Introducing UMSIC Middleware Services

Jussi Laakkonen, Tommi Kallonen, Kari Heikkinen, Jari Porras

Lappeenranta University of Technology
P.O. Box 20
53851 Lappeenranta, Finland
firstname . lastname @ lut.fi

Abstract

Service orientation is an acknowledged approach for its capability to abstracting computing from the underlying hardware and software layers. Furthermore, service orientation can also be seen as dominant paradigm for application development. As future networks are required to behave in a dynamic manner, sufficient coordination at abstract level of middleware deployments is required. Peer-to-Peer networking has shown capability in absorbing service orientation and dynamic behavior. In this paper, services for Peer-to-Peer networking in UMSIC project are introduced.

Index Terms: middleware, service, UMSIC, application development, PeerHood, Peer-to-Peer.

I. INTRODUCTION

Middleware technologies are becoming increasingly important in distributed computing systems. Middleware is defined as the software layer that lies between the operating system and the applications. For networked applications, new knowledge is needed e.g. to enable application-optimized communication technologies, and generic abstraction of middleware that shall dynamically react on changes of the network environment by adapting the chosen settings and protocols to application needs, thereby significantly improving usability and deployment. Issarny et al. [1] show that especially in complex software systems, middleware has proven capable to deal with the ever increasing complexity of distributed systems in a reusable way. Attractive features of middleware have made it a powerful tool in the software system development practice. Hence, middleware can be seen as enabler for creation of methods and related tools for middleware-based software engineering. The development of middleware-based software systems can select applicable Software Engineering (SE) methods and tools, e.g. Service oriented middleware. The evolution of component based programming is going towards the service orientation paradigm that supports the development of distributed software systems in terms of loosely coupled networked services.

The dynamic execution in the future networking environment is continuously increasing and requires coordination with nodes that are in a likelihood unknown during very short time period. In addition adequate coordination at abstract level need to be available to application developers by middleware provision of corresponding overlay network or application. One of the promising approaches is peer-to-peer networking [1].

This paper introduces our own existing implementation of Peer-to-Peer network, PeerHood [2] in Chapter II and the UMSIC project, including the usage scenarios is described in brief in Chapter III. In Chapter IV the reasons for using Peer-to-Peer and service oriented approach for the UMSIC project application is presented. In the same chapter the services provided by the UMSIC middleware are introduced including the hierarchical structure of services. In the final chapter, Chapter V the main content of this paper is gathered.

II. PEERHOOD

PeerHood [2] is an implementation of Peer-to-Peer neighborhood and communications concept in mobile environment which enables proactive discovery of devices and their services from neighborhood whilst providing means for communication. PeerHood is targeted to Personal Trusted Devices (PTD) and it is designed to be a transparent networking module in between the network layer and the applications.

PeerHood consists of daemon, plug-ins and library for application as shown in Figure 1. PeerHood daemon performs the main operations of PeerHood and runs as a background application. PeerHood daemon communicates with application and plugins via socket connection. Different plug-ins are implemented to support different network technologies, these plug-ins are dynamically loadable modules and can be used by other PeerHood components. Currently there are network plug-ins for Bluetooth, WLAN (Wireless Local Area Network) and GPRS (General Packet Radio Service). The PeerHood library is provided for applications to use. It is the interface to be used when interacting with PeerHood daemon and it also provides a common interface for several wireless technologies.

