Web-based Adaptive Application Mobility

Dan Johansson and Karl Andersson

Department of Computer Science, Electrical and Space Engineering

Luleå University of Technology

SE-931 87, Skellefteå, Sweden

Email: {dan.johansson, karl.andersson}@ltu.se

Abstract—Application mobility is achieved when migrating an application with its code, states and all related information from one device to another during its execution. As traditional applications are being challenged by apps, we believe it is time to rethink the notion of application mobility and explore the concept from a web perspective. More specifically, this paper aims to address the challenges associated with application mobility through mapping these against features within modern web technologies, analyzing their ability to meet the specific requirements of application mobility. An architectural proposal is described where we deploy application mobility using web technologies. We show that the emerging HTML5 standard along with related APIs and frameworks can provide an environment for delivering application mobility that strongly meets the requirements of support for offline work and heterogeneous environments.

I. INTRODUCTION

Today it is hard to imagine IT usage without mobility. The number of active mobile-broadband subscriptions worldwide now exceeds 1.2 billion, with 45% of the population covered by 3G or better [1]. Smartphones and other mobile devices constitute platforms for mobile services [2] while improving user mobility. They also extend the mobile infrastructure, taking the role as access points providing WiFi, 3G and/or 4G hotspots. This emerging mobile landscape of new small and powerful devices combined with far-reaching wireless networks and ever-growing bandwidths has both enhanced and brought on new challenges to all kinds of mobility aspects, e.g. service mobility (making services available regardless of devices or networks used), session mobility [3] (seamlessly keeping media streams active although switching user location, network and/or device) and of course terminal mobility.

The ever increasing amount of modern handsets such as smartphones and tablets run third-party mobile apps. Many of these apps are native, often due to performance reasons. However, the emergence of HTML5 [4] and related mobile frameworks has created a belief that web technology eventually will become indistinguishable from native counterparts [5].

Mobility can be achieved in many ways. One specific type of service mobility is when migrating a whole application with code, states and all related information from one device to another during its execution. This is called application mobility [6]. Usually the act of moving applications in this way is carried out in three steps: the *suspension phase*, when the application is paused and its states stored; the *migration phase*, moving the actual application from the original host to a new

one; and the *resumption phase*, restarting the application on the new host, letting the user carry on using the application where he left off [7].

There are many examples of projects addressing application mobility. Most solutions rely on centralized distribution of applications (MDAgent [7]; Sparkle [8]; Gaia [9]; MSP [10]) and/or code loading (Roam [11]; SAMProc [12]), when a central server is used to store different versions of the application. Other projects display decentralized solutions, using multicasting as means of communication and signaling (e.g. Hydra [13], A2M [14]), or peer-to-peer technology in combination with Mobile IP [15]. These projects are all examples of native application mobility, native as in based on machine code and manifested in compiled software applications. As shown in Section II of this paper, most of these systems are also specialized rather than standardized, small-scale (sometimes restricted to a local network) and does not strongly support heterogeneity in devices or platforms.

As traditional applications are being challenged by apps, we believe it is time to rethink the notion of application mobility and explore the concept from a web perspective. More specifically, this paper aims to address the challenges associated with application mobility through mapping these against features within modern web technologies, analyzing their ability to meet the specific requirements of application mobility. An architectural proposal is described where we deploy application mobility using web technologies.

The rest of this paper is structured as follows: Section II will summarize related work on application mobility and present the challenges associated with this field. Section III describes the features of the HTML5 standard and some related frameworks that we then (in Section IV) map against the challenges of application mobility. In this section we also present our architectural proposal, deploying a platform for application mobility by using web technologies. Section V contains a discussion concerning future work and also concludes the paper, summarizing our work.

II. RELATED WORK

A. What is an application?

Traditionally, applications have been defined as software running on a computer helping the user perform specific tasks, contrary to e.g. system software. With the emergence of mobile apps, the definition has been extended to include software running on smartphones and tablets. These applications, or

apps, can be either native or web-based. Web apps are coded in languages interpretable by a common web browser. Commonly, these applications have been designed following a three-tier architecture consisting of presentation (user interface), logic (server) and data (data management), all relying on programming models and languages of very different types. It is argued [16] that this complexity already has been reduced significantly by the emerging web programming standards.

