not available.

3.5.3 Objectives

Expeerience and JMobiPeer are very closed in their conception. We distinguish between the two middleware only when necessary since most of the functionalities are common to Expeerience and JMobiPeer.
Both systems aim to adapt JXTA to mobile ad hoc networks, in order to support P2P applications on mobile ad hoc networks. As a consequence, the main objectives are the same than in JXTA: interoperability, ubiquity, platform independence. In order to deal with MANET requirements, the JXTA core is adapted and new services are introduced when necessary. Nevertheless, the compatibility with JXTA networks is ensured.

3.5.4 Architecture

The two projects may look like JXME: a JXTA adaptation for mobile devices (see Section 3.1). However, the most important difference with JXME is that Expeerience has been designed for mobile ad hoc networks, while JXME was only available in its proxied version, only suitable for wireless networks with an infrastructure.Architecture
The middleware are organized in layers like JXTA or JXME. In Expeerience, the core layer contains the transport protocol. It introduces an adaptation of TCP managing intermittent connections. The JXTA transport protocol works over the basic TCP transport. The service layer offers different JXTA protocols, adapted to mobile environments. It also provides a new service managing code mobility.

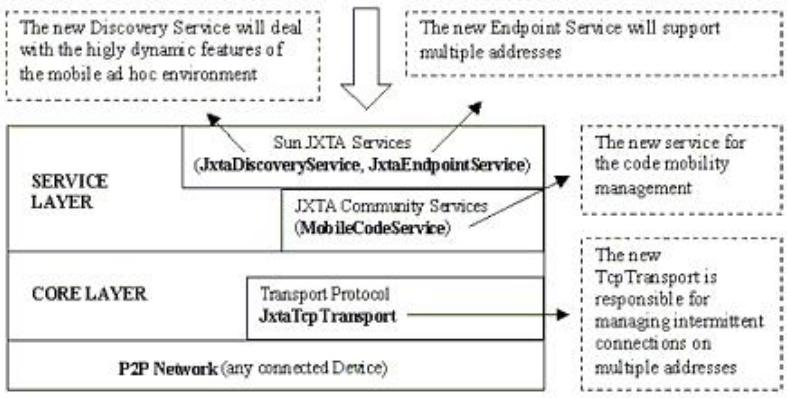

Figure 4: Expeerience architecture [Bisignano, 2004a]

The first difference between Expeerience and JMobiPeer is that in JmobiPeer all layers are J2ME-compliant. The middleware itself is organized into two layers; a third layer represents the applications running over the middleware. The core layer contains a virtual messenger which implements the communication core. The virtual messenger supports the HTTP, TCP and datagram protocols. The endpoint service is the basis of the JXTA protocols and supports an endpoint routing protocol and a propagation protocol. In the core layer, we also find the protocols managing the peer identities, the peer groups and the advertisements. The service layer offers higher level services such as pipes management or resources discovery.

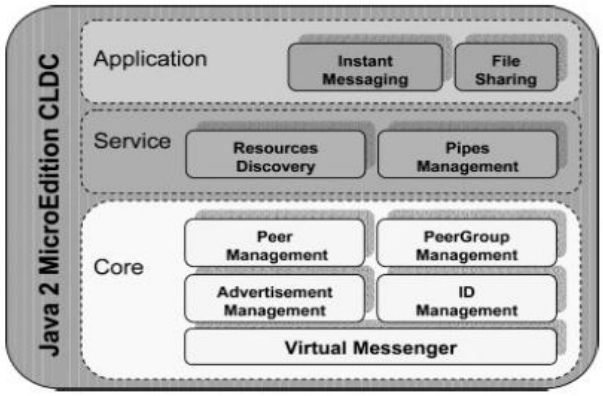

Figure 5: JMobiPeer architecture [Bisignano, 2005]

3.5.5 Device management

Expeerience does not provide solutions for constrained device management. The similarities with JXTA are important and the new services added do not deal with resource consumption except for the advertisement storage as we will seelater.
JMobiPeer is based on J2ME, specifically conceived for mobile devices with low capacities. This implies that the middleware better exploits the limited resources of the mobile devices. However, JMobiPeer does not provide specific features to deal with resource consumption.

3.5.6 Communication

3.5.6.1 Communication model

The two middleware rely on classic protocols: TCP for Expeerience and HTTP, TCP and datagram for JMobiPeer. JMobiPeer extends the possibilities offered by J2ME by using a light HTTP server in order to accept incoming connections.
Over these low level protocols, the middleware provides an endpoint routing service which manages the connection between two peers not directly connected. This protocol searches for a path, replies to path request and routes the message. The choice of such a reactive routing protocol was made in order to limit the consumption of memory by routing tables.
At a higher level, the propagate service allows to send messages within a group. The group notion is fundamental in JXTA and these two middleware. Every peer belongs to at least one group and all communications are performed within a group structure. This allows sending messages with a limited propagation range.

3.5.6.2 Resource identification

All resources (peers, groups, pipes, services …) are uniquely identified , using the same model as in JXTA.

3.5.6.3 Shared resource management

All resources can be advertised in the network. The advertisements are XML-formatted messages. J2ME does not support XML, but a light XML-parser is integrated in the middleware. The advertisements are stored in a binary format in the peers local cache, in order to reduce the amount of memory used. The storage is organized as a vector of three fields: the binary format of the advertisement, the class of membership, the TTL of the advertisement.

3.5.6.4 Resource discovery

In order to discover the available resources, a peer must find the corresponding advertisements. When looking for advertisements, a peer sends a query message in the network. The other peers reply by sending the advertisements matching the query. When the advertisements are received, the requesting peer stores them for future use. In order to use a service, the peer uses the information of the advertisements to find the owner of the service. After that, it sends a query to the owner and begins to use the service. It can also download the service thanks to code mobility. We will present these features later.

3.5.7 Logical network organization

3.5.7.1 Degree of distribution

Every peer in the network plays the same role. The various kinds of peer found in JXTA are not available in the two middleware. They only support edge peers, offering the same functionality as in JXTA.

3.5.7.2 Peer groups

The peer groups are the same as in JXTA. The group is an important structure in the middleware; the authors insist that it is a key point. Every peer belongs to the NetPeerGroup by default. The services are associated to a peer group. It is possible for every peer to create and join groups. The middleware provides the peers with membership functionalities.

3.5.7.3 Localization

The two middleware do not provide functionalities for localization.

3.5.7.4 Proximity/Neighborhood

The two middleware do not support proximity discovery.

3.5.8 Security

The security management is performed as in JXTA. No additional functionality is implemented in the middleware, but it provides support for adding a specific security policy.