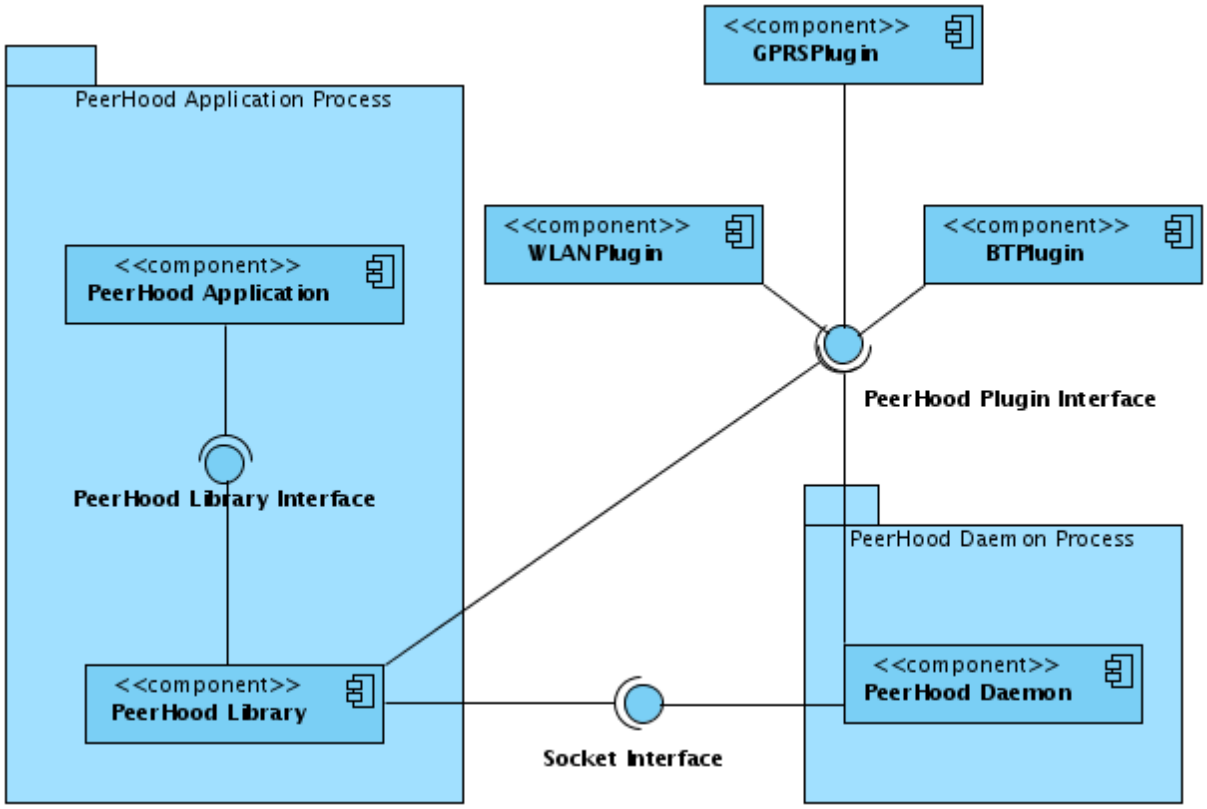

Figure 2. PeerHood components

PeerHood offers following functionalities:

- Device discovery
- Service discovery
- Device monitoring
- Service advertising
- Connectivity between PeerHood enabled devices and services

- Data transfer between PeerHood enabled devices
- Seamless connectivity

The process of service discovery in PeerHood is a two way operation [3]. First the information about nearby device is requested by the PeerHood daemon which contains the information of the device, its services, network techniques available and a list of neighboring devices. After this the application using PeerHood can get the list of services from daemon and use preferred service if found. The actual device and service discovery is specific for each networking technology that is used.

III. THE UMSIC PROJECT

UMSIC [4] is a FP7 project funded by EU, purpose of the project is to improve inclusion and reduce isolation in groups of children, especially where children have attention deficiencies or whose language is different from that of the host country. These goals of this project are going to be achieved through musical activities [5] which are enabled by using modern, mobile technology for creation of musical material in different social contexts. The social contexts include the usage of the application alone, with pair, small and large groups in school or on free time, including communication and collaboration via Internet. As application is used in learning situations in school class methods for controlling, monitoring and helping a child are provided for teacher or other authoritative person to use.

A. The product of the project

The product of the project, JamMo (Jamming Mobile) application, answers the challenges and requirements with the help of UMSIC middleware [6]. The UMSIC middleware contains means for secure communication with the help of PeerHood in service oriented manner and provides features for manipulating musical material, i.e. create, modify, share etc. JamMo application relies on top of the middleware utilizing functions provided. The UMSIC project relies on a mobile interactive product – the JamMo that provides a networked music making environment for children aged from 3 to 12 years. It is being designed for the Nokia Maemo devices, e.g. N900 [7], which provides a high resolution touch screen display (800 x 480 pixels) in a compact size and a stereo audio. Underlying Linux operating system (Maemo) enables the use of innovative and high-quality open source components, such as Clutter [8] for hardware accelerated graphics and GStreamer [9] for media-handling. The implementation of the JamMo software follows open source software development (OSSD) principles and is open for third-party extensions. JamMo is licenced under GPL (Gnu General Public Licece).

B. UMSIC scenarios

JamMo product with UMSIC middleware will be used in several different usage scenarios [5]. The scenarios and the functionalities needed differ in the user groups. In most cases JamMo will be used in classroom environment, but the usage outside school is also supported. The designed users of JamMo range from 3 to 12 in age. The scenarios for younger users are simple and the offered functionalities increases as the age of the users increases.