Charland and Leroux [5] discuss native mobile applications versus web apps. It should be noted that the term mobile application in their definition means application running on a mobile device. They state that building different versions of applications for all the different platforms is both difficult and expensive. Looking at the different platforms, they all have different SDKs (software development kits), making the possible deployment context very heterogeneous in terms of supported devices, tools, build systems and APIs. The only common denominator these platforms have is the browser. As such, the browser becomes a natural choice if one wants to deploy applications over a wide range of devices and platforms. Another result of their research is that the performance of native applications compared to web based ones is negligible or even unnoticeable unless the application has a heavy reliance on 3D games or image processing.

The fast growing importance of JavaScript is underlined by Kienle [17], Charland and Leroux [5] and others, and clearly manifested in the fact that currently all major HTML5 APIs are based on that language. It is interpreted in contrast to compiled languages like Java and Objective C and brings client side functionality. The code is run by the browser, making it truly platform independent.

B. Projects Targeting Application Mobility

Most existing projects target native application mobility. In the Roam system [11], applications can run on heterogeneous devices connected to the system and migrate at runtime between them. The applications must be designed following a very specific component-based programming model, creating so called roamlets. Roamlet class byte code can then be downloaded from a central server and adapted to fit the new device. The SAMProc [12] project works with self-adaptive mobile processes in a similar way, however using a decentralized system infrastructure based on peer-to-peer. Both projects are examples of systems for application migration based on code loading.

In a system called Sparkle [8], states are captured and migrated between devices using what the authors call a universal browser, specifically developed for the Sparkle system. This allows for Java applications to be restored on a new host, using the Java reflection technique. Other centralized systems are Gaia [9], MDAgent [7] and MSP [10], which all make use of a migration server as a means of managing the redistribution of applications within the system. Hydra [13] and A2M [14] both make use of decentralized migration platforms, removing the migration server from the architecture. Both systems are however very limited in scope, bound to a

Local Area Network. An extension of the latter system [15] proposes a solution to the scope limitation adding Mobile IP, allowing mobile nodes in different networks. The system is however still limited in terms of heterogeneity, using Java applications and a Windows platform and a small range of devices.

One of the earliest examples of projects targeting web application mobility is an unnamed system by Bandelloni, Mori and Paternò [18]. The system allows the user to interact with an application while switching devices. Different versions of an interface are created by a migration server, allowing the application to be accessed through mobile and heterogeneous devices. The main functionality of the application is still maintained by the central server, making the very mobility of the application limited, as only the interface is migrated and adapted. Bellucci et al [19] extend this research, focusing on how to preserve states in JavaScript code that is part of migrated web applications. A migration control panel is deployed on every potential host device. When acting as an application host, the control panel periodically announces itself to a migration server, where it can get information about available target devices. It can then connect to a migration server that calls up and adapts the user interfaces for the new device.

Summarizing earlier projects, we can see that although presenting good solutions for achieving different versions of application mobility, all projects also have limitations. All native projects are non-standardized, i.e. specialized in the sense that they require certain software and/or programming models to support application mobility. Some systems can only be deployed locally, that is in an environment where the system owner can control all routers and/or a Local Area Network. Last, full heterogeneity, e.g. being able to run the applications effectively in an environment consisting of devices of different brands, running different operative systems, is not achieved in most of the reviewed systems.

C. Requirements for Application Mobility

A set of requirements for systems providing application mobility is presented by Johansson, Åhlund and Åhlund [15]. These are clustered into five main topics.

1) *Application Distribution and Identification*: The method of distributing the application (and/or its related data) must be integrated with the system. It should contain means to identify the application and regulate who the owner is and who can use the application and under what circumstances. If net connection goes down, the user must be able to continue using the application.

2) *Context-awareness and Context Quality*: As application mobility works cross-device/cross-environments awareness of the use situation – the relevant *context* – is of high importance. Availability of new hosts, their capabilities and constraints, location and other pieces of context must be used by the system to make good decisions regarding e.g. where to migrate an application and when. As context data is often heterogeneous in terms of units and scale and also the rate in which it is

collected, achieving context quality is also very important. Context data should be interpretable, comparable and up to date. How this data should be stored and distributed is also something a designer of a system for application mobility must take into account.

3) *Seamlessness*: When moving an application from one device to another, the migration phase must be quick. Preferably, the user should experience the migration as seamless with regards to user expectations and the context.

