Just in time Entertainment Deployment on Mobile Platforms

Laurent Lefèvre
INRIA /LIP (UMR CNRS, INRIA, ENS, UCB)
Ecole Normale Supérieure de Lyon
laurent.lefevre@inria.fr

Jean-Marc Pierson
LIRIS (UMR CNRS, INSA)
Institut National Sciences Appliquées
Jean-Marc.Pierson@liris.cnrs.fr

Abstract

Managing the deployment of Games on mobile phones can be really complex. Developers have to create multiple versions of the applications, even if they write it in Java. There is thus a need for generic tools to help programmers in the task of building mobile games, that means to optimize their packaging and deployment on the end-users phones. Active and programmable networks allow deployment of dynamic new services for data transport. In this paper, we describe how to use an active network solution to perform dynamic packaging and deployment of applications on cellular mobiles and to face heterogeneity of platforms. [1] *We describe the overall architecture, discuss further possibilities of the platform as well as experimental results.*

Keywords: mobile game deployment, active network, mobile platforms, network emulation

1 Introduction

All modern mobile phones integrate now a Java Virtual Machine. These JVM allow providers to propose applications working on heterogeneous mobile phones (without having to redo some specific development and to adapt them individually for specific features). Most of the core of the application remains the same while only some small parts of the code have to be adapted to the specific features of mobile phones : the memory available, the exact version of the JVM, the layout of the components, the graphics for instance are such things the final application should take into account.

The Mobile Information Device Profile (MIDP) is specially designed for mobile phones. Applications written from the MIDP profile take account of the specific features of mobile phones. The specificity of the API provided by the mobile phone constructors limit the portability and force providers to create different versions of a same mobile application, for each model of mobile. In the games entertaining field, it is even more crucial, since games usually exploit the very limits of resources capabilities. In order to efficiently exploit devices features, providers propose for instance their own APIs dedicated to games which limit portability of applications.

Active Networks[9] allow service providers to inject customized dynamic services into the programmable network equipments. The creation of new services is an original way to think about development and deployment of customized modules to perform computation within the network.

In our project, we propose to benefit from active and programmable networks by deploying active nodes on data path to efficiently adapt streams on the fly. This research follows three main goals :

- to reduce development costs and the complexity for managing a version of a game for each mobile class. The active node will construct the application and adapt the resource files on the fly;
- to reduce the usage of bandwidth and interactions between clients and applications server, by providing a cache storage facility on the data path;
- to efficiently support deployment of games without adding too much latency on real networks.

In this paper, we present the architecture of an active service deployed inside the Tamanoir Execution Environment. We will focus here on the dynamic creation of the application. This service constructs and deploys on the fly games from the different class files and graphic resources in order to adapt them to target mobile phones.

This paper is organized as follows. Section 2 reminds the reader with the standard deployment of applications on mobile phones. Section 3 focuses on supporting on the fly packaging inside the programmable network equipments. Section 4 presents our first experiments on a local platform. We discuss some deployment aspects in real frameworks in

[1] This work is supported by Funds of Region Rhone Alpes on the collaboration between 3DDL (3 Degres De Liberte) company, LIRIS Laboratory and INRIA RESO team.

section 5, as well as other features being deployed on the platform.

2 Deployment of java applications on mobile platforms

The development of one single game in a mobile environment is not comfortable mainly due to heterogeneous terminal equipment. A mobile infrastructure depends on a multitude of features. To solve this problem various solutions have been considered by the main companies (Motorola, Ericsson and Nokia). Currently, the commonly used solution is the J2ME pack (Java 2 Micro Edition[1]). J2ME allows the downloading of applications in the portable phone.

2.1 Mobile Information Device Profile

Large number of companies of mobile phones associated to develop this standard, that allows to use the Java technologies on the mobile phones. Applications written from the MIDP profile name themselves *MIDlets[3]*. These are similar by structure to the *applets* or the *servlets*.

The Over The Air provisioning method (OTA [2]) is a part of the standard MIDP. It allows to recognize, install, actualize and eliminate some MIDlets on the mobile.

OTA provisioning works as follows : first a mobile phone sends a WAP request for a JAD (Java Application Description) file. The request is sent to a Web server through a WAP gateway. The Web server sends back JAD package to the mobile phone. Then the phone fetches the JAR (Java Archive) package defined in the JAD file from the Web server. The phone Java Application Manager (JAM) installs the package. After installation, the phone may send an optional installation notification to the server.

2.2 Deployed files

Every JAR file includes a *MANIFEST file* that provides information on classes it contains. Manifests support nine attributes, among there exists one entry for every MIDP application of the continuation, containing the name of the application, an optional icon and the name of the class that must be started to execute the MIDlet).

