Context-aware Application Mobility Support in Pervasive Computing Environments

Andreas Åhlund
Umeå University
Department of Computer Science
SE-901 87 Umeå, Sweden
ansand03@student.umu.se

Karan Mitra
Monash University,
Faculty of Information Technology
900 Dandenong Road, Caulfield East,
Melbourne, Victoria, Australia
Karan.Mitra@infotech.monash.edu.au

Dan Johansson, Christer Åhlund and Arkady Zaslavsky
Luleå University of Technology,
Computer Science and Electrical Engineering,
931 87 Skellefteå, Sweden
{dan.johansson,christer.ahlund,arkady.zaslavsky}@ltu.se

ABSTRACT
In the future, application mobility can play a crucial role and prove to be an enabler for next generation distributed applications. Application mobility lets an application follow the users while they roam between networks using several devices. In order to achieve seamless application mobility, several issues need to be considered such as device heterogeneity, GUI-adaptation and application loss. Thus, in this paper our contributions are two-fold. Firstly, we present our novel architecture called the Application Mobility Manager (A2M) which provides seamless application mobility. The proposed architecture is context-aware and decentralized. Finally, we present a novel application called the Mobile YouTube Player which is capable of moving between heterogeneous devices and provide users with seamless video experience. We validate the proposed system through rigorous experimentation and user studies based on the real-world test bed and prototype implementation. The results clearly validate that the proposed system can support seamless application mobility.

Categories and Subject Descriptors
C.2.2 **[Computer-Communication Networks]:** Network Protocols– *Applications*

General Terms
Performance, Design, Human Factors, Experimentation, Verification

Keywords
Application mobility, context-awareness, test bed, user-centric, performance evaluation

1. INTRODUCTION
Due to the recent advances in mobile, distributed and ubiquitous computing, we are witnessing a proliferation of heterogeneous devices and applications. Over the past few years researchers have been focusing towards context-aware and user-centric computing where the boundaries between the users, their devices and applications are diminishing [1]. Recently, application mobility has emerged as a new paradigm that tries to bridge this gap by delivering access to user's applications on-the-move on any devices they prefer. Formally, application mobility can be described as the migration of an application from one device to another [2] as shown in figure 1. This figure shows that the application migrates from one device to another i.e., between a desktop, a mini-PC and a projector, to which the user has an immediate access.

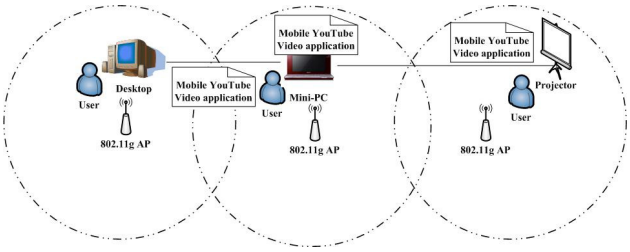

Fig 1. Application mobility scenario: An application moves from one device to another.

In order to provide seamless user experience using application mobility, context-awareness can play a crucial role. In [1], context is defined as a situation of an entity where entity can be anything such as a user, his/her devices, etc. Through context-awareness, devices and the applications can adapt to the user requirements and conditions efficiently in terms of underlying networking technologies and mobility support.

There are several issues that need to be dealt carefully when dealing with application mobility. These are: *a.) Device homogeneity* – most of the prototype implementations such as [3] use Java as the programming language which does not support heterogeneity i.e., different versions of the applications have to be implemented for different platforms, for example, on the desktop PC J2SE is suitable whereas for smaller devices, J2ME is more appropriate. Another problem is that Java does not support execution state mobility. This means that an application will be restarted when it is migrated from one device to another with help of serialization and dynamic class loading. Consequently, in order to resume an application from the same state as it was suspended before, some information concerning its execution state needs to be considered; and *b.) GUI-adaptation* – it is not enough for an

Mobility 2009, Sep 2-4, Nice, France Copyright © 2009
ACM 978-1-60558-536-9/00/0009……$5.00

application to migrate between the devices in order to achieve seamless application mobility. Rather, GUI adaption plays a vital part and needs to be considered in order to build usable applications for heterogeneous devices. For a GUI to be able to adapt to its new host efficiently, it needs to be decoupled from the application logic [4].

This paper aims at providing context-aware application mobility support based on the user and device contexts. In this paper our contributions are two-folds. Firstly, we present a novel context-aware architecture to support application mobility called the Application Mobility Manager (A2M); and finally, we present a novel application called the Mobile YouTube Player supported by A2M which validates our proposed system. A2M is context-aware and decentralized. It is validated through rigorous system performance evaluation and a user study using our test bed and prototype implementation.

