This report presents a lower level specification of the network abstraction and the middleware design architecture for the POPEYE framework. In first place, a brief explanation through the state of the art and group communication is expounded. Among the different approaches, JGroups is chosen as the most suitable platform for reliable group communication. Afterwards, the overlay architecture, which aims to minimize multicast traffic inside the MANET, is depicted. A cluster-based overlay will address this issue by canalizing multicast messages via special nodes, i.e. the superpeers, thus limiting multicast scope to the local cluster. The rest of the document details functional description of the modules in WP4, as well as the architecture and relationships between these modules.

Keywords List

Peer to Peer, middleware, MANET protocols, Ad Hoc networks, overlay
Executive Summary

The purpose of this deliverable document D4.1 entitled "POPEYE lower level Design Document" is to provide a first architectural view of the network abstraction architecture and middleware services addressed by WP4.

This document was developed taking the following documents as references:

- Milestone M4.1, entitled “POPEYE P2P over MANET State Of The Art Analysis Report”, which provides an overview of unicast and multicast routing protocols in a MANET in order to select the most appropriate ones to POPEYE architecture. This document also showed some middleware services just as several simulation environments.

- Milestone M4.2, entitled “Draft POPEYE Lower Level Architecture Description Report”, which provides a draft of the network abstraction architecture and middleware services addressed by WP4. This Milestone document is the start-point of the Deliverable D4.1.

- Another important document used as a reference to this Deliverable is an internal document entitled “POPEYE Module Interface Description” written with the cooperation of all the partners of POPEYE and which lists all the modules that are going to be developed organized by Work Packages and which gives details of every module like the functions or methods provided by each module, a brief definition of the method and the complements that need to be developed.

This document follows the table of contents established during the POPEYE Technical Meeting that took place in Genoa on January 18th-19th, 2007. D4.1 is organized through the following sections:

1. Introduction
2. Functional Description
3. Architecture
4. Relationship with other modules

The first section of the document provides a brief introduction, considering the conclusions of the previous documents and explaining the motivation behind the lower level architecture design.

The second section of this document presents the functional description of each module within the scope of WP4, namely: “Peer Discovery Services”, “Communication Services”, “Naming Services”, “Publish/Subscribe Services” and “Group Management Services”.

The document then presents an architectural view of the different modules, describing both their static architecture and their dynamic behaviour. UML class diagrams and sequence diagrams are included to describe the internal architecture and the dynamic behaviour, respectively.

The last section of the document provides a description of the relationships between WP4 modules and other POPEYE modules.

Furthermore, this document describes and illustrates the achievements made in the first year of the project. The first two phases have concluded: “specification and state of the art analysis phase” and “design research phase” and the key achievements of these phases are the following:
• Specification and state of the art analysis:
  o We have evaluated state-of-the-art solutions for P2P over MANET, taking into account MANET routing alternatives and existing middleware approaches.
  o We have identified several protocols that must be considered for routing issues and P2P services.
  o We have produced the aforementioned Milestone M4.1, which provides an overview of unicast and multicast routing protocols as well as middleware services and simulation environments.

• Design research phase:
  o We have identified performance limitations that require cross-layer communication between routing protocols and the P2P layer.
  o We have explored several interactions between WP4 components (ad hoc routing, overlay routing, multicast communication, group management)
  o We have produced a Milestone document (M4.2) which presents a first draft proposed architecture for WP4.
  o We have identified, specified, designed and started developing several modules in both Middleware Layer and Network Abstraction Layer.
  o The specification and design of these modules include a functional description of each module and a global architectural view of WP4.
  o We also produced a first prototype of Middleware Layer services that provides basic communication mechanisms to the upper layer applications.
  o And finally we have refined the milestone M4.2 to produce the Deliverable document D4.1, which presents the first architectural view of the network abstraction architecture and middleware services addressed by WP4.
## Revision history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description, Editors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>10.05.2007</td>
<td>First version, Antonio F. Gómez Skarmeta (UMU)</td>
</tr>
</tbody>
</table>
## Contributors

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Company</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marcel</td>
<td>Arrufat Arias</td>
<td>URV</td>
<td><a href="mailto:marcelgenis.arrufat@urv.cat">marcelgenis.arrufat@urv.cat</a></td>
</tr>
<tr>
<td>Francisco</td>
<td>Bas Esparza</td>
<td>ENST</td>
<td><a href="mailto:bas@enst.fr">bas@enst.fr</a></td>
</tr>
<tr>
<td>Pedro</td>
<td>García López</td>
<td>URV</td>
<td><a href="mailto:pedro.garcia@urv.cat">pedro.garcia@urv.cat</a></td>
</tr>
<tr>
<td>Antonio F.</td>
<td>Gómez Skarmeta</td>
<td>UMU</td>
<td><a href="mailto:skarmeta@dif.um.es">skarmeta@dif.um.es</a></td>
</tr>
<tr>
<td>Juan Antonio</td>
<td>Martínez Navarro</td>
<td>UMU</td>
<td><a href="mailto:juanantonio@dif.um.es">juanantonio@dif.um.es</a></td>
</tr>
<tr>
<td>Gerard</td>
<td>París Aixalà</td>
<td>URV</td>
<td><a href="mailto:gerard.paris@urv.cat">gerard.paris@urv.cat</a></td>
</tr>
</tbody>
</table>
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCW</td>
<td>Computer Supported Cooperative Work</td>
</tr>
<tr>
<td>CWE</td>
<td>Collaborative Working Environments</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed hash table</td>
</tr>
<tr>
<td>DRM</td>
<td>Delete Replicated Messages</td>
</tr>
<tr>
<td>DYMO</td>
<td>Dynamic MANET On-Demand Routing Protocol</td>
</tr>
<tr>
<td>DYOUM</td>
<td>DYMO implementation of University of Murcia</td>
</tr>
<tr>
<td>GM</td>
<td>Group Manager</td>
</tr>
<tr>
<td>JMS</td>
<td>Java Message Service</td>
</tr>
<tr>
<td>JNDI</td>
<td>Java Naming and Directory Interface</td>
</tr>
<tr>
<td>MANET</td>
<td>Mobile Ad hoc Network</td>
</tr>
<tr>
<td>MMARP</td>
<td>Multicast MANET Routing Protocol</td>
</tr>
<tr>
<td>MP2P</td>
<td>Mobile Peer to Peer</td>
</tr>
<tr>
<td>PDS</td>
<td>Peer Discovery Services</td>
</tr>
<tr>
<td>P2P</td>
<td>Peer to Peer</td>
</tr>
<tr>
<td>WP</td>
<td>Work Package</td>
</tr>
</tbody>
</table>

The present document has been produced in consistence with the definition of terms described in the POPEYE Glossary v1.0 accessible on the POPEYE web site.
# Table of Contents

Executive Summary ...................................................................................................................................................... 2
Revision history.................................................................................................................................................................. 4
Contributors.............................................................................................................................................................. 5
Acronyms .................................................................................................................................................................. 6
Table of Contents .................................................................................................................................................. 7

1 Introduction .......................................................................................................................................................... 9