Stand-alone: Singing and composition games of 3-6-year-old children

The singing game is a simple karaoke application where a child can select a song and the sing along. The singing will be recorded and it can be listened later on. The song can also be sent to teacher. In the composition game child begins by selecting a theme from available possibilities. Each theme has a different look and different musical elements available. The composition happens by adding sound loops to a musical track. There are two tracks; a backing track, which can´t be edited and a track where the loops can be added. On the user

interface the loops are presented by different symbols fitting to the theme. The child can listen to the song that was created at any time with or without the backing track. The child can add, remove or move the sound loops to different parts of the track. After the song is finished it can be listened as whole and it will be saved to songbank on the device. The song can also be sent to teacher if the composition happened in a classroom.

Ad hoc: Composition pair game of 3-6 year-old children

The pair game is a extension to previously presented stand-alone composition game for 3-6-year-old children. The main difference here is that now the players are working as a pair to create a song. Before they start composing they have to select a theme just like before, but now they also have to select a pair (or a pair can be given by teacher). During the composition they now have three tracks in a song, one backing track and one track for each player. Both can add sound loops to their own tracks, edit the tracks and listen to track or song independently. Information about all of their track edits (places of the loops) are transmitted in real-time to their pair, so both can see and listen what the other is doing. After the composition has finished, the song is saved on both devices and can also be sent to teacher.

Public: Inclusive music classroom of 7-12 year-old children

In public scenarios the use of JamMo happens in a classroom where a teacher has a computer with desktop version of JamMo and a video projector to present song creation, playing or group works on a public screen. The children can work in groups up to four persons to create songs. They can create more complex songs than younger ones by editing up to six tracks. They can add sound loops or sound created with virtual instruments to editable tracks. Additionally children can use own samples (probes) they've recorded with the devices but only after the sample has been verified by higher authority to prevent possible misuse. The changes are updated to other members of the group in real-time. The teacher can monitor and control the activities of different groups from his/her computer.

The work in public scenario can also happen in the form of a chain, where one child creates a part of a song and sends it to next (pre defined or randomly selected) who continues. This way to song travels in a chain where each child adds a new part to a song until they all have participated and the song is finished.

Networked: Informal on-line community of 10-12-year old children

The networked scenarios are meant for older children and for non-real-time collaboration. The children can share musical material through on-line community, where they can also create workshops. They work basically as group composition for 7-9 year-old children, the difference is that updates don´t happen in real-time, the updates happen through server when the user logs in. This way the work is informal without the need for teacher participation.

IV. UMSIC MIDDLEWARE AND SERVICES

The usage scenarios presented earlier and the goals of the project set the requirements for UMSIC middleware. The middleware must manage all communications between devices in Peer-to-Peer environment in a way that is transparent to end user. The middleware will take care of actor authentication and authorization, file transfer, data transfer during real-time collaboration etc. The use of PeerHood is beneficial since it can automate device and service discovery and connection establishment. On top of PeerHood we propose several services each in charge of specific tasks. In our approach the UMSIC middleware uses PeerHood to search for other devices with UMSIC services, authenticates them and creates connections when needed without any user input.

A. UMSIC Middleware

The UMSIC middleware should operate on its own without requiring any interaction from the user. The transparency of middleware itself and its operations are requirements in UMSIC project [6]. Therefore the middleware used should be able to maintain itself and the information about surrounding devices and their properties, which can be enabled by using our existing solution, PeerHood [2].

As the users can be assumed to have limited knowledge of the technical issues concerning the product because of their age the self maintainability of the application as a whole is an important issue. Especially when networking is required and connections to other devices are required to be established. Since the approach in UMSIC is to make children work together in small groups without any centralized servers Peer-to-Peer is a natural choice for communication and connection establishment. It is seen as a natural way to build up needed personal seamless ubiquitous client-server connections [10]. Although there is a teacher present in most of the scenarios and the teacher uses a server designed for monitoring and controlling devices of children the actual group works are happening in Peer-to-Peer manner. The teacher's server can be thought as a quiet participant in the group work and in some cases, informal learning and chain work scenarios for instance, the devices children use are communicating only in Peer-to-Peer manner. Especially in chain work where the composition is forwarded to next participant when that particular participant is discovered from the network neighborhood.

In most of the scenarios described earlier the devices are located in same small area, e.g. classroom and therefore most of the communication is happening in some local or personal area network. The current implementation of PeerHood [3] supports both of these network types with the help of different networking plugins. In these kinds of networks centralized servers are not necessarily needed [11] and with the help of Peer-to-Peer connectivity clients can share own services and establish connections to services provided by others. PeerHood supports both of these operations, including automatic and manual selection of services [3]. The automatic service selection simplifies the process and manual selection causes more inconvenience. As requirements of the UMSIC project suggest [6], the service selection should be automated or either done by the middleware at program level.

