Journal of Universal Computer Science, vol. 14, no. 1 (2008), 88-122
submitted: 6/11/06, accepted: 27/7/07, appeared: 1/1/08 © *J.UCS*

Integrating Service-Oriented Mobile Units to Support Collaboration in Ad-hoc Scenarios

Andrés Neyem
(Department of Computer Science, Universidad de Chile, Chile
aneyem@dcc.uchile.cl)

Sergio F. Ochoa
(Department of Computer Science, Universidad de Chile, Chile
sochoa@dcc.uchile.cl)

José A. Pino
(Department of Computer Science, Universidad de Chile, Chile
jpino@dcc.uchile.cl)

Abstract: Advances in wireless communication and mobile computing extend collaboration scenarios. Mobile workers using computing devices are currently able to collaborate in order to carry out productive, educational or social activities. Typically, collaborative applications intended to support mobile workers involve some type of centralized data or services, because they are designed to work on infrastructure supported wireless networks. This centralization constrains the collaboration capabilities in ad-hoc communication cases. This paper introduces the concept of Service-Oriented Mobile Unit (SOMU) in order to reduce such limitation. SOMU is an autonomous software infrastructure running on a computing device; it is able to be integrated to ad-hoc networks and it can interoperate with other mobile units in ad-hoc collaboration scenarios. In addition, the paper presents the challenges faced when designing and implementing the SOMU platform. It also describes an application developed on SOMU.

Keywords: Service-Oriented Mobile Units, Web Services Platform, Middleware for Mobile Groupware, Ad-hoc Collaboration Scenarios.
Categories: C.3, C.5, J.7.

1 Introduction

Fast development in the area of information and communication technology and especially in broadband Internet access and mobile computing has changed the established ways of communication, learning, entertainment and work in professional and private lives. Mobile computing devices and wireless communication capabilities have become useful to support mobile work anywhere. Examples of suitable places are parks, coffee shops, shopping malls, airports, universities, schools, hospitals, administrative offices and factories. Particularly, these technologies have allowed workers to labor outside of the office and accomplish their activities while they are on the move. A mobile worker is often conceived as a person executing tasks anywhere and anytime, using mobile computing devices with wireless communication capabilities.

Mobile workers require capabilities for data synchronization and collaboration with other people. They have frequently some uncertainty about which will be the next collaboration scenario and its characteristics as well. Therefore, they need autonomous, flexible and interoperable collaborative solutions independently of the availability of centralized resources or communication infrastructure. When two or more mobile workers meet, their physical location should not be a limitation to collaborate.

Collaboration activities involving mobile workers can be supported by mobile networks, also called MANETs (Mobile Ad-hoc NETworks) [Aldunate et al. 2006b]. However, solutions using MANETs have to be designed considering the features of these networks, such as small communication range, dynamic topology and high disconnection rate [De Rosa et al. 2005].

Currently, most collaborative applications intended to support mobile workers involve some type of centralized data or services because they are designed to work on infrastructure supported wireless networks (access points). Typically this centralization constrains the collaboration possibilities in ad-hoc communication scenarios [Buszko et al. 2001, Aldunate et al. 2006a]. In order to reduce such limitation, this paper introduces the concept of Service-Oriented Mobile Unit (SOMU). SOMU is an autonomous software infrastructure running on portable computing devices. It can be integrated to ad-hoc networks and it may interoperate with other mobile units through standardized services. Each SOMU is able to provide and consume services from/to others units and it does not depend on centralized data or services to support mobile workers' activities. Collaborative mobile applications developed on this middleware inherit the SOMU capability for interacting among them almost in any communication setup.

Next section presents two application scenarios and introduces the main challenges to be met when designing a solution to support mobile ad-hoc collaboration. Section 3 presents and discusses related work. Section 4 describes the design decisions made to deal with the requirements of these collaborative applications. Section 5 describes the way to overcome the stated challenges with SOMUs. Section 6 describes an application scenario where the ad-hoc collaboration was supported by an application developed on SOMU. Section 7 presents the conclusions.

2 Requirements for Mobile Ad-hoc Collaboration

There is a large variety of work scenarios where mobile ad-hoc collaboration can be supported [Aldunate et al. 2006b]. This section characterizes a work setting where SOMU is advantageous to support collaboration. Such setting is very demanding in terms of services required from collaborative applications. In order to illustrate the main requirements involved in mobile ad-hoc collaboration, two work scenarios are briefly described below.

- *Disaster Relief:* Activities to resist and recover from natural, hazardous and intentional eXtreme Events (XE) are highly dynamic and demand effective collaboration from a broad range of organizations (police, firefighters, medical

personnel and government agencies). Typically, there are minimal communication services available in this collaboration scenario supporting just voice messages delivery (e.g. a couple of radio channels) [Comfort 2004]. However, collaboration among first responders (mobile workers) is highly required and it needs support for voice and data communication [Ochoa et al. 2007]. These mobile workers need to know basic information about the site and affected buildings (e.g. maps, probable people locations and vulnerable points), exit routes, resources deployed in the area and tasks assignment. Mobile computing devices and MANETs could provide an important support to deliver this information in the affected area [Figure 1.a.].
On the other hand, these mobile workers need to be autonomous, interoperable and be able to have access to several types of shared information to perform the assigned activities. Sometimes they also have to update such information and communicate the updates to their partners, leaders and other organizations in order to support decision-making processes. Most of these capabilities have to be offered by the groupware solutions independently of the characteristics of the physical location where the first responders meet. It means that no centralized data or services should be required to support collaboration among them. Using such supporting technological infrastructure, Government authorities in charge of macro-decisions must be able to access information from the mobile workers, monitor the activities evolution, make decisions and deliver orders and information. Summarizing, in this scenario:

- Dispersed teams work in the affected area.
- Teams do not belong to the same organization.
- Mobile workers (first responders) record the advances and problems in the relief process.
- They are not able to use fixed communication infrastructure.
- They must communicate and share information.

- *Building and Construction:* Each construction site typically has a main contractor. The main contractor in turn outsources several parts of the construction project, e.g. electrical facilities, gas/water/communication networks, painting and architecture. Some of these sub-contracted companies work concurrently and they need to know the advances of each other to plan the execution of pending work. Moreover, all these companies should periodically report the advances to the main contractor; the contractor must coordinate the efforts of the companies. For example, electrical engineers (mobile employees) belonging to a company need to be on the move in order to inspect and record the status of the electrical facilities being developed by the company workers at a construction site. During the inspection, each engineer using a Tablet PC updates the information recording the current status of the electrical facilities [Figure 1.b]. After the inspection and before leaving the construction site, the engineers meet to share the data, review it and check agreement on the updated information. If they detect incomplete or contradictory data, some of them can re-inspect the facilities in order to solve such case. Before leaving the construction site, an electrical engineer shares the updated information with a main contractor's employee, who is in charge of tracking the construction project updates. Typically no wireless communication support is available at the

construction site; however it should not be a limitation for collaboration among them. Similar to the previous work scenario, the collaboration involves:

- Dispersed teams that work at the construction site.
- Teams do not belong to the same company.
- Inspectors (mobile workers) record the advances and problems in the contracted work.
- They are not able to use fixed communication infrastructure.
- They have to communicate and share information.

Figure 1.a: Use of mobile technology in disaster relief scenarios

Figure 1.b: Use of mobile technology in construction scenarios

In both scenarios, the mobile workers need autonomy, interoperability and they also need to be able to collaborate independently of the features of the physical locations. Next section presents the general requirements involved in a mobile application, and section 2.2 specifies those particularly relevant to support collaborative work.

2.1 General Requirements

The computation capability of the mobile computing devices is also an issue to consider when designing groupware solutions to support mobile ad-hoc collaboration. Next, these requirements are briefly explained.

- *Autonomy:* Collaborative mobile applications should work as autonomous solutions in terms of communication, data and functionality. Communication availability in the physical scenario and access to centralized shared data and services cannot be a limitation to support collaboration among these mobile workers. Therefore, self-configurable solutions able to work in peer-to-peer settings are required. These solutions have also to consider the changes in the collaboration context. On the other hand, these solutions should also be autonomous in terms of power supply consumption, however this issue is out of scope of this paper.
- *Interoperability*: Provided mobile workers may belong to various organizations and need to do casual or opportunistic collaboration, their collaborative mobile applications should offer communication, data and services interoperability.

Thus, the applications could interact among them although they were not designed specifically to work together. Similarly, notifications and information delivered by the network could be understood by the applications receiving such message.

- *Shared information availability:* Shared information supporting collaborative applications in these scenarios need to be highly replicated since there are frequent disconnections in wireless networks (even using access points). The groupware application has to keep the coherence of shared information in a peer-to-peer network. It must also synchronize the information when it is updated in parallel and asynchronously. Hence, when a mobile worker requests the most updated piece of shared information, he/she can recover the most updated available piece.

- *Variability of the work context:* Since mobile workers are on the move to carry out their activities, their work context can frequently change. Some attributes, such as MANET topology and the Internet/servers access, will change from one place to the next one. It means the communication and coordination mechanisms embedded into the groupware solution have to consider these context changes in order to provide an effective support for collaboration.

- *Use of hardware resources:* Collaborative mobile applications should operate, in many cases, with constrained hardware resources; e.g., the case in which these solutions need to run on Personal Data Assistants (PDAs). Storage and memory capacity, processing power, screen size, data input and battery life are the most important constraints.

2.2 Requirements for Mobile Collaboration

The issues described in section 2.1. represent basic requirements to deal with. They should be addressed regardless if the application is collaborative or not. Collaborative applications designed to work in these work scenarios have also additional requirements. These requirements are just those derived from the type of work to be supported (i.e. loosely coupled work [Pinelle and Gutwin 2006]) and the features of the work and activity contexts [Alarcon et al. 2006]. Such requirements are the following ones:

- *Discretionary Collaboration*. Loosely coupled mobility means that collaboration with others is (in most cases) not strictly required; instead, workers engage in collaboration when they decide that it is valuable to do so. Since the collaboration processes are sporadic, the team members do individual work most of the time [Pinelle and Gutwin 2006]. This type of collaboration requires session management capabilities.

- *On-demand information sharing*. Workers need to share and synchronize information on-demand [Ochoa et al. 2007, Pinelle and Gutwin 2006]. When information maintained by a worker is shared with the rest of the team, the sharing should be at the worker's judgment so that information can be selectively protected. This feature requires session management capabilities and awareness of users' availability.

- *On-demand information synchronization.* Since several mobile workers perform autonomous and parallel activities, they need instances for information synchronization. This synchronization process is typically on-demand, and it could be attended or unattended [Ochoa et al. 2007]. This capability requires users and/or roles support.
- *Low coordination cost.* Tasks are often strongly partitioned among workers. This partitioning minimizes coordination demands and it allows people to work autonomously and in parallel [Pinelle and Gutwin 2006]. Ideally, the coordination process should be unattended [Ochoa et al. 2007].
- *Awareness of users' reachability*. Mobile workers need to know when a particular user is reachable, because they do on-demand collaboration. Hence, awareness mechanisms indicating user reachability should be embedded in mobile groupware applications.