4) *Heterogeneity*: The number of different possible mobile devices and platforms is vast, and the task of deploying application mobility in this heterogeneous environment is a difficult thing. Nevertheless is the heterogeneity challenge of utmost importance when designing for application mobility. In fact, application mobility is in a way motivated by the existence heterogeneity – if all platforms and devices were equal in their capabilities and mobility support, there would hardly be any reason for moving the application in the first place, as one single device would be enough to address all user needs in the best way.

5) *Usability*: Usability is no specific requirement for application mobility, rather a general requirement for all application development. An application, mobile or not, must be easy to use, intuitive and robust.

III. HTML5 AND RELATED FRAMEWORKS

In 2007, the World Wide Web Consortium (W3C) started to cooperate with the Web Hypertext Application Technology Working Group (WHATWG) on the fifth major revision of the Hypertext Markup Language, HTML5 [4] [20]. The work is still in progress, but all major browser developers constantly provide their latest versions with new support for various parts of the standard [5].

One important motivation for the revision was the dominance of vendor specific technologies such as Adobe's Flash, Apple's QuickTime and Microsoft's Silverlight, threatening the web's status as open, widely usable and platform-independent. HTML5 should provide users with a standard that contains enhanced functionality in terms of audio and video presentation, support for 2D and 3D graphics, local storage and new content-specific elements. These functionalities should be provided without forcing the users and developers to bind themselves to specific technologies and vendors. [21]

When we talk about modern web technology in this article we include more than just HTML5. We also consider the third revision of Cascading Stylesheets (CSS3), and various HTML5 related frameworks and API:s as important parts of the puzzle.

A. HTML5

A specification of the HTML5 language is provided in the W3C working draft [4] along with a delineation of its functionalities.

Among the most fundamental new features of HTML5 we find different kinds of embedded content. The canvas element provides a container for drawing graphs and graphics using

scripts, rendering these graphics locally. Server communication is thus limited to a minimum. Other important embedded content are the audio and video tags. The audio element is used for playing sound and audio streams, while the video element represents media data in the form of movies, video or captioned audio. No specific plug-ins or video players are used, and the idea is to provide non-proprietary multimedia support.

The support for working offline is prominent within HTML5. The browser can keep copies of files that are essential for an application to continue to work even while a network connection is unavailable. Data and code is specified within an application cache that can also contain fallbacks, in case the application needs to adapt its presentation or functionality due to the change in context.

Last, some improvements have been made regarding semantics. HTML5 code can be written in either HTML or the stricter XML and also provided with machine-readable markup in the form of RDFa (Resource Description Framework) as an extension to the HTML5 syntax. This allows transformation of elements visible to humans into data that can be understood by machines, without having to distribute such semantics in separate files.

B. API:s and Frameworks

Sometimes an application can benefit from having some scripts running in the background. Web Workers [20] is presented as a means of running heavy-weight, often long-time tasks without interacting with the user. The idea is to allow for an application to carry out background work without having to interrupt or slow down the user.

Another improvement when it comes to online communication is the Web socket API [20], which allows for full duplex data exchange between server and client. The communication channel is opened over a single socket native to the browser, not relying on plug-ins. This enables bidirectional communication, qualitatively advantageous over older techniques like polling, which are repetitive requests sent at regular intervals, and long-polling, i.e. requests that are kept open for a set period, awaiting a notification that may never come.

The geographical position of a user (in reality a device, although the two are often at the same physical location during usage) can be derived through the W3C Geolocation API [22]. The API itself works as a high level interface, so the actual location data can be received from a number of sources, e.g. user input, Global Positioning System (GPS), the assigned IP address, MAC addresses from Bluetooth, RFID and WiFi or GSM/CDMA cell IDs.

Among other APIs we find Notification and Vibration, making it possible to notify and wake up sleeping devices and also give tactile feedback in the form of vibrations. The File API gives the opportunity to represent file objects from underlying file systems in web applications. We also find support for new modalities such as Speech input. [22]

C. CSS3

Cascading Style Sheets is the language used to describe how web pages should be presented in a browser. The latest version, CSS3 [22], provides new features. Most of them are completely design-oriented, making it easier for developers to design and assign colors, borders, backgrounds, text effects and transitions to the different elements of a web page.

