ABSTRACT

Mobility exists in many shapes; when migrating a running application (code, states and data) from one device to another, one has achieved application mobility. In this article we combine this mobility type with context awareness, defining context awareness supported application mobility (CASAM) as when using context in the act of moving an application between hosts during its execution, to provide relevant information and/or services, where relevancy depends on the user’s task. We identify and present five CASAM challenges, that are used to create requirements for a CASAM architecture. A proposal for an architecture, building on peer-to-peer (P2P) technology in combination with mobile IP, is then presented, addressing the identified challenges, providing a decentralized framework for global application mobility. As such, our architecture differs from earlier centralized and/or locally bound solutions for application mobility.

Categories and Subject Descriptors
C.2.1 [Computer-Communication Networks]: Network Architecture and Design

General Terms
Design, Experimentation, Human Factors, Measurement

Keywords
Application Mobility, Context Awareness, User Study, Mobility Architecture, Mobile IP, Peer-to-Peer Networking

1. INTRODUCTION

There are four traditional types of mobility [14]: terminal mobility, allowing a device to change location and still be able to communicate; personal mobility (or user mobility), when a user can keep his or her user identity irrespective of terminal or network; session mobility [17] (or continuous user mobility), letting the user change location or device and still be able to keep media streams active; and lastly service mobility, making a particular service accessible by the user, regardless of terminal or network. One type of service mobility is when the user works with an application installed on a computer that is not in the user’s close vicinity. This can e.g. be realized through virtualization technology [8]. But mobility can also be extended to the actual application; when migrating the running application from one device to another, one has achieved application mobility. In this paper we will combine this mobility type with context awareness and present a novel architecture for context awareness supported application mobility (CASAM).

To describe the concept of CASAM, we present the following scenario: Alice is sitting in her office, attending a meeting via a videoconference system. The application allows her to see all other attendees that are using webcams, and they can also see her. They can speak with each other through headsets and use the videoconference application for text based communication, drawing and sharing presentations. However, the meeting drags on and Alice has to leave. The system senses this and migrates the video application to her smartphone. On the way to her bus, Alice can still participate in the videoconference through her smartphone Bluetooth headset, though without camera or other easy to reach interaction possibilities save speech and sound. On the bus, the system notices that Alice starts her laptop, whereupon the application is transferred to her computer so that she can continue work with streaming video, headset and keyboard input. Later, at home, the video session might be displayed on her widescreen TV, while the sound can be distributed through her media system speakers, allowing her to experience the best available video and audio quality.

1.1 Definitions

In this section, we will define the important elements of CASAM.

- Application mobility: We define application mobility as moving an application between different devices during its execution. This is usually done in three stages: first, the application has to be paused and have its states captured. Secondly, the actual transmission of code, states and related data is carried out, and finally the application continues running on the new device. These three phases can be named suspension, migration and resumption [23]. Migration can be sender initiated, when the user or the system ac-
In the final sections of this paper we summarize and present a number of requirements for CASAM, informed by the identified requirements. In section 5 we propose an architecture for CASAM, informed by the requirements. In the final sections of this paper we summarize and present our additional ongoing and future work.

1.2 Disposition

This paper is structured as follows: In section 2 we summarize related work. Section 3 contains identified challenges within current research, while a user study in section 4 shows the viability of these challenges and provide us with five requirements for CASAM. In section 5 we propose an architecture for CASAM, informed by the identified requirements. In the final sections of this paper we summarize and present our additional ongoing and future work.

2. RELATED WORK

The area of CASAM is rather young. In 1995, Bharat and Cardelli [3] presented migratory applications as: "a new genre of user interface applications that can migrate from one machine to another, taking their user interface and application contexts with them, and continue from where they left off." Over the last ten years CASAM systems have gone from continuous media applications (sometimes nomadic in its mobility nature), residing heavily upon location as context notion, over systems with greater focus on interface adaptation, to more general and robust frameworks, in regards to functionality and interoperability. Many projects are designed around centralized solutions, which in our opinion takes away one of the main advantages of application mobility: avoiding a constant connection to the network. Ontologies as context representation are common in the more recent systems, as is a design comprising of modules each responsible for different tasks within the CASAM framework, sometimes built into a middleware that can provide services such as migration planning and execution, context reasoning and storage. Table 1 summarizes.