Other CSCW supporting mechanisms such as floor control and synchronous interactions do not make sense to be considered, due to the high disconnection rate of the communication network. This paper introduces SOMU to deal with most of the aforementioned requirements. SOMU is a fully distributed software piece (unit), which can provide and consume services from other units. The interaction among these mobile units is supported by a MANET. Each unit has been implemented as a middleware running on laptops and PDAs. Collaborative mobile applications developed on this middleware are then able to interact among them almost in any communication setup. Thus, mobile workers using such applications can collaborate when there is no stable communication support or even without any communication support at all.

3 Related Work

Several collaborative solutions have been proposed to support mobile workers in specific settings [André and Antunes 2004, Guerrero et al. 2004, Menchaca-Mendez et al. 2004, Muñoz et al. 2003, Zurita and Baloian 2005]. Although these proposals have shown to be useful to support specific collaborative activities, they were not designed as general solutions. Therefore, the capability to reuse these solutions in various work scenarios is relatively small.

On the other hand, there are several interesting initiatives in the middleware area, which propose reusable functions to support collaboration in peer-to-peer networks. One of them is LaCOLLA [Marques and Navarro 2006]. This middleware has a fully decentralized peer-to-peer architecture and provides general purpose functionalities for building collaborative applications. LaCOLLA works well in networks with important signal stability, such as wired or fixed wireless (one-hop) networks. Notice the work scenario for the current paper requires similar functions but for ad-hoc wireless (multi-hop) networks. Furthermore, LaCOLLA is not able to run on PDAs.

Unlike LaCOLLA, the iClouds framework offers spontaneous mobile user interaction and file exchange support in mobile ad-hoc networks [Heinemann et al. 2003]. This framework also provides independence of a server doing a full replication of any shared file, which is appropriate for Mobile Ad-hoc NETworks (MANET).

However, it does not provide support to exchange shared objects, just files. Furthermore, iClouds does not distinguish among copies of the same shared file (e.g. master and slave copies) and it does not support distributed operations on those files either. The functions provided by iClouds are focused just on data sharing.

There are frameworks providing specific functionalities to support mobile collaboration through an API, such as YCab [Buszko et al. 2001]. These frameworks implement their own protocol and they provide the following generic services: session management, text chatting, image view, GPS and client information. Probably, the most popular framework to support peer-to-peer collaboration is JXTA [JXTA 2003]. This framework provides a common platform to help developers build distributed P2P services and applications. Here, every device and software component is a peer and can easily cooperate with other peers. Although JXTA has shown to be useful to support collaboration in peer-to-peer networks, it also requires a wired or fixed wireless network (similar to LaCOLLA). Therefore, it is not well suited to apply it in ad-hoc mobile work settings.

On the other hand, Nokia has developed a services-oriented framework that could be used to support mobile collaboration. This framework includes a set of APIs and an SDK (Software Development Kit) allowing developers to create service-oriented applications that act as consumers of Web services on mobile devices [Hirsch et al. (2006)]. Since mobile applications can just consume services, their autonomy is small because they require a service provider. It is unsuitable for our ad-hoc mobile work scenarios.

Currently, there are several proposals to share information in P2P networks, even considering mobile computing devices [Hauswirth et al. 2005, Neyem et al. 2005]. Typical examples are tuple-based distributed systems derived from LINDA [Gelernter 1985], such as: FT-LINDA, JINI, PLinda, T-spaces, Lime, JavaSpaces and GRACE [Bosneag and Brockmeyer 2005, Handorean et al. 2003, Nemlekar 2001]. Despite the fact these implementations work in P2P networks, they use centralized components that provide the binding among components of the distributed system. XMIDDLE [Mascolo et al. 2002] is another middleware allowing mobile hosts to share XML documents across heterogeneous mobile hosts, permitting on-line and off-line access to data. Nevertheless, these middleware are just focused on data sharing and they do not support the autonomy and interoperability capabilities required by mobile workers.

4 SOMU Design Decisions

The first design decisions try to deal with the general requirements involved in most computer-supported mobile work [Section 4.1]. These decisions establish a basic infrastructure supporting any groupware solution developed over it. Section 4.2 presents the design decisions made to deal with the mobile groupware requirements, considering the features of the selected basic infrastructure.

4.1 Design Decisions to Deal with the General Requirements

The strategy followed by SOMU to deal with the general requirements (presented in [Section 2.1]) involves a fully distributed lightweight platform to share information

resources. By lightweight we mean a platform able to run on a PDA. The SOMU design considered to keep low the consumption of memory and CPU, because these are the most critical components for an application. The others hardware components are easily addressable (e.g. storage) or unsolvable (e.g. battery life). The platform uses communication based on MANETs, Web services and XML-based information. Next sections explain these design decisions.

4.1.1 Ad-hoc Networking

Ad-hoc networking refers to a network with no fixed infrastructure [Aldunate et al. 2006b]. When the nodes are assumed to be capable of moving, either on their own or carried by their users, these networks are called MANETs. The network nodes rely on wireless communication to collaborate with each other. The advantage of ad-hoc networking is that the absence of a fixed infrastructure reduces the cost, complexity and time required to deploy the network. It also allows users to be on the move transporting their communication capabilities [Stojmenovic and Wu 2004]. Although most of these MANETs have a small communication threshold in terms of allowed distance between two mobile workers, routing capabilities can help overcome this limitation. Most MANETs adhere to standard specifications, such as IEEE 802.11b/g (Wi-Fi); therefore the communication interoperability among mobile workers is ensured. Next, a brief explanation of these MANET properties is presented.

No pre-existing infrastructure: Ad-hoc networks are not supported by infrastructure. The nodes in the network use wireless communication for information dissemination and gathering. This lets ad-hoc networks be applicable in several environments, providing communication autonomy and interoperability to mobile workers.

Small communication threshold: The current wireless communication norms supporting mobility have a limited communication threshold. For example, the Wi-Fi threshold is about 200 meters in open areas and 20 meters in built areas. Most groupware solutions need to extend the communication threshold as much as possible to increase the interaction scope; hence, routing mechanisms are required to allow it.

Routing capabilities: Routing support in MANETs is vital not only to increase the communication threshold but also to support appropriate message delivery. In addition, if the groupware application has to deliver messages depending on the users' role, routing becomes critical.

4.1.2 Service-Oriented Computing

The Service-Oriented Computing (SOC) paradigm involves four main components: services, clients, servers and discovery technology (registry). Services provide useful functionality to clients. Clients use services to support functionalities that will be available for users. Servers provide the services to clients. The discovery technology, usually implemented as a registry, enables servers to publish their services and clients to find and use needed services. As a result of a successful lookup, a client may receive a piece of code that actually implements the service or eases the communication with the server offering the service.

Since these components typically are located in different computers, the user mobility jeopardizes the interaction capabilities among them. A failure or

disconnection of a mobile device implies a complete lack of communication between users in a collaborative session and between clients and services whose communication is routed via this device, even if they could communicate directly. For example, if the node hosting a service registry suddenly becomes unavailable [Figure 2.a], the advertising and lookup of services become paralyzed even if the client and server remain connected. A similar problem occurs when the advertisement of a service is still available in the lookup table, but the service provider is outside the client communication range [Figure 2.b].

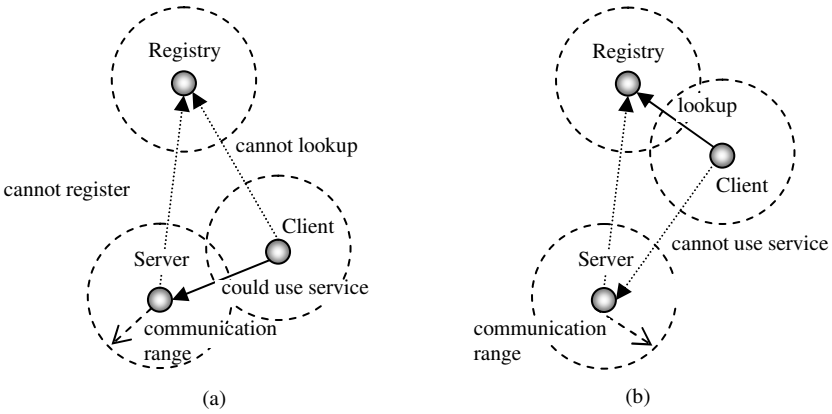


Figure 2: a) The client could use the service, but it cannot discover it because the service registry is not accessible; b) A client discovers a service, which is no longer reachable.

Considering the situations depicted in [Figure 2], it is clear the service-oriented model needs to be adapted to avoid centralized components, e.g. the registry. Model and technologies addressing these issues on MANETs should consider all nodes as mobile units able to provide and consume services from other mobile units. That is exactly the solution implemented in SOMU to deal with the users' mobility and service interoperability. Therefore, the mobile users can collaborate and interact among them on-demand. Since these solutions have to run on a range of mobile computing devices (from PDAs to notebooks), SOMU was designed and developed to be lightweight.

4.1.3 XML-based Information

The eXtensible Markup Language (XML) is a simple, flexible and standard text format to represent and exchange data. XML and its family of technologies provide flexibility to specify formats, modularity (mainly compositionality and reuse), scalability and independence of platforms and applications. The information specified in XML helps to improve data interoperability among software applications. Such information can have associated semantics specifying meanings of the data. It allows mobile workers belonging to various organizations/companies understand the

meaning of the shared information. XML files can be synchronized in order to get updated versions from asynchronously-updated XML files.

The XML data specification architecture represents the foundation layer for establishing the format and structure of messages used in a service oriented solution. XSD (XML Schema Definition) preserves the integrity and validity of message data, and XSLT (eXtensible Stylesheet Language Transformations) is employed to generate various information representations. In other words, XML is used to represent data in a standardized manner and for building a communication framework to bridge the information disparity that usually exists between and within organizations.

4.2 Design Decisions to Deal with Mobile Groupware Requirements

The previous decisions impose restrictions on the viable options to deal with the mobile groupware requirements. Next sections present the decisions made to address these requirements.

4.2.1 Distributed Sessions, Users and Roles Management

Mobile groupware applications must allow multiple work sessions involving users playing several roles. Sessions, users and roles management should be fully-distributed since the workers have to keep their autonomy. The loosely coupled work the users perform in these scenarios requires on-demand collaboration, information sharing and data synchronization; thus, explicit session management [Edwards 1994] should be used in the application design. In explicit sessions, participants must intentionally connect with other clients in order to interchange information or carry out opportunistic collaboration.

Any user in the MANET may participate in more than one session. They access a session sending a request or by invitation. Once a user gets in a session s/he becomes visible, gets a role and can access the shared resources of such a session. A work session is created when the first user is registered as member of it and it is deleted when the last user is unregistered. A session is potentially alive even if no users are currently connected, but there exist registered users.

Every user maintains personal information (such as username, password, full name, etc.) and s/he can have specific access rights over the shared resources according to her/his role. Users have an identifier that allows mobile groupware applications to make the users' rights effective. The rights are related to the role each user has for each session s/he is working on. Typically these rights are related to the user capability to carry out certain operations (e.g. erase, view, modify and recover information) or processes on the shared resources (e.g. backup, message delivery).