This paper is divided into six sections. Section 1 gives an overview of the research domain. Section 2 presents our proposed architecture. Section 3 presents the results analysis. Section 4 presents the related research and finally section 5 presents the conclusion and future work.

2. CONTEXT-AWARE APPLICATION MOBILITY MANAGER

Figure 2 shows our proposed context-aware application mobility solution called A2M. The proposed system is divided into three components i.e., the *Migration Manager* (MM), the *Application Adapter* (AA) and the *GUI Adapter* (GA). The MM can be seen as the system's foundation which provides a generic interface to the applications running on top of it. It is installed on each device and provides the applications with connectivity options, context information and control. The AA encapsulates the applications such as our proposed Mobile YouTube Player, and provides them with the application semantics such as the application state information. It then helps the GA to adapt itself based on the type of device and its capabilities, such as screen resolution. For the scope of this paper, we have used YouTube [5] as the streaming service provider. We have also built a generic Mobile YouTube Player (shown in figure 3) which is encapsulated in the AA. This application can request the videos from the YouTube server on behalf of the users in both HQ and SQ and can migrate between the devices as and when required.

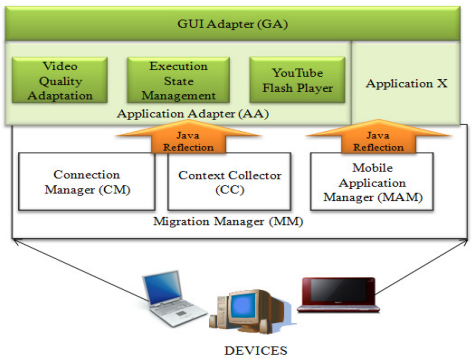

Fig 2. Context-aware Application Mobility Manager (A2M).

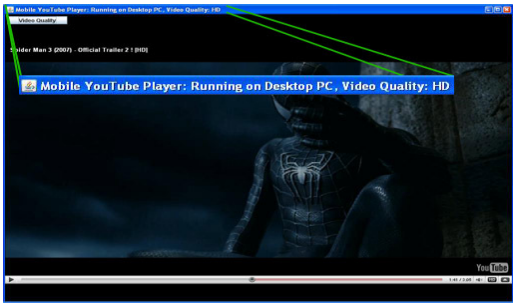

Fig 3. The Mobile YouTube Player.

2.1 Migration Manager

MM is further divided into three modules i.e. the Connection Manager (CM), Context Collector (CC) and the Mobile Application Manager (MAM).

2.1.1 Connection Manager

Device discovery: As mentioned previously that *our proposed architecture is decentralized in nature*. We consider the devices connected to each other in peer-to-peer (p2p) manner. These devices do not maintain any centralized repository to hold information about each other. This was achieved by using multicasting. Here all the devices belong to at least one multicast group. In a multicast group the device discovery is initiated by sending a multicast message to all the devices. After the device discovery, all devices in a particular group can start requesting and retrieving an application from each other.

Server Component: In a dynamic network environment, the behavior of the devices and the applications can change frequently which can lead to disconnections. In case of application mobility, applications can be lost due to such changes. For example, the host machine can shutdown due to low battery, etc. Thus, in such cases, the MM on the host device will send out a multicast message in its group asking for a host that can store and serve the application on its behalf. In this way, at least one host can be selected. The current host device will accordingly choose to send the application to any device in that multicast group on a first-come-first-serve basis. Thus, in this way our prototype *can deal with application loss.*

2.1.2 Context Collector

The Context Collector (CC) is responsible for collecting and manipulating the context pertaining to the device, user and the network environment. For the scope of this paper, we have used user's presence information as the context parameter which is determined using RFID technology. This component enables the devices to know about the user's arrival and departure which can help in triggering the appropriate actions to support application mobility.

2.1.3 Mobile Application Manager

The Mobile Application Manager (MAM) is the most vital component in the Migration Manager (MM). It enables the functionality to control any application (e.g. Mobile YouTube Player) that involves migration and management. The migration of the application is done via a TCP socket connection between the devices. The application migration between the devices is performed by letting the host device first send an application specification containing a string with the application's filenames

and other relevant resources. By receiving the application specification, the requester device will know how many files to expect and how to name them appropriately. Consequently, the next step will be to send the application files. Once the application is transferred from the previous device, to the new device (requester), the connection is terminated. When the application is downloaded on its new device, no prior knowledge of the application will be held. Hence by Java Reflection, the newly downloaded application can be started on the new device from where it was suspended on the previous device.

2.2 Application Adapter

The Application Adapter (AA) adapts the application (Mobile YouTube Player) based on the system's specifications. Once the application semantics are retrieved, it then adjusts the application and maintains its current session state.

2.3 GUI Adapter

The GUI adapter (GA) aims to modify the mobile application's GUI components in order to suit its current host device's user interfaces. This is achieved by equipping each device with a device specific GUI profile, which is responsible for holding the GUI description for a certain application. The GUI profile is then loaded by the GA during a mobile application's resumption phase.

