SOCA (2009) 3:25–45
DOI 10.1007/s11761-009-0038-7

SPECIAL ISSUE PAPER

POPEYE: providing collaborative services for ad hoc and spontaneous communities

Juan A. Botía Blaya · Isabelle Demeure · Paolo Gianrossi · Pedro Garcia Lopez · Juan Antonio Martínez Navarro · Eike Michael Meyer · Patrizio Pelliccione · Frédérique Tastet-Cherel

Received: 30 May 2008 / Revised: 16 January 2009 / Accepted: 22 January 2009 / Published online: 3 March 2009

Abstract Next generation collaborative systems will offer mobile users seamless and natural collaboration amongst a diversity of agents, within distributed, knowledge-rich and virtualized working environments. This ambitious goal faces numerous challenges from the underlying communication infrastructure to the high level application services, with the aim to provide services with the appropriate quality (such as persistence, synchronization, and security). Most currently available tools supporting collaboration address either rather traditional and rigid intra-organizational collaboration scenarios or, at the opposite, completely free and unstructured open communities's interactions. Emerging dynamic, flexible and ad hoc collaboration schemes are hardly or not supported at all. The POPEYE framework offers collaborative services for applications that aim to enable spontaneous collaboration over P2P wireless ad hoc groups, where fixed infrastructure is not a prerequisite, where virtual communities can emerge spontaneously and share data with the appropriate quality of service for business applications (persistence, synchronization, security, etc.).

Keywords Collaborative working environments · Spontaneous communities · Ad hoc networks

J. A. Botía Blaya · J. A. Martínez Navarro
Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia, Murcia, Spain

I. Demeure
Institut TELECOM, TELECOM ParisTech, Paris, France

P. Gianrossi
Softeco Sismat S.p.A., Genova, Italy

P. Garcia Lopez
Department d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Tarragona, Spain

E. M. Meyer
OFFIS, Institute for Information Technology, Oldenburg, Germany

P. Pelliccione (✉)
Dipartimento di Informatica, Università dell'Aquila, L'Aquila, Italy
e-mail: pellicci@di.univaq.it

F. Tastet-Cherel
Advanced Information Technologies Department, THALES Communications, Colombes, France

1 Introduction

Collaborative Working Environments (CWEs) are computer and communications based systems designed to facilitate communication, collaboration, and work by groups, organizations, and societies. The term CWE refers to a broad application and research area encompassing a number of methodologies, technologies and solutions (including research and commercial products) that generally help individuals in communicating, sharing information and coordinating operations during collaborative tasks.

CWE technologies and solutions are typically categorized along three main dimensions (although in practice there are no sharp boundaries between them):

1. *Time*: whether users of the CWE are working together at the same time (synchronous or realtime CWE) or different times (asynchronous CWE);
2. *Space*: whether users are working together in the same place (colocated or "face-to-face") or in different places (distributed or non-colocated);

3. *Scale*: collaboration technologies may be usable by pairs of individuals, by small groups, by medium-size groups, by large organizations, or by entire societies.

Pioneered in the mid-1990s by very popular applications such as IBM's client–server based Lotus Notes and web-based Domino, research in the area of distributed CWE has seen an epidemiological growth over the last decade, thanks to the boost in web technologies (web services) and, more recently, Grid technologies.

Although useful and increasingly put into regular use, current CWE solutions have several limitations and related needs for further research are generally acknowledged. Most systems address either rather traditional and rigid intra-organizational collaboration scenarios or, at the opposite, completely free and unstructured open communities's interactions. Emerging dynamic, more flexible and ad hoc collaboration schemes are hardly or not supported at all. Typically, explicit representation of the collaboration context, of workers' and team's goals and the semantics of underlying business processes are not addressed, and this makes it difficult to ensure the context and process awareness, personalization and collaboration support required in knowledge-rich interactions.

Next generation collaborative systems will offer the mobile user seamless and natural collaboration amongst a diversity of agents within a distributed, knowledge-rich and virtualized working environments. However, this ambitious goal faces numerous challenges from the underlying communication infrastructure to the high level application services. These challenges, depending on the operational need, can be addressed in both technological and scientific terms.

POPEYE [4] is a Specific Targeted Research Project (STREP) in the New Working Environment which aims to enable dynamic, spontaneous, peer-to-peer collaborative group working environments, over heterogeneous mobile ad hoc networks (MANETs), with secure access, support of context information and smart personalization. The result of this project is a framework called POPEYE that offers collaborative services for applications that aim to enable spontaneous collaboration over P2P wireless ad hoc groups, where fixed infrastructure is not a prerequisite and where virtual communities can emerge spontaneously and share data with the appropriate quality of service for business applications (persistence, synchronization, security, etc.). POPEYE integrates a communication platform and context-aware, secure and personalized core services to enable the design and the usability of collaborative applications in such mobile environments. POPEYE offers different kinds of services, such as group management, basic communication and naming services while considering flexibility and the 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 provide the foundations to build higher layer functionalities for the core services level. In order to provide these services, POPEYE meets several requirements: (1) first of all, due to MANET characteristics, mobile and multi-hop scenario must be supported; (2) scalability and low traffic overhead at network layer are key features that must be included to provide an efficient communication platform; (3) network topology awareness allows benefiting from peer locality and avoids generating extra routing traffic—a main concern in mobile networks; (4) group communication is based on this topology awareness to achieve an effective and scalable messaging system.

In summary, the main contributions of POPEYE are:

- *Opportunistic ad hoc networking—meet and join:* support dynamic spontaneous collaborative group working environments with autonomous coordination and knowledge management support. When most of the currently available tools supporting collaboration exploit rigid client-server architecture and rely on a communicationinfrastructure like the Internet, POPEYE's ambition is to get collaborative working free from such constraints. In this setting, POPEYE considers P2P over wireless ad hoc groups, where fixed infrastructure is not a prerequisite, where virtual communities can emerge spontaneously and share data with the appropriate quality of service (persistence, synchronization, security, etc.). In other words POPEYE enables creative usage of networked portable devices without the need of supporting infrastructures.
- *Spontaneous networks—setup working groups quickly and easily:* creative activities and human relations performed among collaborative instruments may get enormous advantages from a cooperative and spontaneous collaboration. Collaborative instruments have to adapt to new needs that can vary spontaneously according to the individual human behavior, e.g., new users entering, existing users quitting the collaboration and users that need to change their role within the collaboration. POPEYE by means of administration mechanisms and cooperation among the nodes maintains service quality and security and offers an automatic discovery and access to services. Confidentiality, integrity, availability and access control with authentication are offered without central administration.

The contribution of POPEYE is then both in the middleware layer, mainly in providing data sharing functionalities with the appropriate quality of service, and in the network layer. POPEYE, in fact, makes use of existing MANET transport protocols, but several modifications and customizations

of these protocols have been done, as will be explained in Sect. 3.4.

The paper is organized as follows: Sect. 2 introduces the technologies that POPEYE uses, Sect. 3 presents the POPEYE framework and its software architecture. The POPEYE implementation is presented in Sect. 4 and Sect. 5 puts in practice POPEYE in the spontaneous collaboration among the participants of a conference, workshop or any similar public event. Related work is discussed in Sect. 6. Finally, Sect. 7 presents final remarks and future directions. A glossary that we built to enable a common understanding of terms used in the paper and in the POPEYE project[1] is attached as an appendix.

2 Background

This section presents the MANET transport protocols that POPEYE uses: the unicast routing protocol is presented in Sect. 2.1 and the multicast routing protocol is introduced in Sect. 2.2.

2.1 DYMO

Many routing protocols have been proposed in the last years to route packets along mobile ad hoc networks. They can be classified in two main categories: *proactive routing protocols* and *reactive routing protocols*. The former group collects routing protocols that attempt to maintain consistent routing information along each node that composes the network. When the topology of the network varies, the information related to the links between nodes is also updated. The second group, by contrast, only creates route to a destination when is required by a source. The creation of the route to the destination begins with a route discovery process to find out the destination. Once the route is established, it is maintained until the destination node becomes inaccessible or the route is no longer needed.

In order to significantly reduce control messages in the whole MANET POPEYE organizes the network in clusters. Unfortunately, proactive routing protocols are not adapt to deal with networks organized in clusters since a peer which belongs to a cluster has no information on the links needed to reach a peer belonging to another cluster. The peer is only aware of the link to his superpeer. It relies on the superpeer for the transmission of the message if the destination is located in a different cluster. Consequently, in the context of POPEYE we have chose a reactive routing protocol that implies less control overhead. 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.

The DYnamic MANET On-demand (DYMO) routing protocol [2], as mentioned in its name, belongs to the reactive protocol group. As such, when a source is interested in sending traffic to a destination node, then it sends route requests throughout the network to find him. Each intermediate node that receives a route request stores a route to the originating node. Thus, when this message arrives at the destination, it will be able to answer to the source directly using a unicast message (route reply) through the reverse route previously created. Thereby, routes have then been established between the originating node and the target node in both directions.

However the source node does not know the complete route to reach the destination, the only knowledge that the node owns is the neighbor who will deliver the message to the next node. All the nodes participating in the active route from the source to the destination maintain their routes and monitor their links so as to react to changes in the network topology. In case of a broken link, the intermediate node will send an error message to the source to notify that the link is broken. Therefore, when the source receives this notification it restarts the discovery route process to get to the destination.

2.2 MMARP

The MMARP protocol (Multicast MAnet Routing Protocol) [28] is designed for mobile ad hoc networks (MANETs). It is fully compatible with the standard IP Multicast model, i.e., nodes from the MANET can interact with nodes from the fixed network in multicast communications without requiring any change. MMARP supports the IGMP/MLD (Internet Group Management Protocol/Multicast Listener Discovery) protocol as a means to interoperate both with access routers and standard nodes. The interoperation with access routers is performed by Multicast Internet Gateways (MIGs) which are ad hoc nodes located just one hop away from access routers. Every MMARP node may become a MIG at any time. The MIG is responsible for notifying the access routers about the group memberships within the ad hoc fringe. This mechanism allows MMARP to interwork with any multicast routing protocol in the access network.

MMARP uses a hybrid approach to build a distribution mesh. Routes among ad hoc nodes are established on-demand, whereas routes towards nodes in the fixed network are maintained proactively. The reactive part consists of a request and reply phase. When an ad hoc node has new data to send, it periodically broadcasts a MMARP_SOURCE message which is flooded within the entire ad hoc network to update the state of intermediate nodes as well as the multicast routes. When one of these messages arrives at a receiver, or at a neighbor of a standard receiver, it broadcasts a

[1] The glossary is also available online at: http://srvweb01.softeco.it/IST-Popeye/site/383/default.aspx.