2 Functional description ........................................................................................................................................... 12
  2.1 WP4 Functionalities overview .................................................................................................................... 12
  2.2 Peer Discovery Services ............................................................................................................................... 12
  2.3 Communication Services ............................................................................................................................. 13
  2.4 Publish/Subscribe Services.......................................................................................................................... 14
  2.5 Group Management Services ...................................................................................................................... 15
  2.6 Naming Services ......................................................................................................................................... 15
  2.6.1 Search Mechanism .................................................................................................................................. 16

3 Architecture ...................................................................................................................................................... 18
  3.1 WP4 Architecture overview ......................................................................................................................... 18
  3.2 Peer Discovery Services ................................................................................................................................ 18
    3.2.1 Peer Discovery Service static view ........................................................................................................... 18
    3.2.2 Peer Discovery Service dynamic view ..................................................................................................... 19
    3.2.3 Superpeer Architecture Description ....................................................................................................... 24
  3.3 Communication Services Architecture ...................................................................................................... 25
    3.3.1 JGroups Protocol Stack modifications .................................................................................................. 26
    3.3.2 Communication Services static view....................................................................................................... 28
    3.3.3 Communication Services dynamic view.............................................................................................. 28
  3.4 Publish/Subscribe Services Architecture .................................................................................................... 29
    3.4.1 Publish/Subscribe Services static view .................................................................................................. 29
    3.4.2 Publish/Subscribe Services dynamic view .......................................................................................... 30
  3.5 Group Management Services Architecture .................................................................................................. 32
    3.5.1 Group Management Services static view ................................................................................................ 32
    3.5.2 Group Management Services dynamic view ........................................................................................ 33
  3.6 Naming Services Architecture ....................................................................................................................... 34
    3.6.1 Naming Services static view ................................................................................................................ 34
    3.6.2 Naming services dynamic view ............................................................................................................. 36
    3.6.3 Search Mechanism Architecture ......................................................................................................... 37

4 Relationship with other modules ..................................................................................................................... 39
  4.1 WP4 Dependencies Overview ....................................................................................................................... 39
  4.2 Peer Discovery Services ................................................................................................................................ 39
    4.2.1 Relationship with security modules......................................................................................................... 39
    4.2.2 Relationship with communication service ............................................................................................. 39
    4.2.3 Context-delivery module ....................................................................................................................... 39
  4.3 Communication Services ............................................................................................................................... 39
    4.3.1 Context-delivery module ........................................................................................................................ 39
    4.3.2 Plugin-based Framework module .......................................................................................................... 39
  4.4 Publish/Subscribe Services ........................................................................................................................... 40
    4.4.1 Workspace management ........................................................................................................................ 40
    4.4.2 Plugin-based Framework module .......................................................................................................... 40
  4.5 Group Management Services ........................................................................................................................ 40
    4.5.1 Workspace management ........................................................................................................................ 40
    4.5.2 Relationship with security modules ....................................................................................................... 40
  4.6 Naming Services............................................................................................................................................... 40
    4.6.1 Data sharing/persistence modules.......................................................................................................... 40
1 INTRODUCTION

The POPEYE objectives are to provide a simple and reliable computing environment for group collaboration over mobile ad-hoc networks.

The POPEYE infrastructure integrates a communication platform and context-aware, secure and personalised core services to enable the design and the usability of collaborative applications in such mobile environments. This platform must offer services in different areas, such as group management, basic communication and naming services while considering flexibility and spontaneous character of mobile ad hoc networks. These services rely on multicast communication, since it stands as the most efficient way to perform synchronous group communication, and they will provide the foundations to build higher layer functionalities for the core services level.

In order to provide these services, the integrated platform should meet several requirements. First of all, due to MANET characteristics, mobile and multi-hop scenario must be supported. Also, scalability and low traffic overhead are key features that must be included to provide an efficient communication platform. Therefore, network topology awareness allows benefiting from peer locality and avoids generating extra routing traffic – a main concern in mobile networks. Therefore, group communication will be based on this topology awareness to achieve an effective and scalable messaging system.

The purpose of this document is, in first place, to show the functional description and the architecture of the WP4 modules in terms of dynamic and static behaviour. Besides, different relationships with other modules are expounded. Before further explanation of these topics, the rest of the introduction will present the state of the art on MANET collaboration middleware.

Afterwards, JGroups is presented as the foundation of the middleware that will provide all communication layer functionalities together with an introduction to the specific overlay topology.

State of the art

Previous to design and development process, study on current middleware for MANET collaboration has been required. In this case, the evaluated middleware approaches are divided in two different groups: a first one including overlay topology related proposals, and a second one embracing programming abstractions and P2P layer basic services. Besides, other more complex CSCW systems have been considered.

Current investigation on P2P middleware is based on two different scopes regarding overlay topology. On the one hand, several unstructured overlays have been adapted to fit MANET requirements. On the other hand, structured overlays like distributed hash tables (DHTs) have proven to be an efficient platform for building distributed applications. Therefore, distributed applications and network services in mobile ad hoc networks (MANETs) could potentially benefit from the deployment of a DHT over the ad-hoc network.

However, most of the evaluated middleware do not have a real implementation since they are just implemented for testing purposes in simulators like ns-2. Tuple spaces implementations [PMR99], as well as other middleware solutions not directly addressed to MANET (JXTA, p2pCM [PGM05]), do not consider the multi-hop scenario and lack communication efficiency. In terms of portability, adaptations of well-known and successful Internet systems like XL-Gnutella [CGT05], BTM [RaS06], and DHT implementations for MANET [ZaS05] [PDH04] rely on a specific routing protocol, thus making them not suitable for our purposes.
Secondly, and after analyzing the different approaches, we state that there is no mature middleware for P2P over MANET, neither at overlay level nor at collaborative level. Mature P2P middleware like JXTA cannot be directly applied in MANETs due to its excessive communication overhead (TCP connections, XML-based protocols). In terms of mobility, most of the approaches being evaluated consider the existence of certain network infrastructure. Furthermore, current collaborative middleware does not fulfil all CSCW requirements like asynchronous and synchronous communication, global virtual data structures or ordered communication channel, among others. However, JGroups toolkit stands as a good alternative, since it provides most of the CSCW required functionalities, such as group membership and communication over a secure and ordered multicast channel, and offers a flexible and extensible implementation.

**JGroups**

In terms of communication in collaborative systems, multicast seems the most natural way to perform group collaboration. However, IP multicast unreliable nature is not sufficient to guarantee most of the desired functionalities. To cope with these needs, JGroups [JGroups] appears to be the most suitable and robust solution. JGroups is a reliable group communication toolkit based on IP multicast but extended with reliability and group membership. It provides a protocol stack architecture in order to suit each application with its specific needs. Moreover, JGroups supports group creation and deletion, joining or leaving the groups, membership detection and notification, detection and removal of crashed members. Group members can send member-to-group messages or member-to-member messages.