4.2.2 Distributed Management of Shared and Private Resources

Every user must have a local private and a shared repository for each session s/he belongs. It allows her/him to share resources on-demand. When a user logs in a session, the user's local shared resources become visible to the rest of the session members. When a user leaves a session, the local (private and shared) resources are kept available for him/herself, by allowing the user work asynchronously.

Since mobile workers have to be autonomous, the resources required by a user to perform an activity should be reachable; thus, they must be locally stored through a

replication mechanism. Replication of resources increases the users' autonomy but it also adds inconsistency. A resource reconciliation process is then required.

Mobile workers connected to a session use a local (private) repository to store the private resources and a shared (public) repository to store the resources they want to share with the session members. The shared repository contains two types of information resources: reconcilable and irreconcilable. A reconcilable resource is a piece of information which can be synchronized with other copies of such resource (from other mobile workers) in order to obtain a consistent representation of it. These resources are shared through data synchronization processes. This sharing approach maximizes the opportunity for distributed asynchronous work and enables mobile workers to modify shared data without restriction.

On the other hand, the irreconcilable resources are those pieces of information that cannot be synchronized. Typically, the system has no information about the internal structure of these files. These resources are shared through file transfer. This sharing mechanism is a solution with low coordination cost.

4.2.3 Context Management

Context is defined here as everything that can influence the behavior of shared workspaces; this includes resources internal to a computing device (e.g. memory or screen size) and external resources (e.g. bandwidth, quality of the network connection, and mobile hosts location and proximity). Context is highly dynamic in mobile scenarios. Mobile hosts may rapidly connect and leave the network. The lookup service is complex in the mobile scenario, and broadcasting is the usual way of implementing service advertisement.

Each mobile groupware application requiring to be context-aware must have a distributed context manager. This manager must store, update and monitor current status of the context in order to adapt the application functionality to changes in the work scenario (e.g., a mobile worker gets isolated or networking support is not available anymore). The context manager can also be used to adapt the system functionality to heterogeneous mobile computer devices and communication scenarios. Furthermore, context information can be used to optimize application behaviors depending on the computing resources availability. Some contextual variables useful in mobile collaboration are: location, relative location, computing devices characteristics and networking support.

The context manager has to be carefully engineered to reduce the use of limited resources, such as battery, CPU, memory or network bandwidth. This manager must provide just a minimal set of functionalities and then it is the application which is in charge of monitoring and adapting its behavior according to its own needs.

5 The Services-Oriented Mobile Unit

SOMU is a lightweight platform supported on MANETs and able to run on PDAs, Tablet PCs and Notebooks. It enables each mobile computing device to produce and consume Web services from other peers. Furthermore, SOMU allows mobile workers to share information in XML and other formats.

Collaborative applications developed on this platform can use the general solutions implemented in SOMU. These solutions concern the management of sessions, users, messages, shared objects and repositories; and partially the work context. Collaborative applications inherit the capabilities for interacting with applications running on others mobile units. It helps developers to focus on the application main goals, freeing them to deal with the low-level interaction support.

The SOMU architecture consists of a set of components organized in two layers and a transversal component [Figure 3]. The layers were named *coordination* and *communication*, and the transversal component, *shared space*. The layers manage the Web services stored in the shared space to implement specific communication and coordination services. These three components communicate with the adjacent one through an API. Since the SOMU architecture is modular, the replacement of components produces minimal impact on the middleware and also on the application supported by the middleware.

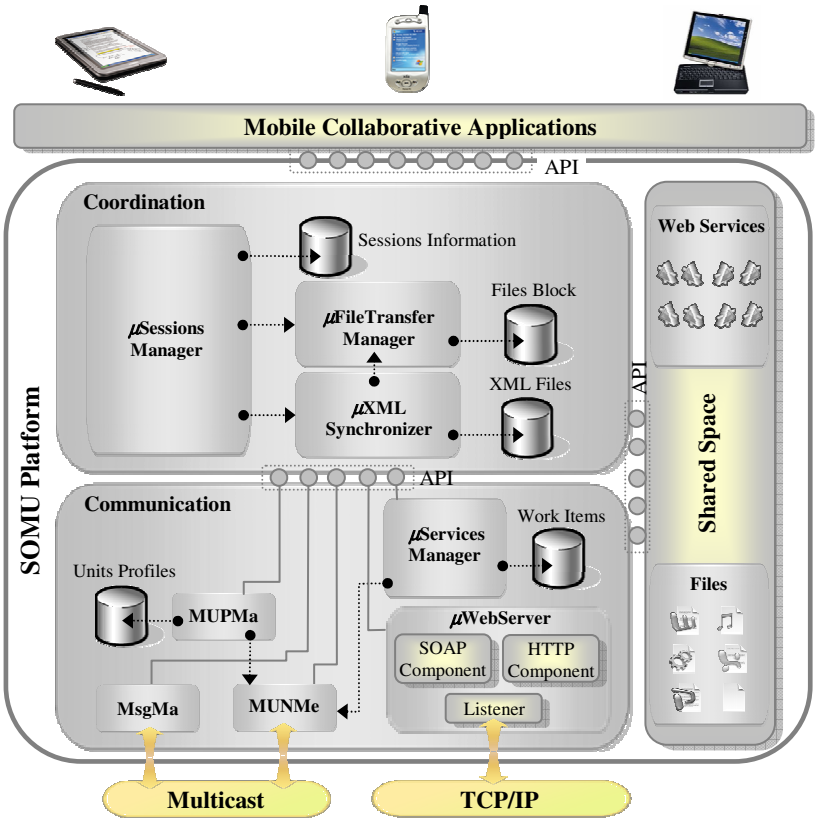


Figure 3: SOMU Architecture

Guerrero and Fuller suggest that basic (generic) services for CSCW are: sessions management, users management, roles management, messages delivery, meta-objects,

repositories (or shared spaces), awareness, floor control and environments [Guerrero and Fuller 2001]. The components included in the SOMU architecture allow dealing with most of these services.

- ***µSession Manager***. This component *manages (creates, updates* and *deletes) work sessions* and their *shared resources*. Since this component records information about *users* and *roles*, it is able to provide users/roles *awareness* information to other users in the same session (if requested by the applications).
- ***Shared Space***. This component implements a *distributed shared space* where data and services can be shared among members of a work session.
- ***µFileTransfer Manager***. This component allows groupware applications to interact with the shared space.
- ***µXML Synchronizer***. This component allows *information synchronization* with a low processing cost.
- ***Profile Manager***. This component provides *device information* for all the types of mobile units making up the current MANET.
- ***Mobile Units Near Me***. This component provides *awareness information* about users' availability and proximity.
- ***Message Manager***. This component disseminates any type of information to one or more receivers. It implements *distributed notifications (or messages delivery).*

Other typical groupware coordination services such as floor control or synchronous messaging do not make sense to be considered in this scenario, because the communication environment has high disconnection rates. Therefore, it is highly probable that mobile users get isolated during some time periods. In summary, synchronous messaging is not recommended, and floor control will require adaptive mechanisms to be effective in this scenario. Adaptive solutions are complex and expensive if they have to run on a PDA with scarce hardware resources. The next two sub-sections explain these components in detail.

5.1 Coordination Layer

The coordination layer is in charge of providing the services required by mobile workers' applications to coordinate the operations on the shared resources (e.g. files, sessions and Web services). This coordination is made individually (per unit) and it generates a consistent view of the group activities. The components of this layer are described below.

5.1.1 µSession Manager

This component records information about users, sessions, roles and shared resources. It also lets users interact with the objects shared by users of a session. The user role is just an informative data, since it is not possible to carry out a distributed control of it. Since the component records users and roles information, it is able to provide users/roles awareness information to other users in the same session.

Access to a session is implemented as a record the µSession Manager writes in the local sessions repository. Then, the users become available in the shared environment (visible) and they can access the session resources. Although the list of available sessions is public, the restrictions to access the session resources will depend on the type of session. The resources of public sessions are visible to any user in the network. The resources of private sessions are visible only to the users connected to them. Users can also create a new session or request access to other existing sessions.

When a user creates a private session, SOMU gives him/her a SessionId, which is not visible to the rest of the MANET members. The SessionId should be sent to the invited users. The invitation and the SessionId will be delivered using multicast as a way to reduce the use of the network. A user can leave a work session indicating that decision to the local session manager. If such user is the only registered one, then the session is deleted.

All reachable mobile units are identified in real time by a SOMU component. This allows users collaborate or share information on demand. When two or more reachable users decide to interact, automatically a session is created in order to isolate the interactions between them and to hide such process from the rest of the mobile units in the MANET.

Every work session has a shared folder in each member mobile unit. This folder stores the files each user shares with the partners. Once a user is in a session, his/her local shared folder becomes visible to the rest of the session members. Shared information specified in XML can be synchronized with the version of a specific partner or the rest of the session members. The attributes of each shared XML file are analyzed and compared to carry out the synchronization process. The µXML Synchronizer component performs that task. Remote shared resources in other formats can be downloaded or remotely accessed using the local session manager.

5.1.2 µFileTransfer Manager

Users produce data as a result of the collaboration process. This data is stored in files that users share to support collaboration. This component provides a transparent way to share these files through multicast transmission among users interacting in a work session. Typically, users browse the remote shared files and decide if there is any file interesting for them. If a user decides to download certain remote file, the µSession Manager creates a download request to the µFileTransfer component [Figure 3]. Then, the µFileTransfer gets information from the Mobile Units Profile Manager (MUPMa), related to the remote unit that stores the file. This information is used to determine the appropriate block size in which the file will be broken down before being transmitted. The block size is relevant to consider because it affects directly the performance of the file transfer process. If the distance between the provider and the consumer of a file is short (i.e., one hop), the block size could be large (16 - 32kb). Otherwise the block size should be reduced to 1-4 kb because the disconnection rate usually loses various blocks during the transmission. If the block size is large and the consumer and provider are not near, then it is likely the retransmission process overflows the link between them. These block sizes were determined through several file transfer tests conducted during the SOMU evaluation process.

Once the size has been decided, the µFileTransfer locally invokes the work items creation to download the remote file blocks. These work items represent the Web Services (WS) invocations, which are handled by the µServices Manager component. The Web services in charge of transferring the remote file blocks adhere to the WS-Attachment specification [Nielsen et al. 2002]. Moreover, they are implemented using the DIME (Direct Internet Message Encapsulation) protocol to encapsulate the messages.

Each file block sent by the remote unit is received by the µServices manager, which notifies the µFileTransfer component. Such component stores the blocks in a temporal local space. When all file blocks are received, the µFileTransfer notifies the µSession Manager component this fact. Thus, the file transfer process concludes.

5.1.3 µXML Synchronizer

It is clear that XML representations of any type of information help increase the data interoperability among software applications. In mobile ad-hoc environments, where disconnections are frequent, mobile users work asynchronously by updating the local XML files. Hence, it is necessary to deal with the data inconsistency of different cached replicas. The µXML Synchronizer provides synchronization functionalities through an application dependent reconciliation process [Neyem et al. 2006]. Currently this component reconciles XML documents; however the process to follow is the same if XML elements need to be synchronized.

The data reconciliation (or synchronization) process is on-demand; therefore each user decides when to synchronize and whom to synchronize with. A user can synchronize or get (via file transfer) shared files only if the users involved in the process are all connected to the same session at the same time.