B. UMSIC services

The purpose of the services provided by the UMSIC middleware is to provide access to software modules and their data which are located inside the middleware. With the help of carefully planned services and service hierarchy automatized enforcing of security can be achieved. With different services the connecting device and its user can be first checked for validity (authenticate) and then the user can be granted an access level depending on the result of the authentication. The access to other services, their functions and data can be then limited by the access level. When the whole procedure is automatized at middleware level the transparency of middleware operations can be achieved. The services are meant for other users of JamMo to use and provide information to authoritative persons, the modules that services are connected to exchange information internally.

The hierarchy for different services is presented in Figure 2. There are five different levels for services (L1,L2,L3,L4 and L5), where L1 (1st level) is the highest and L4 (4th level) and L5 (5th level) are the lowest. Idea is that first a connecting device must gain acceptance from higher level service and only after that lower level service can be used.

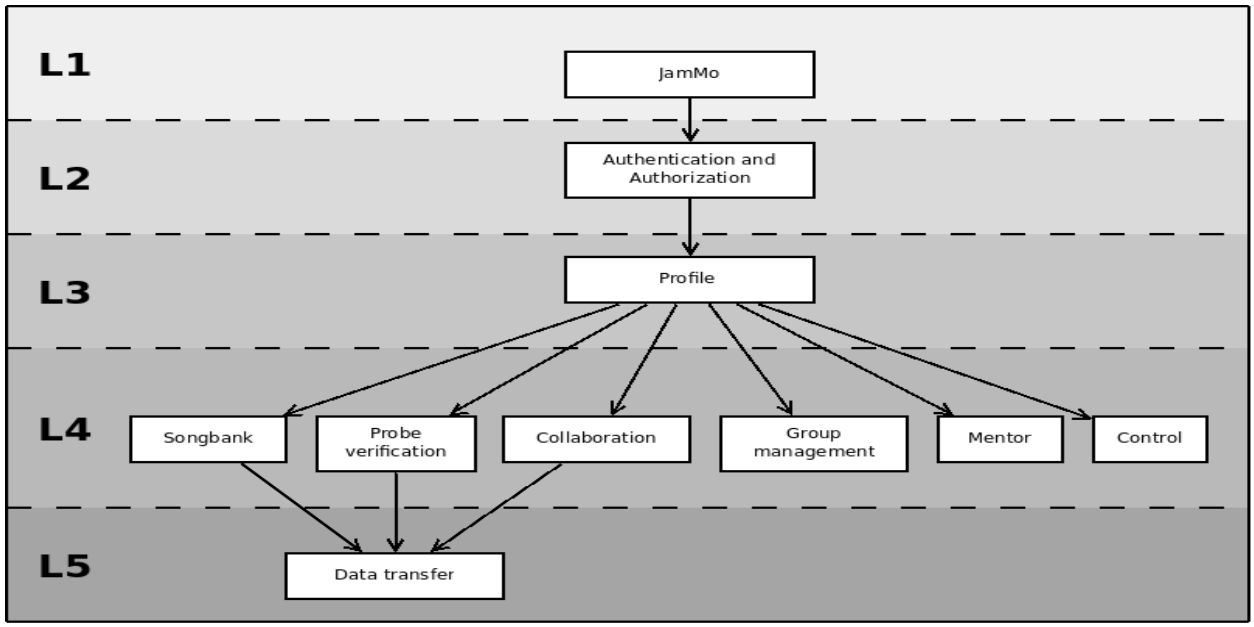

Figure 3. UMSIC service hierarchy

Jammo service

JamMo service is the simplest service in the UMSIC middleware. It is used to detect running JamMo application and the version of the application. Only compatible versions are allowed to interact.

Authentication and Authorization service

Authentication and Authorization service is the second service that is connected to. The validity is checked by verifying authentication credentials which provide the information about validity and type of the user. Based on this user type (there can be 3 different types, UMSIC personnel, teacher and child, where child is the lowest) a access level will be granted and session information is saved on the local device. If the connecting user can not be verified the user is not allowed to use services on lower level and connection is rejected.

Profile service

Profile service will provide information about the user who is using the device. Since the information stored on the device can be sensitive information the amount of information that will be provided for others is based on their access levels. Users with higher access level (e.g. teacher) have access to more information than another user who has the same access level. By the preferences and information provided by Profile service the compatibility between users can be checked. This includes for example checking if the users are the same age or do they have similar hobbies. If the user is accepted access is granted to services on the 4th level. Another purpose of the Profile service is to allow the teacher to push existing profile to the device.

