A Push Infrastructure for Mobile Application Deployment in Mobile Environments

Miguel Borges[1], António Nestor Ribeiro[2], and José Creissac Campos[3]

DI-CCTC, Universidade do Minho, Braga, Portugal
borges.miguel@gmail.com[1], {anr[2],jose.campos[3]}@di.uminho.pt

Abstract. Mobile devices tend to be a synonym of variety. Variety both in hardware capabilities and software act as restrictions to software development and deployment. Other restrictions arise from their condition of mobility, environmental conditions such as bandwidth, coverage availability, lighting and availability of services. In that perspective, this work intends to explore the possibility of a model of application deployment and execution that minimizes these issues - software gets pushed through an infrastructure and interaction between the user and the application is expected to have a behaviour between these two modes: purely client based and purely online based.

1 Introduction

The property of mobile devices that better describes their current technological development status is their heterogeneity. From it, a series of problems common to various technical areas arise. One of them - relating to the interaction with information systems or, more specifically, with applications, lays its foundations in two basic application models. These are in every aspect similar to the ones developed with ”non movable” computers - fat client model and thin client model. Respectively a model based on compiled code that gathers all the application logic in the device, and an online model that keeps all application logic in a remote server and enables the user interaction by means of a web browser.

In a personal computer, the thin client model is basically composed by a browser, by the HTTP protocol and by a markup language. In mobile devices, this model still prevails almost unmodified: a browser that understands a markup language and communicates using a connection oriented protocol. In some recent devices, browsers and the languages used, are exactly the same as in personal computers. Although this approach may seem natural, it is not always beneficial to the mobile devices that mimic the personal computer because of their heterogeneity. One can assume a rather stable and standardized browser that assumes the presence of a mouse, a keyboard, a screen color on a PC. The same assumption cannot be made about mobile devices. Besides, on a PC, connectivity is assumed ubiquitous. That isn’t the case with mobile devices. In a mobile environment connectivity may vary or even be absent in a short time frame. If the markup language is focused on content representation despite the device it’s being shown at, we may be facing a problem - the under usage of interface resources lowers the application’s usability. Also, if connectivity fluctuates along with bandwidth, one cannot count on permanently interacting with remotes machines thoroughly.

The compiled applications that run natively on a device are called ”fat clients” because they aggregate both logic and presentation layers. The instances of this model do not suffer of the inadequacy to the device they’ll run in, since they are developed and compiled for that particular device. This model suffers of rigidity: the code is hard to adapt to a different device in case of need.

In summary, we can see the ”thin client” as versatile but limited in the way it takes advantage of a device’s characteristics, and also communication intensive. On the other hand the ”fat client” that can take advantage of a deeper integration with the device but lacks the flexibility of the thin client’s approach to development and deployment.

2 An Applicational Push Mechanism

This work establishes a model that remains between purely compiled and purely online models; plus, the applications are delivered in an asynchronous way opposite to what we are used to with web based applications - the user is asked to accept an application that’s being ”sent” by the environment instead of explicitly entering an URL to retrieve it. Our vision encompasses an infrastructure responsible for the deployment of applications through push mechanisms [1]. It is the infrastructure resposibility to maintain some state of the applications sent to the user and share it with other similar infrastructures. This mechanism enables further interaction with the same user in a more appropriate way. When pushing the applications, the infrastructure must be aware of the final destination and choose the most appropriate device the user has. After being choosen, the application is deployed. At this point, the user has the ability to refuse a certain application. He/She can also define a set of interests which the infrastructure respects by not pushing the user with unwanted applications. On the user’s end, the applications run in a special container in the chosen mobile device which is responsible for managing pushes, managing application state and security. Together with the infrastructure, both can decide on whether an application is still valid or should be erased/put to sleep/be migrated, similar to what one can expect of a mobile code system [2].

In summary, our work aims to develop a software client to run on the mobile device with the responsibility to manage software pushes. It will incorporate a virtual machine responsible for running the pushed code. The development of an infrastructure responsible for the deployment of an application after a device has been detected is also tackled in this work. It is also our goal to develop an infrastructure responsible for the deployment of application after a device has been detected. Its role will also include the reception of data from the application on the device - whether to forward it to other hosts, or to interpret it, or to store it.