The key characteristic of JGroups is that reliability of multicast communication is a deployment issue, and does not have to be implemented by the developer of the application. The protocol stack can be extended with protocols that handle transport, fragmentation, failure detection, lost messages retransmission, ordering, membership and encryption. It is also possible to add a new protocol in order to implement a desired functionality. JGroups also provides high-level abstractions, called building blocks, such as a Replicated Hash Table, a Message Dispatcher, to provide synchronous communication, or an RPC Dispatcher, to invoke remote methods in all group members. Furthermore JGroups has two ports for the J2ME architecture: JGroups-ME and JGroups CDC.

Although JGroups is not designed for high churn rate networks, its layered protocol stack permits to replace components to adapt to specific requirements. In this line, we aim to develop a generic collaboration middleware that can transparently work on top of different network infrastructures (Wifi AP Infrastructure, Ad-hoc network, Multi-hop networks). By replacing lower level layers we will for example provide specific solutions for a MANET network using the unicast DYMO protocol and MMARP manet multicast protocol.

In fact, we propose to create a lower level layer based on dynamic super-peers in order to support the MANET scenarios presented in the POPEYE project:

**Superpeer overlay topology**

Besides considering the best way to perform collaboration, for the purpose of building a sound collaboration architecture, MANET constraints should also be considered. Since performance is a critical issue on multihop networks, communication must be minimized and enhanced between groups members.

Taking this as a basis, in first place we must say that most collaborative applications (chat, shared whiteboards and other) need messages to be received by different members of a group at the same
time. In order to avoid sending the same message several times through the network, hence lower network performance, multicast messaging stands as a good option to cope with group collaboration necessities. In the POPEYE architecture, MMARP [RSG03] routing protocol provides this multicast functionality at network level. Although it is also possible to provide multicast delivery at application level, its complexity and overhead make network protocol implementation much more efficient than application level multicast. Nevertheless, multicast communication needs of tree maintenance and advertisement messages, which may cause excessive message propagation throughout the whole network, consuming bandwidth and battery from resource-constrained devices.

Collaboration on POPEYE is more probable to be performed between collocated peers, since groups tend to be formed in a nearby area. Thus, most multicast messages will be exchanged between close peers, making unnecessary the construction of full multicast maintenance structures. For this reason, we try to benefit from this behaviour and optimize network traffic by exploiting cross-layer functionalities. Therefore, we propose a superpeer overlay based on cluster communication. Multicast messages will be sent only inside a cluster, in order to minimize propagation of multicast maintenance messages. Then, if multicast messages must be sent to farther locations in the network (i.e. a member of the group is located outside the cluster), the superpeer, which is the responsible of a cluster, will send the message transparently to the other interested peers.

Superpeer selection will be dynamic, self-adjusting and sensitive to device characteristics such as connectivity, reliability, bandwidth, and battery power. Superpeer selection algorithms will manage this procedure and, in consequence, will define cluster size and the precise number of superpeers in order to maximize multicast performance.
2 FUNCTIONAL DESCRIPTION
This section describes the functionality of each of the WP4 modules.

2.1 WP4 FUNCTIONALITIES OVERVIEW

This section provides a brief description of each WP4 module focusing on the module functionality.

Firstly, this work package provides communication mechanisms to achieve an effective and scalable messaging over a mobile ad hoc network. Communication Service module is responsible for offering such communication primitives to middleware services and applications. Since collaborative applications are tightly related to group communication, reliable multicast messaging is proposed to perform 1-to-many communications. Moreover, unicast communication is also supported in order to provide 1-to-1 messaging.

Secondly, taking into account MANET constraints and collaborative applications scenario, an overlay architecture based on clusters and superpeers is proposed. The goal of this overlay is to reduce multicast communication by limiting multicast forwarding within the scope of the cluster. Superpeers retransmit multicast traffic between clusters and are in charge of managing cluster information. Peer Discovery Service module is in charge of maintaining the clusters and the superpeers.

Finally, WP4 provides a basic support of core collaborative services. Basic services like a naming service and a publish/subscribe service provide the basis to develop upper layer services and simple final applications. The lower layer of the POPEYE middleware is constituted by these services along with the group management service, which deals with creation and deletion of groups and user membership information.

Network Abstraction Layer Modules

2.2 PEER DISCOVERY SERVICES

Peer Discovery Service will be responsible for the organization of the network. This service is also used among superpeers to share information about services enabled in the whole network.

Superpeers periodically transmit messages SP_INFO to a well known multicast address (228.8.8.8) announcing their presence to other nodes which wants to join the network. A new becoming peer, with an IP network address and running DYMOUM and MMARP routing protocols, first of all will establish thereby the aforementioned multicast address in order to receive the messages sent by the superpeers. Once the node receives the message he checks the distance (TTL, i.e. Time to live) in hops to the superpeer so as to compare it lately to the distance to other reachable superpeers.

Once the node has checked the distance to the superpeer, he will send an acknowledge message SP_OK to the nearest superpeer and will wait for a confirmation to join a cluster and thus became a peer of the network. If a peer wants to leave a cluster he will send a SP_KO message to the superpeer address to let the superpeer update his cluster information. Afterwards, the superpeer will send a confirmation message to the peer.
Superpeers do not only participate in the network as normal peers, but they are also responsible for managing a cluster, and thereby superpeers need to communicate each other changes in their clusters.

Among other functionalities of a superpeer, as we have shown in above paragraphs, he has to send messages to announce his presence to peers to provide them mechanisms to join his cluster. However, a superpeer also has to establish a different multicast address (224.0.0.8) in order to receive updated information of the network from other superpeers. Information about groups and services available in other clusters will be sent to this address.

Figure 1 shows an example of the proposed architecture. In this case the network is divided into three clusters, with their corresponding superpeers.

Another important issue carried out by this service is the group management, which will be described below in the architecture section.

This service provides the next primitive functions:

- Create a group in the network
- Delete a group in the network.
- Join a group in the network.
- Leave a group in the network.
- Obtain the list of members of a selected group.

2.3 COMMUNICATION SERVICES

The Communication Services module is a network abstraction layer module which provides basic communication mechanisms to the middleware layer. Besides interaction with physical network, this module provides the Communication Channel, a component that offers unicast and multicast communication between members of a group. A member of the group can send a message to a specific member or to all members of the group. The user of a communication channel can receive messages by invoking blocking calls or by managing message events that are fired whenever a message is received.
It is also possible to use a communication channel with a specific identifier that restricts the communication between channels with the same identifier used by different peers. These special channels are called Named Communication Channels. Messages that contain identifiers are filtered on reception and are delivered to the applications or services that have registered a listener with the specified identifier.

Provided functions:

- Create a default communication channel: a channel is created with a default name.
- Create a named communication channel: a channel is created with a specific name.
- Send a message: a message is delivered to the recipient member.
- Send a message to the group: a message is delivered to all the members of the group.
- Asynchronous receive: an event is fired when a message is received.
- Synchronous receive: blocking call. The application blocks until a message is received.

**Middleware Layer Modules**

### 2.4 PUBLISH/SUBSCRIBE SERVICES

In every group, different publish/subscribe channels are available. In this way, users can create topics and then assign subscriptions to them. Subscriptions are needed to receive messages published under a certain topic. To be able to retrieve messages after network disconnection, durable subscriptions are also available.