The reconciliation process is an adaptation of the XMIDDLE algorithm [Mascolo et al. 2002]. In our adaptation, the data replication and synchronization are two fundamental aspects in the reconciliation strategy. The detection of conflicts and support data reconciliation is eased by a versioning mechanism for XML documents and the use of resolutors. Every mobile host maintains two types of XML documents: *versions* and *editions*. Versions contain changes that have been performed locally (without communicating them to the other hosts). Editions are, in a sense, stable versions; they contain changes that have been agreed with another host, after a reconciliation process. We refer to the process of establishing a new edition as releasing a version. Therefore, an edition can have both versions and editions as direct descendants in the version graph, whereas a version does not have descendants, because first it has to be turned into an edition [Figure 4]. Consequently, versions always contain the most recent information.

For instance, Figure 4 shows Host A and Host C have released two editions of the tree (XML files). Currently, Host A works on a modified copy (a version) of the latest common edition. Host B has only a version (it has not reconciled it with other hosts). Shared XML documents are distinguished by an edition identifier. This identifier contains the type of XML document, the edition number (which is the maximum of the two previous edition numbers incremented by one) and the hosts that agreed in releasing this edition. We use the symbol "*" when the creation of the edition does not involve a second host (this is the case of a host that generates the first edition after

the application performed an export operation) or when all hosts have agreed and established a new common edition.

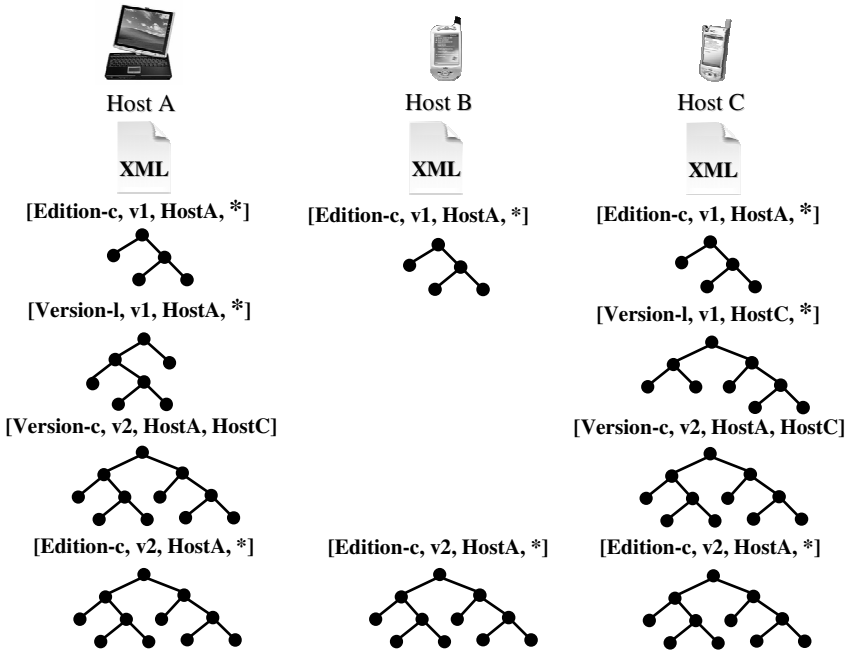


Figure 4: Versions history graph of an XML document

The versioning scheme solves the problem of identifying different editions of shared information in a peer-to-peer network that lacks a central authority to issue edition numbers. In fact, it is possible for two hosts to reconcile a tree they copied from another host, without interacting with the owner (in our case, a possible central authority). The synchronization strategy is suitable for a mobile setting because it applies a typical peer-to-peer solution, without the presence of a host that is able to provide a particular service (in our case, number edition issuing).

On the other hand, the reconciliation protocol is one of the most important design aspects of the strategy implemented in this component. Essentially, its aim is to obtain a consistent version of the same XML document once hosts become connected and agreed with other hosts to synchronize. It is based on XML tree differentiating and merging techniques. In fact, the design goals of this protocol are to minimize data transfers, only transmitting the differences between data structures and, at the same time, to be able to locally reconstruct diverging replicas from a common previous edition on the same host [Neyem et al. 2006]. Then, the result of the reconciliation is propagated to the others host, communicating only the changes performed on the common latest edition. Therefore, this synchronization process has low cost in terms of CPU and bandwidth consumption.

The reconciliation protocol uses a mechanism to control possible conflicts between different replicas. In other words, it can be performed in an application-

specific way. This feature of our strategy is fundamental for a large class of applications. It is worth noting that this component does not implement any particular policy. From this point of view, it is extremely flexible. In fact, programmers can easily develop complex mobile applications that need data sharing, without considering the problems related to disconnections and possible data inconsistencies.

5.2 Communication Layer

The communication layer is typically in charge of providing the support for message interchange among mobile units. This component allows a user to send a message to all users in the MANET, those connected to a session, a specified group of users, or a single user. Based on that, a mobile collaborative application can send messages (notifications, commands, data or events) to other users. This layer components are the following ones: *µWebServer*, *µServices Manager*, *NUNMe* (Mobile Units Near Me), *MUPMa* (Mobile Units Profile Manager) and *MsgMa* (Messages Manager).

5.2.1 µWebServer

The µWebServer component has the capability to expose and consume Web services, and executing HTTP requests from Laptops, Tablet PCs and PDAs. A listener module is responsible for managing client requests on a particular port [Figure 3]. It performs validations and determines the most appropriate supporting components to carry out a request. The supporting components represent the implementation of an Internet protocol, particularly HTTP and SOAP.

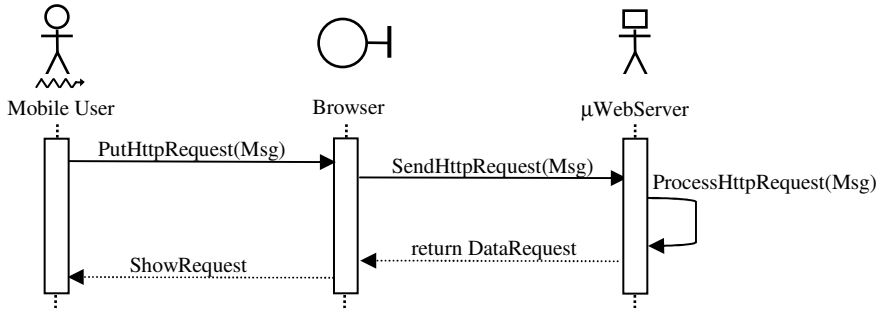
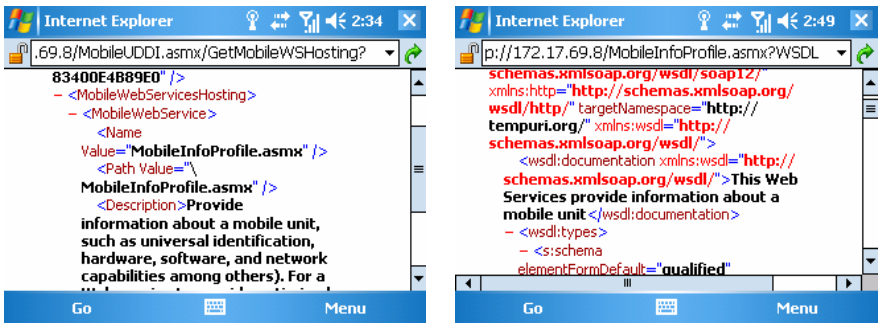


Figure 5: Sequence diagram of a service request over HTTP

The HTTP component supports the processing of HTML, GIF and JPEG Web requests and GET and POST through SOAP components. As client requests are received, the required file is retrieved from local storage. Then, this file is converted into a stream of bytes and sent back to the client mobile unit. Figure 5 shows the sequence diagram to invoke Web services over HTTP GET operations. Figure 6 (a) presents the results of invoking the "Mobile UDDI" Web service (included by default in SOMU), which provides information about all Web services hosted in a remote mobile unit. Figure 6 (b) presents the results of a similar invocation. In this case, the

invoked remote Web service is the "Mobile Info Profile", which informs the WSDL (Web Service Definition Language) document about a mobile unit.



(a) (b)

Figure 6: (a) List of Web services hosted in a remote mobile unit; (b) WSDL of a remote Web service

On the other hand, the SOAP component addresses the requirements of processing Web services remote invocations sent by other mobile units. The current implementation supports GET, POST and SOAP action operations. Typically GET and POST operations are used for browser requests. Meanwhile, SOAP actions are used to identify SOAP packets sent by applications using a particular Web service. Additionally, the SOAP component provides facilities to automatically generate WSDL files from a requested Web service, provides security verification (through WS-Security specification with UsernameToken Profile) to determine the access rights, and provides files transfer with WS-Attachment specification using DIME.

It is worth noticing the solution implemented in the µWebServer was focused on the design of lightweight versions of two key components: the Web services architecture and the component-based Web server. This last module is able to support lightweight Web services extensions and a range of Internet standards protocols. Furthermore, the module hosts Web services in mobile devices, which offers many benefits in delivering of portable software services to mobile users. It allows mobile users to interoperate with their partners in an ad-hoc environment.

5.2.2 µServices Manager

This component is in charge of creating, storing and dispatching work items when an application invokes remote Web services (provided by other mobile units). The work items stored in a mobile unit represent the Web Services (WS) invocations that such unit needs to perform. Each work item is composed of a Ticket, a Mobile Universal Identification (MUI), the WS Proxy, WS Input and WS Output. The ticket is the work item identifier. It is used to communicate the results of a WS invocation to a mobile collaborative application. The MUI identifies each mobile unit and allows the µServices Manager to make direct invocations to WS running on other mobile units. WS Proxy contains the information required to coordinate the invocation and the response of WS provided by other units. WS Input contains the invocation parameters

to be sent by the WS Proxy when it invokes the remote WS. WS Output contains the results of a WS invocation.