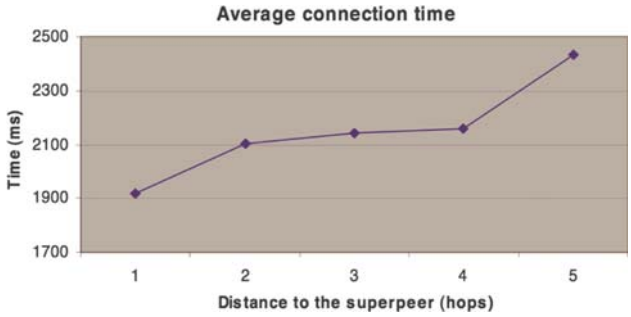

Fig. 1 Average connection time

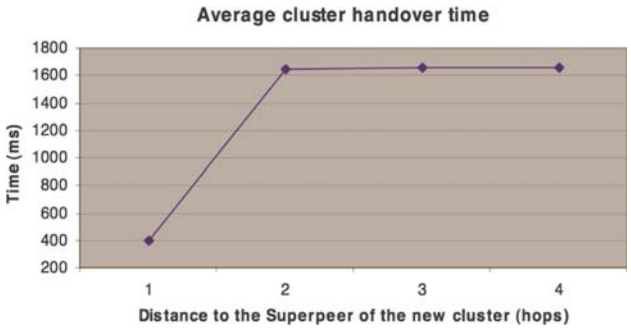

Fig. 2 Cluster handover time

MMARP_JOIN message including the IP address of the selected previous hop towards the source. When an ad hoc node detects its IP address in a MMARP_JOIN message, it recognizes that it is in the path between a source and a destination. It then activates its MF_FLAG (Multicast Forwarder Flag) for the group and rebroadcast a MMARP_JOIN message back to the source.

The proactive part of the protocol is simply based on the periodic advertisement of the MIGs as default multicast gateways to the fixed network broadcasting a MMARP_DFL_GW message. The reception of an IGMP/MLD Query can be used by ad hoc nodes to detect that they are MIGs. The process of creating the path towards the MIG is similar to the one described before. When the MIG receives the MMARP_JOIN message, it sends an IGMP/MLD Report which creates a forwarding state in the multicast router towards the ad hoc network. Once the mesh is established, data packets addressed to a certain multicast group are only propagated by ad hoc nodes which have their MF_FLAG active for that group.

This protocol has been improved to offer information about the clusters, specifically the identifier of the cluster. This information is used to limit the traffic that only involves on cluster to that cluster, not overheading the rest of the clusters. Thus, when a peer is interested to send some traffic to some peers of the same cluster, the information will be only propagated along the cluster but not to the rest of them. When this traffic reach the border nodes of other clusters, they discard these messages preventing the network to be flooded for this traffic. This approach reduces significantly control messages in the whole MANET.

In the following graphs, some performance metrics have been obtained to support the aforementioned statements.

Figure 1 shows the average connection time that requires a peer to connect to a cluster depending on the distance to the closest SP. As we can see, the connection time remains quite unchanged in a cluster that requires from two hops to four hops in order to reach the SP. Thus, the best size of a cluster for this framework is a four hops distance to the SP.

POPEYE is able to deal also with the mobility of their users, reacting to the changes of the topology. Thus, while the user is moving, POPEYE receives information about the closest clusters. A cluster handover process is started if POPEYE detects that the current cluster is not more the best one (due to the proximity of a new one).

Figure 2 shows those results for different distance to the SP. This Figure also supports the statement of the appropriate size of the cluster.

A complete work can be found in [15] and [21] with more details regarding the metrics analyzed and the performance results.

3 The POPEYE framework

The main objectives of POPEYE are to (1) provide an integrated overlay networking architecture that combines the stability and performance of infrastructure networks (when available) with the flexibility and spontaneous character of mobile ad hoc communications, (2) develop a communication platform to provide efficient P2P management and communication primitives, (3) provide higher-level context-aware, secure and personalized core services to facilitate application development by allowing the combination of user preferences with ambience information, such as time, location, user activity, and peers presence.

Applications built on top of POPEYE are intended to be used as easily as possible from the user's point of view. When users of POPEYE applications want to collaborate they just have to start the POPEYE Environment with the selected applications on their wireless device and login to POPEYE.

The device connects to a P2P overlay network which is built upon a mobile ad hoc network (MANET). This happens invisibly to the user. In the next step a user can search for and join existing Workspaces (WS) or he creates a new WS and invites other available users to join him for collaboration. In POPEYE, Workspace is the term used to designate a group of users and the data and applications they share. The users who joined a WS form the group that belongs to the WS. Sharing

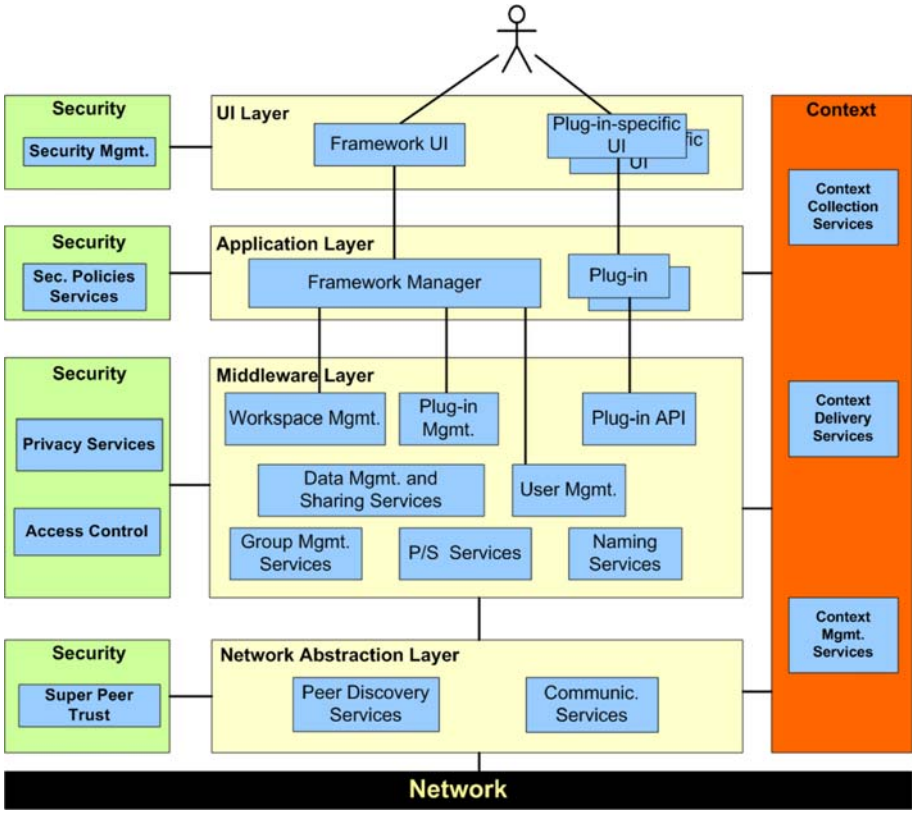

Fig. 3 The POPEYE layered architecture

of data between all members of the group is supported by the shared space which is associated to each WS. The applications the users employ for collaboration (e.g., file sharing, group calendar, whiteboard, etc.) can be plugged into the user's local POPEYE environment at runtime. We call those applications plug-ins and their instances associated to a Workspace and having a specific configuration are named (plug-in-) sessions. The POPEYE system is secure and context aware at all times.

The architecture elicitation process for such a system started with gathering and specification of requirements [5]. This elicitation process was also guided by an end-users requirements workshop in which we gathered requirements expressed by possible POPEYE end users.[2] Based on these requirements, a layered architectural design has been worked out. A layered architecture allows us to organize the system into a set of layers each of which provides a set of services. We chosen the layered model because it supports the incremental development of sub-systems in different layers. In fact, when a layer interface changes, only the adjacent layer is affected.

The POPEYE architecture is composed of 22 components distributed in four layers and two macro-components. Figure 3 shows four different layers: the User Interface, the Application, the Middleware, and the Network Abstraction layers.[3] Security and Context macro-components are orthogonal to the four layers. Each layer and macro-component provides/requires services to/from adjacent layers, through provided/required interfaces. Since each context service is accessed by all POPEYE layers (a real orthogonal context) it is represented by only one component. On the contrary, the security macro-component provides a number of services to the different POPEYE layers.

Relationships among components are always use relationships. Direct connections between single components represent access from the upper layer components exclusively to functions of the connected components within the specific lower layer. In particular the Framework Manager uses services provided by the Workspace Management, the User Management and the Plug-in Management components. At the same time the plug-in API forms a gateway for plug-ins to access only functions of the Middleware layer plug-ins are authorized for.

In the following subsections we focus on each layer and macro-component, and in their subcomponents. The research challenge of POPEYE is both in the middleware layer, mainly in providing data sharing functionalities with the appropriate quality of service, and in the network layer. POPEYE, in

[2] Public Workshop Proceedings are available on http://srvweb01.softeco.it/IST-Popeye/site/376/default.aspx.

[3] The additional physical Network layer is not consider in the remaining of the paper.

fact, makes use of existing MANET transport protocols, but several modifications and customizations of these protocols have been done, as will be explained in Sect. 3.4.

3.1 User interface layer

The user interacts with the POPEYE system by employing the User Interface layer. This layer offers to the users an access to both the POPEYE framework services (e.g., search for users and authentication) and its applications (e.g., chat and file sharing). The *Framework UI* is used for interactions with general POPEYE functions provided by the Framework Manager. This includes all actions concerning Workspace management, plug-in administration and other configurations with regard to the general POPEYE system usage. *Plug-in specific UIs* are individual user interfaces for each application that is installed on the local POPEYE system. For better handling the plug-ins, UIs are packaged with the related plug-in. All components in this layer consume services from Security Management.

3.2 Application layer

The components in this layer implement services to be built on top of the POPEYE infrastructure. They represent the application logic which is accessed by the user through the User Interface layer. The *Framework Manager* represents the real POPEYE Client application core. It constitutes the Controller in a MVC (Model-View-Controller) [3] architectural pattern (where the middleware layer constitutes the Model and the User Interface layer acts as the View). The Framework Manager covers two main roles:

The first role consists in acting as a common entry point for client layer components to POPEYE middleware functionality. Application layer components, such as plug-ins or User Interface components obtain access to lower layers functionalities (such as Workspace handling, shared space management, profile management, security and so on) through this component.

The second role covered by the POPEYE Framework Manager consists in acting as container for managing the life-cycle of plug-in instances in execution on the client application in the context of running sessions.

3.3 Middleware layer

This layer provides the core functionalities of the POPEYE framework. It is organized in eight different modules that make use of the services supplied by the Network layer and Security and Context macro-components.

3.3.1 Data management and sharing services

The Data Management and Sharing Services module is one of the most important and challenging part of the middleware layer since it offers a distributed storage system that allows group members to share data. This system makes use of context information to replicate data in some of the participating devices in order to achieve accessibility from anywhere in the network and the efficient use of the network resources. This module makes use of the super-peer architecture chosen by the network layer. Furthermore, our algorithms make use of the hierarchical network architecture. They take into account locality, for example, to retrieve the data from the closest peer or to decide whether a data should be replicated or not. To consider locality, *Data Management* needs to have some information about the topology. Since clusters are constructed taking into account the proximity of the peers, we use them as a criterion for the locality based algorithm. We provide a refined an improved static view of the module. This module is rather complex since it involves a large number of classes and mechanisms requiring inter modules and inter peer communication. Nevertheless, we provide the users with a well defined set of functionalities available through a unified set of high level interfaces.

