

Architecting Mobile Collaboration at the Middleware Level

Pierre Rust Giovanna Ferrari Titos Saridakis
Software Technology Lab
NOKIA Research Center
PO Box 407, FIN-00045 NOKIA Group, Finland

Abstract

Mobile Internet enables mobile clients to access web-based services and the value of the implied client/server architecture is extensively certified. However, the raising world of hand-held devices with multiple connectivity capabilities needs support for distributed computing, especially in a short physical range context where mobile devices interact with each other and with computers connected to a fixed network (group work, information distribution, gaming, etc). This paper focuses on the requirements elicitation and the derived architecture of middleware services that provide support for collaborative sessions among mobile terminals. Special emphasis is given to the ad hoc networking requirements of collaborative applications as well as the flexibility of the service architecture that integrates value-added services available at web/WAP enabled hosts. The middleware service architecture is based on the CORBA model and a prototype implementation for a collaborative drawing application is used to validate the conceptual design.

1 Introduction

Mobile communications, personal computing and distributed information services have been rapidly merging into what is called *mobile Internet*, a connectivity concept which opens the wired Internet to the mobile world. The popular client/server architecture which has been widely used in Internet-based distributed computing has been also successfully used on the mobile Internet (e.g. WAP [1]). However, enabling full fledged inter-networking of mobile devices and fixed computers requires an additional effort as it has been realized by recent research projects (e.g. VIVIAN/ITEA-99040 [2]). To accommodate the needs for distributed computing in the mobile Internet, widely accepted middleware technologies have specified their "wireless" or "mobile" variants (e.g. wireless CORBA [3]). Following this step, two con-

dition should be met before one can claim that distributed computing is as easy on the mobile Internet as it is on its traditional counterpart. First, the mobile variants of the middleware technologies should be mapped onto the most prominent mobile communication protocols. Second, new middleware services should be specified which meet the requirements for ad hoc networking and nomadic behavior of the mobile Internet users and applications.

This document focuses on the latter task and we concentrate on the case of collaborative applications, an application domain extensively studied in the case of fixed networks and Internet under the name of Computer Supported Collaborative Work (CSCW) [4]. First we elicit the requirements that collaborative applications place upon the mobile middleware (§ 2). Then, we map the elicited requirements onto middleware services and we derive the corresponding architecture for mobile collaborative applications (§ 3) like gaming sessions, group work, information distribution, etc. The resulting service architecture is generic enough to support the collaborative nature of a wide range of applications independently of the interaction frequency, the application data exchanged among collaborators and the access control policies applied to the participating end-users. Then we describe the implementation of the most fundamental services and their integration with a collaborative drawing application in order to put in evidence the validity of our architectural approach (§ 4). The document concludes with a summary of our contribution and a succinct presentation of our plans for the immediate and near term future (§ 5).

2 Requirements Elicitation

In this section we reduce the requirements from the technology, the end-user and the market perspective, which describe a set of properties that a collaborative system should provide, functionalities that it should offer and constraints that it should satisfy.

2.1 Technology requirements

A collaborative application provides a shared space where end-users, after becoming participants in a given session, can communicate application-specific data to each other. In this context the term *shared space* describes the conceptual area where end-users place the application-specific content they want to communicate to each other, *session* is the virtual meeting of a group of end-users devoted to access (e.g. create, edit, review, observe, etc) the contents of the shared space, and *participant* is an end-user who has joined a given session.

In the domain of mobile collaborative applications the technology requirements stem from a simple fact: a platform that enables mobile collaboration must support a diversity of mobile terminals (e.g. laptop computers, hand-held devices like the PSION netBook¹ and the NOKIA 9210², etc) and their associated mobile communication protocols. The latter can be classified in the following categories: wireless connectivity to a local network (e.g. WLAN [5]), short range wireless connectivity (e.g. Bluetooth [6]) and 2G/3G mobile phone network connectivity (e.g. GPRS [7] and UMTS [8]). These mobile communication technologies offer a wide variety of characteristics, e.g. bandwidth that ranges from 56Kbps for GPRS to 11Mbps for WLAN/IEEE 802.11b and coverage that ranges from 10m for Bluetooth to global coverage through satellites for UMTS megacells. The variety of the mobility characteristics of these technologies imply the following requirements on the middleware that provides support for mobile collaboration:

- R1** Switching from one communication protocol to another when certain conditions are met and when the end-user approves the switching without disrupting the session participation.
- R2** Session continuity despite the temporary interruptions the communication may suffer.
- R3** Session must be deadlock free despite the unexpected disappearance of participants.