3.5.9 Additional functionalities

Expeerience and JMobiPeer support code mobility. A peermay download or upload a service by code migration. As a result, a service available on a distant peer can be copied by another peer in order to be executed locally. The services are really shared in the network and not only accessible thanks to the network. As a consequence, the number of messages circulating on the network decreases and the problems of multi-hop connections are avoided as soon as the service has been copied locally.

3.5.10 Discussion

The evolution from JXTA to JMobiPeer and Expeerience to support mobile devices is done in a good way. The use of J2ME allows JMobiPeer to be more compliant with the mobile devices.
The compatibility with JXTA enables bridges with applications running on a fixed network: for example, Internet access is possible for applications running over JMobiPeer.
The architecture of JMobiPeer is well organized. The virtual messenger offers communication protocols encapsulated in a common component in the core. The other protocols using transport protocols are situated outside the virtual messenger.
The code mobility opens new possibilities. In particular, it could be useful when considering energy consumption: remote execution could be a way to save energy provided it does not generate too many messages.

3.6 InfoWare

3.6.1 Main bibliographical references

[Plagemann, 2003], [Plagemann, 2004].

3.6.2 Context of the project

Infoware is a project from the University of Oslo, Norway, in collaboration with Thales Communications AS and the Oregon Health Science University. The project is under development since 2003 and is expected to be finished in 2007. No information is available about the implementation and the source code is not free.

3.6.3 Objectives

Infoware is a middleware for information sharing in mobile ad hoc networks. The goal of the authors is to provide support for emergency and rescue operations with the following requirements:

- The mobile ad hoc network constituted during a rescue operation may be a hybrid network where several devices act as gateways to the Internet.
- The data shared among participants must be replicated in order to ensure their availability at any time.
- The resources must be efficiently used.
- Information access must be controlled following security and privacy policies defined.

The authors describe a complete rescue operation scenario. The different phases of the operation are well described and give an interesting overview of the use of the mobile devices.

3.6.4 Architecture

Infoware is organized as a set of five components which provide services to applications:

- The "knowledge manager" handles ontology, metadata and integrates information from different sources.
- The "distributed event notification" component decouples subscribers and publishers through mediating nodes.
- The "watchdogs" notifies the participants about local events.
- The "resource manager" keeps track of neighbors and their resources. It also includes the replication mechanisms.
- The "security and privacy manager" provides access control, key management for messages signing and encryption. The mechanisms are based on certificates shared before the use of the middleware during the operation.

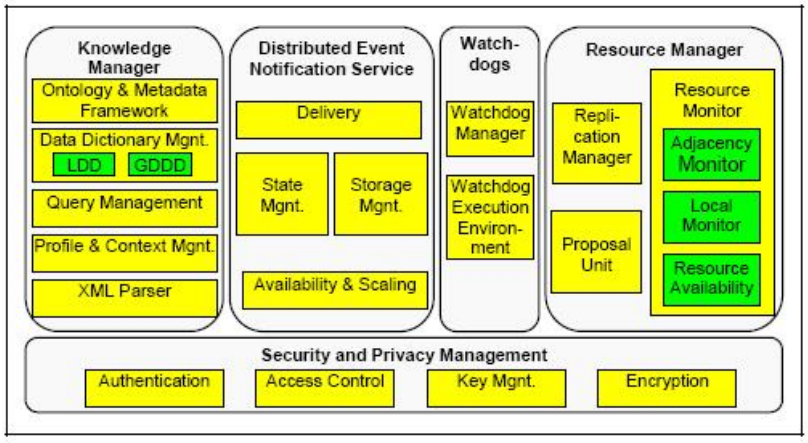

Figure 6: Infoware architecture [Plagemann, 2003]

3.6.5 Device management

Infoware takes into account the low amount of available resources. The middleware is organized into a set of components and is configurable: resource-weak devices only run a subset of the components while more powerful devices may run all components. The Resource Manager (RM) is a distributed service which manages information about resources available in the network. It allows the mobile devices to share their physical resources. For instance, the memory can be shared in order to constitute a common storage space. So it is possible for an overloaded device to store data on another device.

3.6.6 Communication

3.6.6.1 Communication model

The communication follows an event-based model. The peers may act as publishers and subscribers. Peers exchange messages. Every event generates a message which is sent on the network.

3.6.6.2 Resource identification

The resource identification mechanism is not explicitly detailed in the avaible Infoware documentation. However, the Resource Manager deals with physical devices resources as well as software registered as shared resources. The Resource Monitor therefore needs some resource identification mechanism.

3.6.6.3 Shared resource management

The Knowledge Manager (KM) and the Resource Manager (RM) are the two major components that deal with resource management. The Resource Manager allows the devices to monitor their local resources and to share them with the devices in the neighborhood. The information about local resource is frequently updated. The Knowledge Manager is a component that provides high level resource descriptions. The resources are structured and stored thanks to the KM. It also provides global distributed data dictionaries in order to provide a global view of the information shared in the network.

3.6.6.4 Resource discovery

The resource discovery is ensured by the Resource Manager. The RM monitors the local resources available for sharing. The RM also provides functionalities to share the resources between devices in range. The information is disseminated to the other nodes using the Replication Manager component.

3.6.7 Logical network organization

3.6.7.1 Degree of distribution

Every peer plays the same role in the network in order to ensure network robustness. However, several peers connected to the Internet may provide other peers with a connection. Nevertheless, this capability is more a service than a key role in the network structure.

3.6.7.2 Peer groups

Infoware allows the peers to form groups. The groups are used to limit the search for resources to a defined community such as a rescue team. The security manager provides the groups with access control.

3.6.7.3 Localization

Infoware does not provide facilities for device localization. The authors argue that a positioning system like GPS is not interesting in a rescue operation. For example, if the rescue field is underground, the positioning system will not work. Moreover, the field is supposed not to be too extended and the different teams are aware of their location.

3.6.7.4 Proximity/Neighborhood

The notion of neighborhood is used in the resource manager. Considering one node, the closest nodes around are neighbors. These neighbors will be particularly monitored in order to maintain information about available resources. The adjacency monitor is the part of the resource manager which aims to gather information about neighbors. However, the neighborhood is an internal notion and does not constitute an entity directly usable by the applications.

3.6.8 Security

Infoware provides a security manager. It separates the nodes into two groups: the authorized nodes and the foreign nodes. Some information is supposed to be shared prior the use of the mobile devices on the rescue field. The approach adopted consists in using a public key infrastructure (PKI) associated with a common certificate authority. All the messages sent on the network are signed thanks to the key. Thus, the traffic is limited to authorized devices and non-signed messages are not considered by the mobile devices. The security mechanisms are supposed to be transparent to users.