3. RESULTS ANALYSIS

We performed result analysis based on the application performance where we studied the application's suspension, migration and the resumption time in wired, wireless and in mixed (wired-wireless) networking environments. *Suspension time* refers to the time it takes to save the execution state on the host device; *migration time* is the time elapsed from when the application request is sent to when the application has been downloaded on the new host; and *resumption time* refers to the time elapsed from when the Mobile YouTube Player is started on the new device to when the video playback is resumed. We also analyze the results based on the user interviews to further validate the system performance.

3.1 System Performance Evaluation

For performance evaluation three devices were used; a desktop PC, a laptop, and a mini-PC. In case of test 1, the application moves between the mini-PC, the desktop PC and then to the laptop in the wireless environment. In test 2, the application moves between all the PCs in the wired network environment. Finally, in test 3, the application moves in the mixed network environments i.e. from the desktop PC (wired connectivity) to the laptop (wireless connectivity) and from the desktop PC (wired connectivity) to the mini-PC (wireless connectivity). In test 3 the application first moves back to the desktop from the laptop and then moves from the desktop to the mini-PC. In all the cases both HQ and SQ based tests were evaluated except for the cases where the Mobile YouTube application to mini-PC was transferred. This was due to the fact that our mini-PC could not manage the HQ video because of its limited processing capabilities. The total payload communicated between devices for migration was 2.39 Mb, of which the size our developed application was mere 28Kb whereas the remaining size was for the Java plug-in we used [6].The video used during the experiments was a Spiderman 3 trailer with the duration of 3:05 min which was available in both HQ and SQ. Figures 4 and 5 depict the total delay in terms of the summations of suspension, migration and resumption time for all the tests, and all values are mean values of 30 measurements. As can be seen in the figures, the suspension time is not varying much for either the HQ or the SQ. The slightly increased time for the wireless part is due to the communication to receive execution state parameters from the YouTube server.

The migration time is significantly lower in the wired network environment (IEEE 802.3) compared to the wireless counterpart (IEEE 802.11g). Due to significant interference observed by the 802.11g devices in our office environment, we were only able to achieve a throughput of 6Mbps in the wireless network. The resumption time of the application that occurred in the receiving device comprises of the application start-up time and the time required for fetching the application content. The Mobile YouTube Player is downloaded onto the new device and at the same time a request sent to the YouTube server to access the Spiderman video in either HQ or SQ. The resumption time varies slightly depending on the network setup and video quality. The longer time for resumption of the HQ video (compared to SQ) can be attributed to the buffering of bytes needed from the YouTube server to play out the first frame in the Mobile YouTube Player. We will show later in the user study that even with the total delay of nearly 9 seconds, the user does not consider it as a significant disruption to his/her user experience.

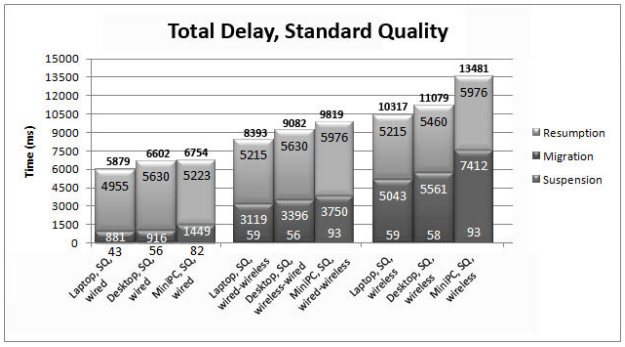

Fig 4. Total delay observed on devices for SQ in wired, wireless and mixed network environment.

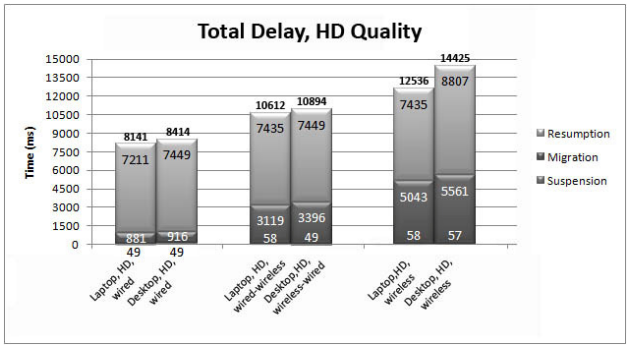

Fig 5. Total delay observed on devices for HQ in wired, wireless and mixed network environment.

3.2 User Interviews

The performance evaluation in the previous section shows the overall delay users will experience using our proposed solution. However, these results are not conclusive of the fact that the overall system performance is acceptable by the users or not.