2.2 User requirements

Based on a collaborative drawing application (i.e. participants drawing on a shared canvas) the remainder of this subsection describes a use case that reveals the user requirements placed on the middleware that provides support for mobile collaboration.

In our collaborative drawing use case, an end-user obtains the right to participate to a given session by registering his interest and obtaining the admission to enter the session. The approval of the admission request depends on some admission policy which is irrelevant for our study and the decision is taken based on the end-user id and the *role* he requests to play in the session. Although the exact set of supported roles is not important for our purposes, we assume the existence of three roles: the *editor* who can create new shapes (application data) and alter existing shapes for which he has the appropriate access rights, the *reviewer* who can only add annotations on existing shapes for which he has the appropriate access rights but he cannot create new shapes, and the *observer* who can only see the shapes for which he has the appropriate access rights but he cannot alter them or create new ones. Users that are admitted in a session can exercise their right to join the session and become participants and users that are not admitted do not see any of the shared canvas contents. A session opens when the first admitted user joins the session and closes when the last participant quits the session.

Besides the control on accessing the shared canvas contents which is based on the participants role, a set of access rights are associated with each shape defining the visibility of the shape in a session. Consequently there is another functionality that describes a control which relates to the access rights of the shapes. The owner of a shape in the canvas is the participant who creates it and he is the one who may specify the access rights to the given shape. Other editors are allowed to edit a shape only if its owner has placed them inside the editor group from the given shape. By setting the access right of a shape, its owner can make the shape visible only to a subgroup of the session participants, those placed in the observer group of the shape. Sessions where the participants do not share any shapes because of the access rights they have set, are automatically terminated since the collaboration overhead does not serve any purpose.

The above use case reveals the following requirements regarding the functionalities that are needed to support mobile collaborative applications:

- R4** An admission functionality which allows end-users to request admission to some session. Acceptance of the admission request depends on an admission policy (which is irrelevant for purposes of this document) and the role the end-user requests.
- R5** A supervision functionality which serves monitoring, addressing and administrative purposes. It

¹www.pSIONteklogix.com/main/netbook.htm

²www.nokia.com/phones/9210/index.html

allows admitted end-users to join a given session and which also checks out participants who wish to quit the session.

- R6** A synchronization functionality which enables participants to view the same shapes (if they have the appropriate access rights) at any given time during a session. This functionality implies a serialization of drawing events which assures that the contents of the shared canvas can be uniquely defined.
- R7** A monitoring functionality which follows the execution of a session and assures that there is some canvas contents shared by at least two participants.
- R8** An access control functionality which assures that a participant is not performing an action that is not allowed by his role.
- R9** An access right marshal functionality that associates access rights to shapes and verifies that shapes are accessed according to these rights throughout the lifetime of a session.

2.3 Market requirements

Requirements are also placed from the stakeholders which dominate the market domain to which the middleware that enables mobile collaboration is targeted:

- R10** The middleware must be attractive for the customers, which implies that it should have a low cost, be ported to the most prominent mobile terminals, and be compatible with popular products for mobile terminals.
- R11** The middleware must allow easy development of 3rd party applications, which implies that it should follow some common distributed programming model with implementations on the popular development platforms (e.g. Java, C++, etc).
- R12** The middleware must enable the painless integration of value-added services which can be offered by telecom operators and ISPs, which implies that the middleware must be extensible and must support dynamic (re)configuration during a session.

3 System Architecture

This section describes the mapping of the requirements and the associated functionalities onto services which we describe by means of their interfaces.

3.1 Mapping requirements to services

The analysis of the collected requirements and the relations among them (an activity not described here) resulted in 8 groups of functionalities. Each of these functionality groups maps to a distinct service or property of the middleware enabling mobile collaboration as elaborated in the remainder of this subsection and summarized in Figure 1.

