DEMAC: A Distributed Environment for Mobility Aware Computing

Christian P. Kunze

University of Hamburg, Department of Informatics
Vogt-Kölln-Straße 30, 22527 Hamburg, Germany,
kunze@informatik.uni-hamburg.de,
http://vsis-www.informatik.uni-hamburg.de

Abstract. For ubiquitous respectively pervasive computing *mobility* is one of the most important aspects. In the past, mobile devices became more and more aware of their location and vicinity and communicated rather loosely with each other. Therefore, mostly *asynchronous* communication paradigms were used in order to decouple temporally message transport.
As such communication mechanisms seem suitable for single communication acts, they may not be sufficient, however, for more complex tasks which consist of sequences of related communication acts. This holds particularly if the resulting operating sequence spans several mobile devices in frequently changing vicinities.
Therefore, the approach taken by the ongoing dissertation project DEMAC aims at a *higher abstraction level* for inter-device communication, especially for more complex user tasks. The concept as developed so far is based on integrating processes into mobile system infrastructures and on distributing their execution over different nodes in the network. For this purpose, a middleware platform for context aware applications is designed which allows for defining process schemas and which uses an interpreter to execute the defined processes in a distributed manner.

1 The Challenges of Mobility in Pervasive Environments

In computer technology in general, two diametrical trends can be identified: On the one hand side computers become more and more powerful and they decrease in size. On the other side, the amount of connected items which are equipped with processing units increases continuously and they penetrate ever more into everyday life [1]. In recognition of this trend, e.g., Marc Weiser formulated his vision of *"The Computer for the Twenty-First Century"* [2] with the final goal to make computers available to users at any time and place, but making this access effectively invisible to them [3]

For these ubiquitous environments *mobility* is one of the central aspects to cope with: Mobile *users* travel around and need access to their programs and data from everywhere and at any time. But also *devices* are mobile and thus able to form dynamic environments which share data and/or services. And even

the *code* can move because it may migrate among and be executed on devices which form the mobile vicinity. However, supporting such mobility aspects on systems level also leads to additional constraints and demands than in more static distributed systems.

This paper introduces the dissertation project "*Distributed Environment for Mobility Aware Computing*" (DEMAC) which aims at developing a middleware infrastructure to support the integration of mobile devices into "classical" distributed environments in order to enable value added and process oriented applications.

The following subsections of this paper introduce the (broader and narrower) problem domains; section 2 addresses related and previous work, and section 3 outlines the DEMAC system software solution. Finally section 4 concludes this paper.

1.1 Integration of Mobility Constraints

Research in *mobile computing* has identified four constraints which are intrinsic to mobility but different to traditional distributed systems: These are the *restrictions of resources* in comparison to static devices, the *increased variability in performance and reliability* of wireless connections, the *finite energy sources* to rely on, and the *hazard of mobility* itself. This caused the insight that mobile systems can not provide as much transparency as systems in more statically wired environments. In contrast, the moving elements have to be aware of the changing vicinity and to react and adapt accordingly [4].

A first focus of Mobile Computing was on timely decoupling the communication of mobile systems because there is no guarantee that clients and servers are reachable at the same time. System support for that lead to middleware systems which mainly used *asynchronous* communication models - like message-passing, tuple space based approaches, or sharing of replicated data [5].

A second important aspect of research based on the implications of mobility constraints is the realisation of *awareness and adaptability*. These systems introduce the principal of reflection in order to make changes in the context available to applications. In such systems, the context describes - for each entity - the pieces of information which specify the situation of the entity and are relevant for its behaviour [6]. This knowledge can either be presented to the entity proactively by using adequate events or passively by providing a query mechanism.

In summary: with asynchronous communication and location awareness some restrictions of mobile clients' resources can be allayed. Namely, an application can retrieve and use services provided by other servers in its environment to extend its own capabilities.

However, all these solutions have in common that they solve the problems in mobile environments just for a single simple task or communication act because they are realised on a relatively low conceptual level: The decoupling is isolated for single interactions and only works under the assumption that clients and servers can eventually connect in finite time again.