Most CASAM-systems (Roam [6]; an unnamed system by Bandelloni and Patern` o [2]; Sparkle [22]; Gaia [19]; MD-Agent [23]; SAMProc [21]; MSP [10]) are centralized, to an extent relying on server-side components. Such an architecture makes the system very sensible to disturbances in Internet connection, thus canceling out one of the most important strengths of application mobility, namely a lesser degree of Internet dependency (for instance compared to the usage of similar cloud applications). Satoh [20] instead adapts a decentralized approach, using an architecture consisting of several computers which form a federation. Within the federation, an application can be dynamically distributed and deployed even while the application is executed. The application can be split onto several devices for input and output, for example a screen showing the GUI, a mobile phone providing input and a computer (without keyboard) for processing. All components consist of collections of Java objects, each with their own identifier and also an identifier of which federation they belong to. The system requires all nodes to be directly connected through...
3. ARCHITECTURAL REQUIREMENTS

Many challenges exist within CASAM. When surveying related work, we can identify several areas that stand out. First of all, there are issues concerning application identification. How users are authorized is an important matter, as is privacy support and control of secondary context [9]. Also, one has to consider how to authenticate the devices, the users and – of course – the applications. In addition, naming and keeping track of applications is important, as is device recognition and the prevention of application loss (and/or the addition of recovery methods).

Another issue of CASAM design is the context-awareness model. How context data is acquired, represented, interpreted and abstracted has great impact of system functional- 

tions with certain, often varying requirements. Design for diverse environments is therefore an important challenge to overcome. In short, enabling the system to sense, think and act is of utmost importance when it comes to supporting application mobility.

All downtime in application mobility must be minimized, or preferably eliminated, to allow a good usage of the system and its services. Migration is often associated with big overheads, making it a challenging task. The topic of seam- lessness must also be seen from a user perspective, as seam- lessness might be experienced different by different users in varying contexts. A user might for example be more tolerant with downtime in a video stream, as she still has to move her eyes from one device to another, but when listening to music, an interruption in a song might be experienced differently and with less patience. This is also what two ITU-T Study Groups [15] state in their definition of Quality of Experience, taking into account “the complete end-to-end system effects (client, terminal, network, services infrastructure, etc)” and the fact that “overall acceptability may be influenced by user expectations and context”.

Finally, when moving an application within or between systems with devices that differ in hardware or software configurations, migration is said to be heterogeneous. This is normally the case for CASAM, where various types of devices, all with different resources, are used to run applications with certain, often varying requirements. Design for diverse environments is therefore an important challenge to overcome. Our earlier work [16] contains a longer and more detailed survey of CASAM challenges.

4. COMPLEMENTARY USER STUDY

Researchers have listed benefits, drawbacks and challenges that need to be considered when trying to achieve sufficient application mobility. However, the involvement of potential end-users has been very poor so far. In this paper we therefore present a complementary qualitative user study which goal was to test the viability of the indentified challenges, and also to see if the users could provide new insight to the problem domain.

The study was divided into three phases and conducted with the help of nine users – four females and five males – at Luleå University of Technology in Skellefteå, Sweden. The tests where carried out individually. Each user was introduced to the concept of application mobility, then we informed about the task that was to be carried out. Moreover, while signing up for the evaluation the users were asked to fill in a questionnaire, where the aim was to gather information about the target group. The questionnaires show that the age of participants spanned from 20 to over 60 years. On a Likert scale ranging from 1 to 7, the users considered themselves to have computer skills between 4 and 7. Additionally, the target group was made up from persons representing administrative professions, students from art courses, technical courses and economic courses respectively, and also researchers and teachers.

During the second phase of the study, each user carried out a specific scenario while using a videoconference application enhanced by context-aware application mobility, being able to migrate between three different devices, using RFID to communicate the user’s position to the system. Our A2M prototype [24] provided the functions needed to set up a multicast network over 802.11g WiFi. The videoconference application was implemented using Flash and Java. Context data was modeled as XML and included user location and
The test commenced in an office, where the user was participating in an e-meeting on a laptop computer. After a while, the participant was asked to go to the library to borrow a book. The user consequently moved the application to a mini-PC and continued the conversation on the way to the library. The third and final step was for the user to bring the book and go home to his/her living room, where he/she migrated the application to a computer with a TV screen. Figure 1 shows the layout of our simulated test environment.

After the scenario, interview sessions were performed, where each user was asked ten open ended interview questions concerning the concept of application mobility, and the experience of testing the prototype. Each session was recorded, to fully capture all answers.