R1 is mapped to the *Connectivity Management Service* which has access to the various communication capabilities of a mobile terminal and allows the seamless transition from one connectivity means to the other. Besides the smooth switching between available connectivity means, the service also allows the detection of enabling conditions for the available connectivity means (e.g. entering the range of a Bluetooth neighborhood or the reach of a WLAN access point) and the notification of the application about this fact. This service also allows the selection of the connectivity means to be used for the mobile communication based on some application defined criteria (e.g. higher bandwidth, lower cost, etc).

R2, **R3** and **R7** are mapped to the *Session Monitoring Service* which monitors the participants of a session to ensure the liveness of a session (e.g. unexpected disconnections of participants do not deadlock the session) and to facilitate the participation continuity despite transient disconnections (e.g. an unexpectedly disconnected participant can re-connect to the initial session without having to ask for admission again). The operations offered by this services are mainly used by other middleware services (e.g. the *Session Management Service* latter in this subsection).

R4 is mapped to the *Admission Service* which handles the admission of users to sessions according to a set of admission policies (which are irrelevant to the scope of this document) that define which user is allowed to obtain what role. Admission is transmitted to the accepted participant (e.g. by an admission ticket) who can then exercise this right by contacting the *Session Management Service* (see below) in order to become a participant in the given session. The *Admission Service* offers also operations to the *Session Management Service* to interrogate the set of admitted users in a given session.

R5 is mapped to the *Session Management Service* which is the cornerstone of the execution of a session. It allows admitted users to join a session after verifying their admission rights with the *Admission Service*. Using the list of session participants and interacting with other services (e.g. *Session Monitoring Service*, *Connectivity Management Service*), this service super-

vises the liveness and the "well-being" of a session. Moreover, this service provides the session data necessary for the *Access Control Service* to accomplish its purpose (see below). The operations provided by the *Session Management Service* include the registration of an admitted user to a session, the interrogation of the role of a participant, the request of the participants list, etc.

R6 is mapped to the *Synchronization Service* which ensures the atomic delivery of collaboration events to all participants of a session, in order to prevent race conditions to result in inconsistent views of the shared space. Despite its apparent conceptual simplicity, this service hides the remarkable implementation complexity of atomic message delivery and atomic broadcast often confronted in distributed systems or imposes the bottleneck of a centralized connection point which serializes the delivery of collaboration events.

R8 and **R9** are mapped to the *Access Control Service* which is responsible for approving participants' actions in the shared space. This service performs two types of controls: using information provided by the *Session Management Service* it verifies whether a user's role permits an attempted action, and based on the access rights of shared space objects it decides whether an attempted action is permitted on its target. The main operation offered by this service is the approval of an action of a given participant on a given object of the shared space.

R12 is mapped to the *Dynamic Configuration Service* which supports all aspects that relate to the on-the-fly configuration of a session to accommodate components which were not known at the time of the session initialization (e.g. dynamic service invocation, download and installation of components during the execution of a session, etc). In general this service may contain a big number of operations related to dynamic configuration functionalities and which may range from simple component downloading or service discovery and dynamic connection establishment to a complete support for mobile agents. In the remainder of this document we do not examine this service.

R10 and **R11** do not map on services as in the previous 7 cases. These two requirements represent a set of implementation directives (or system properties) that developers of the collaboration enabling middleware should guarantee in their product. The satisfaction of these properties depends heavily on the targeted market (e.g. in a wireless CORBA dominated market, the OMA model guarantees both the easy installation of components and their easy integration with other products existing on the mobile terminal).

```

interface connectivityManager {
    switchConnection(newConnection, params)
    detectConnectivity(connectivityDescription)
    requestNotification(condition)
}
interface sessionMonitor {
    monitorParticipant(participantID)
    registeredDisconnected(participantID)
    timeBlockedSession(timeout)
}
interface admissionServer {
    requestAdmission(userID, role, sessionName)
    requestAdmittedUsers(sessionName)
}
interface sessionManager {
    joinSession(userID, sessionName)
    removeParticipant(participantID, sessionName)
    participantRole(participantID, sessionName)
    participantList(sessionName)
}
interface eventSynchronizer {
    atomicDelivery(eventDescription)
}
interface accessController {
    checkAction(participantID, actionDescription)
    checkAccessRight(participantID, objectID)
}