The publish/subscribe mechanism uses the multicast capability of the JChannel to send a message to all members of a group. If a peer is not interested in receiving messages from a certain topic, messages will be automatically discarded. Messages are not sent via the communication channel since this channel is intended to be used just by higher level services.

Durable subscriptions persist to network disconnection, so that if a peer leaves the network due to disconnection or mobility issues, messages published on that topic are delivered automatically to the subscriber when rejoining the group. This can be achieved because all members with a durable subscription on a topic keep all received messages for a certain time. Then, lost messages or previously posted messages can be recovered by asking one of the members of the group.

This publish/subscribe functions are implemented as a subset of the JMS specification. Only the multicast model is implemented, which means that the point-to-point model is not available.

Provided functions:

- Create a topic: allows to send/receive messages on a topic
- Publish a message under a topic: sends a multicast message to all group members.
- Create a subscription to a topic: allows a member to receive messages published under a certain topic.
- Create a durable subscription to a topic: allows a member to receive messages published under a certain topic and to keep them for a certain period of time.
Register a listener to process messages: defines how messages will be processed when they are received under a certain topic.

2.5 GROUP MANAGEMENT SERVICES

Group management services support multiple group creation. Groups are essential for collaboration in POPEYE, since all actions are performed inside the scope of a group.

Groups are created by providing the group name and then, multicast address and port mapping are provided from the underlying services. Each member can join several groups at the same time and they will be notified about event generation whenever other groups are created or deleted. Membership on each group is provided as a list of members that currently belong to the group, together with notification about member join/departure events. Furthermore, reconciliation after network merge will be available for services such as naming and publish/subscribe.

Provided functions:

- Get group members: returns a list with all the members in the current group.
- Add listener for membership changes: allows registering a listener into a group that notifies changes whenever a members joins or leaves the group. These listener methods will be invoked when a Join/Leave event is produced.
- Remove a listener for membership changes: remove one of the previously registered listeners.
- Add listener for group membership changes: allows registering a listener that notifies changes whenever a group is created or deleted. These listener methods will be invoked when a groupCreated or groupDeleted event is produced.

2.6 NAMING SERVICES

Each group in the POPEYE network maintains a decentralized Naming Service which binds names with resources and locations. These resources can be context information sources, registered plug-ins, or other available objects and services. The Naming Service is based on a Replicated Hashtable, i.e. a replicated structure provided by JGroups which replicate each entry of a hash table to all members of the group. The naming service contents are replicated among all members of the group; therefore these contents are automatically available to all members of the same group whenever they are generated.

The Naming Service provides an implementation of a subset of the JNDI specification which offers the following functionalities over a flat namespace:

- Bind a name to an object.
- Bind a name to an object, overwriting any existing binding.
- Unbind a specific object.
- Retrieve the named object.
- Enumerate the names bound in the named context, along with the objects bound to them.
When a new name is added, removed or the bound object contents are changed, the Naming Service fires a NamingEvent. This event can be handled by upper layer services or applications using dedicated JNDI listeners.

Moreover, when a new user joins the group, the current state of the Naming Service is automatically retrieved. The new user obtains the same view of the naming information that is shared by all the peers of the same group. In addition, when two groups merge back, a new view of the naming service is generated and is sent to all peers. This process does not need the intervention of the user because is automatically done whenever a peer joins an existing group or two groups merge back.

2.6.1 Search Mechanism

As described in section 2.1 of deliverable D5.2 (POPEYE Core Services Description), one of the core services of the POPEYE framework is the Data Shared Management, which allows users and other modules to share data (documents and objects) within the same Workspace. Users can see at any time the tree structure of the shared space that holds the data shared by all the members of the workspace. Although this information is useful, it may not be enough to cope with user needs, since most of the time they will be interested in a small portion of these documents which specifically meet their current requirements.

To cope with these needs, the Naming Service will provide a search mechanism to help users find data which interest them.

As explained in deliverable D5.2 (POPEYE Core Services Description), metadata are associated to each data structure. Metadata include attributes set by the users as well as attributes set by the system. Metadata are described as XML files (Figure 2 shows an example).

```xml
<data>
  <title>LIFE OF N. ARMSTRONG</title>
  <author>JAMES R. HANSEN</author>
  <type>TEXT DOCUMENT</type>
  <topic>NEIL ARMSTRONG</topic>
  <description>Biography of American astronaut Neil Armstrong who made history by becoming the first man to walk on the moon</description>
  <owner> </owner>
  <creationDate>03/01/1990</creationDate>
  <accessRights>roro</accessRights>
</data>
```

Figure 2 Metadata instance

Different types of search can be proposed, depending of the kind of information the user is interested in:

- Search by data name: the user specifies a pattern to be found within the document name.
- Search by document attributes: the user specifies a pattern to be found within the metadata attributes of the document.
• Search by document contents: the user specifies a pattern to be found within the content of the document.

Within the POPEYE framework, we will focus on mechanisms to search by data name and by data attributes.

User will have the possibility to search on any metadata field and on any combination of them1.

Regarding the results provided by the search engine, two options will be available. The first option returns a list containing the names of the data that match the query. Since this information may not be useful enough, we provide a second option in which the search engine returns the metadata associated to each piece of data that matched the query.

---

1 In further phases of the development process, we will describe precisely the form of the search queries.
3 ARCHITECTURE

In this chapter we focus on the architecture of WP4 modules. For each module, we describe the static internal architecture and their dynamic behaviour.

3.1 WP4 ARCHITECTURE OVERVIEW

This section is aimed to provide a description of the overall architecture of Work Package 4 modules. The architectural design, the structure and the organization of each module is described in the corresponding subsection. Sub-components of each module are identified and described as well.

First of all, the architecture of the network abstraction modules is presented, describing the Peer Discovery Service and the Communication Service. Special attention is paid to cross-layer adaptations between the multicast routing protocol and Peer Discovery Service. This section also includes the behaviour description of the nodes, whether they are superpeers or they are standard peers.

The network abstraction layer, based on two MANET routing protocols (DYMO and MMARP), and composed by a peer discovery service and a communication service, provides a communication API to support lower-layer middleware services. This middleware includes the Naming Service, the Publish/Subscribe Service and the Group Management Service. We have considered Java standards to implement some of these services. Particularly, the Naming Service implementation is based on JNDI (Java Naming and Directory Interface) and the Publish/Subscribe service follows the JMS (Java Message Service) specification. Although Popeye implementations only consider a subset of these interfaces, the use of standard APIs facilitates the development and maintenance of upper-layer services and applications.

Network Abstraction Layer Modules

3.2 PEER DISCOVERY SERVICES

3.2.1 Peer Discovery Service static view

As explained in the functional description this service is in charge of the organization of the MANET in clusters, defining two kinds of entities in the network. A standard peer which participates in the network and a superpeer which is responsible of the management of a cluster of the network. Moreover it defines the primitives to allow group operations like creating, joining or leaving a group.

We can see in Figure 3 a view of the architecture of Peer Discovery Service implemented using JGroups. It is composed by three main classes corresponding to the main activities done by this service.
First of all, we can see the PDS protocol class which copes with the mobility of nodes. It maintains the Superpeer architecture (superpeer discovery, superpeer communication, cluster handover, etc.). The architecture follows several design patterns in order to simplify the interactions of the classes, for instance, the use of a strategy pattern is used to change the functionalities provided by a peer on the network depending on his role.