3.6.9 Additional functionalities

Infoware does not provide additional functionalities.

3.6.10 Discussion

Infoware brings a solution to a very specific use of mobile ad hoc networks. It results in a set of functionalities that are not found in other middleware. For instance, the global objective of robustness is addressed through security management and data replication.

3.7 MESHMdl

3.7.1 Main bibliographical references

[Herrmann, 2003]

3.7.2 Context of the project

MESHMdl is the middleware developed in the MESH (Mesh Enables Self-organized Hosts) project at the Technical University of Berlin, Germany. The project also involves the development of a simulator for mobile ad hoc networks. . It was started in 2003 and seems to be ongoing. The software is distributed under a free software license and is implemented in J2ME.

3.7.3 Objectives

The main objective of MESHMdl is to allow mobile ad hoc networks to self-organize. In order to do so, the middleware uses mobile agents and tuple space communication. The role of mobile agents is to introduce logical mobility to complete the physical mobility of the devices. In the same way, tuple spaces represent a logical organization of the information. They introduce a high degree of decoupling. As a result, two

communicating devices do not have to be in the same location in order to communicate. The communication is done through the tuple space.

3.7.4 Architecture

MESHMdl is organized into five layers above the "physical" network. This one can be Bluetooth, IEEE 802.11 or even Internet. The five layers are as follows:

- The generic connection layer provides a generic interface to discover and connect to neighbor devices.
- The interaction layer is the communication layer with neighbor devices.
- The space layer provides the decoupling expected by tuple spaces: asynchronous, anonymous, associative communication.
- The agent runtime manages the mobile agent: start, migration, agent repository …
- The agent application layer instantiates mobile agents.

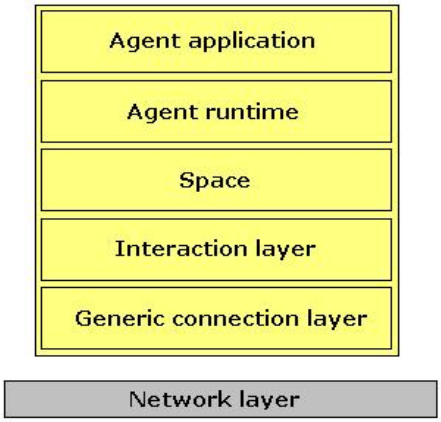

Figure 7: MESHMdl architecture

3.7.5 Device management

MESHMdl does not deal explicitly with device resources management. It only provides information about hardware and software properties of the device, but without managing their restrictions. However, since it is implemented using J2ME, it is well designed for constrained devices.

3.7.6 Communication

3.7.6.1 Communication model

The applications are implemented as groups of mobile agents which collaborate thanks to tuple spaces. The mobile agents can migrate to every node in the network. They are autonomous and do not need controlling entities to take decisions. The inter-agent communication is performed thanks to the tuple spaces. The tuple spaces are used in order to supply the applications with an asynchronous and non-located communication paradigm.

3.7.6.2 Resource identification

MESHMdl introduces the notion of node entries. A node entry contains a universally unique identifier and a series of attributes which describe the hardware and software properties of the device. Consequently, a peer and its properties are identified and are accessible to other peers.

3.7.6.3 Shared resource management

The resources are represented by tuples. A tuple contains typed data items. They are shared in the spaces and can be consulted by the agents. The mobile agents can write into and read from the tuple spaces. MESHMdl also supports information diffusion thanks to Xectors. Xectors allow the devices to disseminate information using other devices in range.

3.7.6.4 Resource discovery

The resource discovery mechanism is not explicitly described. When two nodes become neighbors, they share their entries. As a consequence, and due to the fact the devices are mobile, the information is disseminated. The mobile agents also play an important role in resource discovery. As they are mobile, they can explore the resources represented as tuples and stored in the spaces everywhere in the network. In order to find appropriate information, the mobile agents perform searches using templates.

3.7.7 Logical network organization

3.7.7.1 Degree of distribution

In MESHMdl, every peer plays the same role. The communication is done thanks to mobile agents and tuple and the peers have the same capabilities to use them.

3.7.7.2 Peer groups

MESHMdl does not provide peers with group structure. Nevertheless, the tuple spaces can be considered as group structure. Only the peers using a same space are able to communicate. The spaces are communication areas like JXTA groups.

3.7.7.3 Localization

MESHMdl does not support any localization mechanism.

3.7.7.4 Proximity/Neighborhood

MESHMdl considers the notion of neighborhood. When two devices are in communication range, an engagement protocol is initiated thanks to mobile agents. The two devices exchange their node entries and store them in their local space. When a device leaves the transmission range, its entry is removed from the local space. A peer can send its mobile agents on a neighbor peer in order to disseminate them. The mobile agent stays alive even if the neighborhood link is broken.

3.7.8 Security

MESHMdl allows the peers to exchange data anonymously via tuple spaces. It also ensure the security of inter agent collaborations. These are the two main features for security in MESHMdl.

3.7.9 Additional functionalities

MESHMdl does not provide other particular functionalities.

3.7.10 Discussion

The first advantage of MESHMdl is the availability of source code and the good documentation associated.
The proximity is quite restrictive, but interesting. The possibility for a peer to send its agents on other peers in its communication range permits to disseminate information without flooding the network. This constitutes a good example of the collaboration between logical and physical mobility.
However the article does not describe the basic mechanisms like resource discovery. It focuses essentially on decoupling, agent mobility and the interest of the spaces without specifying how to use these mechanisms.

3.8 Emma

3.8.1 Main bibliographical references

[Musolesi, 2005]

3.8.2 Context of the project

Emma stands for Epidemic Messaging Middleware for Ad hoc networks. The project is developed in the department of computer science in the University College London. There is no detail available about the implementation or the availability of the source code.

3.8.3 Objectives

Emma is designed to provide mobile ad hoc networks with an efficient epidemic messaging protocol. The authors argue that synchronous communication protocols are not suitable to mobile ad hoc networks. They propose to develop an asynchronous protocol, using an epidemic dissemination of the information. The mechanisms are based on an adaptation of Java Message Service (JMS) for mobile ad hoc networks. It uses an event-based model.

3.8.4 Architecture

The architecture is not detailed explicitly in the available documentation. It is centered on an event-based communication scheme. The mobile devices support some kind of routing tables and can manage queues in order to fulfill the point to point model requirements. The events are treated by an unspecified entity.

3.8.5 Device management

Emma does not provide features to manage device resources. Moreover, Emma does not seem mind about resource limitation and takes up a lot of resources by using multiple tables to disseminate the information.