```

Figure 1: Abstract service interfaces

3.2 Mapping services to components

Following the identification of the services provided by the mobile collaboration enabling middleware, their mapping to architectural components provides the first structural view on the system architecture. The mapping of services to architectural components is based on a number of criteria (e.g. minimizing inter-component communication, grouping services that belong to the same unit of failure or level of security, etc) and on the system architect skills. In the remainder of this subsection we do not elaborate on the mapping process; instead we present the coarse-grained service architecture of the mobile collaboration enabling middleware we have studied so far, which is graphically depicted in Figure 2.

The figure shows that the *Admission Service* is mapped to a component by itself, which dissociates the admission to a session from the actual participation to it. The *Synchronization Service* is also mapped

to a component by itself with the intention to integrate it either with the application or with the communication protocol at the implementation design phase. Finally, the *Monitoring, Management and Access Control* services are mapped to the same component. This deployment of services to components provides the following properties:

- The *Admission Service* may reside on some remote location (e.g. a web/WAP server) without compromising the intensive interactions that might take place during the execution of the session.
- The *Synchronization Service* can be integrated with the application or the communication protocol without imposing the same merging of other services.
- The extensive interactions that take place among the *Monitoring, Management and Access Control* services are confined in the frame of a single component.
- The middleware is application independent and capable of supporting a variety of collaborative applications like collaborative drawing, group work, etc.
- The essential concepts of CSCW (including sessions management, roles and access rights) are supported offering a platform that fits the traditional collaborative work needs while the concept of admission, rarely used in the wired CSCW, is especially suited for the mobile world.

4 Prototype Implementation

To validate of our architectural approach we developed the above components in Java and we used a simple drawing application [9] as the front-end for demonstration purposes.

The **Admission** component was implemented as a CORBA server which consults the admission policies from a local file and gives the admission rights in the form of an admission ticket which consists of the encrypted data of the user ID and the role with which it was admitted. The **Admission** component also maintains for each session the list of admitted users which can provide upon request for session administration purposes.

The **Monitor**, **Access** and **Management** component is also implemented as a CORBA server which is directly accessed by users only when they request to join

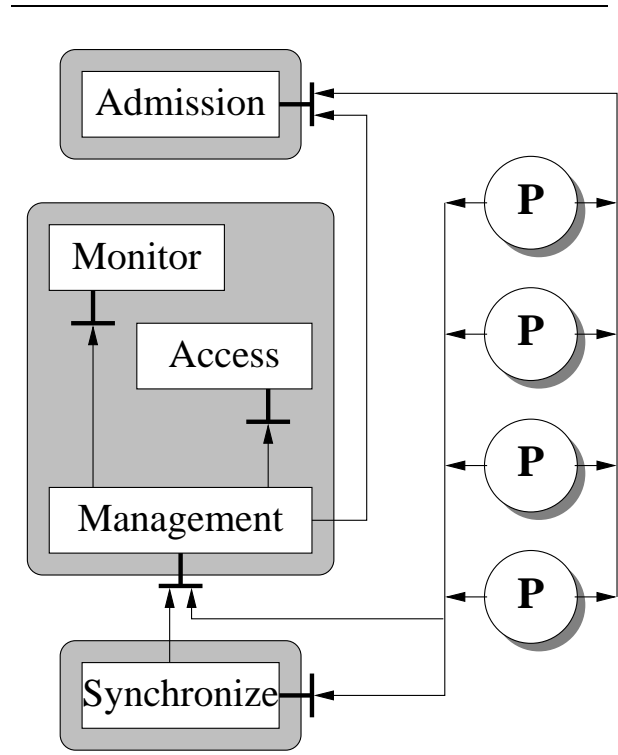


Figure 2: Coarse-grained service architecture

or leave a session. The implementation of the distinct services within this component correspond to different Java classes and the communication among them is implemented as normal method calls and not over CORBA or RMI. Besides the designated functionalities of the contained services, this component also performs some administrative checks, e.g. the same user does not join twice the same session. It is possible however for the same user enter two different session that are both under the supervision of the same **Monitor**, **Access** and **Management** component.