Secondly, the GM (Group Manager) will be responsible for providing group functionalities. Here we can see a class called Group (referring to a group of collaboration) that represents the messages sent between peers and superpeers in order to create, join, or leave a group.

Finally DRM (Delete Replicated Message) protocol detects duplicate packets and deletes them. This is very common in MANETs, where multiple paths may exist among sender and receivers and all nodes may act as routers.

3.2.2 Peer Discovery Service dynamic view

Below we can find several Illustrations that will clarify the behaviour of this service related to the activities done by peers and superpeers.

The following figures show the main activities carried out by the PDS in a superpeer.

First of all, as we can see in Figure 4, the process of initializing a channel in order to prepare himself to receive the messages from other peers and superpeers.
Once the channel is initialized he periodically sends SP_Info messages (Figure 5) to other superpeers to make them aware of his presence. He also receives SP_Info messages from other superpeers of the network to update the information about the whole network. If some messages do not get to their destination, superpeers have mechanisms to resend them.

Group primitives are also provided by PDS. Here in Figure 6, it is shown the messages sent to other peers and also how the superpeer receives a group creation request.

Finally, in Figure 7 it is shown the sequence of messages transmitted in order to disconnect from the network.
In the aforementioned diagrams it is showed the point of view of the superpeer, but in this architecture peers also participate. The following diagrams take into account this point of view.

Every peer, the same as superpeers, has to initialize his channels in order to receive messages from peers and superpeers. The initialization process has the same steps and messages as in the Figure 4.

Once the peer has initialized his channels he will be able to detect the presence of superpeers by the reception of MSG_SP_INFO_Cluster messages. The peer will check the distance to all the reachable superpeers in order to check the nearest superpeer. When the peer knows the nearest one he will send a MSG_SP_OK in order to join the cluster as shown in Figure 8.

The next figure, Figure 9, describes the process carried out by a peer in order to do group actions, like the creation of the group, the sequence of messages to join a group and finally how to leave a group.
Superpeers send periodically aliveness messages in order to announce his presence. Due to peers of a network can move establishing new links between them. A peer can detect that there is other superpeer nearer than the current one so he will join the new cluster and leave the previous one. This process is described by the Figure 10.

Finally, as described in Figure 11, when the peer ends his activity it closes the JChannel by sending disconnect and stop messages.
Since the above diagrams only show the messages among the classes of the implementation architecture of peers and superpeers, here we are going to describe some interactions between peers and superpeers in a network.

First of all, in Figure 12, we are going to show the initial stage in which a peer discover the network and join a cluster.

![Sequence diagram about discovering the network](image)

Figure 12 Sequence diagram about discovering the network

The superpeer periodically sends SP_Info_Cluster messages to the network to announce his existence. A new peer receives his message and joins his cluster sending a SP_OK message. Since the peer is moving he also receives other SP_Info_Cluster coming from a nearer superpeer so he joins the nearer one and leave the farther.

The next figure Figure 13, tries to explain the interactions between peer and superpeer regarding group management from the point of view of a superpeer.
When a peer with certain privileges wants to create a group, he has to send a SP_New_Group message to the superpeer in order to authorize him to create it. The superpeer authorizes the peer sending a SP_OK_Group and updates his service list. Afterwards a peer is interested in the group so he sends a SP_Group message to the superpeer to join it. If the peer is authorized to join the group he will receive a SP_OK_Group message. In this diagram the messages used to leave and delete a group are also shown.

3.2.3 Superpeer Architecture Description

In POPEYE, devices can play dynamically two different roles: standard peers or superpeers, depending on their connectivity or resources. Superpeers are the nodes who have the responsibility of organizing the MANET into clusters formed by standard peers.

When a node joins the network it has to connect to the nearest superpeer in order to know information about the whole MANET, which will be provided by the superpeer. For each cluster there will be one superpeer that will manage and have all the information about the cluster, like membership and group information. This information is shared with other superpeers, and thus peers can join groups created in other clusters.

When a peer wants to communicate via multicast messages to other peer, if this peer is in the same cluster, peers communicate directly. In case the peers are located in different clusters, they communicate through their superpeers.

Superpeers communicate each other using a special multicast address in order to share the information about the services of their clusters, discover new superpeers, and to allow the communication between peers from different clusters. Furthermore, this architecture provides mechanisms to reduce the overhead in the MANET. This is because the messages include information regarding the superpeer of the cluster so if a node receives a message
and the superpeer of the message is different from the cluster of the receiver, then it does not propagate this message on its cluster.

### 3.3 COMMUNICATION SERVICES ARCHITECTURE

The communication channel offers a simple way to send a message to another peer. Basic messaging primitives are offered, so that peers can communicate directly via unicast messages using DYMO, or can use multicast to send a message to all other group members through MMARP protocol as shown in Figure 14. Thanks to JGroups protocol stack, several functionalities such as encryption and FIFO or causal ordering can be supported. Furthermore, messages to all peers located in wireless range (1 hop) can also be delivered. In summary, the functions provided by this service are the following:

- **Send**: This function sends a Unicast message to the specified peer.
- **SendGroup**: This function sends a Multicast message to the specified group.
- **SendCluster**: This function sends a message to all the peers that are collocated in the same cluster the sender is.
- **SendNeighbours**: This function sends a broadcast message that is delivered to 1-hop located peers.
- **OnMsgReceived**: Receives a message. This function can be implemented as an event.

![Routing Algorithm Diagram](image-url)
In Figure 15 it is shown how is divided the network in clusters, the relationship between superpeers and also the communication between peers. For example in this figure peers from different clusters communicate each other using their own superpeer.

### 3.3.1 JGroups Protocol Stack modifications

The architecture of JGroups consists of three parts: (1) the Channel API used by application programmers to build reliable group communication applications, (2) the building blocks, which are layered on top of the channel and provide a higher abstraction level and (3) the protocol stack, which implements the properties specified for a given channel.

A channel is connected to a protocol stack. Whenever the application sends a message, the channel passes it on to the protocol stack, which passes it to the topmost protocol. The protocol processes the message and the passes it on to the protocol below. Thus the message is handed from protocol to protocol until the bottom protocol puts it on the network. The same happens in the reverse direction: the bottom (transport) protocol listens for messages on the network. When a message is received it will be handed up the protocol stack until it reaches the channel. The channel stores the message in a queue until the application consumes it.

The most powerful feature of JGroups is its flexible protocol stack, which allows developers to adapt it to exactly match their application requirements and network characteristics. We will extend the JGroups protocol stack with new protocols in order to cope with particularities of Ad hoc networks, to support the superpeer architecture and to incorporate the operation based on Groups. We have defined two new JGroups protocols: the Peer Discovery Service (PDS) protocol and the Group Manager (GM) protocol.

The PDS protocol is mainly in charge of the superpeers’ coordination, superpeer discovery and clusters handover. Superpeers use a JGroups channel to communicate themselves and
another channel to communicate with peers into its cluster. Standard peers use a channel to receive information from the superpeers of the cluster where the peer is attached.

