# Mobile Access to MPEG-7 Based Multimedia Services

Yiwei Cao, Matthias Jarke, Ralf Klamma, Oscar Mendoza, and Satish Srirama
Lehrstuhl für Informatik 5
RWTH Aachen University
Ahornstr. 55, 52056 Aachen Germany
{cao|jarke|klamma|mendoza|srirama}@dbis.rwth-aachen.de

*Abstract*—**Multimedia information systems have been developed into service-ware. With the paradigms of web services, service oriented architectures (SOA), and Web 2.0 widgets, multimedia has become truly ubiquitous. However, interoperability, scalability, reliability and security are arising challenges at mobile multimedia service development. This paper focuses on the analysis, design, development and evaluation of a middleware that allows access from mobile devices to a bundle of multimedia services. The services are based on the international multimedia metadata description standard MPEG-7. The implementation is based on new generation of service-oriented application servers called Lightweight Application Server (LAS). Mobile web services refers to the fact that mobile servers host web services. A prototype was developed as a proof of concept, showing how to access MPEG-7 based multimedia services from a mobile host and the analysis results of providing MPEG-7 based multimedia services in the form of web services from the mobile host to other mobile devices. An alternative solution is to apply enterprise service bus as middleware. The performance evaluation results of both approaches show the reliable accessibility of MPEG-7 based multimedia services via the enterprise service bus solution.**

***Keywords-mobile data management; MPEG-7; SOA; web services; mobile web services, middleware; multimedia; enterprise service bus***

## I. INTRODUCTION

Developing community information systems on various mobile handheld devices is one of the most promising research areas of future mobile computing. Next to games, location-based services for social networks on the iPhone have been the biggest force in accelerating the changes showing mobile devices can do. Although mobile networks are getting much faster, there is still much challenges in practice. Theoretically 3G cellular, advanced Wi-Fi, and Mobile WiMax provide maximum data rates of 3.6, 100, and 70 megabits per second, respectively [18]. There is often a broad problem between the theoretical and the practical settings, especially in the research area of mobile computing. Therefore, community-based multimedia services are a true performance challenge for mobile data communication and management.

Also the development of middleware for information systems on mobile devices is a challenging. Firstly, it is constrained by limited resource capacity of handheld devices [17]. Secondly, middleware on desktops builds up a framework to support data communication and management in a better way. It brings also additional communication cost. The middleware facilitates hosting mobile web services on mobile devices that enable them to be service providers in addition to consumers. Because of the high costs in developing middleware infrastructure we aim at a high level of interoperability on the data management level. MPEG-7 [12] is a well-established an widely used standard in multimedia data management. However, due to its inherent complexity it was not used in mobile data management that often. With new initiatives like the application profiles the use of MPEG-7 has become much easier, also for mobile data management. New services could be developed and deployed to achieve accessibility to MPEG-7 based multimedia services while being mobile. Since the middleware is capable of publishing services from a mobile device, Peer-to-Peer (P2P) service consumption is also possible. This enables sharing of possible MPEG-7 based multimedia services with mobile peers in P2P manner.

Within the research cluster Ultra High-Speed Mobile Information and Communication (UMIC) in the German Excellence Initiative, this research work focuses on realizing the distributed infrastructure for mobile web services. We use mobile multimedia information systems as scenarios to make clear the future requirements in this area. Examples for such scenarios are mobile health care, mobile tourists guides, and mobile cultural heritage management. As part of the later scenario, we developed a mobile social software called Virtual Campfire [4], Virtual Campfire, is an advanced framework to create, search, and share multimedia artifacts with context awareness across communities. We have chosen this scenario because its high demands for scalability (multimedia information for cultural heritage management may contain image collections, drawings, videos, 3D animations etc.), reliability (information management must be guaranteed also in remote areas and developing countries like Afghanistan) and security (the existences and location of cultural artifacts may not be disclosed to tomb raiders). The framework provides a set of multimedia processor components to communities connecting the many different multimedia data sources. Connections are made context-aware concerning time, place and community [5]. The framework is built on top of services implemented on the Lightweight Application Server (LAS).

The Lightweight Application Server (LAS) hosts a set of web services such as the high-security user management service and MPEG-7 based multimedia annotation, sharing and deploying services. With the core services service designers can create community information systems to facilitate and support professional communities. LAS services access user profiles, multimedia and metadata from database servers. In addition, LAS uses also streaming servers as multimedia repositories. Streaming is a technology creating new demands in bandwidth in mobile communication.

Many community information systems have been built on top of this framework including MIST, a MPEG-7 based non-linear digital storytelling system, NMV, a MPEG-7 multimedia tagging system, ACIS, a GIS enabled multimedia information system, or CAAS, a mobile application for context-aware search and retrieval of multimedia and community members, They proved LAS as a service-oriented application server capable running highly interoperable MPEG-7 based multimedia services. With the increasing users' mobility needs and with the rapid development of mobile technologies, new challenges to establish a mobile LAS framework with distributed LAS services are posed. One major goal is the mobilization and distribution of LAS services.

The challenges for mobilization and distribution of MPEG-7 based multimedia services are as follows. Firstly, there are bottlenecks concerning the multimedia communication between servers and the backend databases. Secondly, the distribution of MPEG-7 based services to a wide range of service consumers is partly limited, because they are not implemented according to the W3C web service standard.

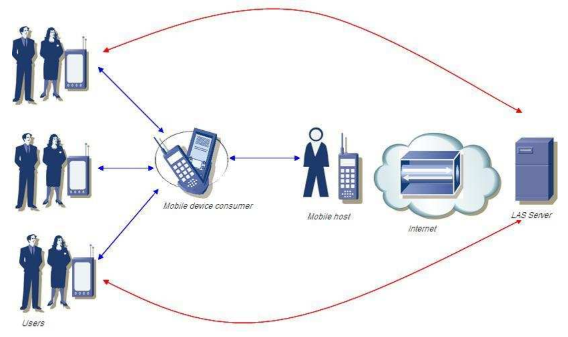


Figure 1. A scenario of mobile LAS services

A scenario is demonstrating how user communities might use mobile handheld devices to share MPEG-7 based multimedia services among themselves in order either to reduce communication costs or to set up an ad-hoc community network (cf. Fig. 1).