The test and user interviews supported the requirements identified in section 3. Users commented on the importance of the system to be always accessible and that it must not break down or lose its connection to other devices when the user does not desire it. Security issues were also discussed, along with the importance of identifying applications. Regarding context awareness and quality some users felt it was important to be in control, while others wanted the system to be able to make its own, well-informed decisions. Different modalities was also considered to be a good thing, having the opportunity to use, e.g., a big screen instead of the small displays of smartphones or ultra-mobile personal computers, if a big screen were to be available. Moreover, migration performance was a recurring issue in the interviews; the system should be fast, give the experience of “good flow” and have low delays when starting up the applications on new devices. During our study, total migration times (suspension, migration and finally resumption of the 2 MB application) varied between 4 and 11 seconds. As shown in figure 2, suspension times were very small, averaging 58 and 96 ms respectively. Migration times averaged 6787 ms for the nine unique migrations between the office and the library and 3959 ms for the migrations between the library mini-PC and the living room. Resumption times were 1169 ms and 1303 ms respectively.

Several of the users also commented that a system providing application mobility should work on different devices and platforms, and be able to handle resource challenges such as battery consumption.

Other, clearly user-centric demands and comments where also added to the existing challenges. These issues could be divided into two main groups; simplicity and social issues. Several users stated that they wanted a system providing application mobility to be simple and intuitive, and also easy to install. The applications should be stable and well-designed, and it should not be expensive to use the system. Some of the users saw application mobility as a good technique for social applications like chats, enhancing availability, and one of them came up with the idea of using it as a platform for gaming. Some concerns were also raised that application mobility would make the user always available for work, in the same manner a mobile phone can make people presuppose that the owner of the phone always will be available to answer. Table 2 summarizes the result of the user evaluation.

5. ARCHITECTURAL PROPOSAL

Supported by our survey of related work and the user study we conducted, we find that an architecture for CASAM should fulfill the following five requirements:

- Must handle application identification
- Must be context-aware and maintain context quality
- Must provide seamless migration, minimizing downtime
- Must cope with diverse environments, i.e. migration between different networks, heterogeneous interfaces, resources, local preferences etc
- Must have high degree of usability
The A2M prototype [24] fulfills some of these requirements, but not all. Therefore we have designed an architecture for application mobility that will act as a foundation when further developing our prototype. This architecture takes all the identified requirements into account, either by directly addressing them through its composition or being general enough to allow customized solutions for specific problems. Starting with peering capabilities for devices: if the system is to be used in a bigger network or between networks, a typical multicasting solution would be insufficient, as multicasting requires routers to be configured to support this communication technique. Therefore, the Peer-to-Peer (P2P) paradigm [4], creating an overlay network of devices used for application mobility, is the preferred solution.

In our architecture, P2P networks for application mobility will be small or of moderate size, and they may be widely distributed geographically. When adding a new node to the P2P network, ways are needed to locate the peers and to join the network. First, all users in a network using application mobility between devices need to register an entry in the local DNS, binding a fully qualified domain name (FQDN) (e.g. person-P2P.domain1.se) to a multicast address existing in the home network. When adding a device to the P2P network, users enter the FQDN and the node requests the DNS for the IP multicast address matching the FQDN. When joining the network, a join request will be sent using the multicast address. All nodes in the P2P network connected to the home network will receive the join message and the one responding first will authenticate the requestor. If successful, the node will be a member of the P2P network. Our architecture enables nodes connecting outside the home network to attend the P2P network by using Mobile IP (MIP) [12]. Node mobility is managed by standard MIP, where a node registers with the Home Agent (HA) and requests the multicasts for the P2P group to be forwarded by the HA. In this way, nodes connecting both in the home network and outside can be part of the P2P network. In our architecture, this also makes it possible for nodes borrowing by a user to join the P2P network, as users in our architecture can register (public) nodes with the HA. The join request (multicast message) from an outside node will be sent through a tunnel from the requesting node to the HA and then forwarded by the HA, both in the home network but also to nodes outside, being a part of the P2P network. In comparison with pure P2P networks, P2P-MIP is proposed with personalization in mind to enable enhanced security, scalability and dynamics. Figure 3 shows an overview of our proposed architecture.

In our implementation, we have chosen JXTA [18] as the P2P platform. JXTA is an open source P2P protocol that uses XML messaging to interconnect devices and create decentralized overlay networks. JXTA has defined capabilities for advertising context information such as peers, groups, virtual connections and services, while virtual hash tables help distributing the data structure. As the protocol is based on XML it can easily be extended with the possible extra information needed to support the requirements for our architecture.