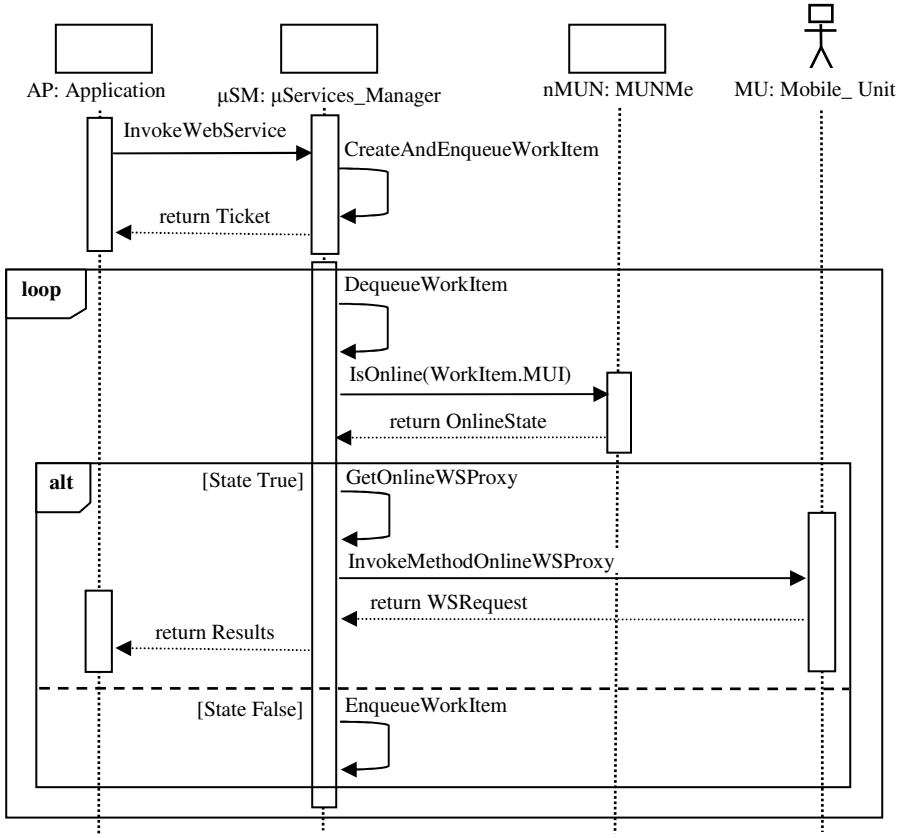


Figure 7: Sequence diagram of a process to create, store and dispatch work items

In order to understand the functionality of this component, Figure 7 presents a possible sequence diagram of a process to create, store and dispatch work items. For instance, let us suppose the mobile unit providing a specific Web service is online. When an application running in another device invokes such Web service, the µServices Manager creates a work item and records it in a queue. The work item is kept there until it is processed or a deadline is got. µServices Manager also creates a proxy client instance, which interacts with the remote Web service. Then, the µServices manager sends the WS request to the remote unit, which process it and returns the results. When the µServices manager of the client receives the results, it notifies to the application, it delivers the results and it removes the work item record.

On the other hand, if the remote mobile unit hosting the Web services is not reachable, the Mobile Units Near Me (MUNMe) component verifies whether the service provider mobile unit gets online. When such unit is online, the µServices manager of the client unit retrieves the work item from the queue. Then, the manager

sends the WS invocation of the remote unit using the proxy functions. After processing the request, the remote unit returns the results back to the client's proxy. Finally, the μServices manager in the client unit returns the results to the mobile collaborative application and it removes the work item from the queue.

5.2.3 Mobile Units Near Me

The Mobile Units Near Me (MUNMe) is the component in charge of discovering and recording the mobile units that are close to the current mobile device. This component classifies the distance between two nodes as: *near*, *reachable* or *unreachable*. Near means the nodes are at one hop of distance from each other. Reachable means the separating distance is more than one hop; and unreachable indicates the nodes are not in the same network or one of these nodes is not available.

This information is updated with the Web services remote invocations performed by each mobile unit, and also with the notifications sent by the remote units. Additionally, a general update process is performed every 30 seconds or on-demand when the user decides that it is a good time to start an interaction with a specific mobile unit. This component uses a multicast protocol to discover units connected to the MANET, and it obtains the name, Mobile Universal Identification (MUI) and the IP address of such mobile units.

5.2.4 Mobile Units Profile Manager

Web services are typically accessed from various kinds of mobile computing devices; therefore, interoperability and personalization play an important role for universal access and application usability. The Mobile Units Profile Manager (MUPMa) stores and manages information related to mobile units, such as the universal identification, hardware resources, operating system and network capabilities. Web services can use this information to optimize the interactions between providers and consumers. Particularly, the μFile Transfer component uses the MUPMa to determine the appropriate block size when sending a file from a provider to a consumer. It affects the application performance and network available bandwidth.

5.2.5 Messages Manager

This component is responsible of sending messages among mobile units adhering to a multicast and unicast strategy. The multicast messages are used by the μSession Manager to maintain updated information about sessions and shared resources located on remote mobile units. Any policy for message delivery that is supported by unicast/multicast should be implemented as part of the Messages Manager.

The message delivery service in SOMU is based on ad-hoc gossip multicast [Haas et al. 2002], because it offers an intermediate solution between the routing and flooding techniques. By using gossip-based multicast, we expect the delivery mechanism achieves high reliability with moderate degradation of performance. This delivery strategy relies on the following three-phases algorithm:

Potential Disconnection Detection: This phase is performed permanently by every node in the MANET. At every time step *t*, each node monitors its 1-hop neighbors. For each 1-hop neighbor located at a distance greater than a certain number of hops,

given as a parameter, the monitoring node records such distance at that moment. At the next time step, the monitoring node will determine the new distance for each previously monitored 1-hop neighbor. If, for each 1-hop neighbor, the distance has increased, then the monitoring node will try to find out if at least one of its other 1-hop neighbors is near to the one under observation. If the monitoring node fails on such task, it will propagate a "potential disconnection" message through the MANET.

Correction: The second phase is also implemented by each mobile unit in the MANET. Once a node detects a potential disconnection, it propagates a message through the network, using gossip-based multicast. Any mobile unit receiving that message will wait a given period of time for the counterpart message; i.e., the message sent by the other node involved in the potential disconnection. If the counterpart message is received by the mobile unit, then it will ignore the situation because it means that there is at least one alternative route connecting both nodes. On the contrary, if the counterpart message is not received by the unit, then it assumes a disconnection is in progress and it will set itself towards that task; i.e., move towards the potential disconnection area.

Maintenance: Each mobile unit in the potential disconnection area tries to detect the presence of the requesting nodes during the third phase of the algorithm. If a unit does not find the requesting nodes after a given period of time, then it sets itself back to Idle state. Nonetheless, if a unit detects the requesting nodes, then it sets its state to *supporting mode*. The unit will remain in such a mode until either the supported nodes are again within communication range or the supported nodes are apart and the disconnection is imminent. In such case, the mobile unit will play the potential disconnection phase.

This three-phase algorithm relies on two assumptions. On one hand, the set of mobile units is comprised of homogenous devices not only in terms of communication capability, but also in terms of mobility; i.e., they are able to move at similar speeds. On the other hand, it is assumed the environment is free of obstacles; i.e., the time for a mobile unit to reach a target location depends only on the distance.

5.3 Requirements vs. SOMU Components

Table 1 presents a summary of the requirements described in section 2, versus the design decisions made. In addition, these decisions are mapped to the SOMU components that implement them.

SOMU Components were implemented in C# using the .NET Compact Framework; however, they can also be implemented using the J2ME SDK for mobile devices. The .NET platform was chosen since it offered rapid prototyping and a rich development environment including live debugging on emulators. The .NET libraries natively support XML manipulation, Web service description and reflection. This allows us to implement basic services for Web services description and discovery.

	Design Decisions / Requirement	Ad-hoc networking	XML-based information	Service-oriented computing	Distrib. sessions, users and roles management	Fully distributed architecture	Distrib. management of private /shared resources	Context management
General Requirements	Discretionary collaboration			+	+		+	+
	On-demand information sharing				+		+	+
	On-demand information synchronization		+		+		+	
	Awareness of users' reachability							+
	Low coordination cost		+				+	
Groupware Requirements	Autonomous	+				+	+	
	Context-aware							+
	Shared information availability						+	
	Interoperable		+	+				
	Lightweight			+				
	SOMU Components	Messages Manager	µXML Synchronizer / µFileTransfer	µWebServer / µServices Manager	µSession Manager	Replicated Data and Services	Distrib. Private / Shared Repository	Mobile Units Near Me / Profile Manager

Table 1: Matching requirements with SOMU components

5.4 SOMU Components Dynamic Interaction

Figure 8 shows the interactions between two mobile collaborative applications, when an application "A" requires downloading a file in a session and it invokes a Web service exposed by a remote application "B". The first step of this interaction requires "A" to make a local request to the µSession manager to download a file stored in "B". The µSession manager creates a download petition to the µFileTransfer (2nd step). The µFileTransfer gets information from the MUPMa related to the remote unit which stores the file. Based on this information, the component determines the appropriate block size to transfer the file between these two mobile units.

Let us assume the file size fits in the file block. In such case the µFileTransfer Manager component creates just one download request for the remote file (3rd step). The µServices manager receives the request and it creates and queues a work item (4th step). Then, this manager asks the MUNMe component if the application "B" is online and within the "A" communication range. If the answer is negative, then the µServices manager waits and retries until it gets a positive answer (5th step).

When the mobile application "B" becomes reachable, the µServices manager of "A" creates the proxy using reflection from the context information. Such information is in the WS Proxy field which is part of the work item. Then, the µServices manager invokes the remote service hosted in "B" (6th step). The invocation is received by the remote µWebServer (7th step). Since the request is a Web service invocation, the µWebServer SOAP component activates the corresponding Web service and it invokes the method implementing such service (8th step).

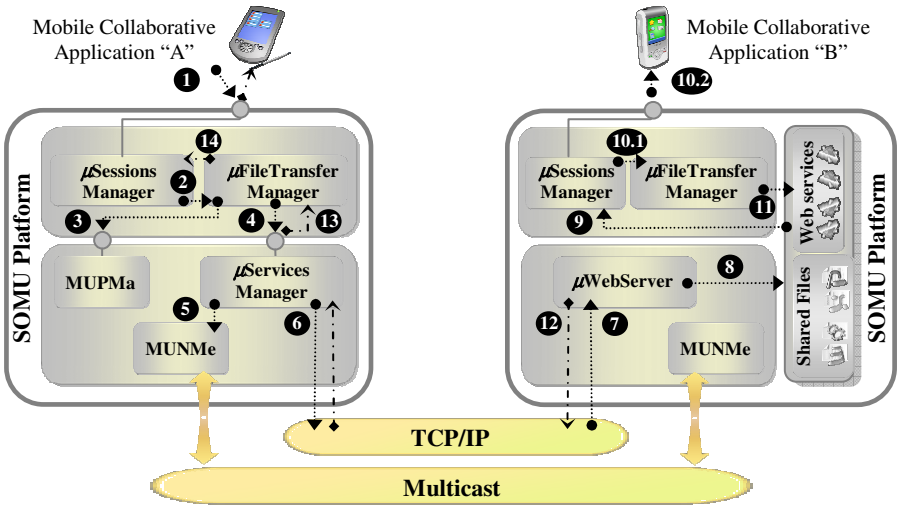


Figure 8: Interactions among SOMU main components

Since the µSession manager previously subscribed this Web service as shared, this service interacts with the µServices manager to determine if the requesting user has the access rights to download the file (9th step). If the answer is positive, then the µServices manager assigns the request control to the µFileTransfer manager (10.1 step). If the application "B" is subscribed to receive the events related to download files, then the µServices manager will send the corresponding notification (10.2 step). The µFileTransfer manager returns the result and control to the Web Services (11th step). The µWebServer returns the results to the mobile application "A" (12th step).

When the µServices manager from "A" receives the results, it removes the work item from the queue and it notifies the µFileTransfer manager indicating that item has finished its processing (13th step). Then, µFileTransfer component notifies the µSession manager the download file request has finished. Finally, this manager notifies the mobile application "A" (14th step).

Next section presents an application scenario showing how the actions taken by a mobile collaborative application are translated into the actions that occur within the Services Oriented Mobile Units. A mobile collaborative application using services provided by the platform was developed in order to test SOMU. This application represents a proof-of-concept and it illustrates the feasibility to use SOMU to support mobile collaboration in ad-hoc scenarios.