Modernizr [23] is a collection of JavaScript libraries, using feature detection to ask a browser which features it supports. This makes it easy to use the new CSS3 when designing for a wide number of browsers and platforms. It also styles HTML5 features and adds fall-back possibilities, should a feature not be supported. Modernizr, which is open source and used by major actors like Google, Twitter and Microsoft, has become the standard feature detector and style adapter for HTML5 web design.

IV. WEB-BASED APPLICATION MOBILITY

The purpose of this paper is to explore to what extent emerging the HTML5 standard along with related APIs and frameworks can provide an environment for delivering application mobility that strongly meets the requirements of support for offline work and heterogeneous environments. We call this approach web-based application mobility.

Architectural requirements are collected from related work (see Section II) and summarized as follows:

- 1) Must handle application distribution and identification
- 2) Must be context-aware and maintain context quality
- 3) Must provide seamless migration, minimizing downtime
- 4) Must cope with diverse environments, i.e. migration between different networks, heterogeneous interfaces, resources, local preferences etc
- 5) Must have high degree of usability

A. Architectural Proposal

An overview of our proposed architecture for web-based application mobility can be found in figure 1. The most important architectural goals consist of replacing native mobile applications with web-based equivalents, moving from specialized systems to methods relying on open and accessible standards, enabling global scale deployment and migration (contrary to systems limited to Local Area Networks). We also have a vision of obtaining full heterogeneity, in our definition be able to migrate applications between devices, both mobile and stationary, of different brands running different operative systems.

In our first version of the architecture we have opted for a centralized solution. This approach makes the system more sensitive to disturbances in Internet access, but we recognize that drawback and will address it as far as current open HTML5 related APIs allow us. The choice of design is further motivated by the fact that – as per date – available APIs cannot support full peer-to-peer functionality. This makes it impossible to build a decentralized system for web-based application mobility without extending the browser or adding other specific software. The centralized web-based approach

on the other hand brings a standardized infrastructure that can easily be deployed as a part of the existing Internet.

Thus, a *migration server* constitutes the central hub in the architecture. It works as a back-end, keeping record of users, applications and devices within a *component registry*. Migrateable applications are stored in an *application repository*.

A mobile application, or *migrateable application*, is the application being transferred from one device to another via the migration server. Applications are coded in HTML and JavaScript. CSS is used to define how to present the application while feature detection (e.g. by using the Modernizr libraries) is presupposed, making it possible to render different designs and functionalities depending on device capabilities. Standard migration functionality is provided through an application called the *migration manager*, deployed on each host node (in that differing from other migrateable applications).

Connected to the migration server are the *host nodes*. A host node is a user device that can receive migrateable applications from the migration server and execute them. Host nodes are registered with the migration server, forming a set of devices, which in turn is bound to a specific user. Users identify themselves with unique user credentials. The host node can be any device with a network interface and a web browser. Laptops, smartphones and tablets are all examples of candidate host nodes. An optional extension to the architecture is what we call *server nodes*, in our definition host nodes that have server capabilities. Preferably, most communication between two host nodes should go directly between nodes, not involving the migration server. E.g. a videoconference application should make use of the Web Sockets API for the bidirectional communication between the participants of that videoconference. This is however only possible in the event of server node addition. It should be underlined that the architectural proposal presented in this paper does not require server nodes, although we want to take to opportunity to open up for such an extension.

B. Application Distribution and Identification

For identification of applications we conveniently use the infrastructure of Internet. Most resources on the Internet can be accessed via Uniform Resource Identifiers (URIs) and web applications are no exception. A default version of every application is always stored in the application repository of the migration server. This avoids the risk of application loss, which could arise as the result of a host node unexpectedly shutting down while hosting the application. As an application is migrated, it will still be stored in its default version but made unavailable to other host devices. There should only be one (1) copy of the application running by each user, and the application repository version should only be treated as a back-up alternative.

However, there is also need for identifying single copies of an application. We use a naming scheme consisting of an application ID combined with a user name. Thus `webapp1@alice` and `webapp1@bob`, become unique copies of the migrateable web application available in the application repository. Naming

Fig. 1. Architectural overview

of devices is carried out in the same way, e.g. device1@alice and device2@alice. The possibility to identify each device becomes important for instance if the device is assigned a new IP number. In the case of a network loss, the migration manager checks the local db for the last recorded IP number and compares that with the current. If these two mismatch, the migration manager on the host node updates the migration server with the new IP number.