3.8.6 Communication

3.8.6.1 Communication model

Emma is a Message-Oriented Middleware (MOM) representing an adaptation of JMS for mobile ad hoc networks. At the lowest level, in order to send the messages between the mobile devices, Emma can use both a synchronous protocol, if available, and an epidemic protocol specifically developed for the middleware. The protocol is detailed in the section "additional functionalities". The communication follows the event-based model. An event is represented by a uniquely identified message. There are two models to disseminate the events in the network: point to point model and publish-subscribe model. The point to point model is based on queues. The events are sent to the queues and stored on the queue host. The queues are advertised in the network using the synchronous protocol in the neighborhood of the queue host. In the publish-subscribe model, a topic is created and hosted by a peer. The peers interested in this topic subscribe to the topic and wait for events. The topic holder sends the events to the subscribers.

3.8.6.2 Resource identification

There is no detail about mobile device identification. However, the events are uniquely identified using a hashing algorithm. Thanks to this identification, the communication protocol can avoid the replicas in the network. The dissemination mechanisms are also based on the event identification.

3.8.6.3 Shared resource management

There is no information about the management of shared resources. The resources are advertised in the network thanks to events. The events are disseminated thanks to different protocols and models as we have seen in a previous part. There is no information about the resource structure, the identification or the storage.

3.8.6.4 Resource discovery

As we have seen, the resources shared in the network are advertised thanks to events. In the point to point model, the events are stored in queues; in the publish-subscribe model they are sent in the network . The queues or the topic hosts are advertised in the network through a synchronous communication protocol between neighbor nodes.

3.8.7 Logical network organization

3.8.7.1 Degree of distribution

The peers generally play the same role in Emma. Every peer can be a publisher or can host a queue. However, regarding a particular topic, the topic holder plays a central role and its disappearance is critical for the topic subscribers. Emma does not give information about the processing of such a disappearance.

3.8.7.2 Peer groups

Emma does not provide mechanisms to constitute groups of peers. However, in the publish-subscribe model, the set of the subscribers to a particular topic can be viewed as a group by the topic holder. Nevertheless, these subscribers do not have the possibility to know each other.

3.8.7.3 Localization

Emma does not provide functionalities for localization.

3.8.7.4 Proximity/Neighborhood

Emma supports the notion of proximity. When two devices become neighbors, they start a so called "anti-entropy protocol". This protocol allows the dissemination of information and the destruction of useless message replicas. However, it is the only use of proximity context done in Emma.

3.8.8 Security

Emma does not provide features for security or privacy.

3.8.9 Additional functionalities

The main functionality of Emma is the epidemic routing protocol. It allows a better diffusion of the events in the network. The epidemic protocol works as follows. First, the message is replicated on peers in the neighborhood of the sender. The replication is done thanks to the underlying transport protocol. The replicas are stored in tables. When two peers enter in each other neighborhood, they compare their tables and exchange the replicas they do not have. When all the recipients of the message are reached, the message is deleted from the tables. An acknowledgment is sent to the sender for the persistent messages. The acknowledgment uses the epidemic protocol to come back to the sender.

3.8.10 Discussion

Emma can be seen as a good communication model that is to be included in a more complete middleware. The epidemic routing protocol is very efficient even if its cost seems to be too high for mobile ad hoc networks. However, we lack details about other basic functionalities besides the communication model.

3.9 MPP

3.9.1 Main bibliographical references

[Schollmeier, 2003], [Gruber, 2004]

3.9.2 Context of the project

MPP is a project of the Munich University of Technology. It was developed during the years 2003 and 2004. There is no information available on the implementation of MPP and it does not seem to be still under development.

3.9.3 Objectives

The objective of MPP is to provide a set of protocols in order to combine P2P networks and mobile ad hoc networks. The two types of network offer similarities in their organization but the differences must be addresses by an inter-layer protocol. MPP is a set of three protocols:

- EDSR (Enhanced Dynamic Source Routing), is a protocol suitable for mobile ad hoc networks at the network layer.
- MPP (Mobile Peer-to-peer Protocol), a protocol for P2P applications at the application layer.

MPCP (Mobile Peer Control Protocol) is a synchronous interlayer protocol which computes information received from the two other protocols and help the two layers to communicate.

MPP is therefore a cross-layering system.

3.9.4 Architecture

EDSR (Enhanced Dynamic Source Routing) works at the network layer. It is a network routing protocol based on DSR with additional request and reply message Functions. However, it does not change the behavior of DSR and is compatible with it. MPCP (Mobile Peer Control Protocol) is a synchronous interlayer protocol. It provides the following functionalities:

- Registration: MPCP allows the services to register at the network layer. This enables EDSR to notify the appropriate service about incoming messages.
- Search: MPCP transmits the search parameters to EDSR in order to reach remote peers.
- Requests: MPCP transmits incoming requests to the different services.
- Responses: MPCP informs the services about incoming responses.

MPP (Mobile Peer-to-peer Protocol) is the protocol at the application layer. It allows peers to directly exchange data. It is responsible for file transfers within the P2P network. It uses HTTP thanks to a light http server.

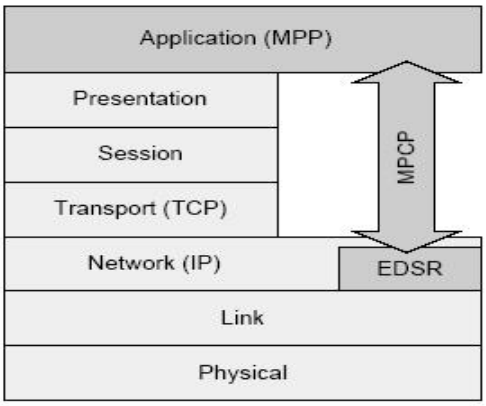

Figure 8: MPP architecture [Schollmeier, 2003]

3.9.5 Device management

The authors do not specify any management of the mobile device resources. None of the three protocols deals with memory or energy. However, the authors mention that

the combination of the three protocols helps in reducing the number of messages sent on the network. It results in a decrease of the energy and bandwidth consumption.

3.9.6 Communication

3.9.6.1 Communication model

The communication is based on an on-demand model. The EDSR protocol provides a set of messages which enable the peers to discover each other and the services shared on the network. At the lowest level, EDSR sends query and reply messages. When receiving a reply message, EDSR transmit it to MPCP which transmits the message to the appropriate application. At the highest level, MPP allows communication between distant peers and can initiate and manage data transfers thanks to the use of http.