Songbank service

Songbank service is, as named, meant for accessing the songs on the device. It provides functionality for searching for available files (songs) on the device, download a particular file from device, get the full list of all available files and saving a file to a server. The saving of a file to a server functionality is available only on a server machine. For transferring the actual data generic Data transfer service is used.

Probe verification service

Probe verification service is used for requesting a verification for a self recorded sample. The service is different whether run on a client device (used by child) or on a server (used by teacher). When run on a server this service provides means for requesting a sample to be verified (sent by child) and on client device it provides functionality for receiving a verification reply for some requested sample (reply sent by teacher). For the actual data transfer generic Data transfer service is used.

Collaboration service

Collaboration service is meant for synchronizing data between group members during the song creation process. The changes others have made are sent to every member of the group either in real time or periodically, depending on the real time requirements of scenario where the group work is done. Open group works (compositions) can be synchronized between users sharing the same access level, the result, being either finished or incomplete can be saved to server. Incomplete works are saved to server for future use in order to allow users to be able to continue the composition afterwards. For transferring larger quantities of data Data transfer service is used.

Group management service

Group management service is used to form communicating groups between users. It offers functionalities to advertise a group, to join a group that was found or part from group. Users on same level have the free will to choose their groups but users with higher privileges (e.g. teacher) can force regular users to act as a group. Also information about current group status is provided to other users requesting the information.

Mentor service

Mentor service is designed to provide way for higher authority (teacher) to get information about the current activities of the user (child). Another purpose is to automatize the process for asking for help if child gets stuck on some task. The request for help is sent to teacher automatically by the software if the module responsible of the cognitive functions notices that child is having problems.

Control service

Control service is meant for gaining full control of the device used by a child. This service can be only used by persons who have higher access levels (usually the teacher). The main usage is in classroom where teacher needs to e.g. help a child with some task or to lock the device for some time period. The service is located only on client devices.

Data transfer service

Data transfer service is a generic service for transferring data securely between devices. It is used for needs of other services and the direct access to this service is restricted. Other services on 4th level must initiate the sequence for transferring data related to that particular service. After this the transferring of data is fully handled by Data transfer service. Data transfer service keeps track of the services which are transferring data and what is the status and direction of the transfer.

V. CONCLUSION

In this paper UMSIC middleware services were introduced. The services provided by the UMSIC middleware are divided to different hierarchy levels based on the responsibility and authoritative requirements. The connected device cannot use the service unless upper level

has granted access to it. The services presented in this paper are derived from the UMSIC project requirements. As the implementation phase of the project has just started it is very likely that the specification of the middleware and affected components, including services will evolve.

The PeerHood implementation enables a variety of services to be created on top of it, as concepts such as seamless connectivity and discoveries can be obtained through PeerHood functionality. One of the major technical results is the development of lightweight middleware for complex scenario settings and use cases. The service oriented approach used in PeerHood is suitable for the needs of UMSIC project.

ACKNOWLEDGMENT

The authors would like to thank UMSIC project.

REFERENCES

[1] V. Issarny, M. Caporuscio, N. Georgantas, A Perspective on the Future of Middleware based Software Engineering, Proceedings of International Conference on Software Engineering, Future of Software Engineering, pp. 244 - 258, 2007, ISBN:0769528295.

[2] Jari Porras, Petri Hiirsalmi and Ari Valtaoja, “Peer-to-Peer Communication Approach for a Mobile Environment”, *Proceedings of the 37th Hawaii International Conference on System Sciences,* 2004.

[3] Arto Hämäläinen, Jari Porras and Pekka Jäppinen: “Service Discovery in Mobile Peer-to-Peer Environment “, 5th Workshop on Applications of Wireless Communications, pp. 21-29, August 2007, Lappeenranta, Finland.

[4] The UMSIC Project homepage, www.umsic.org

[5] The UMSIC Project, "Work Package 1 - Requirements".

[6] The UMSIC Project, "Work Package 2 - UMSIC Architecture Design, Specification and System Integration".

[7] Nokia, "Nokia N900 mobile computer - Technical Specifications.", http://maemo.nokia.com/n900/specifications/

[8] "Clutter Toolkit", http://www.clutter-project.org

[9] "Gstreamer: open source multimedia framework", http://www.gstreamer.org

[10] Tadashige Iwao, Satoshi Amamiya, Guoqiang Zhong and Makoto Amamiya, “Ubiquitous Computing with Service Adaptation Using Peer-to-Peer Communication Framework”, Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03), pp. 240, 2003, ISBN:0-7695-1910-5

[11] Henrik Abramowicz, “Ambient Network Project Description and Dissemination Plan”, http://www.ambient-networks.org/phase1web/publications/D1_A1_AN2_Project%20Description.pdf