1.2 The Need for Decoupling on an Abstract Level

Most existing solutions in the field of *mobile computing* provide their support from an application point of view. They offer mechanisms and techniques to fulfil basic but rather simple tasks (cp. subsection 1.1). However, to step further to the vision of ubiquitous computing the successful execution of the users' (mostly more complex) tasks must come into the focus. For this reason, future approaches to system support for *mobile computing* should be based on such a more abstract and user centric view.

On this abstraction level, complex user tasks can be regarded as sequences or process of related simple tasks from the application centric view which are managed by the mobile client. In consequence, a mobile client must be able to reach and invoke all services which are needed to execute the complex task. And as another implicit consequence, the client must be capable of handling all intermediate results - regardless of their size and relevance to the expected final output. This, however, leads to a single point of failure and a bottleneck during execution time.

If, on such conditions, the mobile device is not constantly reachable while performing a sequence of remote tasks, the execution time can expand fast and will reach an unacceptable dimension quickly. As the client has, additionally, to manage the whole control flow of the complex task, the quantity of possible processes is limited by the capabilities of the mobile device. Since the user, however, is, in mostly cases, just interested in the effects of a process and not in it's execution or intermediate results, the control flow - and with it the complex task - can be transferred to other devices. In such a case, the user should also be able to specify, next to the execution process, non-functional aspects like, e.g., *security* and other *quality-of-service* needs. These requirements should then be taken into account by the remote execution unit and enforced on user's behalf. As this can not be done just using existing techniques additional concepts are needed on the higher abstraction level of complex user tasks. These ones should decouple the execution of processes not only in time but also in space.

2 Previous and Related Work

As mentioned before, much research in the field of middleware systems for mobile computing is based on decoupling the communication of mobile clients and on making them aware of their environment. Most of these systems provide asynchronous communication paradigms and/or detailed descriptions of the device's vicinity. The work presented here uses these results as it is based on message-oriented middleware systems for Mobile Computing - also integrating parts of context-aware and event-based mechanisms.

In addition, such a middleware approach can also make use of current *web service technology* - in particular *workflow definition languages*. At present, we analyse if and to which scale the corresponding languages, e.g. BEPEL4WS or XPDL, are also suited for use in system support architectures for mobile system.

The goal here is to find a compatible subset to allow for integrating the mobile devices seamlessly into existing workflow environments.

Beyond that, this approach has also origins in distributed (multi-) *agent oriented computing*: For instance, research done in this dissertation project so far has demonstrated that it is actually possible to transfer a goal-directed agent system (JADEX) to mobile clients. Although there are, of course, limitations to the devices size it has been demonstrated in this research, that even for PDA-size devices such an approach is possible [7]. This leads to options for integrating *goals* - as described in [8] - as non-functional constraints and, in particular, to the possibility to *deliberate* about different strategies to achieve these goals in the most feasible way which is rather useful for a user centric approach: Different types of goals like, e.g., *achieve, maintain, or query goals*, which define the user's task more precisely, have a direct influence on the execution of the associated process.

As the middleware support proposed in this dissertation project uses remote services as well as delegation of tasks and responsibility in a mobile and distributed environment, there is also a need to ensure the necessary levels of *trust* in such remote services, based on, e.g., *foreign user roles*, and individual user *identities*. Therefore, the DEMAC project also includes some research into digital identity management [9] and, as a consequence, identities should become an integrated part of the context concept of the approach.

Finally, the dissertation project DEMAC is also influenced by previous research in the area of system support for mobile computing in Hamburg which, in earlier years, proposed and demonstrated the use of abstraction concepts for the design of adoption and integration strategies in distributed mobile systems. Also this research indicated that higher abstraction levels lead to fewer device modifications and easier handlings of heterogeneity [10].

3 The DEMAC Approach

The dissertation project DEMAC aims at including a process oriented perception into middleware system support for mobile computing. In this context, process orientation means continuing consequently the decoupling of mobile applications. More specifically, introducing an explicit *description schema* and an *interpreter* for distributed processes in mobility aware computing environments allows for decentralization of the control flow of mobile applications.