3 Framework Description

Our approach covers usage scenarios where the lack of connectivity and abundance of devices collides with the need for rich applications. The lack of connectivity can be implicit - determined by the variables of the environment surrounding the user or explicit - determined by the user (not to use network) or by the application. These restrictions, combined with the richness of applications, imply that the application should carry with

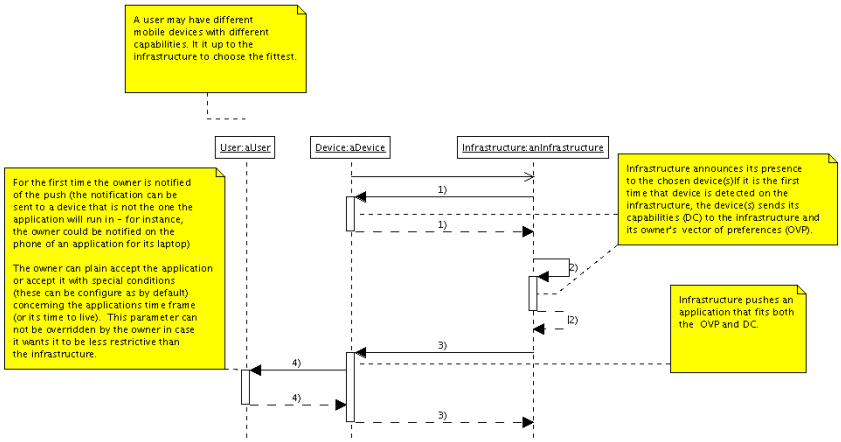

Fig. 1. Detection and deployment interaction model

it all the necessary logic for it to work ofline. These restrictions can map with ”on the move” scenarios as envisaged in [3]. A typical usage might be like: user gets a taxi, has a plain to catch and travels to a different country. While on the move, the user gets pushed with applications matching his/hers interests and uses them in order to book tickets, virtually explore regions, etc. The user can even be pushed with an application that acts as his/hers avatar towards other systems like answering e-mail or phone calls while the user is on the move. These applications can be offered by administrative structures, airports, private stores, police, banks, etc.

The interaction model that captures the needed functionalities is shown in figure 1:

1. Device(s) detected. A user may have different mobile devices with different capabilities. It is up to the infrastructure to decide the fittest.
2. Infrastructure announces its presence to the chosen device(s)
3. If it is the first time that device is detected on the infrastructure, the device(s) sends its capabilities (DC) to the infrastructure and its owner’s vector of preferences (OVP).
4. Infrastructure pushes an application that fits both the OVP and DC.
5. For the first time the owner is notified of the push (the notification can be sent to a device that is not the one the application will run in - for instance, the owner could be notified on the phone of an application for its laptop)
6. The owner can plain accept the application or accept it with special conditions (these can be configured as by default) concerning the applications time frame (or its time to live). This parameter can not be overridden by the owner in case it wants it to be less restrictive than the infrastructure.
7. In case the owner accepts the information, it is downloaded to the chosen device and he/she can start using it.

Also, upon usage there are some variables that are exchanged with the infrastructure - some concerning the application's state and others that are specific to the application.

Additionally, the following behavior is also incorporated in the model: a) the application validity is a function of time and location, e.g., if the user leaves the predefined area of usage or if a certain time elapses, the application gets invalidated and any local or infrastructural cache gets cleaned or the application is set to a sleep state and can be waken up if predefined circumstances are met (location variation; time elapsed). To achieve this, the framework will provide a component that will manage explicit and implicit invalidation when the user leaves a predefined area; b) the variables exchanged with the infrastructure can be shared with other infrastructures in different locations and can be used to refine a different application; c) an application can be ”augmented” within another infrastructure. Data migrates through infrastructures. When leaving a place, the application enters sleep mode and wakes in a new place. Getting to the new place triggers the push of a new appendix to the application based on information retrieved from the previous infrastructure. This feature can leverage its functionality/usability or any other characteristics.

4 Conclusion and Future Work

From the work developed to the moment we think that this approach proves to be adequate to a set of usage scenarios that one can group as ”on the move” scenarios. Considering all the variables such a scenario has - ranging from device variety (both in hardware and in software) to bandwidth restrictions - we believe that an ”in between” approach to the online model and compiled model is the best way to cope with it.

To further understand the concept, we are developing a client and infrastructure prototype. In the future this platform could evolve as an abstract push system (regardless of pushing applications or data). The infrastructure could interact as a broker system on behalf of the user and the application container on the client should be versatile enough to accommodate multiple applications that could aggregate with each other to result in richer and more complex applications.

Concerning security and privacy, issues arise given the fact that an user may be unaware of malicious infrastructures. This problem could be overcome with a certificate policy covering both infrastructures and clients.

References

1. Hauswirth, M.; Jazayeri, M; ”A Component and Communication Model for Push Systems”, ACM SIGSOFT Software Engineering Notes, Proceedings of the 7th European Software Engineering Conference, Volume 24 Issue 6, October 1999
2. Carzaniga, A.;Picco, G. P.; Vigna, G.; ”Designing Distributed Applications with Mobile Code Paradigms”, Proceedings of the 19th international conference on Software engineering, Boston, Massachusetts, United States, Pages: 22 - 32, 1997
3. ”Scenarios for Ambient Intelligence in 2010” ISTAG Final Report Compiled by K. Ducatel, M. Bogdanowicz, F. Scapolo, J. Leijten and J-C. Burgelman, February 2001