A JAD file provides textual information on JAR files, like the location (URL) and size of a JAR file. It can include also specific information concerning the deployment of games. For example, in some Nokia phones Java Application Manager can install a game MIDlet into Games folder instead of the standard Application folder. This is possible if the following Nokia specific setting is defined in the JAD file:
Nokia-MIDlet-Category: Game

Other information can be added in the JAD file, in order to personalize the game itself. For instance, client's personal data being collected during his registration (phone number, date of birth,...) can be added in this JAD file.

2.3 Static deployment and activation of a mobile application on a mobile platform

Two kinds of distant deployment are supported : (1) The user connects to the site of the application server through its mobile phone; (2) The user subscribes to a download operation. He receives a SMS containing instructions for downloading and installing the application.

The user must verify that the mobile phone on which he wishes to download the application is compatible with this service (Fig. 1). The URL indicated in the SMS corresponds to the access path for the JAD file of the game requested, in the following format *http://ApplicationServer/User/MobileType/ ApplicationName.JAD*. Once this JAD downloaded, the mobile phone downloads the JAR file, with the URL provided by the field *MIDlet - JAR - URL* of the JAD file.

As one can figure out, the location of the JAD file is different from one user/mobile pair, since it reflects some personalization, as explained before.

Figure 1 presents the needed operations to allow a downloading of an application on a mobile client :

- 1a: Enrollment, 1b: Downloading application
- 2: Creation of the temporary directory (user/mobile/application) and copy/adaptation of the JAD and JAR files of requested game
- 3: Send SMS via mobile operator
- 4,8,12 : Request for WML then JAD, then JAR files
- 5,9,13 : Request for WML, then JAD, then JAR files
- 6,10,14 : WML, JAD, and JAR files are transmitted to the operator gateway
- 7,11,15 : WML, JAD, and JAR files are forwarded to the client

3 Just in time deployment and packaging of mobile games

This work is done jointly with a mobile applications (and especially advertisement games) provider company *3DDL* which personalizes its applications for each customer (logo integration, adaptation to mobile features...).

The specificity of the API provided by cellulars designers limits the portability, and constrains providers to build and manage various versions of the same application for different mobile platforms. Indeed, the providers usually

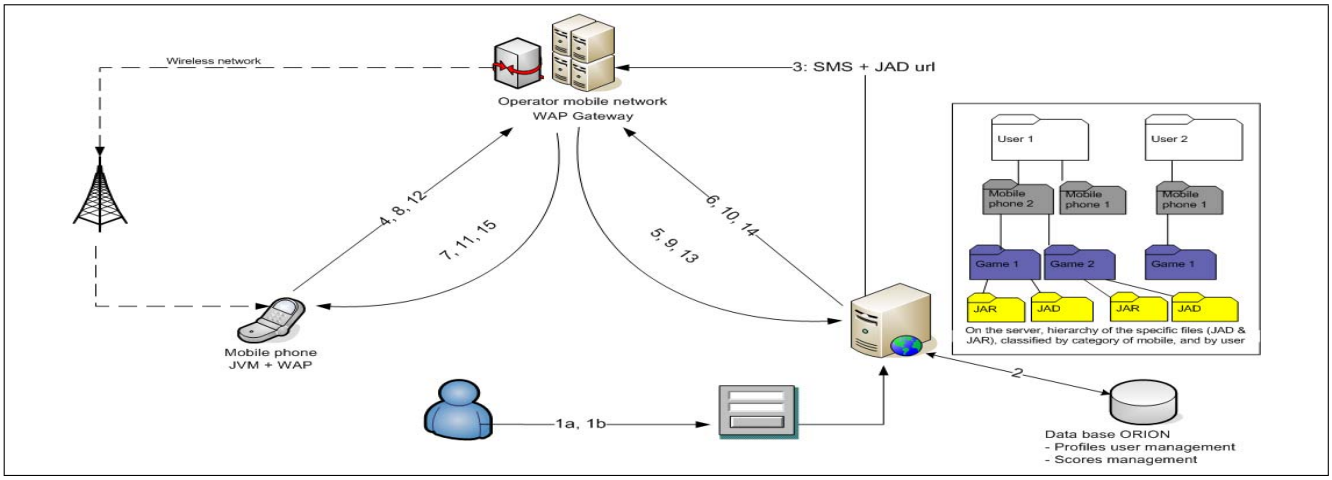

Figure 1. Application deployment without active network support

add some user specific information in the JAD file and store these information in a data base. Moreover, when a user downloads an application, a directory *User/TypeMobile* is created and the JAD and JAR files are copied there. This is an easy and expensive (in terms of storage, management, consistency) way to guarantee the relation between the user, its mobile phone and the version of requested application.

Once the application installed on the mobile, its management is very difficult. Each modification causes new transfers of files (updating application, modifying resource files like logo/advertising,...).

To avoid these consuming operations, we have previously proposed in [8] to process JAD files of the fly inside the network between applications server and terminal clients.