Within the POPEYE framework, users, applications and internal modules must be able to share "data". Metadata can be defined as data containing information about other data (describing its attributes and organization). In POPEYE, we use Metadata to keep information about data to be exploited by the POPEYE system itself and by the applications operating on the POPEYE platform.

Data can be structured in different ways:

- *Simple data*: data not divided into smaller pieces. (e.g.,: a code file);
- *Fragmented data*: data divided into smaller pieces for practical reasons without any specific semantics associated to the fragments (e.g.,: a shared video file can be fragmented into smaller pieces to allow users with scarce memory devices to watch it);
- *Composed data*: logical division of data into meaningful pieces (e.g.,: an encyclopedia article composed of text, images, video and style sheet).

A shared space is a virtual space composed of all the data shared among a group of users, applications and internal modules. A shared space is therefore mapped to a "group". Shared space membership is obtained through the corresponding group membership (a group specifies membership policies that define, for example, who can enter the group). As users may be members of various groups, they may be members of various shared spaces. Each member may access

all data in the shared space as if they were local data, even if they are stored on a remote terminal, provided he has the proper access rights. A shared space is state-full: members may share the same data through different sessions of work without having to specify the access rights on this data for this shared space at the beginning of each session.

A shared space is logically organized as a tree structure, which is independent of the way data are stored on the physical devices. This structure presents the logical organization of the shared space: although several replicas of the same data can coexist, this data is unique in the tree structure.

3.3.2 User management

The User Management module is responsible for managing profiles data of a user. It provides administrative functions for retrieving users and users profiles information within both the MANET and specific Workspaces. Therefore, the module provides methods to create, alter and manage the user's profile information and to search for profiles based on specific information using Context modules. In order to achieve a full set of functionalities regarding information describing the user, the User Management module creates and handles multiple profiles representing him. While a single base profile that contains data the user has marked as "publicly available" is accessible by all other users of the MANET there can be several private profiles that provide further information that is only accessible in the Workspace in which this profile is used to represent the user. These private profiles are managed in association with the Security modules and exclusively contain data that the user has marked as "private" when creating his profile(s). To access the base profile of a user in the MANET, the User Management module has a connection to Context modules via these profiles. Besides, a user is able to search for other users with specific profile information (e.g., users sharing the same interests, having the same job or working at the same company) using a similar algorithm that also involves Context modules in order to get a satisfactory result.

3.3.3 Workspace management

The workspace management module controls the lifecycle of the existing Workspaces and Sessions. The Sessions represent the Workspace-related instances of applications. Shared spaces provided by the Data Management and Sharing Services are assigned to each Workspace to allow the Workspace-wide distribution of data. Workspace Management also handles the profiles of Workspaces and Sessions including, e.g., lists of authorized users and the assignment of manager rights for each Workspace. The profiles are stored in the POPEYE shared space to make them accessible to all peers. All actions occurring inside a Workspace (e.g., membership changes, profile changes and invitation notifications) are distributed locally as well as to all participating peers in the MANET by means of an integrated Workspace Event Model. This Event Model encapsulates the events from different sources regarding the status of Workspaces and Sessions and provides them to other components as unified Workspace Events. Furthermore, this module enables to search for Workspaces in the Network following different criteria.

3.3.4 Group management, publish/subscribe, and naming services

Advanced communication mechanisms offered at the middleware layer are organized in three different modules. First, the *Publish/Subscribe Services* module implements a subset of the Java Message Service (JMS) functionalities. The module offers a fully-decentralized topic-based subscription mechanism, where durable and non durable topic subscriptions are supported. Furthermore, it supports message retransmission of lost messages due to temporary disconnection. As long as a single member of the group is connected, messages published under the topic can be transferred to the node when rejoining the group. It must be considered that since there is no central JMS server, subscription information is partially kept by each peer in the group. In this way, when a node rejoins a group due to a temporary disconnection, it asks one of the members of the group in order to obtain previous messages published in the topic. The naming service also follows a JAVA standard by implementing a subset of the JNDI (Java Naming Directory Interface) interface. This naming service is intended to be used in order to store lightweight data like resource discovery, group and other information. The mechanism to store this information is simple but effective: when changes are performed in the naming service, these changes are sent via multicast to all group members. Since we know this might be a resource-expensive mechanism of delivering data, naming services are supposed to be used to store minimal but critical data. On the other hand, information is rapidly and totally available for current group members.

Besides, a decentralized naming service is also offered by an implementation of the Java Naming and Directory Interface (JNDI). The *Naming Services* module binds a name to a specific resource and at the same time provides event notification whenever naming entries are modified. Built on top of these modules, the *Group Management Services* module supports multiple group creation where members can join several groups at the same time. Each group offers group membership, a replicated name service and different publish/subscribe channels. Furthermore, a notification when groups are created and deleted is offered as well as member join/departure events. Group management services handle all interactions related to group creation/deletion and group membership. This module allows creating logical groups

where members can collaborate. It is necessary that all members willing to collaborate belong to the same group. Initially, all members belong to a default group and afterwards they can create or join new groups. Group information is stored in the lightweight replicated naming service, so that it is available for all members in the network. Regarding group membership, this module also provides information about all connected members in the group. When members leave or join a group, events are triggered and forwarded to the plug-ins that are previously registered to these events.

3.3.5 Plug-in management and plug-in API

As described in Sect. 3.2, the POPEYE collaborative applications are implemented as POPEYE plug-ins. The power and flexibility of the POPEYE software lies in the fact that its plug-in based architecture allows easy extensibility of its functionality depending on the particular needs and requirements of each specific implemented application. Plug-ins represent "pieces" of applications which are used for collaboration, encapsulating specific functionality. They can easily be plugged into the local POPEYE System at runtime. Plug-ins can register themselves or can be retrieved by using a central software component in the POPEYE software, the so-called Framework Manager, which acts as an entry point to the POPEYE platform.

The Plug-in Management module in this layer offers mechanisms to locate, download and install plug-ins. The installation of a plug-in is supervised by the plug-in manager, so that an end user has the ability to search for available plug-ins in the POPEYE environment, to share plug-ins or to receive notifications about shared plug-ins. More specifically, plug-ins that locally reside to the peer are loaded at the system startup. After the startup, at each moment a user can start new instances of already installed plug-ins, can install plug-ins that are in the file system (browsing the file system and selecting the jar file of the plug-in), or can install plug-ins that are shared by another peer and then that reside on another peer. Possible dependencies between plug-ins (a plug-in could need other plug-ins to be already loaded) are taken into account and managed by the POPEYE plug-in management component. Multiple instances of the same plug-in can run on the local POPEYE system (even in the same Workspace). Each of those instances represents an own session with an individual configuration. Finally, the POPEYE plug-in manager provides communication infrastructures for plug-ins. Two kinds of communications are provided: (1) a communication similar to a publish/subscribe mechanism used for instance for plug-ins synchronization (e.g., a plug-in that has to react to modifications of other plug-ins) and (2) a communication, similar to components communication, by means of the plug-in interface (i.e., simply invoking plug-in services). For instance it is possible to implement a map plug-in showing peers geographically dislocated in different rooms of the conference and it is possible interacting on the map to directly open a chat with one of the peers. Extension points technology provided by the plug-in infrastructure permits to implement a plug-in that extends other existing plug-ins adding new functionalities (e.g., adding a new button in a plug-in toolbar with a new associated functionality, or adding a view window in a map plug-in in order to give an overview of the whole map when the map is too big to be readable in its entirety). From the POPEYE client point of view, sessions of plug-ins can be executed in the context of a Workspace from inside the Workspace Explorer.

The plug-in management of POPEYE is based on the CHARMY [26] plug-in infrastructure, a framework and tool for software architecture analysis. A developer who aims to write a new plug-in for POPEYE has only to extend an abstract class (`eu.popeye.pluginManager.plugin.Plugin`) and to implement the inherited method `init`.[4]

The plug-in management of CHARMY is free-based on the Open Services Gateway Initiative (OSGi) Service Platform.[5] OSGi provides functions to dynamically change (without requiring restarts) the composition on the device of a variety of networks. OSGi provides also mechanisms to support the deployment of a bundle, called plug-in in POPEYE, (e.g., installation, removal, update, (de-) activation). Dynamic availability of services provided by bundles requires an application to be capable of dynamic assembly (e.g., publication of the bundle's services, connection between bundles and requested services) and dynamic adaptation (e.g., monitoring of existing services, application reconfiguration). Examples of different implementations of the OSGi Service Platform specification are OSCAR,[6] the IBM Service Management Framework[7] and Siemens VDO.[8] The OSGi platform is also the foundation for the Eclipse 3.0 IDE,[9] managing among other things dependencies and interactions between plug-ins.

Even solutions targeted at Web services could be considered among the most suitable for CWE systems. However, Web services deployment has been primarily studied for wired and infrastructure-based network such as Grids [32]. For this reason, CWE frameworks for ad hoc and spontaneous communities could find in the OSGi-based platform the

[4] For further details and for understanding how simple is to write a new POPEYE plug-in, please refer to the deliverable D6.3 freely accessible in the POPEYE webpage.

[5] OSGi Alliance. http://www.osgi.org.

[6] http://oscar.objectweb.org.

[7] http://www.ibm.com/software/wireless/wsdd.

[8] http://siemensvdo.com.

[9] http://www.eclipse.org/equinox/.

most suitable choice for dealing with the deployment of its services.

The *Plug-in API* represents a well defined interface that encapsulates the set of those middleware functions which are to be accessed by (third party) application plug-ins.

3.4 Network abstraction layer

This layer enables the interaction of the middleware layer with the underlying physical network. As stated before, in POPEYE, the target physical network is a Mobile ad hoc Network (MANET).

MANETs are mainly characterized by the mobility of the nodes, and the scarce bandwidth. The mobility of the nodes causes continuous partitions of the network. Furthermore the scarce bandwidth requires ad hoc routing protocols to create efficient routes between nodes, maintain and reconstruct them whenever they break without generating too many control messages. DYMO [2] (Dynamic MANET On demand Routing Protocol) is the underlying unicast routing protocol that is used in POPEYEs mobile P2P over MANET scenarios. DYMO (see Sect. 2.1 for further details) is defined within the MANET group of the Internet Engineering Task Force (IETF) as the substitute for the well-known AODV [1] protocol. The main advantage of using DYMO is that it is specifically designed to work with heterogeneous devices in terms of capacity and computation power. Moreover, DYMO can scale very well to hundreds of nodes, and it has a very low control overhead. In the area of multicast as it could be relevant to the deployment of service in mobile P2P, we use MMARP [28] (Multicast MANET Routing Protocol). MMARP (see Sect. 2.2 for further details) provides efficient multicast routing inside the MANET and incorporates additional functionalities to deal with the complexity of interoperating smoothly with fixed IP networks.