The GM protocol allows the maintenance of Groups: create, join and leave operations. Each peer uses a different channel for each Group it is participating. The superpeer maintains one channel for each Group with peers participating into its cluster.

![Superpeer and peer architecture](image)

The picture above, Figure 16, shows the channels that maintain peers and superpeers and the JGroups protocol stacks of these channels. The PDS protocol is located in the protocol stack of the channel used to exchange information among superpeers and the channel used by each superpeer to coordinate peers into its cluster. The MWG protocol appears in the channels used for Groups communication.

The picture below, Figure 17, shows how the communication between two peers working in the same Group but situated in different clusters is performed. The collaborative application in peer P1 generates data that are processed by the Group channel and sent to the network. The packet is received by the Group channel of the Superpeer SP1 and passed to the superpeer channel that exchanges the packet with the rest of superpeers. The packet is received by SP2 through the superpeer channel. Since there are peers into its cluster participating in the Group, SP2 passes the packet to its Group channel and sent the packet. The peer P2 receives the packet. The JGroups header contains information about the originator of the packet. Then, the collaborative application in peer P2 receives the packet as it had been directly generated by peer P1.
3.3.2 Communication Services static view

The main component of the Communication Service is the Communication Channel. It is a simple component that provides basic communication primitives to send and receive messages and can be used by applications or upper-layer services. It benefits from the underlying JChannel to send both unicast and multicast messages in a reliable and ordered manner. CommunicationChannel do not use JGroups functions directly but an adapter component (CommunicationAdapter). This component adapts all communication functions to the JGroups API and may be modified if the underlying components are changed.

3.3.3 Communication Services dynamic view

The basic behaviour of the Communication Channel is depicted in the following sequence diagram (Figure 18). The user (applications or upper-layer services) wants to create a communication channel and sends a request to the Group (1). The application can specify a name for the channel, thus allowing multiplexed channels over the same group. The Group class is in charge of creating new communication channels (2) and returns the new channel to the application (3). Then, the application can register its own listeners to the channel (4), in order to receive messages sent to the channel. Received messages addressed to a channel that has been registered in the CommunicationChannel are automatically delivered to the registered listener.
The application can also use the communication channel to send messages to all the members of the group (5) or to a specific member. The CommunicationChannel accesses the CommunicationAdapter (6) to forward the message to the Network Abstraction Layer.

CommunicationChannel also provides a function to enable synchronous receiving. When this receiving method is enabled, messages addressed to the channel are delivered to registered listeners and can also be obtained by invoking a blocking receive function. The time to wait for a message is configurable and if no message is received in this time, the CommunicationChannel raises a TimeoutException.

**Middleware Services Modules**

### 3.4 PUBLISH/SUBSCRIBE SERVICES ARCHITECTURE

The architecture of the publish/subscribe service is based in the JMS (Java Message Service) standard, so it implements a part of the specification. Behaviour tries to be as similar as possible to the indications of the standards, considering, however, restrictions of the MANET environment.

3.4.1 Publish/Subscribe Services static view
The Publish/Subscribe service benefits from the JMS interface to provide a topic-based publish/subscribe mechanism. Since JMS is not fully integrated with JNDI, the TopicConnectionFactory is obtained from a group. Then, the TopicConnection and the TopicSession can be obtained in order to create subscribers, publishers and topics.

On the one hand, all messages are sent through a Topic via a TopicPublisher. On the other hand, in order to receive messages, two different kinds of subscribers can be bound to a Topic. TopicSubscribers just receive the messages whereas DurableTopicSubscriber also keep received messages. Thus, messages can be transferred to other peers in case they need to ask for previous messages when they rejoin the group. The topic abstraction is represented by the Topic class. This TopicListenerManager is in charge of keeping all registered listeners from durable and non-durable subscriptions. Therefore, this manager maintains a list of all existing topics and notifies the subscribers when a new message is published on a specific topic.

3.4.2 Publish/Subscribe Services dynamic view

There are three basic operations performed in this module. The first two operations are referred to basic communication functionalities in the publish/subscribe service.
Figure 20 Publish/Subscribe Services sequence diagram: publish

When a publisher wants to be created, the application requests the TopicSession for a new publisher (1 and 2). When the topic is bound to the publisher, messages can be sent by invoking publish (3) on the selected publisher. The message is forwarded to the communication adapter, which sends a multicast message (4), with the topic identifier, to all members of the group.

Figure 21 Publish/Subscribe Services sequence diagram: subscription

The listener is registered in order to receive messages published under a topic.
On the other hand, messages sent by TopicPublishers are received just for TopicSubscribers. Similarly to the previous case, the application requests the TopicSession for Topic (1) and TopicSubscriber (2) creation. Then, the subscriber registers a listener in the given Topic (3,4), so it is able to receive messages. However, messages are not delivered yet to application until a message listener is bound to the subscriber (8). Therefore, when a message is received (9), the TopicListener forwards the message to the listener registered in the TopicSubscriber, so the MessageListener is invoked (10).

Figure 22 Publish/Subscribe Services sequence diagram: durable subscription

Moreover, DurableTopicSubscriber can be used to avoid losing messages on network disconnection. The behaviour is similar to the TopicSubscriber, except for the PersistentSubscriptions class that keeps all received messages in case they are necessary for other members when they rejoin the group.

3.5 GROUP MANAGEMENT SERVICES ARCHITECTURE

Group management has been built taking the group class as a foundation of the architecture. Several groups can be created simultaneously, which will be managed by the GroupManager. Then each group offers access to the underlying services: communication channel, naming service, etc. Furthermore, the group handle encapsulates all structures needed for group operation.

3.5.1 Group Management Services static view
As it is depicted in the above class diagram (Figure 23), the Group Management Service is handled by the GroupManager class. This class maintains an updated list of all joined groups by the local peer. The Group class is an abstraction of a group of peers and includes basic group functionalities such as FlatInitCtxFactory (used to create the Naming service), the CommunicationChannel (basic communication primitives) and the TopicConnectionFactory (publish/subscribe services). It is worth noting that all the middleware services are always tied to a specific group of peers.

The GroupHandle class is an auxiliary class that maintains specific information of the group (GroupParameters) and gives access to the CommunicationAdapter, the class that provides interaction with the network abstraction. Messages received through the publish/subscribe system are handled by the TopicListenerManager.

3.5.2 Group Management Services dynamic view
Group management offers event generation whenever a member joins or leaves the group. In first place, when the underlying membership service offers a new view of the group (3), the MembershipAdapter computes the events that must be triggered (4,5). Then, the event is forwarded to the application, which previously registered a MemberActionListener (1,2).

When information about group members wants to be retrieved, the application (or another service) asks the group (1) which forwards the query to the membership adapter (2). Finally, the information is returned to the application.