Our architecture allow for the use of existing web security solutions. Data can be encrypted as an option. User credentials can be maintained and emitted from the migration server or added through the use of third-party cloud services (e.g. Google ClientLogin).

The actual application is run with support from application cache, making it possible to continue working with the application (albeit with less functionality) in the case of network connection loss. All states are stored in the local db of the browser, minimizing both the need for server communication and the risk of losing important data, should the connection go down.

Communication between host nodes and the migration server is carried out using standard protocols of the Internet such as HTTP and TCP. When there is need for polling or background communication between the server and the client, Web Workers API can be used to run JavaScripts without blocking the user interface and Web Sockets can allow bidirectional communication if needed.

C. Context-awareness and Context Quality

Context data is stored both on host devices and in the component registry of the migration server. As the outlined architectural model for application mobility is centralized, information about users, devices and applications are also stored centrally. Context data concerning users e.g. contain which devices they own, while device context data cover input and out capabilities, processing power, network interfaces etc. Application data contain information about current host and also a profile describing the requirements of the application, e.g. if it needs a host with certain output capabilities to run properly. As XML is compliant with HTML5, it is the preferred format for context data storage, although the architecture does not limit to this format.

Local db is used to store application dependent data, decreasing Internet dependency when it comes to running the actual application.

Some context data is stable over time, e.g. the size of a screen or camera quality. However, there is also more volatile context data, e.g. current battery level, which can be crucial for making intelligent decisions about where and when to migrate an application. This data must be communicated at the same instant a migration process is being initiated.

To illustrate this, a host initiated migration of an application is depicted in figure 2. It is carried out in the following steps:

- 1) Alice is running webapp1@alice on a semi-stationary host node (a laptop) named device1@alice when a change in work context make her want to go mobile. Thus she initiates a migration process by contacting the migration server.

Fig. 2. Host initiated application migration

- 2) The migration server compares the application profile with the host profiles stored in the component registry. Using a reasoning algorithm – which can be simple or advanced depending on system design (the architecture is not limited to one single algorithm) – it chooses a candidate node for new host, i.e. a node which has capabilities that best suit user and application preferences. In this example the candidate node is a tablet computer named device2@alice.
 - 3) The migration server contacts the candidate node, which in turn compares the application profile with its more volatile context status (e.g. battery level). If the candidate host cannot meet the requirements it must decline the migration request and the migration server must then repeat step 2. Otherwise the migration request is accepted.
 - 4) The migration server contacts the current host node and request its application states and all information related to the app stored in the Local db. These are transferred to the migration server.
 - 5) The migration server migrates the application from its Application repository along with the states and data from the former host node to the new one.
 - 6) The application adapts to the tablet computer according to the description in the Cascading Style Sheets and is resumed. Alice can continue working where she left off on her laptop. Device2@alice is registered as the new host of the application in the component registry of the migration server, and the application cache of device1@alice can be emptied.
- Host initiated migration can be triggered by the user or the system. An example of the latter initiation is if the system notices a dramatic change in context, then Notification API is used for urgent signaling. E.g. if battery is running low on a host device and it needs to initiate a migration process. The migration can also be receiver initiated. The migration process is then carried out in almost the same manner as depicted in figure 2 but with a few important differences: first, obviously

TABLE I
MEETING THE REQUIREMENTS

<i>Requirement</i>	<i>Architectural Features</i>
Application Distribution and Identification	URI, Web Security Solutions, Local db, Application Cache, Web Workers, Web Sockets
Context-awareness and Context Quality	Local db, Notification, Location based services, XML Compliance
Seamlessness	Browser functionality, Web Workers
Heterogeneity	CSS3, Feature Detection
Usability	HTML Familiarity, Browser Interface, Feature Detection, Multimedia Support

it is host node 2 that initiates the migration process; second, host node 1 or its user must explicitly allow the migration of the application. Finally, host node 2 already has knowledge of its volatile context status and can attach this to the initial request, thus eliminating the step where the migration server has to ask for it. Also there is no need for choosing a candidate node as the initiating host node already volunteered for that role.

Context has traditionally been very heavily associated with location, and indeed is location based services a very common feature of modern web apps. This context data can e.g. be used for localizing available host devices adding a geographical dimension, for instance consider migrating a video player application to a computer with a big screen if the system can sense geographical presence to this resource.