- Reducing cost: LAS users Alice and Bob are visiting a museum and want to use the MPEG-7 based storytelling service. Alice's smart phone has a flat rate GPRS contract, while Bob needs to pay a lot for each GPRS data transmission on his PDA. To reduce Bob's communication fee, an e.g. Bluetooth connection can be set up between Bob's PDA and Alice's smart phone. So Bob and Alice can both use different MPEG-7 based services while only Alice uses directly the MPEG-7 based services deployed on the server.
- Ad-hoc community network: A group of cultural scientists are working on excavation of an archaeological site and are equipped with laptops or PDAs that have limited resources. They communicate among themselves via some low-cost ad-hoc network technologies such as wireless mesh networks. They wish to share MPEG-7 based services among themselves, in order to reduce resource occupation on their own devices.

The rest of paper is organized as follows. Section 2 describes basic concepts such as service-oriented architecture concepts and web service technologies. Section 3 presents the mobile host, the LAS server, and the design of two different types of architecture for connecting mobile devices to LAS. The first architecture allows mobile devices to connect to LAS with mobile web services provided by a mobile host. The second architecture allows mobile devices to connect to LAS through an enterprise service bus middleware platform. Section 4 describes the software implementation and evaluation of both approaches. Section 5 concludes the paper with some suggestions for future work.

## II. Related Work

Before discussing the approaches to mobile access of LAS services, we give a short overview of related technologies including Service Oriented Architecture, web services, mobile web services and enterprise integration solutions as middleware.

### A. *Service Oriented Architecture (SOA) and Web Services*

Service Oriented Architecture (SOA) [3] is a component model that delivers application functionality as services to end-user applications and other services, bringing the benefits of loose coupling and encapsulation to the enterprise application integration. The definition of a Service Oriented Architecture is highly related to business/enterprise applications. From the technical point of view, SOA is like any other distributed architecture in that it enables you to build applications that use components across separate domains boundaries, however, SOA is also unique in that it incorporates those factors that are critically important to business: service reliability, message integrity, transactional integrity, and message security [8].

During the last years, Service-oriented design is becoming more popular. Service orientation retains the benefits of component-based development (self-description, encapsulation, dynamic discovery and loading), but there is a shift in paradigm from remotely invoking methods on objects, to one of passing messages between services. A service is generally implemented as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled, message-based communication model.

Meanwhile web services have evolved as the best means of achieving SOA. Web services are distributed software components which can be accessed over the Internet using well established web mechanisms and XML-based open standards and transport protocols such as SOAP and HTTP (HyperText Transfer Protocol). Public interfaces of web services are defined and described using W3C (World Wide Web Consortium) based standard, Web Service Description Language (WSDL). Web Services are moving Internet from *program to user, business to consumer (B2C)* interactions to *program to program, business to business* interactions [7].

### *B. Mobile Applications and Mobile Web Services*

Concurrent to the developments in SOA domain, the capabilities of high-end mobile phones and PDAs (Personal Digital Assistant) have increased significantly, both in terms of processing powers and memory capabilities. The smart phones are becoming pervasive and are being used in wide range of applications like location based services, mobile banking services, ubiquitous computing etc. The higher data transmission rates achieved in wireless domains with Third Generation (3G) and Fourth Generation (4G) technologies and the fast creeping of all-ip broadband based mobile networks also boosted this growth in the cellular market. The situation brings out a large scope and demand for software applications for such high-end smart phones.

Although mobile commerce poses to bring tremendous opportunities, we have to be cautious and understand the risks. Historically, technology adoption was never a smooth or linear process in cellular domain. Only recently, the Java technology is emerging as one of the most important enablers for mobile enterprise solutions. All major mobile device vendors, including Nokia, Motorola, Siemens, Samsung, Fujitsu, Mitsubishi, NEC, Panasonic, and Sony, have adopted J2ME as part of their core strategy for future smart devices. Major wireless carriers such as NexTel, SprintPCS, and ATT have committed to support Java devices and applications on their networks. Moreover, the communication between mobile nodes involved proprietary and application- and terminal-specific interfaces.

To meet the demand of the cellular domain and to reap the benefits of the fast developing web services domain and standards, the scope of the mobile terminals as both web services clients and providers is being observed. Mobile web services enable communication via open XML web service interfaces and standardized protocols also on the radio link. To support the mobile web services, there exist several organisations such as Open Mobile Alliance (OMA), Liberty Alliance (LA) on the specifications front; some practical data service applications such as OTA (over-the-air provisioning), application handover etc. on the commercial front; and SUN, IBM toolkits on the development front. Thus, though this is early stages, we can safely assume that mobile web services are the road ahead.

Mobile terminals accessing the web services cater for anytime and anywhere access to services. Some interesting mobile web service applications are the provisioning of services like e-mail, information search, language translation, company news etc. for employees who travel regularly. There are also many public web services accessible from smart phones like the weather forecast, stock quotes etc. Mobile web service clients are also significant in the geospatial and location based services [2]. While mobile web service clients are common, the research with providing web services from smart phones is still sparse. The scope of mobile web service provisioning was studied by two projects at RWTH Aachen University since 2003 [15][6], where Mobile Hosts were developed, capable of providing basic web services from smart phones. The Mobile Host was developed in PersonalJava, later upgraded to J2ME, on a SonyEricsson P800 smart phone. Open source kSOAP2 was used for creating and handling the SOAP messages. Once the Mobile Host was developed, an extensive performance analysis was conducted, proving its technical feasibility [15].