The *Peer Discovery Services* module constructs a peer-to-peer overlay topology over the MANET protocols to improve communication efficiency inside and between groups of collaborating nodes. To achieve that, this service uses a cross-layer approach in which the peer-to-peer overlay benefits from topology and communication traffic from the underlying MMARP MANET protocol. This cross-layer approach is essential for achieving an efficient protocol that reflects the MANET multi-hop topology and avoids unnecessary traffic in the MANET. Note that other approaches just construct peer-to-peer networks that ignore the underlying MANET topology and communications.

In order to adapt to collaboration settings, we propose to construct a hybrid peer-to-peer overlay based on super-peers and clusters of collaborating nodes. In each cluster, exactly one distinguished node—the superpeer (SP)—is responsible for establishing and organizing the cluster. The SPs are responsible for sending SP Info messages in their clusters, containing administrative information for the cluster members. They also form an overlay which allows the communication with adjacent clusters.

Finally, the *Communication services* module represents the set of services available for sending messages to other peers. The main part is the communication channel that allows peers to send messages to one or to all the peers in the group. Messages targeted to all group members are sent through an optimized overlay multicast service routed over super-peers. Furthermore, the channel provides mechanisms for synchronous and asynchronous receive. Note that the communication service is the major abstraction used by the middleware layer. The communication layer abstracts all the low level details required to interact with the peer-to-peer overlay.

3.5 Security macro-component

One of the main objectives of POPEYE is to provide a simple and reliable computing environment for collaborative working groups. This objective is aimed by several security mechanisms, that are disseminated at different levels of the POPEYE architecture. The POPEYE security also aims at providing cryptographic material and security feedback to the other services of the architecture. The POPEYE environment has specific characteristics that make the security very challenging. First, the nodes have limited capacities in terms of power, bandwidth and memory that require the security to remain lightweight. Its distributed nature and the lack of a localized server that could act as a single trust authority do not allow us to use classic public key infrastructures. However, it is also characterized by a relative small number of nodes and time-limited sessions, so that the POPEYE security mechanisms are sufficient for reliable computing. The key concepts of the POPEYE security are group management, access control, trust support services and privacy. Access control is used to decide whether an operation is allowed or not, depending on the security policies attached to the operation, the credentials provided and the possible input of the end user. Several certificates are created, exchanged and stored between the users to prove one's identity and permissions. Unlike access control, the trust services aim at offering transparent and trusted relationships among users. They do not output a binary decision, but a trust level that reflects the confidence a user should have in another one. The establishment of a peer-trusted chain follows an iterative process consisting of tracking and analyzing user's behavior.

Here in the following we detail components that compose the security macro-component.

The *Security Management* module is composed of the *Security ToolBox* and the *Global Security Manager*. The Security ToolBox module is in charge of providing basic cryptographic functions while the global security manager is

in charge of managing the security configuration that is not specific to a Workspace. For example, it will be in charge of enabling or disabling security globally, or to tune the different parameters used in Trust Computation modules.

The *Security Policies Services* module is used to manage and retrieve the set of rules that regulate how security is implemented and protects and distributes sensitive information. The rules apply at different levels: Workspaces, groups, data-sharing services, plug-ins and profiles.

The *Privacy Services* module is responsible for managing private information and profiles, and their certification.

The private information consists in all data that cannot be managed by the Context Services (all the information managed by the Context Services is public) and that will be delivered only to the members of common groups, via the profile used in the group.

The *Access Control* module is used at different levels in the POPEYE software:

- *Network level*. If this option is enabled via the POPEYE global security management service, a POPEYE credential is required before a POPEYE end user uses any other access control function. This credential can be retrieved by different means (for example it could be distributed via an USB key, or pre-configured on POPEYE devices). However, we have implemented the use of the POPEYE captive portal, since it was the best solution in the context of the conference scenario described in Sect. 5.
- *Collaborative work*. The security related to collaborative work between POPEYE users (that is creating and entering Workspaces) is also carried by the Access Control module.

Captive portal is the implemented solution of POPEYE. The captive portal performs a password-based authentication before delivering the POPEYE credential. The main service proposed by the captive portal client side is thus to load a POPEYE credential into the POPEYE software. Other functions (less important from a technical point of view) are the specification of the trusted third party (that is the certificate of the captive portal), and the validation of the POPEYE credential. The credential generated via the captive portal takes the form of an X509 certificate of the public key of the user, signed by the trusted third party authority.

The *Super Peer Trust* module computes the trust of the different peers of the network and the relation between super-peers and how they trust each other. This module is in charge of computing trust for other POPEYE profiles and users. Depending on whether or not the use of multiple profiles is enabled (or if we keep one unique profile per user), the implementation of the trust computation mechanisms differs in some points. As explained before, super-peers are the nodes of the network in charge of a group of nodes nearby located (cluster), being the nodes responsible of managing the communication between clusters. So these nodes are really important in the ad hoc network, not only because of the communication between clusters but also because they will also be in charge of the organization of their own cluster. This module is also in charge of ensuring to a super-peer that another super-peer is not a malicious node trying to modify the behavior of the network on his own interest. Super-peers compose a group called security group. This group is the one in charge of providing the credentials in case a peer needs them. When a connection between peers is established, using requests and replies, both of them are signed by the senders. However, the only way to check and validate this sign is having the credential of the sender. So the peer will request this information to the security group. After receiving the corresponding credential the node will be able to validate the sign of the other peer and trust it. When a new peer is becoming a new super-peer it will send a *SP_REQ* (super-peer request) message signed with its private key. The other super-peer will receive this message and will validate the information about the becoming super-peer. If the data is validated it will also send the group symmetric key to the new super-peer. In this way both of them will be able to communicate with each other without disclosing the data to other nodes of the network. This step can be generalized to the case where more than one super-peer exists in the network. Another alternative can be applied to this scheme. The alternative consists in a voting scheme to decide if a new node is going to be part of a group. More precisely, in the first steps of this alternative scheme the group key is split among several nodes (super-peers). When a new peer is becoming a super-peer it will also send an *SP_REQ* message and the current super-peers will answer with a vote. If the node receives enough votes, applying a math formula it will be able to build the group key and therefore able to participate in the super-peer group. This scheme is an alternative in case the first super-peer, which is the one who owns the information, disappears from the network.

3.6 Context macro-component

The POPEYE project, as well as most CWEs [12,23], must be aware of the environment, preferences, state and equipment of the user. All information, such as the location of the user, the device used to access the system, the topics of interest of the user, and the group of users interested in the same topics, is contextual information. We deal with this information in the form of context so as services offered in this framework should adapt to changing conditions (e.g., the data sharing service may adapt to changes in network topology) and be proactive (e.g., a new user entering the system may be notified of other users or groups sharing interests depending on his own profile) [18].

This implies that POPEYE must be context-aware to be able to deal with this information. In a typical POPEYE scenario, several users equipped with different devices access the system in a non-deterministic fashion, and they handle the same semantic framework of information.

POPEYE contextual services follow an event-driven architecture. The entities of the system can be seen as either producers or consumers (or both) of context. Contexts coming from the environment can be generated by several sources, such as devices, users, sensors, software modules and even external applications. Each entity capable of generating context is called *Context Source*. Users have a profile with contextual information (such as the people they know, the topics they are interested in and the social relationships with other users) and sensors deliver data periodically. The main goal of the context source abstraction is to wrap the entity generating context in a common semantic framework, so the generated context can be readable and understandable by the rest of the system. In order to do this, POPEYE has a set of ontologies, which sets a homogeneous frame of reference of the concepts handled by the system.

One of such ontologies models context itself. Contextual information structure is modeled by using a context hierarchy and an internal structure for context information. For such purpose, the OCP (Open Context Platform) [25] has been reused and adapted for the POPEYE framework. In such ontology, it is stated that the user is the main entity with context. And this piece of information may be as complex as needed, taking into account that it may include an aggregation of environment, location, personal and social information, among others.

The other ontology related with context is the domain ontology. In this case, a new ontology which includes particular details on POPEYE real scenarios has been created by reusing the basic pervasive systems ontology already included in OCP. This domain ontology includes all the basic building blocks to populate user's context structure.

Ontologies used in POPEYE are expressed and managed in OWL, thus they rely on the RDF model based on triplets for the expression of knowledge about concepts and objects of the world. They are implemented by using the Jena[10] ontology manager and ORE[11] enables context reasoning by the definition and execution of rules.

Now, it is clear how context information is modeled and stored. In the following it is introduced how this information is generated and delivered.

Once declared, a source of context information starts generating contextual data. This contextual data is delivered to the system. Based on the users' personal profile and preferences or interests, they may be interested in being aware of changes or updates in some contextual information. A user may be interested in knowing when something changes on shared documents. In order to do this, and following the event architecture, the users register as listeners of some contextual information. Whenever a listened context changes, all listeners are notified of the updating in the contextual information. The key difference of the context-awareness treatment in POPEYE is the collaborative nature of the system. In POPEYE, the user is not just an isolated entity, but a collaborative member of one or more groups. As such, we must take into account not only his personal context, but his *group context* (i.e., the contextual information that arises from him as a member of a collaborative framework of users). This introduces a new paradigm for contextual information treatment, in which the context is not only seen as owned by an entity, but as a dynamic information unit of the group. The contextual information also allows us to get advantage of the semantics of the information. Semantic relationships can be exploited to offer a better service to the users.

The *Context service* has three modules, Context Collection Services (CCS), the Context Delivery Services (CDS) and the Context Management Services (CMS):

- the module *Context Collection Services* is responsible for collecting both the data on the context of each user in the peers and the communication with the management module contexts (Context Management Services) in the superpeer;
- the *Context Delivery Services* module has the functionality of obtaining contexts through communication with the Context Management Services. The CDS can perform searches of context, register as listener to be notified of changes in a given context, or request a specific context. Therefore, when the CDS receives a message, it possibly notifies the event to their own peer listeners;
- the *Context Management Services* are executed in a superpeer and this module manages all contexts in the cluster in which belong. The contextual communication with other clusters is build with the data share module. In the CMS we can search some context, read a context by its identify, etc. The CMS has a server behavior, i.e., it waits for connections of CCS/CDS modules.

4 POPEYE implementation

This section describes the implementation of the POPEYE framework as available at the end of the project. The actual implementation of POPEYE does not support mobile devices even though the conceptual part of the work can be reused for further implementations optimized for mobile devices such as PDAs. However, it is important to note that POPEYE is

[10] http://jena.sourceforge.net/.

[11] http://ore.sourceforge.net.

defined to work in mobile contexts in which the mobility is on walking speed.

When the user clicks on the POPEYE client icon on his laptop desktop, a start-up console window is displayed; it prompts the user for login information (username and password).

Registered users may login through this console and non registered users may proceed with the registration.