D. Seamlessness

Migrating an application from one device to another should be done with minimum delay while still maintaining functionality, states and related data (be it the contents of a text document or an ongoing audio/video session). As we use standard web technology for rendering web applications, presenting the user a first view should not take longer than accessing and downloading a normal web page. Running JavaScripts will inevitably take longer time than it native counterparts, but our hypotheses is that this will be countered by a shorter migration phase compared to native application mobility.

Addition of history based decision-making could further decrease the total migration time, as this could shorten step 2 in the migration process (as depicted in figure 2). Finally, Web Workers could be assigned to prepare migration, working in the background gathering context data and making preparatory decisions which can be used in the case of a migration initiation.

E. Heterogeneity

Heterogeneity has always been a big challenge for designers of application mobility. Being able to deploy an application on a new device regardless of platform or capabilities has been regarded as impossible, although many ideas have been adopted to address this matter, at least to minimize the heterogeneity constraints. The web environment, including the emerging CSS3 standard, turns the tables on this matter. A web application can, in theory, be deployed on such heterogeneous devices as stationary computers, smartphones and tablets, interpreted by browsers run on diverse platforms such

as Android, iPhone OS or Windows without changing the code. The heterogeneity challenges are – if not overcome – addressed on a broad scale by the new web technologies.

Feature detection makes it possible to ask the browser which functionality it supports and use this in the decision process of where to migrate an application and/or how to adapt the application to the new host.

Altogether, this allows us to capitalize on the biggest advantage of application mobility, that is always running an application on the best available device according to the current context.

F. Usability

Usability is of importance for all application development. Systems and applications need to be robust, easy to use and intuitive. By creating an architecture for application mobility using common web technology, much is won in terms of environmental familiarity. The browser acts as the common interface for all migratable applications, providing the user with an intuitive GUI. This however does not guarantee a good design of the actual application and its UI, but the simple recipe of HTML5, CSS3 and related APIs give developers a good foundation to create applications when it comes to defining structures, presenting them and providing functionality. Open source templates in combination with feature detection give designers easy means of adapting their applications to different platforms, making the GUIs familiar and intuitive to use for the users, expecting different functionality and design e.g. on their iPhone compared to if running the web app in a laptop with a Windows platform.

As multimedia services are considered among the most important today, HTML5 video and audio support will add functionality to web applications that make them useful. In the end, user acceptance will depend on use motivation.

V. CONCLUSION AND FUTURE WORK

Our main goal with this paper was to address the challenges associated with application mobility through mapping these against features within modern web technologies, analyzing their ability to meet the specific requirements of application mobility. An architectural proposal was described where we deployed application mobility using web technologies. We showed that the emerging HTML5 standard along with related APIs and frameworks can provide an environment for delivering application mobility that strongly meets the requirements of support for offline work and heterogeneous environments. A summary of how the challenges were met can be found in table I.

When it comes to future work, we intend to develop and demonstrate a full scale proof-of-concept, based on the architecture presented in this paper. A user study will be conducted along with quantitative measurements of – in particular – migration times. Our intention is also to compare these upcoming results with other solutions for application mobility where measurements have been conducted.

Finally, replacing the centralized solution in this paper with a peer-to-peer infrastructure could improve mobility aspects. Thus, this is a prioritized future work. As open web-based peer-to-peer APIs emerge, this could well be possible in a near future.

ACKNOWLEDGMENT

This work was supported by the NIMO (Nordic Interaction and Mobility Research Platform) project [24] funded by the EU Interreg IVA North program.