### 5.1 Application identification

Our naming uses an address comprised of the P2P domain and the application identification, in the form of application_id@P2Pdomain. Adding and removing devices to and from the P2P overlay network must be made in a secure way, preventing application theft. To address the theft issue, the joining process uses encrypted control messages. This is also the case for control messages managing mobility of applications. Our proposed solution uses symmetric encryption where users need to enter a shared secret key in all devices intended to join the P2P network. To avoid application loss due to devices (hosting applications) shutting down, there are no special mechanisms needed beyond having a node in the P2P network that is "always" available in the network. A desktop computer equipped with UPS and a fiber connection can take such a role. Battery powered computers sense that battery is ending and will request to move an application to another device before shutting down. The nodes in our P2P network can be both private and public devices. In the case that a user wants to add a public device, like for instance a big screen, it must first be authenticated to join the P2P network.

### 5.2 Context awareness and context quality

We propose a decentralized context model. The nodes keep profiles about all the applications, with prioritized suggestions regarding input and output capabilities, network requirements and processing power. The system also contains information about all devices. This information is added in a profile for each device. Both these profile types are available at every node in the P2P network, in the form of distributed hash tables.

Certain context data is crucial for the possibility to migrate an application. E.g. the new device must have a certain amount of free disk space to host the application, otherwise it should not be regarded as an alternative for migration. The same goes for devices which are critically low on battery. This type of context data must be gathered at the same time as the user or the system decides to initiate a migration. Less crucial context data can be stored in the

<table>
<thead>
<tr>
<th>Table 2. Summary of User Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported requirement</td>
</tr>
<tr>
<td>Accessibility, robustness, must not lose identity, must not lose connection, security, durability, must not lose current work</td>
</tr>
<tr>
<td>Application Identification</td>
</tr>
<tr>
<td>User control, adaptability, good decision-making, different modalities, good decision-making</td>
</tr>
<tr>
<td>Design of the Context awareness</td>
</tr>
<tr>
<td>Model &amp; Context Quality</td>
</tr>
<tr>
<td>Fast, good flow, low delays</td>
</tr>
<tr>
<td>Seamlessness</td>
</tr>
<tr>
<td>Must work on different devices, battery consumption, must work on different platforms</td>
</tr>
<tr>
<td>Heterogeneity</td>
</tr>
<tr>
<td>Simple, easy to install, well-designed applications, intuitive, cheap, always available for work, social platform, gaming platform</td>
</tr>
<tr>
<td>Usability &amp; Social Considerations</td>
</tr>
</tbody>
</table>
P2P hash tables and updated in intervals depending of the volatility of data.

Within our proposed architecture, a sender initiated migration of an application (informed by context awareness) will be carried out in the following steps: 1) a migration is triggered directly (for instance by the user) or indirect (by signaling within the system), whereupon 2) a lookup is made in the distributed hash tables to find a candidate device meeting the requirements of the application and the user in terms of input and output and so on. Then 3) a notification is sent to the chosen candidate host that the application is about to move. The application’s profile is then 4) compared with crucial properties as free disk space and battery power, and the candidate host device then 5) give answers on to what extent it can fulfill the application’s demands, i.e. answers yes or no, whereupon 6) the migration is either carried out, or, in the case of a negative answer, the process of finding another candidate host begins anew with step 2 in the migration sequence. This simple yet dynamic negotiation process (also shown in figure 4) gives a better quality to context, in that it always produces the most recent answer to how suitable a certain device is for running the application that is to be migrated at a particular time, weighted against demands for seamlessness. More or less static information is stored in the distributed hash tables (which are updated as often as the system allows, not disturbing network or application performance) while important dynamic context information (e.g. network quality [1] or battery power) is collected and considered when actually contacting the host candidate.

A receiver initiated migration would be carried out in almost the same way as the sender initiated migration presented above, but with one exception in the early migration sequence: migration is triggered through the device shown as Node 2 in figure 4, and then a migration request is sent to Node 1, skipping the whole process of comparing and choosing a candidate node. When requesting receiver initiated migrations this parameter is already known. However, we do have to ask the distributed hash table for the application profile, as we do not know where the application is currently stored before migration.