### 3.6 Naming Services Architecture

#### 3.6.1 Naming Services static view

The naming service architecture is based on the Java Naming and Directory (JNDI) specification, which is a standard Java interface to build naming and directory services. By using a well-known standard, we provide an easy-to-use module to the upper layers of the architecture. The main component of a JNDI implementation is the Context, an interface which provides access to a set of resources bound to names. All the registered resources are stored in a data structure provided by JGroups called ReplicatedHashtable. Context maps all
modifications of the naming space to the ReplicatedHashtable, which is also responsible for communicating the modifications to all the members of the group. The ReplicatedHashtable is, basically, a transparent replicated structure among all the members of the group.

Moreover, JNDI provides standard events (NamingEvent) to notify applications of the changes that take place in the Context: name addition, name deletion and changes in the bound resources. Applications have to implement a NamingListener in order to receive such events.

Each Naming Service, as well as other middleware services, is attached to a specific group. Therefore, the group component must provide a function to obtain its associated Context. This function returns a factory (FlatInitCtxFactory) that allows the creation of an initial context for the current group.

![Figure 26 Naming Services class diagram](Image)

As it is depicted in the Figure 26, the Naming Service provides event notification when a registered entry changes or a new entry is added. The Context implementation consists in a flat context without name hierarchies (FlatCtx). The naming information is replicated among the peers of the group, i.e. all the peers have the same updated information. The underlying mechanism to ensure this replication is the ReplicatedHashtable, which is in charge of notifying naming changes invoking the methods provided by the Notification interface.

The PopeyeInitialContextFactory class provides methods to create the Naming Service for each group of peers. This Naming Service can access to group-related services and information through the GroupHandle component, e.g. to register listeners that allow receiving messages addressed to the NamingService.
### 3.6.2 Naming services dynamic view

The Figure 27 shows the Naming Services sequence diagram. When an application or an upper layer service that use the naming service wants to register a resource, it invokes the bind method on the naming service (1). As the naming service is based on a replicated hashtable, a put operation (2) must be invoked on the replicated data structure, which, in turn, sends a put message to all the group members (3).

When a put message is received, the replicated hash table receives the corresponding event (7) and updates its data structure with the new information. It is worth remarking that any peer could be the source of the received multicast message, including the same peer that receives the message. Once the Naming Service receives the corresponding notification event, it is in charge of creating and dispatching a new detailed event (8). This NamingEvent may include the new name and its old and new bound object, if they are available. Applications that have shown its interest in a certain type of Naming Service changes will receive this event.

Related operations like rebind and unbind present a similar behaviour, spreading the modifications to all members of the group using multicast messages.

On the other hand, the diagram shows the behaviour of the lookup operation. This operation returns the resource bound to a certain name. When the Naming Service receives this type of query, it performs a get operation on the ReplicatedHashtable, which returns the answer if available. So, all queries over the Naming Service are locally solved and do not initiate a network operation.
3.6.3 Search Mechanism Architecture

**Search Mechanism static view**

Figure 28 represents the class diagram of the Search Mechanism.

As explained before, Context is the interface to a set of resources bound to names (which doesn’t have anything to do with the WP5 Context modules). It is used by the DataSearch class, which represents the engine search, to find out the data that matches the user query.

Once the query is performed, it is passed to Pertinence, which sorts the result depending on the user preferences.

![Figure 28 Search Mechanism Class Diagram](image)

**Search Mechanism dynamic view**

Next sequence diagrams shows the way search operations are performed. The result of the query will depend on user requirements.

Sequence diagram in Figure 29 depicts this mechanism. The “Query” parameter will specify whether a simplest search just based on data name is requested or an advanced one based on metadata. The user will also decide whether metadata should be returned or just a list of data names. Finally, user will request whether to sort the result based on his preferences or not.
This diagram shows all the steps involved in an advanced search where the metadata is returned to the user and it is sorted based on his preferences. Some steps may be omitted.

As can be seen in previous figure, the metadata is retrieved from the Sharing Space. We are evaluating the possibility of using the JNDI functionality that allows object sharing inside the replicated hash table to store the metadata. In that case, the metadata will be stored locally. The final decision will be explained in next WP4 deliverable (D4.2: POPEYE lower level Architecture Definition).

Figure 30 shows the operation used to sort the data taking into account the user preferences.
4 RELATIONSHIP WITH OTHER MODULES

In this section we focus on the interaction between WP4 modules and other modules.

4.1 WP4 DEPENDENCIES OVERVIEW

Since Work Package 4 is located in the bottom-most layer in the POPEYE architecture, it is the basis that will underpin core services and final applications. Therefore, several dependencies arise between WP4 and WP5–WP6 modules.

Besides, transversal modules like security or context services present dependencies at several levels. Although they may use the same APIs that WP4 provides to core services and applications, they provide specific functions to WP4 modules. Security will provide access control policies to the group management module and information about superpeer trust to the network layer. Likewise, context modules will provide information in order to compute superpeer selection.

Network Abstraction Modules

4.2 PEER DISCOVERY SERVICES

4.2.1 Relationship with security modules

The communication between peer and superpeer requires certain security abilities because if a peer does not have a right certificate it must not be able to join a cluster. Moreover the group list of the MANET sent by the superpeer must be signed by the superpeer. This is why the Peer Discovery module will need several security functionalities to check the certificate and sign the list.

4.2.2 Relationship with communication service

Depending on the communication the Peer Discovery module uses the Communication Service module to send a message to a group of peers, using MMARP or to send a message directly to a superpeer using DYMO.

4.2.3 Context-delivery module

Peer discovery services may need profile information in order to build efficient superpeer selection algorithms.

4.3 COMMUNICATION SERVICES

4.3.1 Context-delivery module

This module will use communication service functionalities in order to disseminate contextual data whenever changes on contextual information must be notified.

4.3.2 Plugin-based Framework module

Application modules may use all provided functions in communication services if they are necessary for the selected application.
**Middleware Services Modules**

### 4.4 PUBLISH/SUBSCRIBE SERVICES

#### 4.4.1 Workspace management
Workspace management can use publish/subscribe functionalities in order to keep track of events occurred in the workspace.

#### 4.4.2 Plugin-based Framework module
Application modules may use all provided functions in publish/subscribe if they are necessary for the selected application.

### 4.5 GROUP MANAGEMENT SERVICES

#### 4.5.1 Workspace management
Group Management Service provides a list of members for each Workspace and offers functions to access a Group or to create a new one.

#### 4.5.2 Relationship with security modules
Group management module will need several functionalities from security modules. In first place, access control lists (ACL) will be needed when new users connect to a group. Thus, group management module will verify if the user has the corresponding rights to access the selected group. Furthermore, group creation rights must also be checked before users can create new groups.

### 4.6 NAMING SERVICES

#### 4.6.1 Data sharing/persistence modules
In order to build data sharing and persistence modules, group management offers group membership. Furthermore, naming service offers resource binding to associate data with a unique identifier.

#### 4.6.2 Plugin-based Framework module
Application modules may use all provided functions in the naming service if they are necessary for the selected application.

#### 4.6.3 Search Mechanism

**Data Management and Sharing Services**

The Search mechanism will request the tree structured list of data from the Sharing Space. It will also request the metadata associated to given data.