5.5 Results of Performance Tests

This section presents the preliminary results of tests applied to SOMU in laboratory simulations. These tests involved Wi-Fi communication and PDAs with a CPU speed of 624 MHz 64 MB RAM / 128 MB ROM. The devices were stationary during the test and they were deployed within a large room. The distance between two devices was no longer than 3 meters.

Download Performance: This test corresponds to sequential invocations of a Web Service (WS) from one PDA to another one. The invoked WS downloads files from the remote device provider. The communication protocols used to do it were POST, GET and SOAP. The response time was measured at the consumer side. Figure 9 shows the time spent by the WS in the downloading process is an almost linear function of the file size. The download time is low because SOMU is running on a PDA.

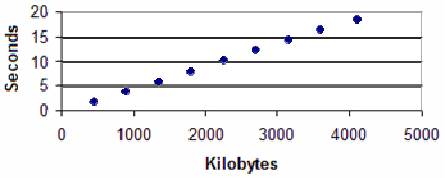


Figure 9: Download time as a function of file size

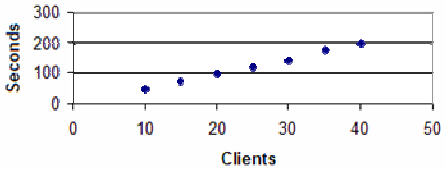


Figure 10: Download time as a function of number of concurrent invocations

Performance of Concurrent WS Invocations: This test measures the average response time of a WS running on a particular PDA, when it is concurrently invoked by several remote clients. The time was measured at the consumer side. The invoked WS downloads a file of 1MB. Figure 10 shows the results, which are similar to the previous test: a) the download time is almost linear, but it now depends on the number of clients invoking the WS; and b) the SOMU performance is still acceptable.

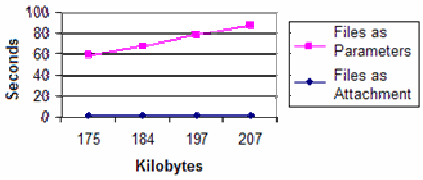


Figure 11: Download time as function of file size for two results returning strategies

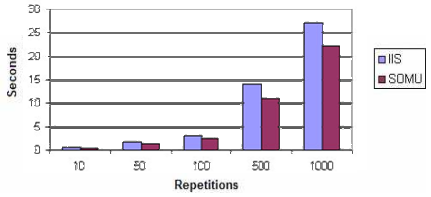


Figure 12: Average response time of a WS. Service provided by SOMU and MS Internet Information Server

Performance based on the Download Mechanism: This test compares the downloading performance depending on the mechanism used to return the results to the WS consumer. In this case, the results were returned as both attachments and parameters. Figure 11 shows the download time is much better when the results are returned as attachments. As it was mentioned in section 5.2.1, SOMU implements WS-Attachments for PDAs.

WS response time: This test compares the average response time of a WS, when such service is provided by SOMU and by Microsoft Internet Information Server (IIS). The test was done using a notebook as service provider because IIS cannot expose WS in PDAs. The clients were implemented using threads running on the same computer.

The invoked WS validates a small XML file and returns a boolean response. The results [Figure 12] show SOMU has better performance than IIS in WS processing.

6 Application Scenario: Disaster Relief

This section presents a mobile groupware application developed on SOMU to support first responders in urban disaster scenarios. Next the key requirements of this application are described. Most of them match the requirements presented in section 2. Then, the groupware solution is briefly introduced and it is shown how the SOMU components interact satisfying the specified requirements.

6.1 Application Requirements

An urban area can be seen as an interconnected system (public utilities, transportation systems, communications, power systems, homes and office buildings) where a failure can potentially affect many people. When a disaster affects urban areas, two key issues have to be addressed to mitigate it. The first one is to control the cascading effects on the interconnected systems [Godschalk 2003, Stewart and Bostrom 2002]. The second one is to keep a communication infrastructure available. This infrastructure should be wireless because of the mobility of first responders, and it should provide digital communication because it allows transmitting voice and data, and also routing messages on the network [NCTAUS 2004, Canos et al. 2005]. These capabilities allow distributed decision-making and coordination of efforts done by organizations participating in disaster relief activities [Comfort 2004]. Summarizing, an interoperable and digital infrastructure that provide *mobile ad-hoc communication* is required, and the communication based on MANETs represents an interesting option.

Typically, the composition of a disaster relief mission involves police, firefighters, medical personnel and government agencies. Police is in charge of isolating and securing the affected area, firefighters are the initial responsible body for protecting human life and physical infrastructure, medical personnel are responsible for healthcare of the affected people, and government authorities, usually located at a command post, are responsible for coordinating the efforts [Comfort 2004]. These organizations need to specify in an interoperable format all information they use, because it should be aggregated and shared with other organizations. In other words, they need *information interoperability* and *synchronization capabilities*.

Initially, first responders moving around the disaster scenario work in *autonomous* groups; however, they should coordinate their activities and share information to make local decisions as efficient and effective as possible. Since they are not able to give much attention to the application during the rescue activities, they *collaborate* and *share information on-demand*. These interactions are possible if each first responder using the application has *awareness of other users' reachability*. Once two or more workers meet and decide to collaborate, the *cost of collaboration and coordination* should be *low*, because they do not have much time for such tasks.

First responders are able to use just small mobile computing devices, e.g., PDAs, because they need to be on the move to carry out the activities. Therefore, the applications running on these devices must be *lightweight*.

6.2 Application Functionality

The application developed on SOMU, named MobileMap, is a kind of GIS that runs on PDAs and PC. The application allows first responders *work autonomously* and they can *access and update shared geographical information* about the disaster area and resources deployed/available to support the mitigation process. This information is depicted on a map allowing a visual identification of the current location. The information is presented in several layers [Figure 13] and contained in various XML files to deal with the *information interoperability* requirement.

The owner of the layer decides whether the information in that layer will be accessible on a *shared, public or on-demand* basis. The mobile workers and decision makers (managers) deployed in the disaster area are the users of this information. Notice these people do not belong to the same organization and they do not visualize the same information, as explained below.

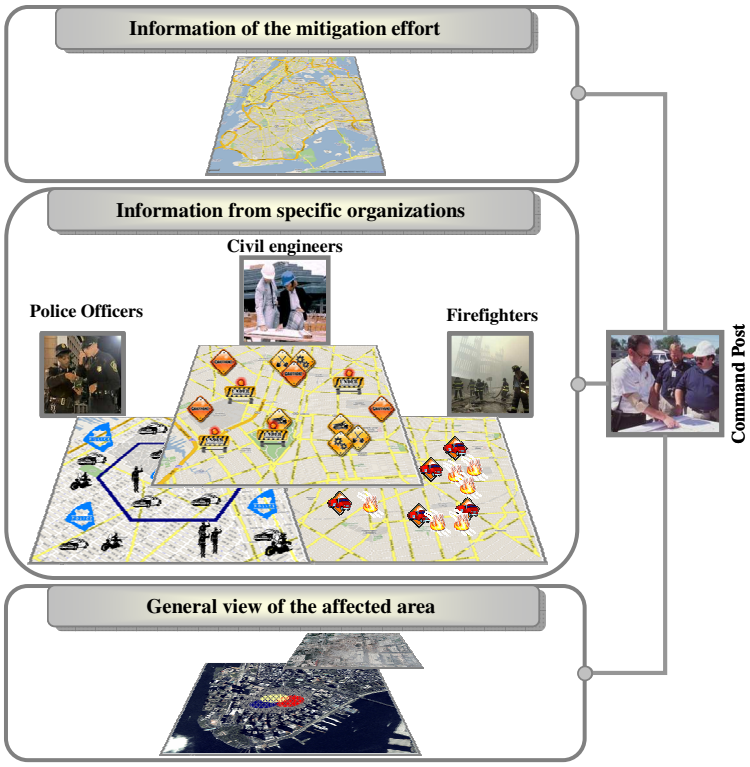


Figure 13: Representation of information in the software system

The lowest information layer shows a general view of the affected area. The middle layer is composed of maps with information particularly relevant for an organization. Thus, the Police information layer contains the officers' location, isolated area, entrance/exit routes, and task force assignments. The firefighters'

information layer includes the group locations, places where search and rescue activities were done, available/deployed equipment, evacuation routes and task assignments and priorities. Finally, the civil engineers information layer contains the stability of the affected civil infrastructure, location of available/deployed heavy-weight equipment, additional maps, vulnerable points and tasks assignment.

The highest information layer represents the intra-organization information. This layer can combine public information from the organizations participating in the relief effort and following up the mitigation process. Typically, the command post uses information from this layer to support the decision-making process.

The application allows first responders to *update and synchronize these information layers on-demand*. The reconciliation process is automatic and based on resolutors; therefore, it can be considered as a *low cost mechanism*. Furthermore, the application allows mobile users to detect neighbors (*awareness of users' reachability*) and interact with them. Each mobile unit is autonomous in terms of data and services required by the application.

6.3 Application Execution

Let us consider the following situation. A firefighter team needs to get updated information related to the stability of the physical infrastructure of an affected area, because they need to conduct search and rescue activities in that place. The most direct way to get updated information is to request it to the civil engineers evaluating the area. They record such information in the civil engineers information layer.

Some firefighter team members use MobileMap to get information from civil engineers, other partners and the command post. When a firefighter wants to get updated information from civil engineers, he/she submits a request to the SOMU platform through the application. The platform translates the request into one or more work items. Since the communication in the disaster area is based on MANETs, the SOMU middleware needs to be aware of the presence of civil engineers in the neighborhood in order to process the work item.

The *Mobile units near me* component can notify the μService manager running on the firefighters units the occurrence of a civil engineer being reachable. Then, this manager synchronizes the local information with the civil engineer information getting an updated view of the disaster area stability [Figure 14]. Using the updated information, these first responders can make better decisions about where and when to conduct the search and rescue activities. The decisions made and the results of the search and rescue activities are recorded in the firefighters' information layer. Now, updated shared information is available.

The process of information synchronization can be triggered and performed in background or as an user assisted process. These strategies can be useful in this scenario because civil engineers need to know which buildings have been already searched in order to choose other buildings to perform the search and rescue activities.

The synchronization process uses not only the Web services provided by SOMU by default, but also other Web services created just for this application. One of these Web services is used by the μXML Synchronizer component to reconcile XML files.

6.4 Information Synchronization

When a civil engineer becomes reachable, firefighter units are notified. Therefore, the µSession manager creates an XML synchronization request to the µXML Synchronizer. This last component creates a work item through µServices manager and invokes a remote Web service exposed by the civil engineer's mobile unit. The invocation includes the version of the information layer the firefighters have. As soon as the µWebServer in the civil engineer unit receives the request, it launches the SyncXML service to process the request.