3.9.6.2 Resource identification

MPP does not provide an explicit identification of shared resources. However, the applications have to register in order to work. The registration might identify the application in order to allow MPCP to transmit the incoming messages. But no detail is given.

3.9.6.3 Shared resource management

The shared resource management is not detailed in the available documentation. The application is identified by a service of MPCP and can send requests in the network using MPCP and EDSR. The resources are not advertised in the network and the communication scheme uses an on-demand model.

3.9.6.4 Resource discovery

When a peer looks for specific resources, it sends a request containing the wanted type of service. The messages are transmitted thanks to EDSR. When a peer holds a resource that matches the request, it replies to the requesting peer. MPCP notifies the requesting application that a service is available on a specific peer. MPP can initiate a communication between the requester and the service owner.

3.9.7 Logical network organization

3.9.7.1 Degree of distribution

Every peer plays the same role in the network, there is no hierarchy.

3.9.7.2 Peer groups

MPP does not support group structures. The whole network is always taken into consideration.

3.9.7.3 Localization

MPP does not provide features for localization.

3.9.7.4 Proximity/Neighborhood

MPP does not provide features for proximity discovery.

3.9.8 Security

MPP does not provide security features.

3.9.9 Additional functionalities

MPP does not provide additional functionalities.

3.9.10 Discussion

MPP provides an interesting point of view about the association of P2P networks and mobile ad hoc networks. The set of three protocols provides a good solution for communication between the physical and the logical layer. It provides the applications with a good abstraction of the changing topology of the physical network. However, MPP looks like a communication middleware, providing interlayer communication feature but no facilities to exploit the networks.

3.10 MIN

3.10.1 Main bibliographical references

[Yan, 2004]

3.10.2 Context of the project

MIN is a project from the Tirku Centre for Computer Science (TUCS) and Abo Akademi University in Turku, Finland, in collaboration with the Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands. The only documentation available is an article published in 2004. Objectives
The motivation of MIN is to study the convergence of peer-to-peer and mobile ad hoc networks technologies. The authors argue that the two technologies have many similarities. For instance, dynamic network topology, multi-hop connection or routing protocols are performed in a similar way in both technologies. The main objective is to build a structured peer-to-peer network upon the basic connectivity provided by mobile ad hoc networks. In the first version, the authors have chosen to focus on self-organization and integrated routing.

3.10.3 Architecture

MIN is composed of multiple blocks organized into three layers. MIN relies on a network layer providing basic and advanced network services. These services are not specified in the documentation.
Over the network layer, which provides network services over the physical network, we find the link layer. It is composed of three components:

- The *Network Manager* is the manager of node connections.
- The *Awareness* allows peers to be aware of their context. It includes node awareness and message awareness.
- The *Interaction* concerns communication links between peers.

The application layer provides three services:

- The *connect service* establishes a connection between hosts.

- The *lookup service* allows the peers to lookup the contents of the peer-to-peer network.
- The *exchange service* allows peers to exchange data.

Finally, the routing stands between the link layer and the application layer. That is called the integrated routing by the authors.

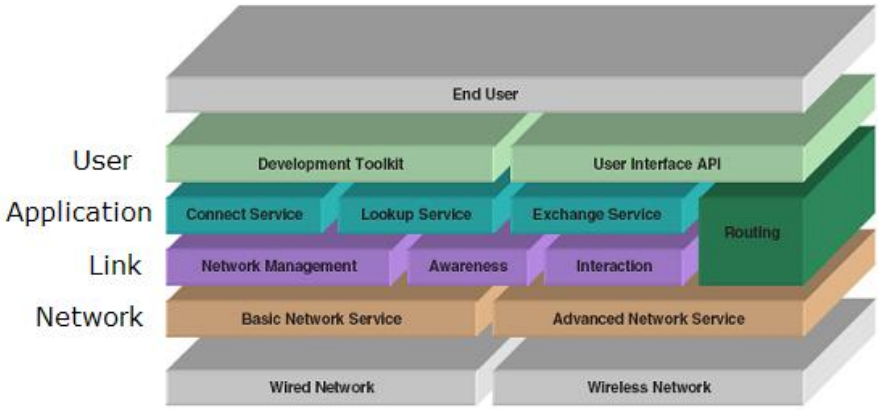

Figure 9: MIN architecture [Yan, 2004]

3.10.4 Device management

Min does not provide functionalities to deal with the limited resources of the mobile devices.

3.10.5 Communication

3.10.5.1 Communication model

At the lowest level, MIN uses an unspecified protocol in order to allow the peers in radio range to communicate. The network layer provides network services, but no details are given. MIN implements a component called Connector. We can imagine that it allows synchronous connections between peers in radio range. The integrated routing is not clearly explained in the documentation. At the link layer, the different services use essentially the communication between neighbors. The messages, whose format is not specified, are disseminate using a flooding protocol.

3.10.5.2 Resource identification

The peers have an identifier. However, no details are given neither about the nature of the identifiers nor about the identification of other resources.

3.10.5.3 Shared resource management

A lookup service and an exchange service are involved, but there no details are given about the resource identification, the replicas management and the advertisement of new services.

3.10.5.4 Resource discovery

The resource discovery is performed at the application layer. After connecting to the network thanks to the connect service, a peer uses the lookup service in order to

lookup the contents in the network. The lookup messages are sent in the network using flooding. When a peer shares a resource that matches the query, it transmits the information to the sender. There is no resource advertisement in the network.

3.10.6 Logical network organization

3.10.6.1 Degree of distribution

The peers play the same role in the network. There is no special function concerning the network organization or the communication organization.

3.10.6.2 Peer groups

MIN does not provide group functionalities.

3.10.6.3 Localization

MIN does not provide functionalities to localize the mobile devices.

3.10.6.4 Proximity/Neighborhood

MIN uses the notion of neighborhood as part of the node awareness functionality. The node awareness is divided into local awareness and remote awareness. In the local awareness, a peer discovers other peers in the radio range. The neighbors are used to update the knowledge about the network topology. The remote awareness allows a peer to find a specific peer in the network, using its ID.

3.10.7 Security

MIN does not deal with security issues in mobile ad hoc networks.

3.10.8 Additional functionalities

MIN does not provide additional functionalities.

3.10.9 Discussion

Min provides a minimal set of services for service support in mobile ad hoc networks. It is communication oriented . Other features concerning network organization are not provided (or are not detailed in the documentation).
The flooding search must be improved in order to satisfy the bandwidth, memory and energy consumption constraints.
The global architecture is hard to understand. The message awareness is used to detect the message type and can report a broken link. However, the integrated routing component can do the same, depending on the message type.