5.3 Seamlessness

Seamlessness, as in instant migration, is a utopia that cannot be completely reached. Sometimes, it might not even be of importance. The focus should lie upon giving a good enough experience of seamlessness, and this might vary from situation to situation. In cases where we can suspect a high need of seamlessness, the process in figure 4 might be overridden, and instead replaced with a history based migration decision. If a user has migrated her mini PC MP3-player onto her home media center seventeen times out of seventeen the last week, there might be little need for a negotiation process to decide that a seamless migration based on historical data would be preferred over a slower migration based on recent data. As the distributed hash tables containing the profiles for applications and devices are updated from time to time, the negotiation process can be shortened and simplified.

However, the experiments presented in section 4 (see figure 2) showed us that suspension and resumption had less importance on migration times compared to the actual migration, which in general caused the biggest time consump-
Figure 4. Sender initiated migration
tion. Using context awareness to decide upon the best available network should therefore be an important feature in an architecture enabling CASAM.

5.4 Heterogeneity
Heterogeneity is both a blessing and a curse; different devices, operative systems and applications makes it hard to migrate applications and have them run on different platforms. Then, on the other hand, heterogeneity is a prerequisite for application mobility that benefits by different interaction capabilities; application mobility is not an end in itself, but rather something that is triggered when the situation is favorable for a migration to take place. The rationale behind a migration is that the application should always run on the best available device, the notion of "best" being decided by the user and/or the system. Our standpoint is that the application should be migrated with its complete execution state. A good thing about this is that only one version of the application will be running at the same time, which would eliminate the problem of version tracking. The drawback is that heterogeneous devices often does not supply the same software platforms. Java applications, for instance, might be vendor-specific platform independent, but the Java Virtual Machine is not.

To this date, there are no good solutions for coping with heterogeneity drawbacks in larger P2P networks. Standardization within networks and communication is being carried out by organizations such as IEEE and the ITU-T, but there will always be platforms not compatible with standards. Storing different versions of a program on a centralized server is a possibility, but then the desirable decentralized architecture feature cannot be used. Although we expect the problem with heterogeneity to diminish as operative systems are getting smaller (see e.g. [13]) and smaller devices are getting more powerful, making standardization easier, we want to recognize the problem. In a small scale P2P network though, heterogeneity will not be a big problem as the user actively can decide which devices to buy and add to the system.

5.5 Usability and social considerations
Designing good applications and application mobility solutions is, as with all design, a task in itself. Involving users in the design process is an important element in our ongoing work.

As users in our study emphasized social applications and tools for interpersonal communication as particularly suited for the concept of application mobility systems, we think that it would be advantageous to do prototyping with this in mind. When developing applications to demonstrate CASAM, we have created a mobile mediaplayer, a videoconference system and a simple multiplayer game. Portable workspaces might also work well with our proposed architecture, migrating applications and data that can be used offline on a new, preferably mobile, host.

6. FUTURE WORK
An early implementation of our proposed architecture already exists, building on a modified version of the A2M prototype along with the JXTA framework. Future work will consist of finalizing the implementation of our proposed architecture. Adding intelligence to the migration decisions will also be an important matter. Performance measures will be carried out, and the planned laboratory tests will be complemented with a case study involving real mobile users in an iterative design and evaluation process.

7. CONCLUSION
We have defined CASAM as "Using context, when moving an application between different devices during its execution, to provide relevant information and/or services, where relevancy depends on the user’s task." When designing for CASAM, there are several challenges to tackle: application identification, design of the context awareness model and context quality, seamlessness, heterogeneity, and also usability and social considerations. An architecture supporting this kind of mobility must be able to meet these challenges.

Our proposed architecture has a decentralized P2P basis. A combination with MIP enables access to the P2P network, even if it is widely distributed geographically. Application identification is solved by unique application ids. Context data is distributed so that all nodes can store or quickly access information about other applications and devices in the network. Also, the challenge of heterogeneity is acknowledged, although only partial solutions can be adapted. Our architecture differs from earlier centralized and/or locally bound solutions for application mobility.

Users in our study emphasized social applications and tools for interpersonal communication as particularly suited for the concept of application mobility systems. When developing applications to demonstrate CASAM, we have created a mobile mediaplayer, a videoconference system and a simple multiplayer game.

8. ACKNOWLEDGMENTS
This article has been compiled as part of the Mobile and Open Service Access project (MOSA, 2008) and the NIMO project (Nordic Interaction and Mobility Research Platform, 2011), which are both supported by EU structural funds. The authors would like to thank the participants in the user study exploring the subject of CASAM.

9. REFERENCES