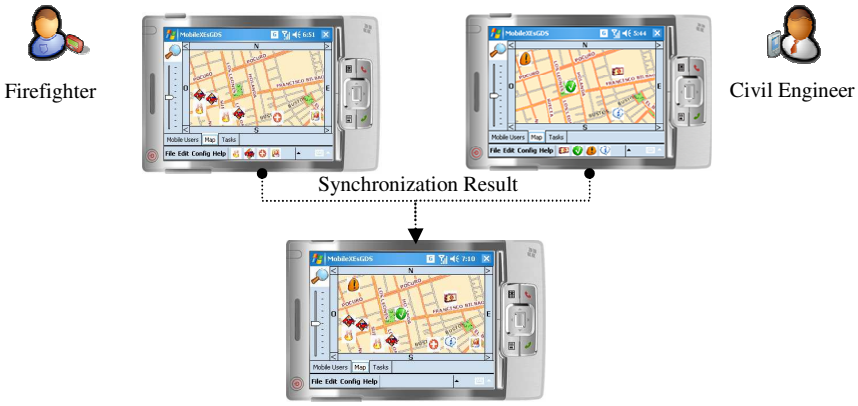


Figure 14: Information Synchronization between a firefighter and a civil engineer

Since the information the firefighter has is outdated, a local process is launched in the civil engineer mobile unit to retrieve the updated information from the local layer. These updates to the firefighter information are retrieved based on the versioning mechanism used by the synchronization protocol. As a result, an XML file indicating the information updates is sent to the firefighter µServices manager. This manager delivers that information to the µXML Synchronizer, which is in charge of reconciling both files. This component uses a mechanism (based on a resolutor) to detect and solve possible conflicts between different file replicas and obtain a consistent version of the same XML Document. Finally, µXML Synchronizer notifies the µSession manager that the synchronization request has finished and µSession manager notifies the mobile application in order to refresh the information shown on the screen.

6.5 Preliminary Application Results

An experiment was designed and performed at the premises of the Department of Computer Science of the University of Chile in order to evaluate SOMU capabilities. The floor space used in the experiment is about 1200 mt2 (60 x 20 meters). This physical infrastructure includes several concrete walls; therefore, the network signal has interruptions in many sectors when MANETs are deployed there. The isolated

areas depend on the number and location of the network nodes. That scenario resembles many real locations where urban search and rescue processes occur.

Three interaction cases were studied. The experiment's main goal was to assess the SOMU capabilities to support mobile collaboration through the Web services exposition and consumption using PDAs. The application used in these exercises was MobileMap (described in sections 6.2 and 6.3).

6.5.1. Experiments Design

Two mobile workers and also a number (between 0 and 7) of stationary PDAs were involved in the exercises. Each mobile worker (service consumer) had to request a synchronization of an information layer, implemented using an XML file, with his partner (service provider). The stationary PDAs acted just as intermediary to support the process. The following six variables were measured during these experiments:

File size: The size of the file to be synchronized was predefined. It involved nine values: 20, 50, 100, 200, 500, 1000, 2000, 5000 and 10000 kb.

Transfer rate: The synchronization process (explained in [Section 5.1.3]) is performed by a Web service, which involves two file transfers between the consumer and provider nodes. The "transfer rate" variable represents the average throughput for both operations.

Synchronization time: This variable represents the average time spent in the synchronization process. The time unit was the second and the corresponding variables were measured at the consumer side.

Number of hops: This variable indicates how many hops were used (average) to support the file transfer processes.

Waiting time: This is the elapsed time for a PDA from the instant a mobile user (potential provider) becomes available to the time the provider completes the reception of the request sent by the consumer PDA.

Number of tries: This variable indicates how many times a stored request was sent from consumer to provider, until the transfer was successfully completed.

Next, the three cases used in the study are presented. Then, section 6.5.2 discusses the obtained results.

Case I: Seven PDAs are deployed in the experimentation site and they are kept stationary forming a MANET [Figure 15]. Two additional PDAs are then deployed in the same area, and they are used by two mobile workers. These users walk by the corridor (indicated with dashed lines) keeping an opposite location to each other and moving at a constant speed (1 m/s). The pathway is about 80 meters long. Each user device runs a background process that randomly modifies an XML file. This file is an information layer of the MobileMap application. The users complete 10 laps modifying files for each file size. The stored values are averages for each file size.

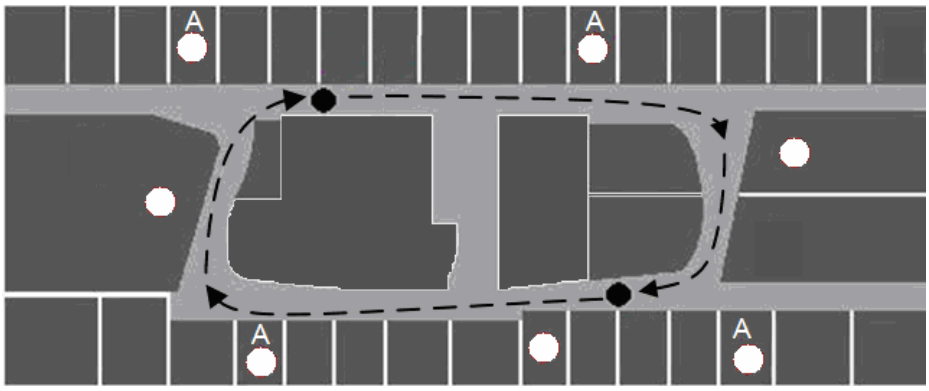


Figure 15: Map of the experimentation site

Case II: This case is similar to the previous one. The purpose of this case is to check performance with a reduced number of hops. Thus, just four stationary PDAs are used (labeled with "A" in [Figure 15]). The mobile workers' movement and the process carried out by them is the same than for case I.

Case III: This last case involves just two mobile workers. No stationary PDAs are deployed. The obtained results in each case are presented below.

6.5.2 Obtained Results

Figure 16 shows the results of transfer rate versus file size. In case I, the transfer rate falls from 157.3 to 93.2 kb with increasing file size. However, it seems to be stable for file sizes larger than 1Mb. A possible explanation for this is that small files are fast to synchronize, e.g. the synchronization of two 200kb files requires just six seconds. Therefore, the networking environment does not change much in that period and the path supporting the communication is usually the same during the whole process.

When the file size is between 200 kb and 1 Mb, the transfer rate falls. The cause for this behavior may be that two or three communication paths are required to carry out the process. Finally, the synchronization process requires at least 40 seconds for a file over 1Mb; thus, several interim points have to be used to connect a consumer with a provider. In such case each mobile worker will walk about 40 meters from the invocation point to the one where the results are received. The stability of the results shown in Figure 16 probably occurs because the transfer uses the same communication paths for all file sizes over 1 Mb.

The file transfer rate for case II is lower than in the previous case, but it is stable independently of the file size. The obtained performance stability may be a consequence of the fact that there are just few and similar paths available to connect a consumer with a provider.

Finally, case III has transfer rates only for files sizes lower than 100 kb. This is because communication between mobile workers is possible just when both are crossing the central corridor of the building. Since no interim transfer states are stored

by the application, the only files able to be synchronized in such short time period (about 3-4 seconds) are those with size smaller than 100kb. The synchronization of larger files will transfer just a portion of the file and then, it will unsuccessfully re-try once and again. The *Number of tries* variable reflected the occurrence of these events. The experiment shows SOMU is fast to detect peers proximity and react based on it. It also shows the WS requests can be effectively queued/unqueued by the *µServices_Manager* depending on the availability of a service provider.

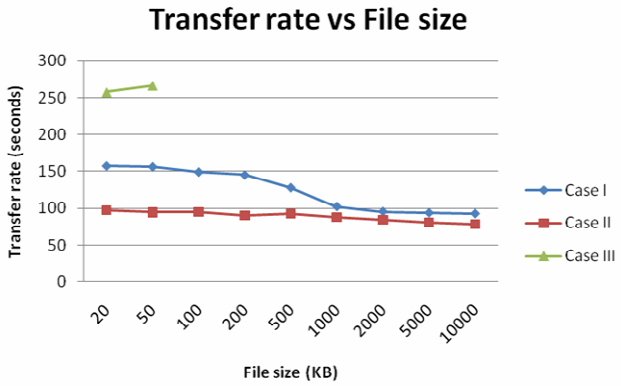


Figure 16: Transfer rate vs. file size

On the other hand, the duration of the synchronization process also depends on the file size and the number and location of stationary PDAs deployed in the area. Figure 17 summarizes the results obtained for each case.

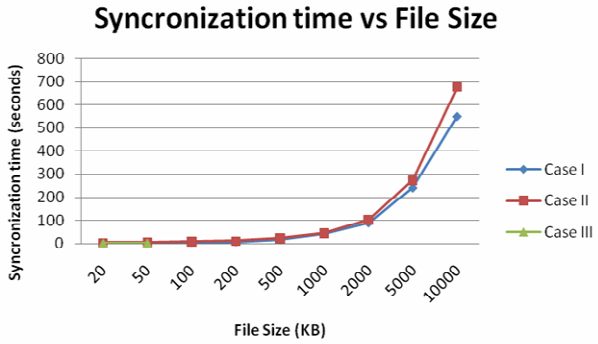


Figure 17: Synchronization time vs. file size

The synchronization time presented in Figure 17 includes: waiting time, file transfer time and reconciliation time. The first component is variable, but it is always less than a second. The file transfer time is highly dependent on the file size and it may be very large. Finally, the reconciliation time is stable, taking between 0.3 and 5 seconds depending on the file size. Summarizing, the performance of the groupware

services in mobile ad-hoc network is highly dependent on the implemented file transfer (and routing) solution.

The number of synchronization tries in case I was between 1 (for 20 Kb files) and 1.2 tries average (for 10Mb files). It seems to be a consequence of the large number of PDAs routing the packages. In case II, the number of tries was similar to the previous one; its values are between 1.2 and 1.6. This indicates the routing algorithm is good to get a dynamic path between consumer and provider, although the number of interim node is small. The number of tries for case III is just 1 for files up to 50 kb, and infinite for larger files. It is worth to notice these values are reachable just in scenarios with a high disconnection rate, and when a few number of workers are available. In any other situation, the results should be better.

The number of hops used to communicate the consumer with the provider in case I was between 2.7 and 4.2 hops. And in case II those values were between 3.2 and 4.3. In case III the only option available is 1 hop.

These results show that SOMU is a context-aware platform able to react quickly and adapt its communication mechanisms in order to provide groupware services to applications developed on it. The platform hides all these mechanisms to the user and software applications that use its services. It makes it easy to be used by developers. Furthermore, SOMU has shown that it is able to expose and consume Web services on PDAs, even when these devices are on the move. The platform is also able to synchronize XML files located in small devices of nomad users. The results show the performance of these functionalities is highly acceptable.

In addition, because of the platform does not include any kind of centralized component, the solutions implemented on it will be more robust than those involving centralized elements. These features make SOMU-based solutions easy to deploy and put into production in several work scenarios, such as police operatives, computer supported learning and health care activities. Finally, these solutions allow users to carry out activities involving high mobility, because they are able to run in lightweight computing devices, making an efficient use of the networking capabilities.

6.5.3. System Usability

The system was also evaluated by experts from the 6^{th} and 8^{th} Santiago (Chile) firefighters units during June 2007. These experts are the official urban search and rescue trainers for Chilean firefighters, and police/military officers. Five PDAs were used to evaluate the system functionality, performance and usability. The first important conclusion indicates the system is ready to be used at least in small urban incidents (fires, chemical spills, small collapses). The system functionality was considered useful to support urban search and rescue activities. The main observations were related to the icons design shown on the user interface.