4 Comparison

In this section, we attempt a comparison and a synthesis of the systems previously described. We proceed functionality by functionality, following the outline used to describe the systems.

4.1 Communication

4.1.1 Communication mechanisms

We noticed that a majority of the studied middleware are "Message Oriented Middleware" (MOM), which means that communications among peers are done through message exchange. It is the case of Emma, Proem, JMobiPeer, Jxme, MIN and Steam. These MOMs are often improved by the inclusion of more elaborate communication schemes such as events or advertisements.
Advertisements are used in middleware inspired from Jxta such as Jxme and JMobiPeer. A peer sharing a resource creates a message advertising the resource and sends it in the network. Peers may store the received requests in a local register. A peer looking for resources, sends a request in the network. When a peer receives a request, it looks in its local register for an advertisement that matches the request. If he finds one, he tells the requesting peer to contact the owner of the matching advertisement.
Steam, Proem, Infoware and Emma are event-based middleware. The use of events is associated with filtering services: peer may subscribe to a particular event type, event subject or event sender. In "publish/suscribe" systems, events are notified only to their subscribers. This is in particular the case in the Emma system.

Some middleware use mobile agents as one of their communication features. In MESHMdl, the agents deposit tuples in tuple spaces and move to these places to consult shared data. This allows the peers to communicate anonymously (because the tuples are anonymous). Direct exchanges between peers or agents are not allowed. In Selma, the agents may move to marketplaces that are places of exchange but may also communicate with other agents through messages. Apart from MESHMdl, all the studied middleware are message-based.

4.1.2 Underlying protocols

In order to communicate in a MANET, the devices need to implement at least two kinds of protocols: a routing protocol and a transport protocol. The middleware is very often running on top of the routing and the transport protocols.

The routing protocol is rarely specified; it may be a proactive or a reactive protocol. We can understand it since the middleware just relies on the protocol and remains independent from them. Emma specifies that it uses an existing routing protocol or can use its epidemic dissemination in order to replace it. However, a "cross-layering" approach allows the middleware to integrate the routing protocol and to take advantage of its information. MPP is an illustration of this approach. It provides three protocols: a routing protocol, an application level protocol and a protocol acting as intermediary between the routing and the application level protocol. The advantage of this approach relies in the adaptability of the protocols behaviour.

Regarding the transport protocol, some middleware, like Jxme, JMobiPeer and Proem, operate on top of a transport protocol such as UDP, TCP or HTTP. These protocols bring a lot of improvements: for example, HTTP allows resuming interrupted transfers. However, contrary to UDP, TCP and HTTP are not well suited for

MANETs as they are connected protocols. They are more costly and may alter the global communication performance.

4.2 Shared resources

4.2.1 Resource management

As explained before, by "resource management" we mean resource identification and the means implemented to advertise them in the network.
The identification of the resources is made in a distributed way. It is important to guarantee that two distinct resources will not have the same identifier. The middleware may, for example, use identifier generators based on hashing functions or on long random byte strings. It is also possible to use MAC addresses, which are unique, in order to perform peer identification. The hashing algorithm allows many peers to generate a unique resource identifier available on several peers which do not know each other. It is the approach chosen by middleware such as Emma and Steam. The second solution works on the principle that a long random string is statistically unique. It is the case in Jxta, Jxme and JMobiPeer. However, it would identify the same resource differently, depending on the host that generates it. Finally, let us note that several middleware do not explain their way for identifying the resources.

In order to advertise the resources in the network, the strategy generally relies on advertisement messages sent in the network. The message format is variable. Some use XML and offer the advantage of a common structured format but they are heavy. It is the choice of Jxta, JMobiPeer and Infoware. Other middleware use a binary format that is lighter but does not offer the same structuring possibilities. It is the case of Jxme, Proem, Steam and Emma.

Finally, let us note that the concept of proximity is used in the shared resource management in some middleware like Steam. The policy is to take into account only the resources available locally in order not to overload the network by employing multi-hop communication. The same applies to groups of interest that make it possible to restrict the advertisements made on the resources to the most interested peers.

4.2.2 Resource discovery

The resource discovery is carried out in different ways. Some middleware propose protocols specifically dedicated to resource discovery. In Proem, the peers discover each other thanks to the presence protocol and the associated service. Steam provides a component that detects the peers in the neighbourhood in a transparent way. Infoware provides a component that regularly updates the resources shared by the peers and discovers those available in the neighbourhood.
In the other studied systems, the resource discovery is performed thanks to specific requests. The peer interested in a resource must initiate a search in the network. A peer sharing a resource corresponding to the request calls the requesting peer and the communication can start. In the "publish-subscribe" model, the requesting peer subscribes to an information flow (becomes a subscriber) and waits for the messages coming from the source (the publisher). However, there is always a resource discovery step in order to find the available sources. The resource discovery generally

works "on-demand" when a peer is seeking the sources and subjects available in the network. In the case of systems based on mobile agents, like Selma, the searches are carried out by the agents.

4.3 Fault tolerance

In mobile ad hoc networks, the peers may disappear in an unpredictable way due to the fact they move away, or because the battery runs low and the terminal "dies" or because the terminal is switched off. Specific mechanisms must therefore be designed in order to react to the disappearance of a peer. We may want to distinguish between two kinds of disappearances: predictable ones such as the lack of energy and unpredictable ones such as when a mobile device does not have any more neighbours and is "out of the network". The studied middleware do not bring a satisfactory solution to these situations.

Proem, for example, provides a presence protocol. When a new peer joins the network, an event is broadcast in the network in order to inform the other peers. The presence is associated to a TTL (Time To Live). When a peer disappears, the TTL expires and other peers may consider the peer leaved the network.
Emma implements the publish-suscribe model. In Emma a peer may be a persistent subscriber or not. If a persistent peer disappears the messages sent to him are stored while waiting for its return. However, nothing is proposed to address situations in which a disappearing peer acts as a sender.
Mobile agents based systems such as Selma are naturally robust to disappearance. The agents move in the network by "jumping" from peer to peer and may remain active in the network even if their owner disappears. Moreover, agent replication may be used to improve fault tolerance.

JMobiPeer and Expeerience introduce code mobility. If a peer anticipates that it is going to be cut from the network, for example because its energy level goes below a threshold, it may decide to move services to a remote peer.

4.4 Groups – proximity

Group management and proximity management are often intertwined mechanisms, which explains why we discuss the two issues in the same section.
The study shows that groups may be either established based on common interest or on geographical proximity.