Mobile Hosts enable seamless integration of user-specific services to the enterprise. Moreover services provided by the Mobile Host can be integrated with larger enterprise services bringing added value to these services. For example, services can be provided to the mobile user based on his up-to-date user profile. The profile details like the device capabilities, network capabilities, location details etc. can be obtained from the mobile at runtime and can be used in providing most relevant services like maps specific to devices and location information. Besides Mobile Hosts can collaborate among themselves and bring value to the enterprise. [16]

### *C. Enterprise Integration*

Nowadays enterprises are not limited to the physical boundaries of an organization. The proliferation of open systems and open IT infrastructures increases the possibility of interoperability not only between platforms, runtimes, and languages but also across enterprises. When our concerns shift from networked systems to networked enterprises, a whole lot of opportunities open up to interact with enterprise applications and the opportunities are endless.

However, many of today's companies have Line of Businesses (LOB) and systems within a single enterprise that were not intended to interact together. Moreover these enterprise networks generally deploy disparate applications, platforms, and business processes that need to communicate or exchange data with each other. The applications, platforms and processes of enterprise networks generally have non-compatible data formats and non-compatible communications protocols. This leads to serious integration troubles within the networks. The integration problem extends further if two or more of such enterprise networks have to communicate among themselves.There are basically four different alternatives for integrating such disparate applications.

*Point-To-Point*: One of the first methods used to integrate applications has been by using point-to-point integration solutions. Under this scheme a protocol or format transformer is built at one or either end between a pair of applications. One of the advantages of this scheme is that both applications have good knowledge about each other thus getting to a tight coupling. The principal disadvantage of this scheme is the difficulty to integrate a new application to the system due to the high number of protocols or format transformers which have to be implemented. This architecture is very popular solution for small scale integration problems.

*Hub-And-Spoke*: This architecture is also known as the *message broker* and it provides a centralized point where all applications connect to by using *lightweight connectors*. This centralized point is called hub or broker. The connectors act as adapters and translate data and messages between different applications to canonical formats. Two of the advantages of the hub-and-spoke architecture are that all the message transformation and routing is done by the hub and the number of connections for the integration is reduced with respect to the point-to-point architecture. Unfortunately like any centralized architecture, the centralized hub becomes a single point of failure.

*Enterprise Message Bus*: an enterprise message bus is an infrastructure of communication where the integration between applications can be done in a platform and language independent way. It is composed by a message router and publishing (subscription) channels where applications use request and response queues to interact with each other via a message. Two new concepts are introduced here. They are applications which provide services, called *providers*, and applications which consume services, called *consumers*. Consumers write request to the request queue and providers listen to the request queue waiting for request to their services. The result messages are then added to the response message queues.

*Enterprise Service Bus*: By using Enterprise Service Bus integration [10], applications do not communicate to each other directly, they communicate by using a SOA based middleware backbone. ESB basically consists of a set of service containers that are interconnected with a reliable messaging bus. The ESB supports multiple integration paradigms in order to fully support the variety of interaction patterns that are required in a comprehensive SOA between these service containers. So it has support for *service-oriented architectures* in which applications communicate through reusable services with well-defined and explicit interfaces, *message-driven architectures* in which applications send messages through the ESB to receiving applications and *event-driven architectures* in which applications generate and consume messages independently of one another.

Thus, we adapted an ESB based middleware to integrate LAS services as the other approach in comparison to the Mobile Web Service solution.

## III. Design of Mobile Access to LAS Services

Due to the current increasing development of mobile devices, technologies and connectivity, everyday a lot of end users are demanding to have more access and capabilities on their own mobile devices. Most cell phones have audio and video features which are used by people for fun, business and research.

### *A. The Light Application Server (LAS)*

First of all, we give an insight into the concept and architecture of LAS. Since a couple of years different kinds of internet communities, either of interest or of practice have emerged. All of those communities use different applications to interactively cooperate, coordinate and communicate. A vast number of systems offering community services was created. Some of them were commonly used in many systems (e.g. calendar, forums, polls, etc.), while others were tailored to a community-specific task. But what if all of those communities and their tools could be maintained centrally? What if communities could extend their service portfolio and share it with other communities? This is, where LAS comes into play. LAS is a lightweight application server designed as a community middleware that is capable of managing users and multiple hierarchically structured communities along with their particular access rights as well as a set of services accessible to users. LAS also includes security management to guarantee access control. LAS is implemented in Java. A community application can make use of the offered services by simply connecting to the server and then remotely invoking service methods. A very prominent feature of LAS is that the server functionality is easily extensible by implementing and plugging in new services resp. components. The main intent of this document is to encourage and enable you to develop and use own LAS extensions tailored to the needs of your communities.

LAS serves as a server and perform many tasks. LAS is capable of managing users and multiple hierarchically structured communities along with their particular access rights as well as a set of services accessible to users. LAS also guarantees access control by including security management. LAS is a platform independent JAVA implementation that can be combined with different tools and communities. A community application can make use of the offered services by simply connecting to the server and then remotely invoking service methods. A very prominent feature of LAS is that the server functionality is easily extensible by implementing and plugging in new services respective components [14].

Development of mobile technologies during last years is demanding the creation of mobile social software that is able to run on mobile devices and to offer the same functionality like the offered by desktop applications. However, in practice, some weaknesses of LAS become evident. Even though, LAS is a reliable application server offering MPEG-7 multimedia services, it is not pure web service architecture, it was not designed under the SOA paradigm and important aspects like scalability and distributed services were not taken into account.

Quality of service and performance problems have been observed recently by LAS users. For many years, LAS has been used on top of traditional networks infrastructures for providing the services required by social software such as Virtual Campfire, now some researches on mobile technologies focus on the problem of offering LAS services to mobile social software by using new affordable communication technologies.

In technical terms, the LAS Java API based on J2SE can be used to extend the server functionality by adding three basic element types: *LAS Connectors* realize client-server communication using a particular protocol. LAS come with connectors for HTTP and SOAP, while the current developed SOAP connector just provided limited support to Web Services. *LAS Components* are internal elements encapsulating functionality for common tasks shared by services or other components. *LAS Services* define the actual functionality that LAS offers to its clients including core services and service mash-ups. A service defines public methods which can be remotely invoked by clients via one of the connectors (cf. Fig. 2).