3.1 Decoupling via Delegation

The main idea to decouple applications on a higher user centric abstraction level as discussed in subsection 1.2 is realised in this project by introducing a *lean description* schema and a *distributed execution mechanism* for processes into the middleware approach. This allows for delegation of the responsibility to perform parts of a complex task to other nodes which can then perform the actual subtask in a more suitable way. Based on such a spatial and temporal decoupling strategy

the control flow remains close to the service execution unit. In addition, with a late service binding strategy, the mobile environment is able to support many (parallel) jobs - even if the initiating device is not reachable or turned off. Such mechanisms do not yet exist in current middleware approaches where just the data transmission between client and service provider is decoupled by the use of asynchronous communication techniques.

But since, in a user centric view, the execution is distributed, the application as well as the user must be able to express *functional* and also *non-functional constraints* like, e.g. security needs or quality-of-service parameters. Decoupling via delegation provides such possibilities to enforce these requirements on behalf of the user - even remotely - with appropriate security (and other) functions even on unknown or suspect servers.

3.2 The DEMAC System Architecture Components

As *system support* for mobility aware computing, DEMAC proposes extensions to an abstract middleware architecture which are based on four services components (see figure 1) which are presented and described shortly in the following.

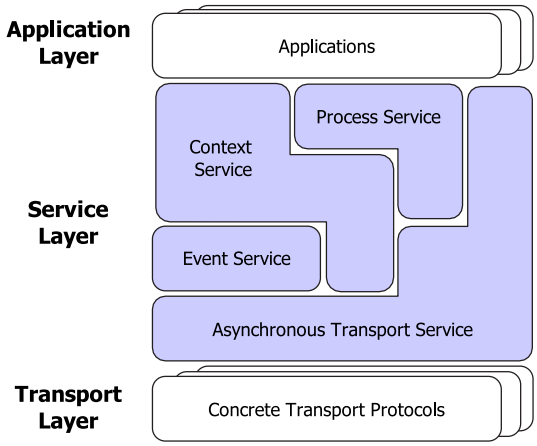

Fig. 1. Overview of the DEMAC middleware architecture

Asynchronous Transport Service The *asynchronous transport service* forms the communication basis for all services and applications in the DEMAC architecture. This service abstracts from concrete transport protocols, like TCP/IP, Bluetooth, or IrDA. On top of these the DEMAC transport mechanism provides the ability to send and receive messages. As the transport service should work asynchronously, messages are referenced by unique identifiers and delivered to

higher services or applications by using the "observer pattern". That means applications waiting for messages can register at the transport service and will be informed about incoming messages.

As the service is independent from the transport protocol it uses its own addressing schema. The used addresses are bound to a device and translated into the concrete protocol specific address by the transport service. If the device is reachable by different protocols, non-functional aspects like e.g. quality of service attributes can be used to make an optimal choice.

Event Service The *event service* is used to provide announcements of changes in the device's internal and external states to the context service. A modification of an attribute in the device, e.g. the loss of a connection or other quality of service parameter, is transformed into an event and passed to all interested local and remote clients. If external services should be informed the asynchronous messages of the transport service are used to notify the remote event service. Receiving such a foreign message the remote service generates a local event which then is passed to its local context service.

Context Service The *context service* collects and maintains all information about the context of the device. It acquires its knowledge either by events from the event service or by direct message exchange using the transport service. But towards the entities which uses the service it filters and partitions the information and provides only the amount they need. These are next to quality of service parameters also information about reachable devices and their services, location parameter and data about other users and their identity.

Process Service The *process service* realises the integration of process management into the DEMAC architecture and thus the decoupling by delegation discussed in subsection 3.1. It is comprised of two parts: The first one is a *definition of a schema* in order to describe the execution process as well as the users and applications non-functional demands. Using this schema, an application is able to define a sequence of services, intermediary results which must be achieved, and constraints for the execution. Thereby the services are referenced by abstract handles to keep the definition short. The second part of the service is an *interpreter* for process definitions. This unit has the duty to resolve and execute processes. It can either invoke the service locally or delegate the process to a remote process service. When delegating a process the description and all necessary data is transferred to the remote unit by the use of the transport service. Thereby the process service relies on the information provided by the context service to find a device providing the needed service and to enforce the non-functional demands and constraints.