The application usability was evaluated simulating the actions that firefighters must do during two small urban emergency situations: a fire and a car accident. The features of these emergencies were obtained from real cases that were occurring in the city. The details of those events were received through the radio system located at the Santiago Alarms Center.

Finally, the experts evaluated the application performance as acceptable. Therefore, during the next weeks, MobileMap will be used as a pilot experience by the 6th firefighters unit in real small urban incidents.

7 Conclusions and Future Work

Mobile computing devices and mobile ad-hoc wireless networks (MANETs) offer a wide range of new collaboration possibilities for these mobile workers. However, the design and implementation of the mobile collaborative solutions for ad-hoc scenarios imply to deal with several key requirements, such as: autonomy, interoperability, shared information management, context-awareness, and low resources use.

Most frameworks and platforms proposed to support collaborative activities of mobile workers use some type of centralized data or services. This centralization jeopardizes the application capabilities to support collaboration in ad-hoc communication settings. This paper presents a platform called SOMU (Service-Oriented Mobile Unit) intended to support the collaborative activities carried out by mobile workers in ad-hoc scenarios. Unlike previous related works, SOMU proposes a fully decentralized architecture to share resources allowing mobile devices to act as autonomous units. The results presented in section 6.5 show this approach not only is feasible, but also it is more robust than the centralized one when MANETs are involved. In addition, this approach is able to present good performance even when interoperability issues are involved.

The SOMU platform provides solutions to deal with most requirements of the mobile groupware applications in ad-hoc environments (presented in [Section 2]). Moreover, SOMU lets mobile computing devices expose and consume Web services in order to ensure services interoperability among them. Moreover, the platform uses communications based on MANETs and XML-based information as a way to provide data and communication interoperability among mobile units. Because the platform is replicated at each mobile unit, users can work autonomously and collaborate on-demand. These capabilities are relevant to support mobile collaboration when there is no stable communication support or no communication at all.

SOMU was implemented as a lightweight middleware running on laptops, Tablet PCs and PDAs. The platform provides a basic foundation for the development of mobile collaborative applications. It intends to increase the technical feasibility of solutions in the area and to reduce the development effort of MANET-based mobile collaborative applications. Although these issues have not been fully analyzed yet, the initial findings support these hypotheses.

Future work includes, in the short future, formal experimentation to study the possible contributions and limitations of SOMU and the consequences on the applications developed on it. As a second step, the functionality of SOMU will be extended to integrate (by default) P2P sessions management, standard WS discovery mechanisms (such as the WS-Discovery specification), and enabled support for the new protocols stack that includes WS-Addressing, WS-Trust, WS-Federation, WS-Eventing and MTOM (Message Transmission Optimization Mechanism) Attachment.

Acknowledgements

This work was partially supported by Fondecyt (Chile), grant Nº: 11060467 and 1040952 and by MECESUP (Chile) Project Nº: UCH0109.

References

[Alarcón et al. 2006] Alarcón, R., Guerrero, L., Ochoa, S., Pino, J.: "Analysis and Design of Mobile Collaborative Applications using Contextual Elements"; Journal of Computing and Informatics, 25, 6 (2006), 469-496.

[Aldunate et al. 2006a] Aldunate, R., Ochoa, S., Pena-Mora, F. Nussbaum, M.: "Robust Mobile Ad-hoc Space for Collaboration to Support Disaster Relief Efforts Involving Critical Physical Infrastructure"; ASCE Journal of Computing in Civil Engineering, American Society of Civil Engineers (ASCE), 20, 1 (2006), 13-27.

[Aldunate et al. 2006b] Aldunate, R., Larson, G., Nussbaum, M., Ochoa, S., Herrera, O.: "Understanding the Role of Mobile Ad hoc Networks in Non-traditional Contexts"; Proc. IFIP Int. Conf. on Mobile and Wireless Comm. Networks, Santiago Chile (2006), 199-215.

[André and Antunes 2004] André, P., Antunes, P.: "SaGISC: A Geo-Collaborative System"; Proc.of CRIWG'04, Lecture Notes in Computer Science 3198, San Carlos Costa Rica (2004), 175-191.

[Bosneag and Brockmeyer 2005] Bosneag, A.M., Brockmeyer, M.: "GRACE: Enabling collaborations in wide-area distributed systems"; Proc. of WETICE'05, Workshop on Distributed and Mobile Collaboration (DMC), IEEE CS Press, Linkoping University Sweden (2005), 72-77.

[Buszko et al. 2001] Buszko, D., Lee, W., Helal, A.: "Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration"; Proc. of ACM Int. Conf. on Supporting Group Work (GROUP), ACM Press, Colorado USA (2001), 5-14.

[Canos et al. 2005] Canós J.H., Borges M.R.S., Alonso G.: "An IT View of Emergency Management"; IEEE Computer, 38, 12 (2005), 27.

[Comfort 2004] Comfort, L.: "Coordination in Complex Systems: Increasing Efficiency in Disaster Mitigation and Response"; Int. J. of Emergency Management 2, 1 (2004), 62-80.

[De Rosa et al. 2005] De Rosa, F., Malizia, A., Mecella, M.: "Disconnection Prediction in Mobile Ad hoc Networks for Supporting Cooperative Work"; IEEE Pervasive Computing, 4, 3 (2005), 62-70.

[Edwards 1994] Edwards, K.: "Session Management for Collaborative Applications"; Proc. of CSCW'94, ACM Press, (1994), 323-330.

[Gelernter 1985] Gelernter, D.: "Generative Communication in Linda"; ACM Transactions on Programming Languages and Systems, 7, 1 (1985), 80-112.

[Godschalk 2003] Godschalk, D.: "Urban Hazard Mitigation: Creating Resilient Cities"; Natural Hazards Review, ASCE, August (2003), 136-146.

[Guerrero and Fuller 2001] Guerrero, L.A., Fuller, D.: "A Pattern System for the Development of Collaborative Applications"; Group Decision and Negotiation, 43, 7 (2001), 457-467.

[Guerrero et al. 2004] Guerrero, L., Pino, J., Collazos, C., Inostroza, A., Ochoa, S.: "Mobile Support for Collaborative Work"; Proc.of CRIWG'04, Lecture Notes in Computer Science 3198, San Carlos Costa Rica (2004), 363-375.

[Haas et al. 2002] Haas, Z., Halpern, J., Li, L.: "Gossip-Based Ad Hoc Routing"; Proc. of IEEE Infocom'02, June (2002), 1707–1716.

[Handorean et al. 2003] Handorean, R., Payton, J., Julien, C., Roman, G.: "Coordination Middleware Supporting Rapid Deployment of Ad Hoc Mobile Systems"; Proc.

ICDCS'03, Workshop on Mobile Computing Middleware, IEEE CS Press, Rhode Island USA (2003), 363-368.

[Hauswirth et al. 2005] Hauswirth, M., Podnar, I., Decaer, S.: "On P2P Collaboration Infrastructures"; Proc. of WETICE'05, Workshop on Distributed and Mobile Collaboration (DMC), IEEE CS Press, Linkoping University Sweden (2005), 66-71.

[Heinemann et al. 2003] Heinemann, A., Kangasharju, J., Lyardet, F., Mühlhäuser, M.: "iClouds: Peer-to-Peer Information Sharing in Mobile Environments"; Proc. of Euro-Par'03, Lecture Notes in Computer Science 2790, Klagenfurt Austria (2003), 1038-1045.

[Hirsch et al. (2006)] Hirsch, F., Kemp, J., Ilkka, J.: "Mobile Web Services: Architecture and Implementation"; Nokia Research Center. John Wiley & Sons Publisher, (2006).

[JXTA 2003] JXTA Project, 2003, http://www.jxta.org.

[Marques and Navarro 2006] Marques, J., Navarro, L.: "LaCOLLA: A Middleware to Support Self-sufficient Collaborative Groups"; Computing and Informatics, 25, 6 (2006), 571-595.

[Mascolo et al. 2002] Mascolo, C., Capra, L., Zachariadis, S., Emmerich, W.: "XMIDDLE: A Data-Sharing Middleware for Mobile Computing"; Journal on Personal and Wireless Communications, 21, 1 (2002), 77-103.

[Menchaca-Mendez et al. 2004] Menchaca-Mendez, R., Gutierrez-Arias, E., Favela, J.: "Opportunistic Interaction in P2P Ubiquitous Environments"; Proc.of CRIWG'04, Lecture Notes in Computer Science 3198, San Carlos Costa Rica (2004), 349-362.

[Muñoz et al. 2003] Muñoz, M.A., Rodriguez, M., Favela, J., Martinez-Garcia, A.I., Gonzalez, V.M.: "Context-aware mobile communication in hospitals"; IEEE Computer, 36, 9 (2003), 38-46.

[NCTAUS 2004] National Commission on Terrorist Attacks Upon the United States: "The 9/11 Commission Report", Dec. (2004), http://www.9-11commission.gov/report/index.htm.

[Nemlekar 2001] Nemlekar, M.: "Scalable Distributed Tuplespaces"; MSc. Thesis. Department of Electrical and Computer Engineering, North Carolina State University, Chapter 5, (2001).

[Neyem et al. 2005] Neyem, A., Ochoa, S., Guerrero, L., Pino, J.: "Sharing Information Resources in Mobile Ad-hoc Networks"; Proc. of CRIWG'05, Lecture Notes in Computer Science 3706, Porto do Galinhas Brazil (2005), 351-358.

[Neyem et al. 2006] Neyem, A., Ochoa, S., Pino, J.: "A Strategy to Share Documents in MANETs using Mobile Devices"; Proc. of ICACT'06, Phoenix Park Korea (2006), 1400-1404.

[Nielsen et al. 2002] Nielsen, H. Christensen, E., Farell, J.: "WS-Attachments Specification"; Technical Report IBM, June (2002), http://www-106.ibm.com/developerworks/webservices/library/wsattach.

[Ochoa et al. 2007] Ochoa, S., Neyem, A., Pino, J., Borges, M.: "Supporting Group Decision Making and Coordination in Urban Disasters Relief Efforts"; Special Issue: Diverse Landscape of Decision Support System Applications, J. of Decision Systems 16, 2 (2007), 143-172.

[Pinelle and Gutwin 2006] Pinelle, D., Gutwin, C.: "Loose Coupling and Healthcare Organizations: Deployment Strategies for Groupware"; Computer Supported Cooperative Work, 15, 5-6 (2006), 537-572.

[Stewart and Bostrom 2002] Stewart, T., Bostrom, A.: "Extreme Event Decision Making Workshop Report"; Decision Risk and Management Science Program NSF, June (2002).

[Stojmenovic and Wu 2004] Stojmenovic, I., Wu, J.: "Ad-hoc Networks"; IEEE Comp., 37, 2 (2004), 9-74.

[Zurita and Baloian 2005] Zurita, G., Baloian, N.: "Handheld Electronic Meeting Support"; Proc. of CRIWG'05, Lecture Notes in Computer Science 3706, Porto do Galinhas Brazil (2005), 341-350.