LAS user-, object- and session manager build up the so called LAS core services. *Session Manager* maintains a session for a user connection and dispatch the invocation to the respective service, if the user has sufficient access rights for accessing service methods and possibly security objects that are used inside service methods. *LAS user manager* maintains users, hierarchically organized groups of users and user roles defining access rights to service methods. *LAS object manager* maintains security objects of arbitrary types and their access rights.

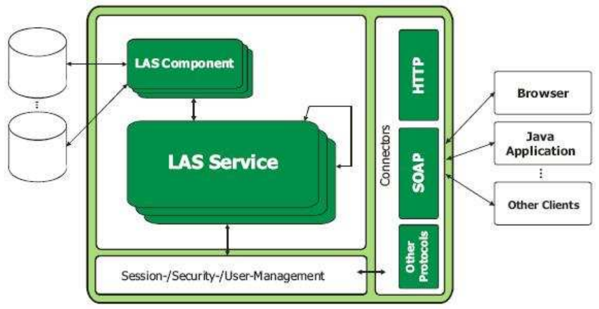


Figure 2. LAS Architecture [5]

Based on the technical requirements of LAS platform, an application to access LAS services needs to be developed, including:

- Requirement elicitation of LAS Mobile services;
- Compatibility analyses of LAS service messaging constructs with the existing middleware;
- LAS Mobile Web Services;
- LAS Application Midlet;
- Mobile LAS Communication Module;
- LAS Data Management Module.

The services deployed on the middleware could be synchronous or asynchronous in nature based on the application requirements. Among other requirements, this should be identified during the requirement elicitation phase of the project.

On the LAS servers, the base requirements on MPEG-7 based multimedia services are as follows. There should not be one monolithic MPEG-7 service, but a set of services, each of them offering functionality for a given domain of the MPEG-7 standard (e.g. multimedia content, collections, classification schemes, etc.). Since MPEG-7 is XML-based, the basic functionality of an MPEG-7 LAS service, i.e. its methods, is to work on XML documents conforming to the MPEG-7 XML schema. MPEG-7 documents should be stored in an XML database, that supports execution of XPath, XQuery and XUpdate expressions. An MPEG-7 service should offer support for the following basic tasks:

- Creation of new MPEG-7 descriptions
- Query for complete or partial MPEG-7 descriptions
- Modification of MPEG-7 descriptions

Each MPEG-7 XML document must be valid against the MPEG-7 XML schema at all stages. This includes, that after having performed modification operations on a valid document, the resulting document is valid again. Work of an MPEG-7 service developer should be supported by tools, that guarantee validity.

MPEG-7 based multimedia services refer to services such as nonliear digital storytelling, multimedia upload, download and tagging/annotation services. They all make use of the the aforementioned basic MPEG-7 services. They are deployed on the LAS server and are a sub category of the LAS services. To enable the access of those services. We proposed two types of architecture in the following subsections.

## *B. Accessing LAS Services through Mobile Web Services*

Since LAS as of now supports the access of services only across the HTTP protocol, the invocation of these services from mobile phones becomes a little tricky. While studying mobile web service clients and providers, we have taken the design decision that the exposure and access of services from/to mobiles would be only through web services and WSDL. Mobile web services basically enable communication via open XML web service interfaces and standardized protocols also on the radio link, where today still proprietary and application- and terminal-specific interfaces are required.

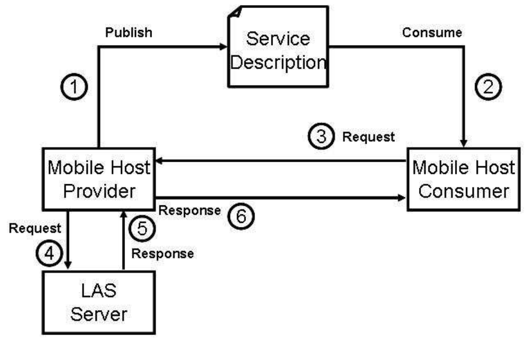


Figure 3. The architecture to access LAS with Mobile Web Services

Following this design decision, Fig. 3 shows the proposed architecture that will help in connecting to LAS services with mobile web services. In the figure there are two smart phones, one of them acting as a mobile web services provider while the other acting as a mobile web services consumer. Here basically the Mobile Host provides a web service interface of the LAS services, for the external nodes. Under this architecture only the Mobile Host directly connects to the LAS and it is the only access point to the LAS services from mobiles. Theoretically the Mobile Host can be replaced by a standalone node that provides web service interfaces for the LAS services. Since we are also interested in personalized services and user specific profiles, we were still proceeding with Mobile Host. Services provided by the Mobile Host are published using WSDL in order to enable the consumer to find those services and invoke them. Once the services are found, the consumer can bind to them and use them. Srirama et al. have studied alternatives for mobíle web service discovery using the features of P2P networks. The discussion of the solution is beyond the scope of the paper and can be found at [15], for enthusiastic reader.

### C. Mobile Access through ESB

Every day the usage of LAS at our chair of Information Sytems Group becomes more frequent and the number of users is continuously increasing. That is demanding a high performance and scalability of LAS which is putting it to the test. Most users are asking for LAS services in a distributed way which offers a better performance and response times.

If one takes into account that today LAS is offering its services to multiple users who have different purposes, it would be good to think about LAS like a set of servers, acting as a cluster, with each of them specialized in a specific field and offering services related to that field. However, if many users are interested in the same kind of services, the server will be overloaded and its performance will degrade and users will be unsatisfied. That is why this subsection introduces enterprise service bus based architecture like a middleware solution which provides distribution of LAS services from different LAS servers under a service oriented architecture point of view.

The main advantage of an ESB middleware solution is its transparency to users and programmers on the client side as well as on the server side. Inside LAS there are no necessary changes to do in order to couple it with an ESB and users of LAS only need to connect to a single point, the ESB, in order to access any LAS server they are interested in. This solution adds also the common advantages offered by ESB architecture, such as support for multiple integration paradigms (i.e. Service-oriented architectures, Message-driven architectures and Event-driven architectures), guaranteed delivery, control centralization and distributed processing.