This console provides a top-level menu for the POPEYE system. It displays basic information about the user that has logged in to the system through this POPEYE instance. The functionalities offered to the user are:

Open or create a Workspace. When the user selects this function, the system discovers the Workspaces currently available in the MANET and lets the user choose which Workspace to join. This is done through a dialog box. The dialog box also allows the user to create new Workspaces. Once an existing Workspace is chosen or a new one is created, a window panel appears below the main console giving access to the application component named Workspace Explorer, which allows the user to browse and interact with the main elements that constitute a Workspace activity. More precisely the procedure of subscription to new Workspaces is the following: the user selects the "search Workspaces" command which opens a search dialog allowing him to select search criteria. As soon as the desired Workspace has been identified, the user tries to open it. The system notices this and asks the user if he wants to send the Workspace administrator a request for subscription. If the user answers "yes" such request is sent to the Workspace administrator. The local POPEYE client instance of the current user then opens a Workspace Explorer window (temporarily "locked"). As soon as the Workspace administrator receives the request for subscription, he may decide to accept it or not. In case the request is rejected, the system notifies the current user and the Workspace Explorer window is closed. Otherwise, the "subscription accepted" notification is sent to the current user and the corresponding Workspace Explorer window is unlocked.

User preferences. A window with user's profile information is displayed and the user can browse and modify his own profile. When this option is chosen, as in the previous case a window appears below the main console toolbar. In this case the window contains the User Profile explorer, to allow the browsing and modification of the profile data items. This is the same window as in the create account command above.

Local settings menu option. Lets the user browse and modify preferences and settings related to the local instance of the POPEYE client.

Search profiles. Opens a window querying the context and displays the search results. From the result list it is possible to perform some operations, such as inviting a user to a Workspace.

Figure 4 depicts the general layout of the Workspace Explorer window. As shown, each opened Workspace (i.e., a Workspace the current user has joined) owns a separate tab in the Workspace Explorer; this allows easy navigation on multiple Workspaces without making the user interface too complicated. Each tab has a structure that allows access to the Workspace elements: the active plug-in sessions, the subscribed users and the Workspace profile (settings). As can be seen on the left hand-side of Fig. 4 the plug-in session view is organized in three different categories of plug-ins: plug-in sessions active in the local terminal (and other sessions of the same plug-in can be started just clicking with the mouse) plug-in sessions active in the Workspace (and a session plug-in can be started on the machine just clicking with the mouse) and finally plug-in that can be started from a jar file that resides in the file system. Moreover, a message area will appear below the main window area to show incoming messages or system notifications.[12] Finally the right-hand side part of the window hosts the loaded plug-ins.

5 Putting POPEYE in practice in the context of international conferences

In this section we put POPEYE in practice in the context of international conferences. We developed a set of applications on top of the POPEYE middleware in order to show how POPEYE can facilitate mobile interaction, networking and collaboration among the participants of a conference, workshop or any similar public event. To better set the context, let us consider the following scenario:

"Pedro is a researcher in the area of distributed systems taking part in an important conference on P2P issues. The conference is organized in Château Villette, a huge Castle in the countryside near Versailles in France. The Castle is characterized by the almost complete absence of communication infrastructures.

Pedro arrives at the reception desk and physically authenticates as conference participant. He receives his proceedings book, a badge, a collaborative tool and a signed certificate. Pedro switches on his laptop and, by means of the collaborative tool, in the user list he can distinguish conference participants from conference organization members. Pedro contacts an organizer of the conference by an instant message tool and asks him for specific information about the conference's technical setting for giving his talk. Pedro searches for users with similar interests, matching academia users with enterprise users in order to prepare a project proposal in his research area subject. Pedro enters the P2P work group to discover other conference attendants.

[12] More details on these elements are included in the deliverable D6.2 freely accessible in the POPEYE webpage.

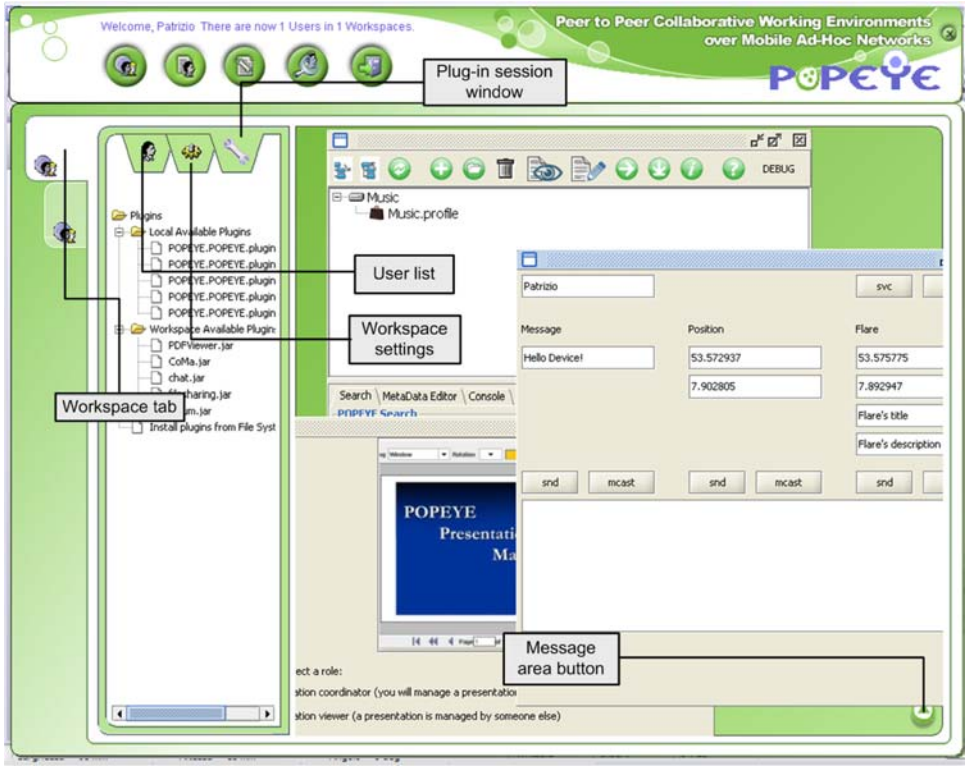

Fig. 4 Workspace Explorer window

Pedro's peer obtains additional collaborative tools provided by conference organizers and by work group users. These additional tools become available in Pedro's tool list. When entering the work group, Pedro is redirected to the conference portal (HTML/WML) where he can access all information regarding the conference such as the program, venue and city maps, city guides, and list of participants. He can also list the online users connected to the work group by using the presence tool.

All "shareable" documents stored in the participants' end devices are automatically made visible in Pedro's space. Pedro looks for documents that deal with P2P on MANETs. In particular, he is interested in documents that were edited by his former colleague Perdita. Pedro starts browsing the documents and, although not all of them are available by direct network connection, the system makes them visible and accessible to Pedro through the distributed network just as if they were in a single virtual space. A participant may create, read and possibly modify a shared document provided that he has been granted the corresponding privileges. Collaborative editing sessions may also be conducted. Pedro decides to create a new shared document in which he will collect some of the participants' feedbacks from the presentations they attended. He leaves the document open for modifications by other participants. Some participants prefer to prepare their comments "off-line" and to insert them in the document when they are happy with them. Others make their comments visible to all as they are in the process of editing them. Some participants chat together through the chat service offered by the system to discuss the details of the contributions during the editing sessions. When the conference ends, the conference work group could continue as an Internet work group, or otherwise users could transfer some information to their Internet private work groups."

Analyzing this scenario we selected a set of plug-ins to implement in order to put POPEYE in practice in the context of this case study. The most important plug-ins we implemented are presented in the following.

File sharing plug-in

The "Data Management and Sharing Services" module of POPEYE (see Sect. 3.3) provides a virtual common space for each of the Workspaces where users (by means of plug-ins) and other internal modules may share files and objects. The contents of each of these shared spaces are distributed within the different devices in the network so single peers do not need to maintain locally a copy of all the shared items. However, all the users taking part in the same shared space, maintain a global view and do not need to know the physical location of the files. The contents of the shared spaces follow a tree structure so that users can create directories and sub-directories to better organize their files. In addition, the system stores certain attributes (called metadata) associated to the data. While some of these attributes can be modified by the users (e.g., author, description), some others are set by the

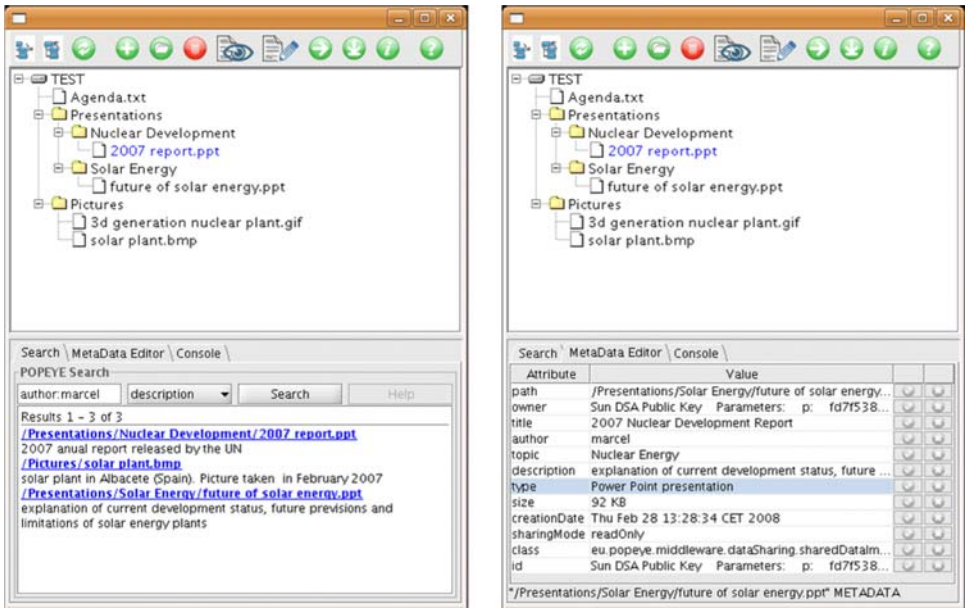

Fig. 5 File sharing plug-in

system (e.g., creation date). In either case, users can use them to perform queries which allow them to identify documents they are interested in. These search mechanisms are provided by the "Data Search Mechanisms". The File Sharing plug-in is a graphical application providing access to all the services provided by the "Data Management and Sharing Services Module" and the "Data Search Mechanism". Figure 5 shows the file sharing plug-in.

Messaging plug-in

The POPEYE Messaging plug-in allows all members inside a group to communicate by exchanging group and private messages. The plug-in graphical interface is similar to a chat room. In first place, all the members of the group who have launched the plug-in are listed in the membership list. Besides, messages sent to the scope of the whole group (i.e., sent to all the members of the group that launched the plug-in) are shown in the main board, common to all participants. Furthermore, by selecting one participant from the membership list, it is also possible to send a private message, so only the sender and the receiver can follow (see) the conversation.