3.3 State of the Project

The DEMAC project is still in an early state where just the presented abstract architecture and the correlations and interconnections of single services have been solidified. Now, more detailed research and specifications for each service and a first prototypical implementation are in progress. After this basic implementation the system support services will be extended incrementally. In this stage, also other state-of-the-art middleware techniques shall be considered, especially agent technologies like goal representation and deliberation to optimise the execution of the user centric process. To get a perception in which scale the agent-oriented techniques benefit such a process oriented middleware for mobile computing, major parts of an existing (multi-) agent platform has been transferred to mobile devices [7]. With this implementation the use of speech act communication and deliberation of goals in mobile scenario is analysed and the results are transferred to the DEMAC middleware approach. For an *example application process*, an insurance company environment is used which covers managing claims resulting from traffic accidents with mobile devices in a remote and distributed fashion. This scenario could demonstrate major parts of the advantages of the DEMAC architecture because it spans several mobile users (assessors, insurance agents etc.) and static back office systems like the insurance company or the garage.

4 Conclusion

This paper argues that existing approaches for mobile computing deal with the limitations of mobile systems on a communication level which is too low for many (especially complex) user applications. They basically decouple device communication in time only and support the ability to be aware of the device's context. By combining theses techniques, mobile clients may be able to make use of services provided in their vicinity in a more application adequate ways. However, such device extensions are only suitable for single communication acts; for sequences of remote service requests, however, more abstract concepts (and a corresponding system support) are needed. The DEMAC approach as presented in this paper solves this problem by integrating processes into a mobile middleware platform and delegating the control flow of the whole remaining process to remote executing units. In addition, such a delegation can be done recursively such that the execution of the process is consequently distributed and spatially decoupled from the mobile device. The assumption is that in cases where users are just interested in the effects of their respective processes they usually do not care about any details of the process execution as long as their functional and non-functional goals are achieved. This is supported by the DEMAC middleware by integrating non-functional constraints and options for goal representation into the (traditional) process execution. All this lifts the main perspective within the approach from a focus on rather basic application support to a more abstract user centric view in which user's tasks can be executed in most feasible and cooperative ways.

References

[1] Mattern, F.: The vision and technical foundations of ubiquitous computing. Upgrade **2** (2001) 2–6

[2] Weiser, M.: The computer for the twenty-first century. Scientific American **256** (1991) 94–104

[3] Weiser, M.: Ubiquitous computing. IEEE Computer Hot Topics (1993)

[4] Satyanarayanan, M.: Fundamental challenges in mobile computing. In: Proceedings of the Fifteenth ACM Symposium on Principles of Distributed Computing. (1996)

[5] Mascolo, C., Capra, L., Emmerich, W.: Middleware for mobile computing (a survey). In Gregori, E., Anastasi, G., Basagni, S., eds.: Networking 2002 Tutorial Papers. Volume 2497. (2002) 20–58

[6] Lehmann, O., Bauer, M., Becker, C., Nicklas, D.: From home to world: Supporting context-aware applications through world-models. In: Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom 04). (2004)

[7] Harbeck, M.: BDI-Agentensysteme auf mobilen Geräten. Master's thesis, University of Hamburg - Department of Informatics - Distributed Systems and Information Systems Group (2004)

[8] Braubach, L., Pokahr, A., Lamersdorf, W., Moldt, D.: Goal representation for bdi agent systems. In Bordini, R.H., Dastani, M., Dix, J., El Fallah-Seghrouchni, A., eds.: Proceedings of the Second International Workshop on Programming Multi-Agent Systems. (2004) 9–20

[9] Baier, T., Kunze, C.P.: Identity-enriched session management. In Lamersdorf, W., Tschammer, V., Amarger, S., eds.: Building the E-Service Society: E-Commerce, E-Business, and E-Government, IFIP, Kluwer Academic Publishers Dordrecht (2004) 329–342

[10] Müller-Wilken, S., Lamersdorf, W.: Jbsa: An infrastructure for seamless mobile systems integration. In Linnhoff-Popien, C., Hegering, H.G., eds.: Proc. 3rd IFIP/GI International Conference on Trends towards a Universal Service Market (USM 2000). Lecture Notes in Computer Science, Ludwig-Maximilians-University Munich, Springer Verlag (2000) 164–175