Imagine a set of LAS servers, which could be different instances of the same implementation or servers completely different from each other. Now consider a group of Mobile Hosts or mobile web service clients trying to access many services from those LAS servers. The current way to achieve this goal would be configure each Mobile Host with the specific connection to the right LAS server based on the services offered by it. What is proposed here is implementing an ESB middleware solution in charge of receiving requests from many Mobile Hosts and processing connections to different LAS servers which are able to response those requests successfully (cf. Fig. 4). The primary advantage of such an approach is that it reduces the number of point-to-point connections required to allow applications to communicate with LAS servers. This, in turn, makes impact analysis for major software changes simpler and more straightforward. By reducing the number of points-of-contact to a particular application, the process of adapting a system to changes in one of its components becomes easier.

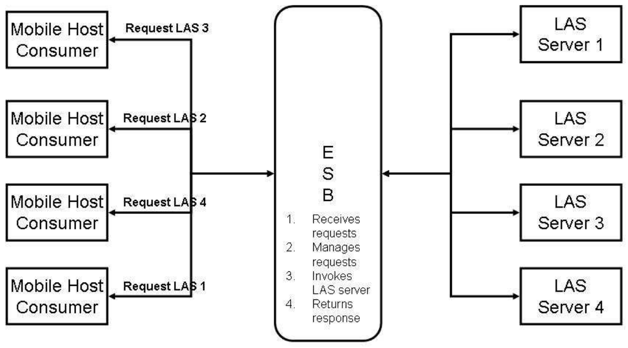


Figure 4. Architecture of accessing LAS services through ESB

Under this new architecture the ESB should be able to replace all direct contact between Mobile Hosts and LAS servers, so that all communication takes place via the bus. As described in Figure 4, the ESB must receive all the requests from Mobile Hosts and apply a management process to select the right LAS server to be called. Once the LAS server has been selected, the ESB must call the appropriate service. As soon as the response from the LAS server is received by the ESB, it has to encapsulate the response and forward it to the Mobile Host requester. This is typically accomplished through the use of an enterprise message model. The message model defines a standard set of messages that the ESB will both transmit and receive.

## IV. Implementation and Evaluation

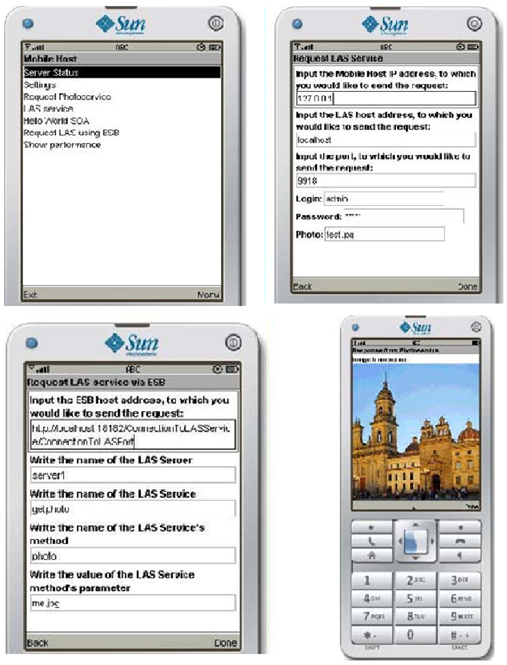
Since this research work is in principal divided into the development of a connection to LAS services with Mobile Web Services and the development of a connection to LAS services through an ESB, the evaluation is also described separately for each part. An evaluation of the photo service functionality was also performed and is presented like a point of comparison to see the effect of adding a connection to LAS server to the basic Mobile Host architecture.

### A. Implementation of Both Types of Architecture

With the approach of invoking mobile web services, by creating a request, the mobile application is creating the object in charge of preparing the SOAP message read later by the web service provider. The web service requester needs to encapsulate into a SOAP envelope all the parameters needed by the Mobile Web Service invoked. The SOAP envelope is sent to the web service provider that invokes the Mobile Web Service and returns its response. The Mobile Web Service response is encapsulated into the SOAP message and returned to the mobile application requester. At the end, the mobile application requester is able to process the response received.

With the approach of invoking the MPEG-7 based multimedia service through an ESB, OpenESB is an open source platform for business and enterprise integration and for

Service Oriented Architectures [9]. The wide range of components included in this platform involves communication with back-end systems and ESBs and document transformation, process transaction, among others. All the components are interconnected with a messaging bus called the Normalized Message Router (JBI bus). All the message exchanges performed in OpenESB are done using standardized message exchange patterns (MEP) based on abstract WSDL. By creating a request, the mobile application is creating the object in charge of preparing the SOAP message read later by the ESB. The SOAP message is sent to the Enterprise Service Bus. The ESB must read internally the SOAP message and get the information needed. That information is passed to an Enterprise Java Bean, created in this case to perform the LAS server selection and the connection to the server selected. Once the LAS server returns the response, it is encapsulated by the ESB into the SOAP message that will be returned to the mobile application. At the end, the mobile application requester is able to process the response received.



Request LAS Service

Input the Mobile Host IP address, to which you would like to send the request: 127.0.0.1

Input the LAS host address, to which you would like to send the request: localhost

Input the port, to which you would like to send the request: 9918

Login: admin

Password: *****

Photo: test.jpg

Request LAS service via ESB

Input the ESB host address, to which you would like to send the request: http://localhost:18182/ConnectionToLASService/ConnectionToLASPort

Write the name of the LAS Server: server1

Write the name of the LAS Service: getphoto

Write the name of the LAS Service's method: photo

Write the value of the LAS Service method's parameter: me.jpg

Figure 5. Screenshots of the mobile clients

The prototype is run on the Sun Wireless Toolkit emulator (cf. Fig. 5). Users are able to check whether the Mobile Host server runs properly by clicking the menu of Server Status. Users can set the Mobile Host in debug mode to see all messages on the console and configure the IP address of the Mobile Host, using the settings menu. By using the option of the LASViaESB menu, the MPEG-7 based multimedia service is invoked via the ESB, instead of via the Mobile Web Services.