The POPEYE Messaging plug-in demonstrates basic functionalities provided by communication services. First, in order to achieve group communication, the plug-in creates a new communication channel. Only the members of the group that created this communication channel will receive messages from the channel. Therefore, two of the most important functionalities of the channel are used:

- *sendGroup:* the message is received by all members;
- *send:* the message is received only by one member.

Messages are handled by each node considering whether they are group or private messages, so they are shown in the main board or in a separate window, respectively. Besides, each member advertises himself when entering the chat room, so the other members can update the membership list. Apart from this, the messaging plug-in also allows encrypted communication to ensure message confidentiality. If encryption is needed, the plug-in retrieves the Workspace shared key to encrypt the messages. Therefore, no person outside the Workspace or the POPEYE application is able to intercept messages and process them.

Shared presentation plug-in

Within collaborative environments, it is very common to perform talks and presentations using a set of slides in order to reinforce the information given by the speaker. To do that, a slide projector would be necessary. This fact collides strongly with the spirit of POPEYE, where spontaneous networks can be created without any previously established infrastructure, not even a slide projector. To be even more strict, imagine the situation where an event is held in the open air: we would not even have a wall where slides could be projected! Taking these facts into account, we implemented a plug-in allowing the management and visualization of presentations. We called it *PDFViewer*, due to the format of the supported files. In brief, with this plug-in, the speaker controls the visualization of his presentation on the audience devices, i.e., he decides which file is to be presented and the page number to be displayed at any moment. The audience devices are merely passive and follow all the orders received from the coordinator.

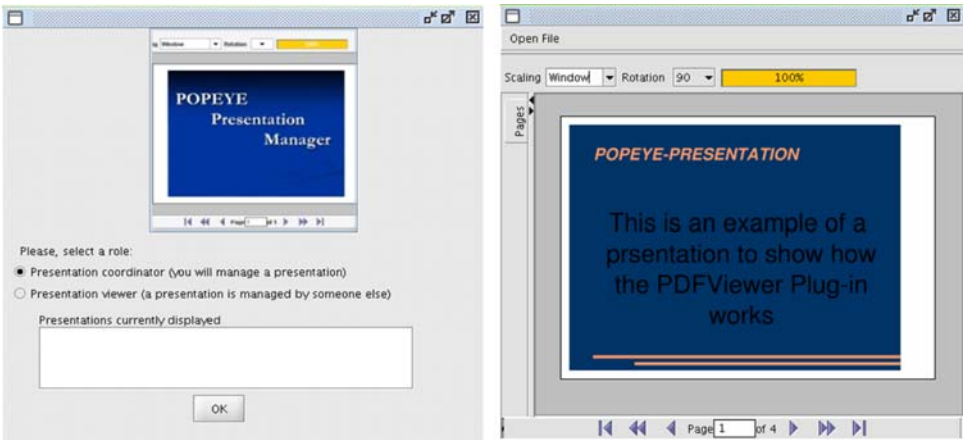

Fig. 6 PDF viewer plug-in: initial window and coordination mode window

This plug-in makes uses of the following POPEYE modules:

- Naming Service is used to announce the name of the presentation and the identity of its coordinator;
- Data Management and Sharing Services are used to share the displayed presentation so it is available to all the devices taking part;
- Communication Services are used to communicate the changes in the page number currently displayed.

Figure 6 shows the PDF Viewer plug-in: initial window and coordination mode window.

Voting plug-in

The voting plug-in implements a simple voting protocol to determine the preference of members of a Workspace on some subject. It allows any member to define a poll, with a question and some options. All the other members can then either choose one of the given options or "dismiss the issue", signalling they do not care. Results and some statistics are collected by the proponent of the poll, who can then publish aggregated results to all the other members. The proponent may also choose at one point to stop the poll, meaning no more answers will be accepted.

The voting plug-in makes use of both group multicast messages and unicast messages in the various stages of the poll. The poll proposal and results publishing is done in multicast by the poll proponent, while single preferences are sent in unicast mode to the proponent. This allows for some weak anonymity scheme, as only aggregated results are spread around. However, this plug-in has only demonstrational purposes and is by no means adequate to real voting.

Forum plug-in

The forum plug-in is a means of sharing information and opinions in a threaded, consistent way such as it is done on a Web forum. A Workspace member can browse in the hierarchical data, write, read and reply in a threaded way. This plug-in makes use of data sharing, keeping thus persistence of data within the Workspace lifespan.

Collaborative map plug-in

The Collaborative Map plug-in (CoMa) provides a map to allow support in orientation based tasks for members of Workspaces especially appropriate for larger outdoor environments. The map has different visualisation layers for presentation of different types of information. First of all, the current position of the user is gathered from a position receiver and shown on a geo-referenced map. Furthermore, the current user's position is sent to the other Workspace members to enable them to see the position on their device. The position data is currently based on GNSS (Global Navigation Satellite System) signals. In the future, this positioning mechanism can be replaced or complemented by other systems such as indoor positioning (e.g., via RFID-tags or WLAN signal strength).

The map layer and the user position layer are displayed as well as the positions of other members within the user's Workspace and specific landmarks or points of interest (POIs), that can be defined and sent by the Workspace members. This allows, for example, members to define meeting points and show them to others. By clicking on the symbols on the map, the user can get further information about them (e.g., see the profile of a user or a description of a POI) or can send short text messages to other members. Figure 7 shows the collaborative map plug-in.

5.1 Simulating the conference scenario

In order to show what happen in practice when using POPEYE, similarly to what happened in Glasgow, Scotland (April 1, 2008) during the 6th edition of the Minema

Fig. 7 Collaborative map plug-in: users in workspace and a meeting point displayed on a map

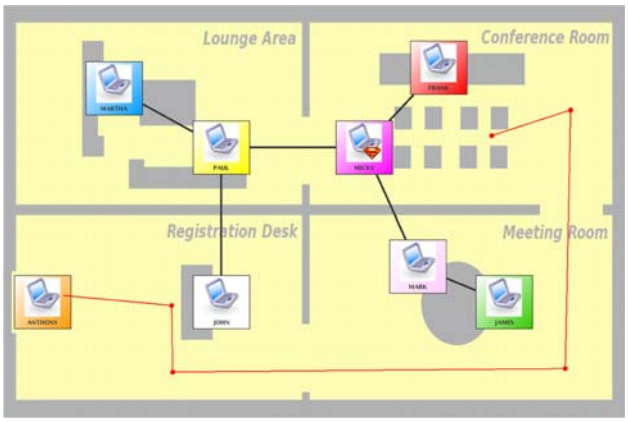

Fig. 8 Simulation tool: the mobility model

workshop,[13] we make use of a simulation tool that allows the user to choose a configuration and to consequently trigger a network re-configuration. This simulation tool[14] refers to real POPEYE peers that run on real machines and is used to simulate the peer mobility.

The Minema demonstration was organized in an eight nodes scenario. The initial configuration of the network was as shown in Fig. 8.

This scenario is organized in order to make a node move through 3 rooms in meeting scenario. The node *ANTHONY* enters the Registration Desk room; after making the registration he walks through the door to the Meeting Room. In the Meeting Room he says hello to his colleagues *MARK* and *JAMES*. Afterwards he goes to attend *FRANK*'s conference in the Conference Room. The links between him and the other nodes changes as follows: first, when he enters to the Registration Desk he approaches *JOHN*, the person in charge of the registration, therefore he connects to him. Then, when he gets to the Meeting Room he loses the direct link with *JOHN*, and establishes direct connection with *MARK* and *JAMES*. Finally, when he moves to the Conference Room he disconnects from *MARK* and *JAMES* but he links to *FRANK*. Table 1 summarizes the network modifications during the execution of the scenario.

In this case, the scenario uses the user defined mobility model so we can see the path. The node follows the red lines path shown in Fig. 8.

The user can also set a different path just clicking on the scenario. When the user clicks a red point is displayed and the node moves to it when the mobility is started. If more points are added straight lines are materialized between one point and the next one. When the mobility is started, the node follows the created path. The user can create or delete links between nodes, add nodes (when a new node is added to the scenario, a new entry on the table located on the right is created. From this table, user can specify the node IP and MAC addresses), change node names, save the network configuration, load a previously saved topology, check the accessibility of a node (all the peers that are accessible are displayed in their original colour and the ones not accessible, in red), etc.

Conference attendants located in the same conference room may form a cluster. In fact, they will probably collaborate in the same Workspaces. Nevertheless, collaboration with attendants in different clusters is still possible using the correspondent super-peers. Note that the super-peer architecture is self-adjusting and dynamic. Figure 9 shows a mobility model with nodes distributed in 5 clusters into the simulation tool.

6 Related work

The aim of this section is to discuss POPEYE-related work. We first present frameworks similar to POPEYE, and we then detail related work for the most innovative parts of POPEYE: Network Abstraction (Sect. 6.1), Security aspects (Sect. 6.2), and Context management (Sect. 6.3).

The WORKPAD (An Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios) project[15] aims

[13] See webpage: http://www.cs.kuleuven.be/conference/minema2008/demos.html for further details.

[14] The simulation tool is an open source project that has been released at Source Forge under a LGPL license. The source code as well as some screenshots and documentation is available at http://sourceforge.net/projects/manetconfig.

[15] http://www.workpad-project.eu.

Table 1 Summary of the network modifications

Step	Current links	Created links	Deleted links
1	JOHN	–	–
2	JOHN	–	–
3	MARK, JAMES	MARK, JAMES	JOHN
4	FRANK	FRANK	MARK, JAMES
5	FRANK, MICKY	MICKY	–

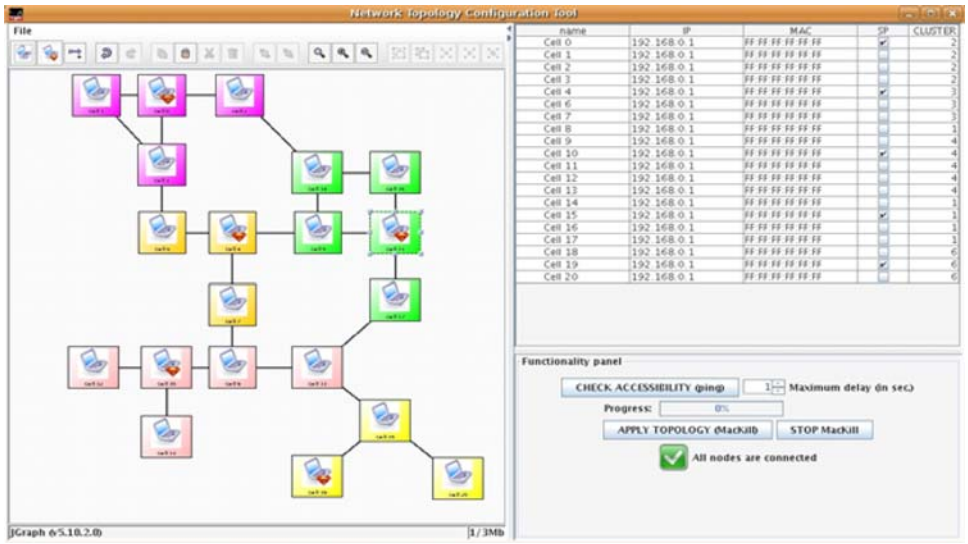

Fig. 9 Mobility model with nodes distributed in five clusters

at developing an innovative software infrastructure for supporting collaborative work of human operators in emergency scenarios. Even if this project is strongly related to POPEYE, WORKPAD works in higher level layers (such as BPM, workflows) and could benefit from the POPEYE framework. Moreover this project seems more focused on collaborative environments in which each team member is equipped with handheld mobile devices (PDAs) and in which the adaptiveness to connection/task anomalies is fundamental.