Steam and Selma are examples of middleware in which groups are related to peer proximity. The idea is to facilitate and favour the exchanges among neighbour peers by building neighbour groups. The peers communicate primarily within these groups. The assumption is that that local information is more relevant than remote information. This naturally limits the messages transmission range and therefore the communication cost is reduced. The groups of proximity are highly dynamic because of the peers mobility. Access control mechanisms are often associated with these groups.

The groups of interest give more freedom to the peers. They allow peers located anywhere in the network to form groups. We name them "group of interest" because their principal use is to make it possible for peers interested in the same activity or subject to communicate within a restricted group. Group management may involve

access management mechanisms. Jxme, JMobiPeer and Proem implement groups of interest. In Jxme and JMobiPeer, the groups are virtual entities to which a peer may choose to belong to or not. The main benefit of the group lies in the limitation of the messages range that it offers. Let us note that in these middleware, inspired by Jxta, the concept of group is central because communication relies on the group concept. Finally, in Proem the groups are called communities and correspond to what we have called "groups of interest".

4.5 Device management

When used in the context of mobile ad hoc networks the devices (PDA, laptops and cell phones) often operate on their batteries. Therefore they have limited energy resources, but are often also limited in terms of memory and computing power capabilities. Mechanisms to limit energy and memory consumption have been proposed. For example, in order to save energy, the operating systems manage the luminosity of the screen and the frequency of some processor can be adjusted. However, a great part of the resource consumption is due to the applications and to the network operation.
The control of the resource consumption can be obtained by reducing the number of messages sent in the network as it is done in MPP, Selma and Steam. This reduces the amount of bandwidth and computing time used as well as the energy consumption. The reduction of the number of message sent is obtained, for example, by using the concept of proximity which reduces the transmission range.
The use of J2ME (Java 2 Mobile Edition) in JMobiPeer naturally limits the resources consumption because J2ME was especially conceived for mobile devices with low capacities.
Infoware uses software components in a modular approach that makes it possible to adapt the middleware behaviour to the device capacities. For example, it is possible to load only a reduced number of components in order to limit resource consumption.

It should be noted that some middleware ignore the potential mobile devices capability constraints . It is the case, for instance, of Emma whose epidemic protocol is an important resource consumer.

4.6 Security

Security is only very seldom taken into account in the studied middleware. The principal reason is the complexity of security in ad hoc mobile networks: the lack of a central control device does not make it possible to check the identity of a user. The protection of shared contents is also lacking. It is possible for devices to intercept a communication and to exploit the data contained in the messages. The messages must be encrypted. As the devices share data, the user's privacy has to be enforced by hiding unshared data from other users' sight.

Some of the studied middleware propose solutions for security management. For example, MESHMdl proposes to exchange the data in an anonymous way within shared tuple spaces. JMobiPeer offers components to support various security protocols such as SSL. Finally Infoware proposes the most complete approach by using PKI keys and by encrypting the messages. This effort on security is justified by the nature of the medical data shared in Infoware.

5 Conclusion

The growing interest for mobile ad hoc networks leads to design middleware in order to provide the applications with distribution facilities. In this survey, we have presented an overview of the solutions exploited in the middleware. We have identified a list of useful functionalities which could be incorporated in a middleware for MANET and presented each studied system following this list. Many of the functionalities listed are not addressed by a majority of middleware, but they constitute original features in other middleware.

The first point we can notice after this study is the youth of the research field. All the middleware studied in this paper were developed within the past five years. All come from university laboratories, there is no industrial middleware or application for the moment. Many of these middleware are not developed anymore. However, it is interesting to notice that projects associated to JXTA, like JXME or JMobiPeer, are still supported.

We can notice some similarities between the studied middleware. The communication scheme used in the studied middleware is almost the same. The majority uses message-oriented communication. The messages are often "improved" by events. This scheme suits very well to MANETs as it supports asynchronous communication. Other methods, like remote procedure call, are more complex to use in MANETs because of the lack of reliability between the mobile devices.
Our study also pointed out several functionalities, which are common to the middleware. Most of them constitute the middleware core in order to allow the mobile devices to communicate and share resources. These functionalities are shared resource identification, resource advertisement and discovery, communication protocols, group structure management.

There are also lacks of functionalities, common to a large majority of middleware. The documentation available for each middleware is often light. It is difficult to conclude if the proposed solutions are not complete or are not completely described. Nevertheless, they lack two main functionalities in a very large majority of the middleware: security and resource management.
Security is a major issue for MANETs. The wireless communication is very vulnerable; there is a lack of confidence between the mobile devices. These problems are clearly identified by many authors. However, no middleware proposes a solution to address the security issue.
Resource management also constitutes an important issue for MANETs. The devices have limited resources (energy, memory, CPU, bandwidth) and their management has to be performed in a distributed manner. The middleware seems to be the good layer to enforce resource manager. However, we noticed that in many middleware, the issue is clearly identified, but never addressed.

To conclude on this study, we can argue that no middleware for MANETs provides the applications with a complete set of functionalities. The most useful functionalities, concerning communication for instance, are provided, but some important lacks remain. The research field is always very active and solutions to address security and resource management issues exist. It is reasonable to think that in the near future new middleware for MANETs will be proposed, providing the applications with full functionalities.

6 References

[Abolhasan, 2004] 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).

[Bisignano, 2003] Bisignano M., Calvagna A., Di Modica G., Tomarchio O.: "Expeerience: a Jxta middleware for mobile ad-hoc networks". Proceedings of the Third International Conference on Peer-to-Peer Computing, p. 214 - 215 (2003).

[Bisignano, 2004a] Bisignano M., Di Modica G., Tomarchio O.: "An infrastructure-less peer-to-peer framework for mobile handled devices". Special Issue of European Transactions on Telecommunications on P2P Networking and P2P Services, Vol. 15, Issue 6, p. 599-612 (2004).

[Bisignano, 2004b] Bisignano M., Calvagna A., Di Modica G., Tomarchio O.: "Design and development of a Jxta middleware for mobile ad-hoc networks". Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks (2004).

[Bisignano, 2005] Bisignano M., Di Modica G., Tomarchio O.: "JMobiPeer: A Middleware for Mobile Peer-to-Peer Computing in MANETs". Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops, p. 785 – 791 (2005).

[Chetan, 2004] Chetan S., Al-Muhtadi J., Campbell R., Mickunas M. D.: "A Middleware for Enabling Personal Ubiquitous Spaces". Proceedings of UbiSys: System Support for Ubiquitous Computing Workshop at Sixth Annual Conference on Ubiquitous Computing, Nottingham, England (2004).

[Chetan, 2005] Chetan S., Al-Muhtadi J., Campbell R., Mickunas M. D.: "Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing". Proceedings of the IEEE Consumer Communications & Networking Conference, Las Vegas, USA (2005).