Thus, in this section we present the results of an initial user study, where the aim was to evaluate the system's performance and the overall concept of using a mobile video player from the user's perspective. The evaluation was scenario based, where the users got to carry out a specific task while using the system, followed by an open ended interview session with the participants. In our user tests, the population size comprised of 10 participants, including 7 male and 3 female users. The population was a mix of both staff and student members. The lowest age was 20 while the highest being 60. Because of the fairly small target group, the evaluation was done with qualitative open-ended interview questions, mainly since more users are needed to draw any statistical conclusions from a quantitative evaluation. From the study we can conclude that the users were quite satisfied with our proposal. It's worth noticing that during the demonstrations, the users did not seem to notice the mean resumption delay of nearly 9 seconds (HQ video). In regards to the general usefulness of the system and in particular, the Mobile YouTube player, we got positive responses as well. Most of the users liked the proposed system. They believe it has a potential to be used in everyday lives such as in-home scenario where a user could move from one room to another while watching a video.

4. RELATED WORKS

Chu *et al.* [3] designed and implemented a seamless application framework called "Roam" to let the developers build multi-platform applications that are able to run on heterogeneous devices. Their solution is centralized with a proactive migration approach. Also, the application loss problem has not been taken in account. Siu *et al.* [7] implemented a state migration mechanism in the Sparkle Pervasive Computing Environment. Their technique covers state capturing at application level as well as the usage of context to adjust the application state to suit the new environment after migration. Their solution is decentralized with a proactive migration approach, and supports device heterogeneity. The loss of application problem is not considered. Yu *et al.* [8] describes a mobile agent based paradigm for enabling an application to move with the user between devices. During the analysis, applications were moved from one PC to one Laptop. Indications showed that their solution is implemented in Java and are web service based which means that their solution is centralized. The system is sensitive to user's execution context. However, in their proposal it is uncertain whether their solution supports device heterogeneity since the experiments are conducted on a Desktop and Laptop PC.

Compared to the related work presented in this section, our proposal is decentralized and is based on p2p networking approach. Thus, there is no centralized authority to maintain state information. Using multicasting and by using the server component, A2M ensures that there are no application losses. Our proposal is context-aware which determines user presence and then sends the application to device closest to the user. Finally, we also validate our proposal using the user study which is not present in the related research.

5. CONCLUSIONS AND FUTURE WORK

In order to provide seamless user experience to the end user, 'anytime' and 'anywhere' kind of services will need to be supported on the network infrastructure. We believe through application mobility that this goal can be achieved. However, there are several challenges that need to be tackled. Issues such as device heterogeneity, GUI-adaptation and application loss can hamper user experience in pervasive computing environments. Thus, in this paper we propose a novel context-aware architecture called A2M. A2M addresses the aforementioned issues and presents itself as a capable architecture to provide application mobility. It is decentralized and is based on the p2p network paradigm. Thus, it minimizes application loss. It is context-aware as it can sense user presence and then route the application to the device closest to the user. We validated the proposed system through extensive performance evaluation and the user study conducted via our live prototype implementation. To the best of our knowledge, ours is the first proposal which presents the results based on user evaluation. In future, we also plan to include other types of applications such as Web browser and PDF readers.

6. ACKNOWLEDGMENTS

Our thanks to the EU structural funds (Objective 2) and the Swedish savings bank foundations for funding this project.

7. REFERENCES

[1] A.K. Dey and G. D. Abowd, "Toward a better understanding of context and context-awareness, GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology. ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf," 1999.

[2] T. Koponen, A. Gurtov, and P. Nikander, "Application mobility with hip," in *HUT T-110.557 Research Seminar on Telecommunications Software*, 2004, pp. 50-62.

[3] H. Chu, H. Song, C. Wong, S. Kurakake, and M. Katagiri, "Roam, a seamless application framework," *Journal of Systems and Software,* vol. 69, pp. 209--226, 2004.

[4] F. M. David, B. Donkervoet, J. C. Carlyle, E. M. Chan, and R. H. Campbell, "Supporting adaptive application mobility," in *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, 2007, pp. 896--905.

[5] "YouTube online video provisioning service," [ONLINE] http://www.youtube.com/, Access Date: 14-06-09.

[6] "JFlashPlayer," [ONLINE] http://www.jpackages.com/jashplayer/, Access Date 22-05-09.

[7] P. P. L. Siu, N. Belaramani, C. L. Wang, and F. C. M. Lau, "Context-aware state management for ubiquitous Applications," in *Embedded and Ubiquitous Computing*, 2004, pp. 776--785.

[8] P. Yu, J. Cao, W. Wen, and J. Lu, "Mobile agent enabled application mobility for pervasive computing," in *Ubiquitous Intelligence and Computing*, 2006, pp. 648--657.