For the system test and evaluation, it is important to analyze not only the fulfillment of requirements but also the application performance. Detailed challenges are discussed in depth in [1]. In mobile environment users need reasonable response times from central servers even with limited hardware resources. Most mobile devices have limited resources, low processing capacity and low data transmission rates, but users want their mobile devices to response as well as a laptop or personal computer and the application developer must try to satisfy as well as possible that desire.

### B. *Evaluation of Connection with Mobile Web Services*

Due to the simple functionality offered by the mobile application developed, the set of tests performed to evaluate the viability of the application were focused on measuring the performance of the application under different circumstances.

In order to test the performance of the connection to LAS with Mobile Web Services, the following test case was developed. Two Mobile Hosts were executed on the emulator provided by the Wireless Toolkit 2.5.2. One Mobile Host was acting like server while the other one was acting like client. During the test case the Mobile Host client requested each picture to the Mobile Host Server that connects to the LAS service and gets the picture. The request round trip of that process was measured and is analyzed in the present section. In order to have a stable base for the analysis, all the requests were repeated 10 times and the first request to the server was always omitted to avoid measuring the initialization process on the server side. This case test was designed in the same way as the case test designed for the Photoservice functionality in order to be able to compare them.

The first variation observed after including the connection to LAS Services to the basic Mobile Host architecture is the growth in the size of the request. This increasing amount is almost 178 bytes on average due to the overhead caused by the LAS URL, the listening port, login and password that must be informed to the Mobile Host in order to create a successful connection. Even though this kind of request could be designed in a different way, an overhead in the request is almost inevitable because additional information must be transmitted.

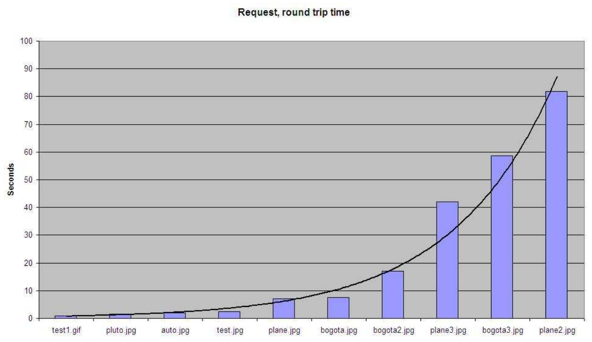


Figure 6. Round trip time of request handling using Mobile Web Services

Since the response obtained from the LAS server is exactly the same as the one obtained from the Mobile Host in the Photoservice case test, no variation in the size of the response is observed, and the same almost linear behavior is obtained. This is due to the fact that no additional information must be added to the response when it comes from the LAS server to the Mobile Host server and only the encoded picture is transmitted.

Comparing the required time for processing the response between the Photoservice functionality and the LASService functionality, an average increase of only 0.037 milliseconds is noticed which is negligible and one might assume that no increase is observed. That variation might be caused due to the environment used for testing, where the LAS Server, Mobile Host server and Mobile Host client were running on the same computer and processing time must be shared between all the applications.

The most important result of this case test is the total time elapsed since the request is sent by the Mobile Host client until the picture is received. This time is the time perceived by the final user and is one of the decisive factors for a user to select an application. The introduction of a connection to a LAS server with Mobile Web Services using a Mobile Host server as middleware is not a viable solution, at least it is not a viable solution for the transmission of pictures from Mobile Hosts to other mobile devices. The average increase in the round trip time of the LASService request with respect to the round trip time of the Photoservice request is 2.95 times greater, which means that a user has to wait 151 seconds more for a picture of 96.5Kb. The response time obtained for connections to LAS services with Mobile Web Services are not competitive in the real market, where users expect high performance applications and response time relatively short. Fig. 6 shows how it takes for each picture to be received on the Mobile Host client.

### C. *Connection Through ESB*

Due to the new functionality implemented on the Enterprise Service Bus, additional to the existing functionality on the Mobile Host side, three different tests were performed for this architecture. A set of unit test was developed to ensure the correctness of the ESB and the EJB implemented; a set of performance tests similar to those performed on the Mobile Host for testing the viability of the architecture from the user point of view, it means, focusing the attention on response times; and a set of tests for showing the load balancing done by the ESB using two instances of a LAS server for attending requests.

In order to test the correctness of the Enterprise Service Bus and the Enterprise Java Bean implementation, a *composite application* (CA) was developed in NetBeans. The JBI Composite Application (CA) project system allows the developer to create a Service Assembly (SA) artifact that contains all the subprojects as Service Units (SUs). The SA can then be deployed to the JBI component containers on the JBI Meta-container. It provides a project folder with subfolders on the local file system for storing project specific design and configuration data.

The ConnectionToLASPort consumer endpoint connects to the ConnectionToLASPartnerLink provider endpoint in order to call the service offered. Once the EJB processes the request, uses its consumer end point to return the response. That response is obtained by the ConnectionToLASPort by using its provider endpoint.

Once the correctness of the functionality of the Enterprise Service Bus was tested, a set of tests was designed to measure the performance of the architecture managing access to LAS services from mobile devices. The design follows the same schema used for testing the performance of the Photoservice functionality and the LASService functionality. In this case test, a Mobile Host acting like client and without running its internal server tries to access a LAS service by using a middleware architecture, in this case the Enterprise Service Bus.

Since the ESB does not add any additional content to the response obtained from LAS, the size of the response is also very similar to the size of the responses for the case tests analyzed. No important variation in the time for processing the response is observed either with respect to the other case tests, where only a difference of more or less 0,03 milliseconds is presented.

Even though the size of the response for all the functionalities (Photoservice, LASService and LASViaESB) is very similar, two important advantages regarding the request round trip time is observed. The first advantage is that the request round trip time increases linearly with respect to the picture size, contrary to the behavior observed for the Photoservice functionality and the LASViaESB functionality, where the increase was always exponential. The other advantage is the short time elapsed from the request is sent to the response is received. In this case the average time for the request round trip was 3.65 seconds, a much lower time than 22.08 seconds required in the Photoservice functionality and even lower than 65.24 seconds required in the connection to LAS service with Mobile Web Services (LASService functionality).

The advantages of using ESB as middleware to support LAS scalability are listed as follows:

- Transparency: No changes on the client side or on the LAS server must be done. The ESB is able to use the existing mechanisms on LAS for connecting to it and consume its services.
- Scalability: ESB is already an architecture highly scalable that follows all the integration patterns and standards existing on the market.
- Performance: Tests demonstrated that an ESB acting like a middleware connection between clients and LAS servers keeps a high performance and is even more efficient that having a direct connection between a Mobile Host and LAS.
- SOA paradigm: The introduction of an ESB for connecting to LAS brings a world of opportunities and introduces LAS to the Service Oriented Architecture paradigm.

The MobSOS test bed for multimedia community services [13] was used to see what was happening inside each LAS server during the test phase. Due to the fact that MobSOS was originally designed for monitoring only one LAS server, it was necessary to create two MobSOS databases and monitor each server separately. For measuring the load balancing done by the ESB in a randomly way, each picture was requested 10 times, therefore, one hundred requests were sent to the ESB from a mobile device and the results obtained using MobSOS are summarized in Table 1.

TABLE I. LOAD BALANCE BETWEEN TWO LAS SERVERS

| Measure | Server 1 | Server 2 |
|---|---|---|
| ***Number of connections*** | 52 | 48 |
| ***Total rrequest bytes*** | 7775 | 5788 |
| ***Average request bytes*** | 55.53 | 56.19 |
| ***Max. request bytes*** | 113 | 113 |
| ***Total response bytes*** | 3370982 | 2368598 |
| ***Average response bytes*** | 24078.44 | 22996.09 |
| ***Max. response bytes*** | 136258 | 136258 |

Even though the load balancing implemented on the ESB was only a random selection between two different servers, the statistics obtained from MobSOS show how equitable is the distribution of processing. 100 connections were established and almost a perfect distribution between both servers was obtained (52 percent for server 1 and 48 percent for server 2).

The request and the response with the maximum number of bytes is the same on both servers because it corresponds to the biggest picture requested. Other measures taken by MobSOS, such as the number of bytes requested, the number of bytes responded, and the average of those values, show how well distributed is the load on each server. Further load balancing criteria could be created to produce interesting results.

## *D. A Comparison between Mobile Web Services and ESB*

This comparison allows seeing how using an Enterprise Service Bus as middleware for connecting mobile devices to LAS services improves the response time and makes the developed application a viable solution for the real world requirements.

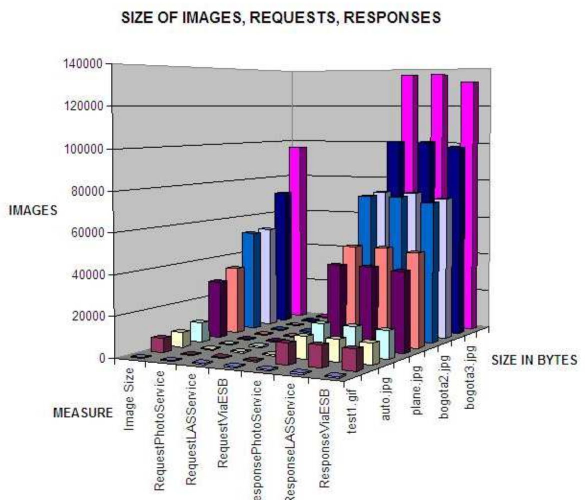


Figure 7. Test results of request round trip time through ESB

Fig. 7 shows the size of the images used during the testing phase, the size of the requests formed for getting the picture in each case test and the size of the response returned for all the implemented functionalities (Photoservice, LASService and LASViaESB). Even though the similarity between the data shown in Fig.7, it must be highlighted that connecting to LAS services through the implemented ESB reduces the average size of requests in 16 bytes and the average size of responses in 1537.7 bytes with respect to connecting to LAS services with Mobile Web Services.

Using an ESB like middleware for connecting to LAS from mobile devices reduces the average response processing time in 46 percent with respect to the connection done with Mobile Web Services using a Mobile Host like middleware.

The advantages of using ESB as middleware for connections to LAS services instead of using Mobile Web Services can be found clearly. The ESB keeps the response time perceived by the user almost constant in comparison with the exponential growth perceived when a Mobile Host and Mobile Web Services are used. The response time perceived by the user is inside competitive ranges for mobile applications (1-4 seconds). There is a low relation between the size of the picture and the response time perceived by the user (cf. Table 2).

TABLE II. A COMPARISON OF USING MOBILE WEB SERVICES AND ESB AS MIDDLEWARE

| Factors | Mobile Web Services | ESB as middleware |
|---|---|---|
| ***Performance*** | Response time is proportional to response size and increases exponentially | Response time increases linearly and at a low rate |
| ***Scalability*** | Not scalable, only one LAS server is connected | Scalable, distributed LAS services |
| ***Implemen-tation*** | Details must be known by clients | WSDL is used to publish and find Web services on the ESB |
| ***Viability*** | Response time makes it unfeasible | Short response time makes it attactive for users |

## V. CONCLUSIONS AND FUTURE WORK

This paper presents the ongoing research results of realization of mobile access to MPEG-7 based multimedia services. Two different kinds of architecture are applied: Mobile Web Services and Enterprise Service Bus as middleware.