REFERENCES

- [1] ITU. *The World in 2011 – ICT Facts and Figures*. URL: [http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures 2011.pdf](http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures%202011.pdf). 2011.
- [2] K. Andersson & D. Johansson. Mobile e-Services Using HTML5. To appear in *Proceedings of the 37th IEEE Conference on Local Computer Networks (LCN)*. 2012.
- [3] D. Johansson. Session mobility in multimedia services enabled by the cloud and peer-to-peer paradigms. In *Proceedings of the 36th IEEE Conference on Local Computer Networks (LCN)*. 2011.
- [4] W3C. *HTML5. A vocabulary and associated APIs for HTML and XHTML*. W3C Working Draft, March 2012. URL: <http://dev.w3.org/html5/spec/>. 2012.
- [5] A. Charland & B. Leroux. Mobile application development: web vs. native. In *Communications of the ACM* 54, 5, 49-53, May, 2011.
- [6] D. Johansson & M. Wiberg. Conceptually advancing "application mobility" towards design: Applying a concept-driven approach to the design of mobile it for home care service groups. In *International Journal of Ambient Computing and Intelligence*, vol. 4, no. 3, pp. 2032, 2012.
- [7] Y. Zhou, J. Cao, V. Raychoudhury, J. Siebert & J. Lu. A Middleware Support for Agent-Based Application Mobility in Pervasive Environments. In *Proceedings of the 27th international Conference on Distributed Computing Systems Workshops*. ICDCSW. IEEE Computer Society, Washington, DC, 9, 2007.
- [8] P. P. L. Siu, N. Belaramani, C-L. Wang & F. C. M. Lau. Context-Aware State Management for Ubiquitous Applications. In *Embedded and Ubiquitous Computing*. Yang, L. T., Guo, M, Gao, G. R, Jha, N. K. (Eds.). International Conference on Embedded and Ubiquitous Computing, 776-785. Aizu, Japan, 26-28. August, 2004.
- [9] A. Ranganathan, C. Shankar & R. Campbell. Application polymorphism for autonomic ubiquitous computing. *Multiagent Grid Syst.* 1, 2, 109-129. April, 2005.
- [10] K. Hojgaard-Hansen, H. C. Nguyen & H. Schwefel. Session mobility solution for client-based application migration scenarios, *Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, pp.76-83, 26-28, January, 2011.
- [11] H. Chu, H. Song, C. Wong, S. Kurakake & M. Katagiri. Roam, a seamless application framework. In *J. Syst. Softw.* 69, 3, 209-226, January, 2004.
- [12] H. Schmidt, R. Kapitza & F. J. Hauck. Mobile-process-based ubiquitous computing platform: a blueprint. In *Proceedings of the 1st Workshop on Middleware-Application interaction: in Conjunction with Euro-Sys 2007 (MAI '07)*, vol. 224. ACM, New York, NY, 25-30. March 20, 2007.
- [13] I. Satoh. Self-deployment for Distributed Applications. In *Scientific engineering of distributed Java applications: third international workshop* (N. Guelfi Ed.). Berlin, Springer, 48-57. 2004.
- [14] A. Åhlund, K. Mitra, D. Johansson, C. Åhlund & A. Zaslavsky. Context-Aware Application Mobility Support in Pervasive Computing Environments. In *Proceedings of the International Conference on Mobile Technology, Applications and Systems Conference, ACM Mobility*, Nice, France, September 2-4, 2009.
- [15] D. Johansson, A. Åhlund & C. Åhlund. A MIP-P2P Based Architecture for Application Mobility. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia*, Beijing, China, December 7-9, 2011.
- [16] M. Laine, D. Shestakov, E. Litvinova & P. Vuorimaa. Toward Unified Web Application Development. In *IT Professional*, vol.13, no.5, 30-36. 2011.
- [17] H. M. Kienle. It's About Time to Take JavaScript (More) Seriously. *IEEE Softw.* 27, 3 (May 2010), 60-62. 2010.
- [18] R. Bandelloni, G. Mori & F. Paternò. Dynamic generation of web migratory interfaces. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services (MobileHCI '05)*, 83-90. 2005.
- [19] F. Bellucci, G. Ghiani, F. Paternò & C. Santoro. Engineering JavaScript state persistence of web applications migrating across multiple devices. In *Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems (EICS '11)*. ACM, New York, NY, USA, 105-110. 2011.
- [20] WHATWG. *HTML Living Standard*. URL: <http://www.whatwg.org/specs/web-apps/current-work/multipage/>. 2012.
- [21] S. J. Vaughan-Nichols. (2010). Will HTML 5 Restandardize the Web? In *Computer*, vol. 43, no. 4, pp.13-15, April 2010.
- [22] W3C. *All standards and drafts*. URL: <http://www.w3.org/TR/>. 2012.
- [23] Modernizr. URL: <http://modernizr.com/>. 2012.
- [24] NIMO. *Nordic Interaction and Mobility Research Platform*. URL: <http://nimoproject.org/>. 2012.