**User Management**

The result of the queries will be sorted depending on the user preferences, therefore, the user profile of the user will be requested from the User Management module so pertinence can be computed.
5 CONCLUSION

In this deliverable, we have provided an architectural view of the network abstraction and the lower level middleware addressed by WP4. We have described this view in terms of functional description, internal architecture design, and relationships between other POPEYE modules.

As a big picture, we aim to develop a generic collaboration middleware that can transparently work on top of different network infrastructures (Wifi AP Infrastructure, Ad-hoc network, Multi-hop networks). To achieve this goal, WP4 is offering two main layers: the middleware layer and the network layer. By replacing the network layer we can adapt to the underlying network infrastructure.

Concerning the middleware layer, we have described the different modules which this layer is composed by: Naming Services, Publish/Subscribe Services, or Group Management Services. These modules provide basic services to the upper middleware services designed by WP5 and are based on Network Abstraction modules that offer basic communication primitives. It is worth noting that we have considered well-known standards like JNDI and JMS to implement some of these services. In addition, several UML diagrams are included to illustrate the internal architecture of the modules and their dynamic behaviour.

Concerning the network layer, we have presented a cross-layer solution based on dynamic super-peers that tries to reduce traffic overhead in the overall MANET. This solution uses the DYMO unicast protocol and a modified version of the MMARP MANET multicast protocol.
6 REFERENCES


APPENDIX A – MODULE INTERFACE DESCRIPTION

WP4 modules interface

Module name: Publish/Subscribe Services
Description: Publish/subscribe channels are bound to a Group. In this Group, topic-based filtering message communication can be established between the set of peers. Event persistence and data conciliation on merging groups are provided.

Provided functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Consuming modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish</td>
<td>Send a message to the members subscribed to that topic</td>
<td>Topic</td>
<td>Data Sharing, Context Collection Services</td>
<td></td>
</tr>
<tr>
<td>Subscribe</td>
<td>Allows users to receive messages about a selected topic</td>
<td>Topic</td>
<td>Data Sharing, Context Collection Services</td>
<td></td>
</tr>
<tr>
<td>Unsubscribe</td>
<td>Stops receiving messages from a selected Topic</td>
<td>Topic</td>
<td>Data Sharing, Context Collection Services</td>
<td></td>
</tr>
</tbody>
</table>

Required functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Providing module</th>
</tr>
</thead>
</table>

Module name: Group Management Services
Description: the Group Management Services handles group related functionalities. Peers can join a group to start communicating with other group members and can retrieve information about membership and existing groups in the whole MANET.

Provided functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Consuming modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>createGroup</td>
<td>Allows users to create a group</td>
<td></td>
<td>Group</td>
<td>Access Control.</td>
</tr>
<tr>
<td>joinGroup</td>
<td>Allows users to perform group operations: subscribe to a Topic in a group, send a message to all the group members, etc if you are authenticated.</td>
<td>Group, credentials</td>
<td></td>
<td>Access Control</td>
</tr>
<tr>
<td>leaveGroup</td>
<td>Lose communication capabilities in the given Group</td>
<td>Group, credentials</td>
<td></td>
<td>Access Control</td>
</tr>
<tr>
<td>getMembers</td>
<td>provides information about the members of a group</td>
<td>Group, credentials</td>
<td>A list of the members of the Group</td>
<td>Access Control</td>
</tr>
<tr>
<td>getGroups</td>
<td>Provides information about the created Groups</td>
<td>(functionality restricted to the default Group)</td>
<td>A list of the existing Groups</td>
<td></td>
</tr>
<tr>
<td>addMemberActionListener</td>
<td>Allows to register a listener that offers the methods onMemberJoin/Leave/Change, that will be invoked when a Join/Leave/Change event is produced in membership</td>
<td></td>
<td>Workspace management</td>
<td></td>
</tr>
</tbody>
</table>
### Required functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Providing module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Module name: Naming Services

**Description:** the Naming Services provides basic functionality to bind a resource or service with a qualified name

### Provided functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Consuming modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind</td>
<td>Sets the correspondence between a resource identifier and a qualified name</td>
<td>name, resource</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lookup</td>
<td>Gets the resource bound to the given name</td>
<td>name</td>
<td>the resource</td>
<td></td>
</tr>
<tr>
<td>unbind</td>
<td>Removes the correspondence between a resource identifier and a qualified name</td>
<td>name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>search</td>
<td>Searches the resources whose name matches the given pattern</td>
<td>Pattern</td>
<td>A list of resources</td>
<td></td>
</tr>
</tbody>
</table>

### Required functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Providing module</th>
</tr>
</thead>
<tbody>
<tr>
<td>listData</td>
<td>Give the subtree structure under the Path</td>
<td>SharedSpaceID, Path</td>
<td>Shared Space tree structure</td>
<td>Data Management and Sharing Services</td>
</tr>
<tr>
<td>getMetaData</td>
<td>Access to the metadata</td>
<td>SharedSpace, DataID</td>
<td>Metadata</td>
<td>Data Management and Sharing Services</td>
</tr>
<tr>
<td>getUserProfile</td>
<td>delivers a user profile</td>
<td>user</td>
<td>profile</td>
<td>User Management</td>
</tr>
</tbody>
</table>

### Module name: Peer Discovery Services

**Description:** This discovery service provides information about the superpeer overlay.

### Provided functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Consuming modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>getLocalSuperpeer</td>
<td>provides local information about the corresponding superpeer to the local node</td>
<td></td>
<td>Member information about the superpeer</td>
<td></td>
</tr>
<tr>
<td>isDistantHost</td>
<td>tells whether a host is farther than 2 hops or not</td>
<td>host</td>
<td>a Boolean indicating is the selected host is more than 2 hops away</td>
<td></td>
</tr>
<tr>
<td>getTopologyInfo</td>
<td>gives general information about the MANET topology</td>
<td></td>
<td>general characteristics about the MANET</td>
<td></td>
</tr>
</tbody>
</table>
Required functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Providing module</th>
</tr>
</thead>
<tbody>
<tr>
<td>getProfile</td>
<td>Allows a member to obtain its profile. This function can be useful for the superpeer selection.</td>
<td></td>
<td>Profile</td>
<td>Profile Service (WP5)</td>
</tr>
</tbody>
</table>

Module name: Communication Services
Description: The Communication Services provides several primitives to send messages to the network members.

Provided functions:

<table>
<thead>
<tr>
<th>Function name</th>
<th>Description</th>
<th>Parameters</th>
<th>Returns</th>
<th>Consuming modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>send</td>
<td>Unicast message to a selected member</td>
<td>member, message</td>
<td></td>
<td>Potentially, all higher level modules</td>
</tr>
<tr>
<td>sendGroup</td>
<td>Multicast message to a selected group</td>
<td>message</td>
<td></td>
<td>Potentially, all higher level modules</td>
</tr>
<tr>
<td>sendAll</td>
<td>MANET broadcast message</td>
<td>message</td>
<td></td>
<td>Potentially, all higher level modules</td>
</tr>
<tr>
<td>onMsgReceived</td>
<td>Receives a message. It can be implemented as a listener.</td>
<td>message</td>
<td></td>
<td>Potentially, all higher level modules</td>
</tr>
</tbody>
</table>

Required functions: