Mobile services based on client-server or P2P architectures facing issues of context-awareness and heterogeneous environments

Robert Schmohl
Technical University of Munich
Department of Informatics
85748 Germany

Uwe Baumgarten
Technical University of Munich
Department of Informatics
85748 Germany

Abstract ***With the advent of mobile computing and mobile services becoming more widespread and more complex, the development process of those services requires an ever more important focus on numerous design issues. The heterogeneous environment, reaching from different networks and devices to different software platforms embedded on such devices, implies difficulties in making a mobile service broadly available. Another interesting aspects is, that mobility and personalization enable the creation of location based services which may be coupled with the user's context additionally. Thus, this circumstances require, that a mobile service provide an adequate location- and context-awareness.***

This paper enumerates some fundamental aspects of mobile service design summarized from current research results. At this, we focus on the aspects of heterogeneity and location-/context-awareness. Beginning with regarding the mobile services' environment, we consider the requirements of a mobile service and discuss the architectural design options. Subsequently, we explain how the issues of heterogeneity and context awareness impact the design process, and we present concepts how those issues are currently being addressed.

Keywords: mobile services, unstable environments, heterogeneity, context awareness, mobile p2p

1 Introduction

A mobile service can be defined as a radio-communication service between mobile and land stations, or between mobile stations [1]. The recent advancement of mobile computing in both hardware and software have enabled the commercial exploitation of value generated by such services on various levels [2]. However, the growing complexity and demand for mobile services imply a number of issues.

The recent advancement of mobile computing has also led to the emergence of a large variety of different networks and even a larger variety of mobile devices roaming in such networks. Due to this heterogeneous environment, mobile services can hardly be made generally accessible. Consider a SymbianOS operated smartphone roaming in a GSM network and a Windows XP driven laptop registered in an 802.11 wireless LAN, both trying to access the same service. Although the UI, while accessing the service, may be similar on both devices, the implementation will be different, since the service is accessed by two entirely different platforms in two different networks of different types.

Additionally, the technological advancement regarding both devices and network capabilities allows the design of more and more complex mobile services. Especially the possibilities to track a user's position the to adapt to his personal preferences require mobile services to handle both the location- and context awareness [3], resulting from location tracking and personalization respectively, more and more efficiently.

Recent research has spawned a number of middleware solutions to address both issues of heterogeneity and location-/context-awareness. After describing the background situation in section 2, we identify the requirements of a mobile service and discuss the architectural design options to develop it in sections 3 and 4, respectively. Subsequently, sections 5 and 6 analyze the mentioned problems of heterogeneity and context awareness, and present the current solutions how to approach them. Section 7 concludes this paper with a summarized result of the set of problems discussed here.

2 Background

This section briefly introduces the technology, that mobile services can be developed on. We discuss both the standards and the environment where these standards are implemented.

2.1 Technological standards

Networks Networks are the backbone systems for mobile services, providing the communication between the components providing the service and those accessing it. The existent networks and standards quite differ in size, reaching from satellite communication systems to Bluetooth picocells.

Mobile devices Mobile devices are primarily the accessors of mobile services, providing the users with an interface to use those services. Mobile devices are usually employed for the network purpose, i.e. cell phones are used for mobile communication in GSM networks.

That for, embedding mobile services on the devices depends on the devices' characteristics, such as available APIs, installed software, operating systems and present hardware specifications. However, the devices' properties are very heterogeneous. This implies challenges when making mobile services compatible on a highest possible amount of devices (discussed in section 5).

2.2 Environment

Unstable environments Mobile services are generally embedded into *unstable environments* [4], that can be described as such for the following reasons:

- *Vulnerable infrastructure* Mobile services are run in networks, that cannot guarantee unhindered provision of services. Since mobile clients may be moving, and since network coverage may be limited, the availability of a service may be limited as well (concerning reachability and sufficient bandwidth) and connections are likely to become transient [4].
- *Physical surrounding* Unstable environments can also be characterized by the physical conditions of a mobile client's surrounding - such as noise or lighting conditions - since it has impact on the usability of a mobile service [4].
- *Changing user context* The aspect of mobility implies the users' contexts being in a constant state of flux [4]. A user's context can depend on any imaginable user-personalized characteristic, such as personal preferences, level of trust, et cetera [3]. Together with the aspect of mobility, this volatility causes the context to shift continuously. This issue will be discussed more closely in section 6.
- *Wireless discovery* This aspect describes the issue of the "management of many small computing nodes comprising a large complex system" [5], comprising of two subordinate issues. First, a mobile node must be able discover other mobile nodes, and second, it must be able to communicate adequately with the them.

Mobile devices Another aspect of the environment to be taken into consideration is the mobile clients' hardware specifications, whereas we have identified two subordinate aspects. First, the spectrum of mobile hardware is very heterogeneous implying the problem to uniformly provide a service (see section 5). Second, because of the aspect of mobility, the hardware capabilities are bound to certain limitations, making it necessary to constraint the services adequately.

The hardware constraints especially address the following problems [5]:

- *Power management* Mobile devices have limited power supplies, so that all operations are bound to low power consumption, which can be divided into three categories: processing, storage and communication. In each of those categories, controlling power is achieved by either reducing performance or shutting down components when they are idle.
- *User interface adaption* The size of mobile devices imposes physical difficulties in utilizing their user interfaces. This - for example - applies for small display sizes and only a few input keys.

3 Requirements on a Mobile Service

This section briefly summarizes the requirements applying for the design of mobile services. First, we identify the quality issues, that differ from those applying for conventional services. Subsequently, we present design requirements implied by the characteristics of unstable environments discussed in section 2.2

3.1 Service Quality

A widely acknowledged approach on measuring service quality is to distinguish between the service process and the service outcome [6]. While the service outcome reflects its functionality, the service process can be measured by a number of aspects defining whether it meets the customer's expectations. Projecting this approach on mobile services, the following aspects significantly influence the process quality [4]:

- *Reliability* meaning that services are accurate and current.
- *Responsiveness* requiring the service to minimize response times. Together with the reliability requirement, this especially relevant since the clients' mobility may result in time-critical situations.
- *User interface* designed according to the applying restrictions (see section 2.2), hence required to be simple, but efficient.
- *Security* is relevant, since communication is wireless.
- *Customization* requiring the presentation of information in a format, context-specific to the user.

It is recommended to verify both service process- and service outcome quality through user participation [4].

3.2 Handling System Challenges

An essential requirement when designing mobile services is to create reliable mobile services running in unstable environments, which were described in section 2.2. We are now about to discuss how unstable environments impact the service quality aspects from section 3.1 and how those aspects can be provided to an adequate level of satisfiability.

Vulnerable infrastructure The condition of the infrastructure coheres with the aspects *reliability* and *responsiveness*. Improving reliability can be achieved by two basic approaches. The first approach comprises of minimizing failures by implementing highly reliable components during the development phase. The second approach handles the behavior when failures actually occur during runtime. This includes adequate handling of failure issues and embedding redundancy by making the service available on different communication channels (ubiquitous access). In turn, ensuring service provision at diminished levels - such as low bandwidth or transient connections - improves responsiveness [4].

Changing user context Addressing the problem of constantly changing user context comprises of regarding several service quality aspects. Having a mobile service quickly adapting to the changing user context improves *responsiveness*. Optimizing the *user interface* can be achieved by considering the target environment (light, noise, et cetera) and by displaying information according to the current context. Further, a mobile service should be *customizable* to ensure access from different types of devices. And last, but not least, the user context should be handled confidentially if needed, providing a certain level of *security* by the service [4].

Wireless discovery This issue recommends the employment of adequate technologies, such as UPnP [7] and Rendezvous [8], that aim on managing devices in a network, including the integration of new devices.

4 Architectural Issues

In this section, we explore the architectural options of the mobile service implementation. We briefly describe the client-server and P2P approaches, and we introduce the reader to the middleware concept, which is essential for handling the issues heterogeneity and context awareness, to which we will refer later in this paper.

4.1 Client-Server Architecture

This design approach suggest that mobile devices representing the clients access the service, which is provided by servers in the network's back end. Concluding, a client-server based mobile service has to be composed of different software components for servers and clients, each realizing the service's functionality on the respective hardware.

Servers There are two basic options *where* to place a server: either inside or outside the network's backbone. Servers placed inside a network are usually only used for proprietary network services, such as SMS in GSM. Placing a server outside the network implies placing it in the internet, since most wireless networks have gateways to the world wide web.

The software components, running on servers, are called *service platforms*, providing the mobile services to its clients. Speaking of web based mobile services, there are plenty of technologies, that support the development of those platforms, which - in this case - basically function as web servers. There is a wide spectrum of web frameworks, such as Struts and Turbine provided by the Apache Software foundation [9], or Spring MVC [10] - all facilitating the development of service platforms by offering from-the-shelf-solutions for aspects like validation, business logic et cetera.

Clients The *client software*, running on client devices, realizes the access functionality to the mobile services' service platforms, and thus, the mobile services themselves.

Regarding clients, we can differentiate between *thin clients* and *fat clients*, which basically describes the necessity of extra software for accessing the mobile service in question. Thin clients are devices, whose delivery status software payloads are sufficient for accessing the service. The lack of dedicated client software components makes the mobile service generically accessible for all devices with the base software configuration in question. For example, in a GSM network, a web based mobile service can be accessed by the device's integrated browser, which eliminates the necessity of proprietary client software. However, thin clients are always bound to the restrictions of their delivery status software payload.

In turn, fat clients are equipped with extra software, especially tailored for the mobile service in question. Therefore, the spectrum of possibilities is accordingly higher than that of thin clients, since the functionality of the extra software is only bound to the underlying API, such as the SymbianOS API [11], J2ME [12], or Microsoft .NET [13].

Fat clients can also act as mobile agents. Mobile agents excel themselves in acting autonomously. They are generally able to handle transient connections by communicating asynchronously, handle requests autonomously and provide a certain level of security [3].

4.2 Peer-to-Peer Architecture

Introduction The alternative to the client-server-paradigm is to embed mobile services into an environment where mobile devices do not necessarily require dedicated servers, making mobile devices users and providers of services at the same time [14]. Such *Peer to Peer* based mobile services profit from autonomy and self-organization, minimizing any administrative effort by moving from a *content located at the center* model to a *content located at the edge* model [15].

The current architecture of P2P networks is commonly called the *3rd generation of P2P*, which presents a good base for the development of P2P based mobile services [15].

Design criteria for P2P based mobile services Since the communication techniques of P2P differ from the client-server paradigm, the requirements on a mobile service in such an environment are slightly differently weighted.

The first issue to be mentioned is the communication between the peers. Since mobile P2P systems are decentralized and peers usually interact during brief physical encounters, communication links are highly transient, forcing the implementation of communication protocols, that can handle frequent disconnections and reconnections [14].

Another issue is the demand for interoperability between P2P networks. It should be possible to extend a mobile service to communicate with any of the many existing P2P protocols. This includes the parallel usage of P2P protocols both on the abstraction level of the mobile service and the mobile device itself [15].

Mobile P2P middleware To employ a P2P based mobile service, the according software to access it must be available on the client device (as mentioned in section 4.1). This software must be responsible for handling all communication issues with the P2P network. However, one step further is to design this software as a middleware platform, handling the access to the P2P network on the lower side and providing access to the P2P network for applications on the upper side (see figure 1) [15, 14].

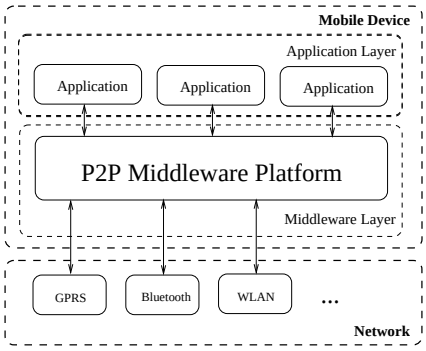

Figure 1: P2P Middleware Architecture

Such a middleware platform basically masks all communication aspects to the applications accessing it. This includes the following aspects [15, 14]:

- managing connections in the physical network
- controlling all communications according to the available P2P protocol(s)
- managing the presence of neighboring peers
- control the exchange of data with other peers
- providing an interface for applications accessing P2P networks

4.3 Middleware

Middleware solution have proved to efficiently solve problems occurring in the mobile computing sector. In this paper we discuss the possibilities to handle the problems of heterogeneity and context-/location-awareness by employing middleware. But before we discuss those issues in sections 5 and 6 respectively, we sum up a few architectural requirements for middleware solutions:

- *Separation of policies and implementation*, meaning that policies provide a high level definition of the behavior of a system, whereas the implementation defines how this behavior is achieved [3, 16].
- *Placement*, dealing with the question *where* to physically place the middleware - which can be on the server machine or mobile devices, either acting as clients or peers. Placing the middleware on the mobile devices recommends the employment of mobile agents, since those work autonomously and efficient, especially in distributed systems [17]. It is further to be reminded, that mobile services relying on a client's delivery status software (as stated in 4.1) must outsource any middleware solutions from the client.

5 The Problem of Heterogeneity

Mobile computing is characterized by a high level of heterogeneity, since there are only a few standards, that are obeyed by the device manufacturers, software developers and network providers. This counteracts the goal to maximize both availability of mobile services and interoperability between heterogeneous devices.

In this section, we present a common approach by employing middleware to address this issue. After abstracting the problem, we discuss the requirements and realization issues of such a middleware framework.

5.1 Abstracting the Problem

In mobile environments, heterogeneity can be divided into aspects concerning hardware (physical), software (logical) and architecture [17].

Hardware heterogeneity reflects the presence of devices with different capabilities. We can split up this issue into two subordinate aspects. First, devices of the same type can have different capabilities, such as a primitive cell phone does not have MMS capabilities in comparison to a modern smartphone. The second aspect of hardware heterogeneity is the presence of different networks. As an example consider devices in a GSM network trying to communicate with an Iridium satellite phone. Software heterogeneity describes the presence of different operating systems and applications running on mobile devices. Architectural heterogeneity is achieved when network interconnections among devices do not respect any common architecture.

Each of these heterogeneity aspects recommends the employment of middleware in order to provide seamless communication.

5.2 Middleware Approach

Requirements A middleware framework providing interoperability is placed between devices or networks, so that it can handle the communication between heterogeneous devices in heterogeneous networks by making the devices available to each other and *translating* their communication. This task can be split up into the following aspects [18]:

- *Transport-Level Bridging* involves the translation of protocols and data types.
- *Service-Level Bridging* implies the demand for quickly recognizing new devices in the network and make them available for use with other devices.
- *Device-Level Bridging* involves the translation of device semantics and the acquisition of device specific data in a *device database*.
- *Future Evolution* requires the framework to be able to accommodate new device types.

A vital requirement to handle heterogeneity is the premise to confine it. This makes a *device capability database*, which includes the specifications of a set of devices and/or device types, indispensable. Such databases are generally available in commerce. An ambitious freeware project, WURFL, is even offering it licence-free [19].

Design Aspects We can divide the design considerations into four dimensions, which are discussed below [18]:

- *Translation Model* specifies how device semantics are translated according to the device database. This can be done in two ways: First, the translation can occur *directly*, implying the advantage of minimal loss of semantics, but requiring the implementation of a translator for *each* pair of devices. The *mediated* translation represents the alternative approach, introducing a device independent intermediary space, where device semantics have an intermediate representation. This requires only one translator per device making this approach well scalable, but implying a greater risk of loosing semantics.
- *Semantic Distribution* specifies the visibility of devices. This aspect also comprises of two possible approaches based on the choice of action regarding the translation model. The direct translation approach implies *scattering* representations of a device across platforms enabling it to be visible by all other devices. The mediated translation approach requires *aggregating* device representations in the intermediary space, hiding them from all other devices.
- *Intermediary Semantics Granularity* specifies how devices are represented in the intermediary space. *Coarse-grained* representations summarize a device's operations and semantics. Any pair of devices is compatible if their operation and semantics match. The opposite approach by defining representations *fine-grained* comprises of projecting the device semantics on a set of communication endpoints. Compatibility of devices is determined by examining compatibility of their communication endpoints.
- *Location of Interoperability Layer* This issue describes the question where to place the translation logic at runtime. Either, it is placed *at the edge* of the network, meaning on the devices themselves, or it is embedded into a dedicated node in the *infrastructure*.

It is to be remarked, that some of these approaches are dependent on each other, so other design choices may not be mutual compatible [18]. As an example, we can state the direct translation approach being incompatible to all design choices including intermediary representations.

Representation The representation of mobile services is not trivial, since it is bound to the restrictions of the heterogeneous environment of mobile devices. A common approach is to have the middleware applying a *presentation service* to tailor the output specifically to the target device. As stated in section 4.3, this service can be placed on the mobile device or centrally.

The actual process of creating the output is commonly relying on the information in the device capability database.

6 Location- and Context-Awareness

There are two basic issues associated with the mobility aspect of electronic services: handling both the user's location and context. Both are normally constantly changing, calling for dynamic adaption of services to handle them. From the perspective of a mobile service, the awareness of both aspects present a different level of visibility. Location awareness defines the visibility of the user's physical position, whereas context awareness defines the visibility of the set of resources associated with the user [3].

This section will analyze both aspects and - as in section 5 - discuss a common approach based on middleware.

6.1 Location awareness

The location can be defined as "the property of physical position of users, devices and resources" [3]. The following aspects are to be considered while handling location awareness.

Infrastructure There are two basic ways of determining a device's position. Either the device detects its position and propagates it to whom it may concern (i.e. its server or another peer), or a server keeps track of its clients by monitoring their movements. The latter case is only feasible in cellular networks (i.e. GSM) by remembering the IDs of the cells the clients roam in. Satellite based detection systems (i.e. GPS) function as positioning systems for devices, keeping the infrastructure completely unaware of the devices' positions and leaving no other choice but the devices to propagate their positions.

Information delivery The basic approach to specify which location based information to deliver is to define *notification areas* and *client specifications*. Notification areas are geographical polygons, indicating what information is available where. This means, that a user only receives information associated with the notification area, that he is currently located in. Client specification allow a further refinement of the information, specifying *what* information is desired and *how* it's delivered. A GSM subscriber, for example, can choose only to receive tourist information about his current location (*what*) via WAP-Push (*how*) [20].

The delivery technique of such information associated with a notification area can be distinguished between delivery on request, and delivery on entering a notification area. While the first option is always triggered by a client request, the latter requires constant monitoring of the client's location to detect his entrance into a notification area, that triggers to push information to the client device. Therefore the time interval of track client devices must be chosen accordingly to satisfy the ability to reliably detect their entrance into a notification area. Figure 2 summarizes the workflow of location based notification [20].

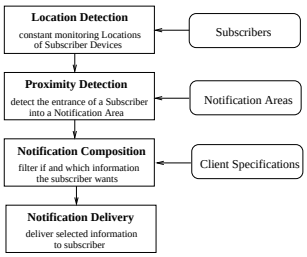

Figure 2: Workflow of location based Notification

6.2 Context awareness

Introduction The context of the user is defined "as a group of available resources and services considered relevant for the service provisioning" [3]. For example. this can affect personalized subscriptions, access restrictions, et cetera. Just like the location may change quickly, so can the context. In fact, location changes often imply changes of the context. As described in section 6.1, a change of location may make different location-based data available, which may be personalized for the user, implying a change of his context. Concluding, context aware services need to adapt to a quickly changing environment, which they cannot control [21].

Context Models To enable context awareness in a mobile service, contexts need to be abstracted to context models, specifying context information and their relationships. The most common context models are either object-oriented or ontology based [17]. Object oriented realization profit from simplicity and easy deployment. Ontology based approaches are more powerful offering rich expressiveness and evolutionary aspects of context modeling. However, they require performance draining ontology engines [17].

Middleware support To project the context on the context model, context aware middleware uses software components called *context sensors* to capture the context. This is basically labeled as context access, which is influenced by the following aspects [17]:

- *Communication* specifies the gathering of context information across the network. This happens either asynchronously or synchronously.
- *Distribution* specifies if context information can be probed locally or if distributed context is supported.
- *Updating* describes how context information is updated, leaving the possibilities of updating frequently or intelligently on occurrence of certain conditions.
- *Context service placement* specifies whether the middleware's context processing is placed centrally or distributed.
- *Context evaluation* describes the middleware's evaluation process, comprising of two approaches: either frequently (eager) or on demand (lazy).
- *Context querying* specifies how the middleware queries the context over the context repository. This aspect specifies to whom which context information should be available.

Additionally to those aspects, when designing context aware middleware, it is recommended to identify roles of *context producer*, *context service* and *context consumer*, and assign those to network entities [17]. This consideration may additionally help capturing the context.

Closing, it is to be remarked, that the issues of handling context and handling heterogeneity (as discussed in section 5) are closely related and often handled together by middleware approaches.

7 Conclusion

As presented in this paper, the environment characterized by the mobility aspect induces significant issues concerning the design of mobile services. We have shown, that the current approaches to employ middleware solutions to handle numerous issues, such as heterogeneity, changing user context and P2P communication - to mention a few - provide adequate solutions.

The future work will certainly focus on the refinement those concepts in respect to future technology developments.

References

[1] Insitute of communication services at boulder, co, usa. http://www.its.bldrdoc.gov/.

[2] Jari Karvonen and Juhani Warsta. Mobile multimedia services development: value chain perspective. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 171–178, New York, NY, USA, 2004. ACM Press.

[3] Paolo Bellavista, Antonio Corradi, Rebecca Montanari, and Cesare Stefanelli. A mobile computing middleware for location- and context-aware internet data services. *ACM Trans. Inter. Tech.*, 6(4):356–380, 2006.

[4] Els van de Kar, Sam Muniafu, and Yan Wang. Mobile services used in unstable environments: design requirements based on three case studies. In *ICEC '06: Proceedings of the 8th international conference on Electronic commerce*, pages 302–308, New York, NY, USA, 2006. ACM Press.

[5] Roy Want and Trevor Pering. System challenges for ubiquitous & pervasive computing. In *ICSE '05: Proceedings of the 27th international conference on Software engineering*, pages 9–14, 2005.

[6] C. Gronroos. Service management and marketing. a customer relationship management approach. *Chichester, Wiley. 2nd ed.*, 2001.

[7] Universal plug and play. http://www.upnp.org.

[8] Rendezvous, apple's automatic discovery mechanism for computers, devices and services on an ip network. http://developer.apple.com/macosx/rendezvous/.

[9] Apache software foundation. http://apache.org.

[10] Spring framework and spring web flow. http://www.springframework.org.

[11] Symbian developer network. http://developer.symbian.com/.

[12] The java me platform. http://java.sun.com/javame/.

[13] Microsoft .net framework. http://www.microsoft.com/netframework/.

[14] Gerd Kortuem. Proem: a middleware platform for mobile peer-to-peer computing. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):62–64, 2002.

[15] Erkki Harjula, Mika Ylianttila, Jussi Ala-Kurikka, Jukka Riekki, and Jaakko Sauvola. Plug-and-play application platform: towards mobile peer-to-peer. In *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 63–69, New York, NY, USA, 2004. ACM Press.

[16] R. Wies. Policies in network and system management - formal definition and architecture. *J. Netw. Syst. Manag. 2, 1.*, 1994.

[17] Ricardo Couto A. da Rocha and Markus Endler. Evolutionary and efficient context management in heterogeneous environments. In *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–7, New York, NY, USA, 2005. ACM Press.

[18] Jin Nakazawa, H. Tokuda, W.K. Edwards, and U. Ramachandran. A bridging framework for universal interoperability in pervasive systems. In *Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on*, pages 3–3, 2006.

[19] Luca Passani and Andrea Trasatti. Wireless universal ressource file. http://wurfl.sourceforge.net.

[20] Jonathan P. Munson and Vineet K. Gupta. Location-based notification as a general-purpose service. In *WMC '02: Proceedings of the 2nd international workshop on Mobile commerce*, pages 40–44, New York, NY, USA, 2002. ACM Press.

[21] Gruia-Catalin Roman, Christine Julien, and Qingfend Huang. Network abstractions for context-aware mobile computing. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, pages 363–373, New York, NY, USA, 2002. ACM Press.