As the front-end for our demonstration we used a simple drawing application implemented on Java and using CORBA for communication. The architecture of this application is server-based: each participant performing an action on some shape on the virtual canvas is actually calling a method on the application server which in turn notifies all participants (after obtaining their IDs from the *Management Service*) about the performed action. The presence of an application server which serializes all communications among session participants simplifies our prototyping effort by alleviating the need of providing the *Synchronization Service* which has associated a serious implementation complexity. In order to avoid problems related to the

presence of a centralized server which represents a single point of failure that can block the whole session, we have modified the application so that the application server resides in one of the participants' mobile terminal and any of the participants can launch a new application server in case the current one fails.

Each session participant runs a CORBA client and the interactions of participants is based on the *Model-View-Controller* design pattern [10]. The *View* part consists of a container responsible for drawing the content on the canvas which is notified whenever this content (the *Model*) is altered. The *Controller* part is implemented by a set of very basic tools that can be used to manipulate the canvas contents. The most important part of the client application is the *Model* part, which consists of a list of shapes that is a replica of the one maintained by the application server. Each time a tool try to manipulate some canvas content the corresponding operation is relayed to the application server, which broadcasts the change (if approved by the *Access Control Service*) to all the replicas.

We have run a set of tests to verify the correctness of the application and the middleware services. The tests were performed on a network of Windows 95/98 laptops and PSION netBooks and NOKIA 9210s running Symbian OS R6 and connected by a combination of WLAN, Bluetooth and serial cable links. The behavior of the modified drawing application and the middleware services was stable under a number of regular test cases, disconnection scenarios and failure schemes. The testing of the demonstrator will be completed near the end of this year when the Bluetooth connectivity kit for NOKIA 9210 will be available and which will permit us to execute the demonstrator in an all-wireless network.

5 Conclusion

In this document we have argued about the motivation for support at the middleware level for mobile collaboration and we have presented an outline of the conception (requirements elicitation and analysis, service and architectural component description) of the middleware services which meet that need. Our prototyping activity has also been summarized showing how painlessly the developed middleware services can be used with an ordinary collaborative application.

Conceptually, our work does not contribute any innovation to the domain of collaborative systems. What makes our work interesting is the combination of collaborative aspects of distributed systems with wireless connectivity technologies for mobile terminals and the special attention that have attracted issues

related to the communication continuity and connectivity switching. The practical interest of our work will rise when hand-held devices like the PSION netBook and the NOKIA 9210 will have a commercial strength Bluetooth connectivity support and the need for supporting mobile collaboration will become more apparent than ever before.

Our plan for the immediate future is to extend our CORBA based middleware with a set of *bridge* components which will allow the participation to a session of users that use the RMI or the socket version of the same drawing application. To complement this effort for supporting a set of heterogeneous communication means, the part of the *Dynamic Reconfiguration Service* (see **R12**) that is responsible for the seamless transition from WLAN to Bluetooth communication is in its final implementation phase.

References

- [1] The WAP Forum. *Wireless Application Protocol (WAP)*. www.wap.com
- [2] The VIVIAN Consortium. *Opening Mobile Platforms for the Development of Component-based Applications*. www-nrc.nokia.com/Vivian
- [3] Object Management Group. *Wireless Access and Terminal Mobility in CORBA*. www.omg.org/cgi-bin/doc?telecom/01-03-01
- [4] Association for Computing Machinery. *The ACM Conference on Computer Supported Collaborative Work (CSCW)*. www.acm.org/pubs/contents/proceedings/series/cscw
- [5] ISO/IEC 8802-11; ANSI/IEEE Std 802.11. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification*, 1999. www.ieee.org/ieeexplore
- [6] The Official Bluetooth Site. *Bluetooth Specification v.1.1*. www.bluetooth.com
- [7] GSM World. *General Packet Radio Services (GPRS)*. www.gsmworld.com
- [8] UMTS Forum. *The Universal Mobile Telecommunications System*. www.umts-forum.org
- [9] M. Hughes. *Networking our Whiteboard with Java 1.1*. www.javaworld.com/javaworld/jw-11-1997/jw-11-step.html
- [10] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns*. Reading, Massachusettes, 1995.