In [29] authors present a framework for collaborative working environments which consists of four layers: resource layer, middleware layer, upperware layer, and application layer. Similarly to POPEYE, the upperware contains plug/play facilities for adding new functionalities. Differently from POPEYE, which is specialized in ad hoc and spontaneous networks, the upperware layer contains also intelligent agents for coordinating and controlling multiple middleware techniques (such as Grid computing, Web-services, etc.).

In [22] authors aim to propose a generic collaborative working environment. This work is part of the ECOSPACE project.[16] Particular attention is paid to interoperability purposes, i.e., users should not be aware of different technologies which underlie the integrated platform. This paper proposes a layered architecture based on Service Oriented Architecture (SOA), and more concretely in the use of web services. The architecture is divided in the following three layers: Services,composite collaborative services and Applications. The services, in the lower layer, are the ground for the composite collaboration services that follow some useful interaction patterns defined by the use of business process languages. The POPEYE software architecture was somehow influenced by the software architecture defined in ECOSPACE.

The main objective of the CASCOM project[17] is to implement, validate, and trial a value-added supportive infrastructure for semantic web based business application services across mobile and fixed networks. Thus, CASCOM is not focussed on collaborative working environments but some aspects of P2P infrastructures and related security aspects could be somehow related. In the following we detail related work for each aspect of POPEYE and we discuss these aspects.

6.1 Network abstraction

Several protocols have been proposed to address routing in mobile ad hoc networks. They are mainly classified as unicast

[16] ECOSPACE "Integrated Project on e-Professional Collaboration Space" (FP6 IST- 035208).

[17] http://www.ist-cascom.org.

and multicast protocols with regard to the number of intended receivers. Unicast protocols can follow a reactive or a proactive routing approach. DSR, AODV [1], and DYMO [2] are examples of reactive unicast routing protocols. Reactive protocols find routes on demand, by flooding the network with route request packets. On the other hand, proactive protocols like DSDV, WRP and OLSR, maintain updated routes by distributing routing tables periodically. There are many unicast routing protocols in the literature, but currently, the IETF MANET working group is considering for standardization a reactive protocol (DYMO) and a proactive protocol (OLSRv2 [10]).

Multicast routing protocols for MANET are mainly classified in mesh and tree-based protocols, depending on how they propagate data. Whereas mesh-based protocols may have various paths between any source and receiver pair, tree-based approaches consider only a single path. ODMRP is an example of a reactive mesh-based protocol. MMARP [28], the multicast protocol that we use in POPEYE, is a mesh-based protocol as well, and its distribution structure is similar to the one used by ODMRP. Examples of tree-based approaches include MAODV and MOLSR, based on AODV and OLSR unicast protocols, respectively.

Whereas MANET routing protocols are essential to tackle multi-hop communication, during recent years several peer-to-peer layers have been designed to be deployed on top of a mobile ad hoc network in order to benefit from the similarities between MANETs and peer-to-peer networks. These overlay layers are usually based on successful Internet systems and are ported or adapted to fit MANET requirements. For instance, some unstructured approaches have been proposed, like BitTorrent for MANETs [27] or XL-Gnutella [11]. The latter proposes a cross layer adaptation of Gnutella that facilitates the interaction between overlay peers and the underlying routing protocol by means of an event system. However, these systems are focused on file sharing and do not provide a good basis to develop collaborative applications. Likewise, JMobiPeer [7] is a JXTA compatible framework designed for J2ME CLDC environments. JXTA is the most mature P2P framework and provides interoperability and platform independence, allowing connection between heterogeneous devices. Hence, JMobiPeer benefits from these characteristics and introduces new features like a routing layer and emulation of the multicast functionality to adapt JXTA to mobile environments. Nevertheless, JXTA architecture does not take into account locality and communication may incur in adding high overhead due to XML-constructed messages.

6.2 Security aspects

The missing support from any infrastructure and spontaneous nature of wireless ad hoc networks brings along a number of security challenges. The network nodes and networking equipment are not controlled by one administrator or organization. The presence (and especially the permanent availability) of a generally trusted third party, like a certification authority, cannot be assumed. Theoretically, mutually unknown nodes may spontaneously join and collaborate in one network. In practice, this statement can be mitigated: the participants of a conference, for instance, have to physically register at the beginning and may be issued certificates for their devices during this administrative proceeding. Therefore, a node's legitimacy to participate in a network or an application may be verifiable. Complementary mechanisms as trust computation, security framework for group management and access control mechanisms, and distributed intrusion detection allow one to secure both the functionalities and the messages exchanged within a wireless ad hoc network.

Trust computation is based on normalized trust metrics locally maintained in every node for its neighbors or interaction partners. Recommendations from partly trusted third parties may either be used to initialize the trust metric for an unknown node, or be considered for the continuous updates of that value. In a transient trust model, trust values are assigned to direct neighbors in a "trust graph" and are calculated along all possible paths and averaged, which implies a complete view of the network [30]. In a community trust model, trust values are computed with the common history, or common knowledge of the group [20].

Security frameworks for MANET comprise systems using a trust metric, and adaptations of classical PKI architectures. They generally aim at the creation of a group of trusted nodes or a trust domain, in which nodes will mutually rely on the networking function offered by their peers and may engage in collaborative applications. In centralized schemes [19], all users or nodes have to be authorized by a central entity on initialization. Therefore, users cannot generally join groups or obtain access rights during system runtime, or need to contact the central entity. In hybrid schemes [6], initialization is also done by a central entity authorizing all nodes or users present at system start, but newcomers are able to join with full rights without immediately contacting the central entity. In decentralized schemes [16], if a central entity may be required to communicate with a small number of nodes during initialization, all other operations work without this entity.

In MANETs, intrusion detection is essentially done locally, since each node has to protect itself. This local view may be refined with information from other nodes. However, an IDS relying on wrong information from other nodes may open new vulnerabilities (especially for DoS attacks). Different detection modes have been proposed: misuse, anomaly and specification-based detection [17]. The first scans for patterns of known attacks, the second regards interactions more in general, and the third only allows well-defined exchanges. The information exchanged among different nodes may also

take different forms: from pure broadcasts of the assumed probability that an attack is under way over mobile agents to a closely synchronized distributed system establishing the overall system state.

6.3 Context management

P2P networks are gaining momentum as frameworks for collaborative, context-aware applications and services. Some examples include Edutella [24], Groove[14], COMPASS [31] or SpeakEasy [13]. As they become more widely adopted and demanded, a need arises for adding the advantages of context awareness to such collaborative systems in order to offer a faster, more efficient and personalized service to the user, independently of the resources, devices and underlying technologies used. In the frame of context-awareness, SpeakEasy [13] proposes a collaborative P2P system, with resource, network and content sharing as the main goal. It defines a common set of interfaces each component must fulfil in order to communicate in an homogeneous way with other components in the network. Communication is reached by means of transmitting mobile code to components willing to access the services of another component. In order to reduce network traffic, our approach is based in the transmission of homogeneously formatted contextual information, instead of having to transmit all application code needed to access a device. Besides, in our approach all relevant information is treated as contextual information, not just location and metadata, thus giving chance to a deeper and more rich reasoning process. Chen [9] uses context in a collaborative Filtering e-commerce tool, in which context awareness is used to predict the user's preference, not only from opinions of similar users, but also from feedback of other users in a context similar to that the user currently is in. In this case, the context awareness is limited to the information regarding the actual state of the user, instead of the complex aggregation of historical context data that conforms the real context situation of a user. A similar approach is taken in COMPASS [31]. This time, context is treated homogeneously as all relevant information that can be used to characterize the situation of an entity. The idea is to offer to a user a set of services based on his preferences and in the previous selections and actions of users with similar context. While user's preferences are used as hard criteria, the context of the user is used as a soft criteria. COMPASS works also with a concrete ontology suited for its application domain. Yang [33] proposes a collaborative working environment where context-awareness is employed to set a learning system, with consumers (learners) and producers (knowledge providers) of information. Each collaboration is aimed at this learning process. It uses a concrete OWL ontology, and receives information directly from the user in the way of forms or by context detection and extraction. While the Yang approach has a very solid context collection mechanism, context is limited to some social data and to location, so context is not treated in an homogeneous way, and it is also employed in a concrete scenario.

When giving a collaborative wrapping to a context aware platform, we must be aware of the conditions for a full collaboration (different from mere group work), and design the context information such as collaboration is encouraged and risks avoided. Biström [8] identifies these collaboration issues. In P2P applications, this implies setting the right scope for users to collaborate amongst them, and this must be reflected in contextual information. The conclusions seem to indicate that a proper information structure and metadata is mandatory to produce the infrastructure of a peer-to-peer network that needs to make the information useful for the users. Unlike the above described approaches have a flexible but homogeneous meta-ontology on the bottom of our design. Therefore, we are able to build domain-specific ontologies that can suit concrete scenarios, without imposing a specific data model to applications.

7 Conclusion

In this paper we presented POPEYE, a Specific Targeted Research Project (STREP) European project that focuses on supporting peer to peer collaborative working environments over mobile ad hoc networks. POPEYE aims to give a solution on how to provide support for ad hoc cooperation, with the appropriate quality of service, in situations where the fixed network infrastructure is absent or cannot be used.

The first implementation of POPEYE available as proof-of-concept application has been used as the basis of two demonstration events.

After two years of joint development, the POPEYE consortium has developed an integrated framework for P2P collaboration in MANETs. The POPEYE project has been successful and we developed a running prototype. The prototype is available as open source in Sourceforge.[18] Furthermore, POPEYE software was demonstrated in two live demonstration events. The first one of these events took place in Paris on October 26, 2007 and was open to a restricted group of users (the POPEYE User Group) with the goal of testing and validating the POPEYE framework in a realistic situation. The second event during which POPEYE was demonstrated was a more open, public showcase of POPEYE technologies set up in the context of a project open Workshop organized in conjunction with the International MiNEMA event[19] held in Glasgow on April 1, 2008. The goal of the second demonstration of the POPEYE system was the dissemination of the project results to the wider IST, scientific

[18] http://sourceforge.net/projects/popeye-cwe.

[19] http://www.cs.kuleuven.be/conference/minema2008/demos.html.

and technical communities. Both demonstration events were a success. The results of both the demonstration events are described in this paper and further details are collected into two proceedings documents publicly available on the project web site: http://www.ist-popeye.eu.