[Chlamtac, 2003] 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).

[Clip2, 2001] Clip2: "The gnutella protocol specification v0.4" (document revision 1.2). http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf. (2001).

[Frodigh, 2000] Frodigh M., Johansson P., Larsson P.: "Wireless ad hoc networking: The art of networking without a network". Ericsson Review. (2000).

[Gong, 2001a] Gong L.: "Project JXTA: A technology overview". Technical Paper, Sun Microsystems, http://www.jxta.org/project/www/docs/jxtaview_01nov02.pdf (2001).

[Gong, 2001b] Gong L.: "JXTA: A Network Programming Environment". In IEEE Internet Computing, v. 5, p. 88-95, (2001).

[Görgen, 2004] Görgen D., Frey H., Lehnert J. K., Sturm P.: "SELMA: A Middleware Platform for Self-Organzing Distributed Applications in Mobile Multihop Ad-hoc Networks". Communication Networks and Distributed Systems Modeling and Simulation, San Diego, USA (2004).

[Gruber, 2004] Gruber I., Schollmeier R., Kellerer W.: "Performance evaluation of the mobile peer-to-peer service". Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, p. 363-371, (2004).

[Hayes, 2004] Hayes A., Wilson D.: "Peer-to-Peer Information Sharing in a Mobile Ad Hoc Environment". Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications, p. 154-162, (2004).

[Herrmann, 2003] Herrmann K.: "MESHMdl - A Middleware for Self-Organization in Ad hoc Networks". Proceedings of the 1st International Workshop on Mobile Distributed Computing, Providence, Rhode Island, USA (2003).

[J2ME] Sun Microsystems: "Java 2 Mobile Edition website". http://java.sun.com/j2me/index.jsp

[Jxme] JXME Community: "JXME's website". http://jxme.jxta.org

[Jxme, 2002] Arora A.: "JXTA for J2ME - Extending the Reach of Wireless With JXTA Technology". Technical paper, Sun microsystems, http://www.jxta.org/project/www/docs/JXTA4J2ME.pdf (2002).

[Jxta] JXTA community: "JXTA's website". http://www.jxta.org

[Jxta, 2003] Jxta community: "JXTA Protocol Specification v2.0". Technical paper, Sun Microsystems, http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.html (2003).

[Kortuem, 2001a] Kortuem G., Schneider J., Preuitt D., Thompson T. G. C., Fickas S., Segall Z.: "When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad hoc Networks". Proceedings of the 1st International Conference on Peer-to-Peer Computing, Lingköping, Sweden (2001).

[Kortuem, 2001b] Kortuem G.: "*Proem: A Peer-to-Peer Computing Platform for Mobile Ad-hoc Networks*". Technical Report.

[Kotilainen, 2005] Kotilainen N., Weber M., Vapa M., Vuori, J.: "Mobile Chedar - A peer-to-peer middleware for mobile devices". Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications Workshops. P. 86-90, (2005).

[Ledoux, 1999] Ledoux T.: "OpenCorba: A Reflektive Open Broker". In Proceedings of the Second international Conference on Meta-Level Architectures and Reflection. P. Cointe, Ed. Lecture Notes In Computer Science, vol. 1616. Springer-Verlag, London, 197-214, (1999).

[Meier, 2002] 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).

[Meier, 2003] Meier R., Cahill V.: "Location-Aware Event-Based Middleware: A paradigm for Collaborative Mobile Applications". 8th CaberNet Radicals Workshop, Ajaccio, France (2003).

[Meier, 2004] Meier R., Cahill V.: "Exploiting Proximity in Event-Based Middleware for Collaborative Mobile Applications". First Workshop on Middleware for Network Eccentric and Mobile Applications (MiNEMA), Dublin, Ireland (2004).

[Meier, 2005] Meier R., Cahill V., Nedos A., Clarke S.: "Proximity-Based Service Discovery in Mobile Ad Hoc Networks". Proceedings of the 5th IFIP International Conference on Distributed Applications and Interoperable Systems, LNCS 3543, p. 115-129, Athens, Greece (2005).

[Musolesi, 2005] Musolesi M., Mascolo C., Hailes S.: "EMMA: Epidemic Messaging Middleware for Ad hoc networks". Personal and Ubiquitous Computing Journal. Vol. 9. September 2005.

[Papadopouli, 2000] Papadopouli M., Schulzrinne H.: "Seven Degrees of Separation in Mobile Ad Hoc Networks". IEEE GLOBECOM, San Fransisco, USA (2000).

[Papadopouli, 2001] Papadopouli M., Schulzrinne H.: "Design and Implementation of a Peer-to-Peer Data Dissemination and Prefetching Tool for Mobile Users". Proceedings of the 1st NY Metro Area Networking Workshop, IBM TJ Watson Research Center, Hawthorne, New York, USA (2001).

[Plagemann, 2003] Plagemann T., Goebel V., Griwodz C., Halvorsen P.: "Towards middleware services for mobile ad-hoc network applications". Proceedings of the 9th IEEE Workshop on Future Trends of Distributed Computing Systems, p. 249-255, (2003).

[Plagemann, 2004] 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". *P*roceedings of IFIP Workshop Challenges of Mobility, Kluwer (2004).

[Proem] Proem's website: http://www.cs.uoregon.edu/research/wearables/proem/

[Schollmeier, 2003] Schollmeier R., Gruber I., Niethammer F.: "Protocol for Peer-to-Peer Networking in Mobile Environments". Proceedings of the 12th International Conference on Computer Communications and Networks, Dallas, USA (2003).

[Services] JXTA services home page: http://services.jxta.org/

[Skype] Skype website: http://www.skype.com

[SOUL] SOUL project website: http://www.syssoft.uni-trier.de/soul/

[Van Steen, 1999] Van Steen M., Homburg P., Tanenbaum A.: "Globe: A Wide-Area Distributed System". IEEE Concurrency, 7(1), p. 70-78, (1999).

[Yan, 2004] Yan L., Sere K., Zhou X., Pang J.: "Towards an Integrated Architecture for Peer-to-Peer and Ad Hoc Overlay Network Applications". Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, p. 312-318, (2004).

Dépôt légal : 2007 – 1er trimestre
Imprimé à l'Ecole Nationale Supérieure des Télécommunications – Paris
ISSN 0751-1345 ENST D (Paris) (France 1983-9999)

Ecole Nationale Supérieure des Télécommunications
Groupe des Ecoles des Télécommunications - membre de ParisTech
46, rue Barrault - 75634 Paris Cedex 13 - Tél. + 33 (0)1 45 81 77 77 - www.enst.fr
Département INFRES