The Mobile Web Service architecture makes use of a Mobile Host acting like a server and providing Mobile Web Services to connect to LAS. The function of this Mobile Host is processing requests from other mobile devices that consume its web service where the connection to LAS is encapsulated. The request as well as the response of both the mobile device and the Mobile Host is embedded inside SOAP messages created using the functionality offered by kSOAP. The connection to LAS is completely transparent for the mobile device which makes the request and it only needs to know how to consume the Mobile Web Service offered by the Mobile Host. This architecture is a good solution for mobile devices that do not have a direct connection to LAS but are able to consume Mobile Web Services provided by the Mobile Host. However, the response time for pictures whose size is greater than 1k makes this implementation few attractive for users.

The ESB architecture makes use of the OpenESB technology developed by Sun Microsystems for creating a single access point to LAS servers for multiple mobile devices. In this architecture, the LAS service is accessed directly from the ESB by using an Enterprise Java Bean. In this way the LAS service is encapsulated into a web service that can be consumed by any application that knows its WSDL file. Under this schema, applications that want to connect to LAS only need to be able to establish a connection to the ESB and consume the web service. This architecture follows the entire standards for integrating applications and SOA, and can be easily scalable since the OpenESB was designed from the beginning on with that characteristic.

Both kinds of architecture were put to the test and interesting performance results were found. The ESB solution advances in lowing request response time with higher scalability.

The attempts done in this project is only the beginning of a new research field around the LAS server. As the proof of concept, only the prototype of the Photoservice as one of the MPEG-7 based multimedia services running on the LAS server is tested. Further services including storytelling services, multimedia annotation services, and context-aware services etc. should be distributed via the mobile access in future research work. In addition, the current test data set does not cover a large amount of multimedia data, due to the complexity of multimedia retrieval evaluation processes [11]. Above all, the both types of architecture for mobile access to MPEG-7 based multimedia services need to be evaluted on large multimedia repositories.

## References

[1] A. Barbir, C. Hobbs, E. Bertino, F. Hirsch and L. Martino, “Challenges of Testing Web Services and Security in SOA Implementation”, L. Baresi, E. Di Nitto (Eds.): Test and Analysis of Web Services, Spring, 2007, pp. 395-440.

[2] B. Benatallah and Z. Maamar, “Introduction to the special issue on m-services”, IEEE transactions on systems, man, and cybernetics - part a: systems and humans, 33(6):665–666, November 2003.

[3] S. Burbeck, “The Tao of e-business services - The evolution of Web applications into service-oriented components with Web services”, IBM DeveloperWorks, October 2000.

[4] Y. Cao, M. Spaniol, R. Klamma, D. Renzel, “Virtual Campfire - A Mobile Social Software for Cross-Media Communities”, K. Tochtermann, H. Maurer, F. Kappe, A. Scharl (Eds.): Proceedings of I-Media'07, International Conference on New Media Technology and Semantic Systems, Graz, Austria, September 5 - 7, 2007, J.UCS (Journal of Universal Computer Science) Proceedings, pp. 192-195.

[5] Y. Cao, R. Klamma, M. Hou, M. Jake, “Follow Me, Follow You - Spatiotemporal Community Context Modeling and Adaptation for Mobile Information Systems”, X. Meng, H. Lei, S. Grumbach, H. V. Leong (Eds.): Proc. of the 9th International Conference on Mobile Data Management, April 27-30, 2008, Beijing, China, pp. 108-115.

[6] G. Gehlen, “Mobile Web Services - Concepts, Prototype, and Traffic Performance analysis”, PhD thesis, RWTH Aachen University, October 2007.

[7] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, ”Introduction to web services architecture”, IBM Systems Journal: New Developments in Web Services and E-commerce, 41(2):178–198, 2002.

[8] J. Hasan and M. Duran, “Expert Service-Oriented Architecture in C sharp”, Apress, 2006.

[9] F. Jennings and D. Salter, Building SOA-Based Composite Applications Using NetBeans IDE 6, Packt Publishing, 2008.

[10] M. Keen, S. Bishop, A. Hopkins, S. Milinski, C. Nott, R. Robinson, J. Adams, P. Verschueren, and A. Acharya, “Patterns: Implementing an SOA using an Enterprise Service Bus”, IBM RedBooks, July 2004.

[11] M. Lux, G. Dösinger and G. Beham, “User-Centered Multimedia Retrieval Evaluation Based on Empirical Research”, M. Granitzer, M. Lux and M. Spaniol (Eds.): Multimedia semantics – The Role of Metadata, Springer, 2008.

[12] J. M. Martinez, C. Gonzalez, O. Fernandez, C. Garcia, and J. de Ramon, “Towards universal access to content using MPEG-7”, In Proceedings of the 10th ACM International Conference on Multimedia, ACM Press, 2002, pp. 199–202.

[13] D. Renzel, R. Klamma, M. Spaniol, “MobSOS - A Testbed for Mobile Multimedia Community Services”, 9th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '08), May 7-9, 2008, Klagenfurt, Austria.

[14] M. Spaniol, R. Klamma, H. Janßen, D. Renzel, “LAS: A Lightweight Application Server for MPEG-7 Services in Community Engines”, K. Tochtermann, H. Maurer (Eds.): Proceedings of I-KNOW '06, 6th International Conference on Knowledge Management, Graz, Austria, September 6 - 8, 2006, J.UCS (Journal of Universal Computer Science) Proceedings, Springer, pp. 592-599.

[15] S. N. Srirama, M. Jarke, and W. Prinz, “Mobile web service provisioning”, In AICT-ICIW ’06: Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, page 120. IEEE Computer Society, 2006.

[16] S. N. Srirama, “Mobile Hosts in enterprise service integration”, PhD thesis, RWTH Aachen University, October 2008.

[17] H. Paulino, “Mobile Service Development and Deployment with Remotely Launched Service-Oriented Mobile Agents”, C. Becker, C.S. Jensen, J. Su and D. Nicklas (Eds.): 8th International Conference on Mobile Data Management (MDM 2007), Mannheim, Germany, May 7-11, 2007, IEEE, pp. 412-416.

[18] S. J. Vaughan-Nichols, “The mobile Web comes of Age”, IEEE Computer, vol. 41, No. 11, November, 2008, IEEE Computer Society, pp. 15-17.