The major drawback of the prototype is that we require two MANET transport protocols (DYMO, MMARP) that currently only work in the Linux platform. To solve this portability issue and ease configuration management we began a research line on Application Layer MANET protocols. In this line, we developed a Java implementation of the OLSR protocol (jOLSR) and an optimized overlay multicast over jOLSR named OMOLSR. The overall idea for a future work would be to replace POPEYE's network abstraction layer with this Application layer protocols. This would enable the portability of POPEYE to any platform with a Java Virtual Machine and would also reduce the configuration complexity of the system.[20]

Acknowledgments This work was supported by the POPEYE project: Peer to Peer Collaborative Working Environments over Mobile Ad Hoc Networks. POPEYE is part-funded by the EU under the 6th Framework Program, IST priority Contract No. IST-2006-034241 (http://www.ist-popeye.org). We would like to acknowledge each member of the POPEYE consortium.

Appendix: POPEYE glossary

The purpose of this glossary is to enable a common understanding of terms used in the paper and in the POPEYE project.

- *Authentication, authorization:* http://en.wikipedia.org/wiki/Authentication.
- *Core Services:* Services that support any collaborative working application by providing the necessary infrastructure to share data and context information with users, applications and internal modules. In the POPEYE middleware, these services are implemented by the modules "Data Management and Sharing Services", "Workspace Management", "User Management", "Plugin Management", "Plugin API", "Context Collection Services", "Context Delivery Services" and "Context Management Services".
- *Group:* A group of peers that corresponds to the group of users who are assigned as members to a Workspace.
- *Knock:* The POPEYE-framework provides a programming interface that enables a Plug-in to knock. Which means, the Plug-in writes an entry into a log file and, corresponding to the importance of the entry, tries to direct the users attention towards this entry. Example: When a POPEYE-user is invited to a WS, there is a new entry inserted into his knock-log and there might be a knock-symbol flashing at the bottom of the screen and the loudspeaker might produce knocking sounds.
- *Partner:* Any of the seven Members of the POPEYE Consortium.
- *Plug-in:* A piece of software that is designed to be plugged into the POPEYE-Framework. A Plug-in provides a specific tool for collaborative work like for example: file sharing, videoconferencing, etc. Each Plug-in may be considered as "application".
- *POPEYE framework:* The idea is to design POPEYE as a kind of extensible core system which can be expanded by specific collaborative tools (see plug-in). This core system provides the means to handle groups, sessions, users, members, etc, but it does not provide elaborated tools for specialized collaborative work (like emergency rescue coordination or whiteboard teaching).
- *POPEYE user:* A person who has installed and started POPEYE-software on at least one of his computers.
- *Session:* An instance of a Plug-in inside a WS. It may own a specific configuration.
  - *Example 1:* Harald is member of the tulip lover WS. He creates a file sharing Session and puts all his tulip photographs into the shared file system.
  - *Example 2:* Harald and John and some others are members of a WS named "Rapid Illustrators". The members of this Workspace are interested in illustrators who are very quick at drafting. Today there is a contest between Harald and John. For this purpose there are four Sessions inside this WS:
    - (a) An Instant messaging Session where a referee gives the subject for the drawing;
    - (b) and (c) Two whiteboarding Sessions: Harald (as well as John) has his own whiteboarding session which he uses to show his drawing to the other WS-members who have joined this session;
    - (d) A voting Session where the members can elect the winner.
- *Session-creator:* A Session-creator is a WS-member who created a session. Session-creator are allowed to change the Session's profile and to terminate the Session.
- *Session-member:* A WS-member who has joined a Session.
- *Session-manager:* A session-member that has the additional right to change the session's profile and to terminate the session. There can be more than one Session-manager for a specific session. A session-manager might be a workspace-manager of the containing workspace but he doesn't have to.
- *Workspace (WS):* A Workspace comprises a Group (group membership and access control), shared data and tools (Plug-ins associated to Sessions).

[20] See: http://ast-deim.urv.cat/wiki/OMOLSR for more information.

- *WS-creator:* A POPEYE-user who creates new WS. He also becomes the first WS-manager.
- *WS-manager:* A WS-manager is a WS-member who is allowed to change the WS' profile, terminate the WS, exclude WS-members, give manager-rights to WS-members, and invite POPEYE-users (might be allowed for all WS-members). A WS' creator becomes the first WS-manager of that WS. It is possible that each WS-member is also a WS-manager.
- *WS-member:* A POPEYE-user who has joined a WS.

References

1. Ad hoc On Demand Distance Vector (AODV). http://moment.cs.ucsb.edu/AODV/aodv.html
2. Dynamic MANET On-demand Routing Protocol (DYMO). http://moment.cs.ucsb.edu/dymo
3. Model-View-Controller (MVC). http://java.sun.com/blueprints/patterns/MVC.html
4. POPEYE websites. The home page of the POPEYE project is http://www.ist-popeye.eu, while the source code can be found in http://sourceforge.net/projects/popeye-cwe
5. POPEYE deliverable D2.2 Description of Functional, non-Functional and Technical Requirements. http://www.ist-popeye.org/, 2006
6. Almenarez F, Carbonell M, Forne J, Hinarejos F, Lacoste M, Marin A, Montenegro JA (2005) Design of an Enhanced PKI for Ubiquitous Networks. In: Proceeding of the sixteenth international workshop on database and expert systems applications, Los Alamitos. IEEE Computer Society, pp 262–266
7. Bisignano M, Modica GD, Tomarchio O (2005) A JXTA compliant framework for mobile handheld devices in ad-hoc networks. In: ISCC, Murcia, Spain
8. Biström J (2005) Peer-to-peer networks as collaborative learning environments. Paper presented at HUT T-110.551 Seminar on Internetworking, retrieved October 25
9. Chen A (2005) Context-aware collaborative filtering system: predicting the user's preferences in ubiquitous computing. In: CHI'05, New York, NY, USA
10. Clausen T, Dearlove C, Jacquet P (2007) The Optimized Link-State Routing Protocol version 2, February 2007. Internet-Draft, draft-clausen-manet-olsrv2-03.txt, work in progress
11. Conti M, Gregori E, Turi G (2005) A cross-layer optimization of gnutella for mobile ad hoc networks. In: Proceedings of the MobiHoc05, Illinois, USA
12. Drira K (2000) A coordination middleware for collaborative component-oriented distributed applications. Netnomics 2(2):85–99
13. Edwards W, Newman M, Sedivy J, Smith T, Balfanz D, Smetters D, Wong H, Izadi S (2002) Using speakeasy for ad hoc peer-to-peer collaboration. In: Proceedings of ACM computer supported cooperative work (CSCW 2002), New Orleans, Louisiana, USA
14. Eikemeier C, Lechner U (2004) Peer-to-peer and group collaboration—do they always match? In: WETICE '04, Washington, USA. IEEE Computer Society, pp 101–106
15. Galera FJ, Martinez JA, Sanchis MA, Gomez-Skarmeta AF (2008) Design of a cluster-based peer to peer architecture for manets. In: SCCC'08
16. Gray E, O'Connell P, Jensen C, Weber S, Seigneur J, Yong C (2002) Towards a framework for assessing trust-based admission control in collaborative ad hoc applications, Technical Report, vol 66. Department of Computer Science, Trinity College Dublin
17. Inverardi P, Mostarda L (2005) A distributed intrusion detection approach for secure software architecture. In: Proceedings of the European workshop on software architecture (EWSA 2005), Pisa, Italy. Springer, Berlin, p 168, 184
18. Botia J, Ha Duong H, Demeure I, Gómez-Skarmeta A (2008) A context-aware data sharing service over MANET to enable spontaneous collaboration. In: Proceedings of the 6th international workshop on distributed and mobile collaboration (DMC 2008), WETICE. Rome, Italy, 24 June, 2008
19. Kagal L, Finin T, Joshi A (2001) Trust-based security in pervasive computing environments. Computer 34(12):154–157
20. Legrand V, Hooshmand D, Ubda S (2003) Trusted ambient community for self-securing hybrid networks. Technical report, no. 5027, INRIA Rhne-Alpes, France
21. Martinez JA, Galera FJ, Sanchis MA, Gomez-Skarmeta AF, A cluster-based framework for spontaneous collaboration without infrastructure. In: CCNC'09
22. Martfnez-Carreras MA, Ruiz-Martfnez A, Gomez-Skarmeta AF (2007) Designing a generic collaborative working environment. In: IEEE International Conference on Web Services (ICWS 2007)
23. McKinley PK, Malenfant AM, Arango JM (1999) Pavilion: a middleware framework for collaborative web-based applications. In: Proceedings of the international ACM SIGGROUP conference on Supporting group work (GROUP '99), New York, NY, USA. ACM Press, pp 179–188
24. Nejdl W, Wolf B, Qu C, Decker S, Sintek M, Naeve A, Nilsson M, Palmr M, Risch T (2002) EDUTELLA: a P2P networking infrastructure based on RDF. In: Proceedings of the eleventh International World Wide Web Conference (WWW2002), pp 604–615, Honolulu, Hawaii, USA
25. Nieto I, Botia J, Gomez-Skarmeta A (2006) Information and hybrid architecture model of the OCP contextual information management system. J Univers Comput Sci 12(3):357–366
26. Pelliccione P, Inverardi P, Muccini H (2008) CHARMY: A framework for designing and verifying architectural specifications. IEEE Transactions on software engineering. IEEE computer society, 29 December 2008. doi:10.1109/TSE.2008.104
27. Rajagopalan S, Shen C-C (2006) A cross-layer, decentralized bittorrent for mobile ad hoc networks. In: Proceedings of MOBIQUITOUS 2006, San Jose, USA
28. Ruiz P, Gomez-Skarmeta A, Groves I (2002) Multicast routing for MANET extensions to IP access networks: the MMARP protocol. In: Proceeding of the international workshop on mobile IP-based network developments, pp 75–81, October 2002
29. Su D, Xiong Y, Zheng Y, Ji S (2008) A framework for collaborative working environments. Int J Prod Res 46(9):2363–2379
30. Theodorakopoulos G, Baras JS (2004) Trust evaluation in ad hoc networks. In: Proceedings of the 2004 ACM workshop on wireless security (WiSe'04), pp 1–10, New York, NY, USA. ACM Press
31. van Setten M, Pokraev S, Koolwaaij J (2004) Context-Aware Recommendations in the Mobile Tourist Application COMPASS. In: Proceedings of the interantional conference adaptive hypermedia (AH 2004), pp 235–244
32. von Laszewski G, Foster I, Gawor J (2000) Cog kits: a bridge between commodity distributed computing and high-performance grids. In: JAVA'00: Proceedings of the ACM 2000 conference on Java Grande, pp 97–106, New York, NY, USA. ACM
33. Yang SJH (2006) Context aware ubiquitous learning environments for peer-to-peer collaborative learning. Educ Technol Soc 9(1):188–201