We go here further in this direction. We adapt at the latest possible time the game itself. Thus, we construct on the fly the JAR file that will be sent to the mobile phone. The benefit of this is :

- since the JAR file is constructed on demand, the provider does not have to manage the different packages (that means to store and construct every possible arrangement of class files);
- the class files can be cached along the way to the client and thus can be re-used for other clients;
- when update is needed, only new modified class files and resources are to be retrieved from the provider.

The mandatory parameters of the configuration of the application are not known in advance by developers and are provided at the time of the download operation of the application by the user.

An active node located on the stream path between the mobile terminal client and the application server is able to play the role of content adapter and application packager. The proposed approach is to reuse the same application software for all application deployment steps and for many heterogeneous mobile platforms, without to redesign programs. The stream adaptation can be applied on various parts of the data (Java classes contained in JAR archive, personalized JAD file, resources files : logos, sounds, announcements, animations). We developed an active service deployed on a Tamanoir active node and localized between the servers and the mobile phone of the customer.

Tamanoir Active Node

The RESO team at INRIA developed the TAMANOIR [6] execution environment [2], which allows to open out new services on the network.

We have designed an architecture for a high performance active router capable of being deployed around a high performance backbone: the Tamanoir Execution Environment [6, 7]. Our approach comprises the strategic deployment of active network functionality around a backbone in access layer networks and the provision of a high performance dedicated architecture.

We define an Active Network Execution Environment (EE) as an environment able to *load* and deploy network services. It must be also able to *direct* packets towards the required service thanks to appropriate header filtering.

Active services must be deployed at various levels depending on resources (e.g. processing capabilities, memory consumption and storage capacity) and intelligence (flexibility of the execution environment) they need. In order to provide an adapted EE for each type of service and to limit packets ascent, we design an active node architecture

[2] Tamanoir homepage: http://www.ens-lyon.fr/LIP/RESO/Tamanoir

on four levels: Network Interface card (NIC), Kernel space, User space and distributed resources (see Figure 2).

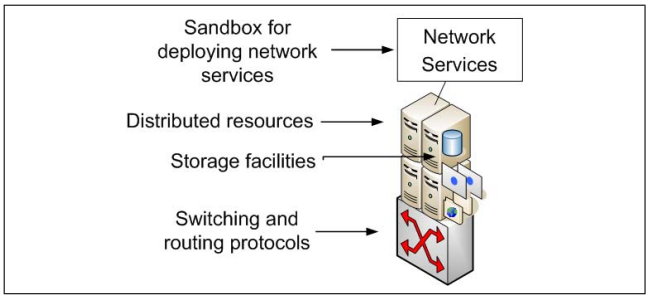

Figure 2. Tamanoir active node

A TAMANOIR node is composed of two main components, TAMANOIRd and ANM (Active Node Manager). The first component TAMANOIRd, is a daemon who turns on a active node TAN (Tamanoir Active Node) and that acts like a programmable active router. The TAN node receives and sends packets while processing tagged data packets with personalized services. Indeed the TAMANOIRd daemon redirects packets assets received toward the adequate service that is launched under shape of one light process (thread). The resulting packet of this treatment is sent to the next TAN node or to the final receiver.

The ANM (Active Node Manager) manages localization and deployment of new active services on the node.

Active network support

By deploying a Tamanoir Active Node near the terminal clients, we propose an enhanced global architecture : The active node is deployed on the mobile operator site; it handles requests from mobile nodes and packages applications to be downloaded. Application server and database remains in the applications providers control.

Figure 3 presents the needed operations to download an application on a mobile client within this infrastructure :

- 1: Register, provide registration profile, request specific game
- 2a, 2b : Send SMS via mobile operator + URL of the JAD file on Tamanoir
- 3, 8: Request for WML and JAD file
- 4: Extraction of the user_agent + identifying user from the URL
- 5: Request for file "Standard JAD" + Sending of user_agent,User_ID, Application_ID to the Tamanoir servlet
- 6: Send standard JAD file, the list of Java class and the resources constituting the application to Tamanoir node
- 7 : Check in the Tamanoir cache if the JAR is already built, otherwise which Java classes are absent/present
- 8: Request Java classes not already present in the cache. Put them in the cache
- 9 : Verify integrity of Java classes
- 10: Create on the Tamanoir node the final game JAR file : compress Java classes, create manifest, put them together
- 11 : Adaptation of the JAD content according to user ID and mobile type (containing size of JAR file)
- 12 : Install game

4 Experimental validation

Active nodes have the capabilities to process packets in addition of the usual routing functionalities. In our fist experiments, we evaluate the benefits of an active node to support the deployment of applications and some scalability issues. To validate this device several measures and tests have been conducted with a network emulator tool (NistNet [5]) which allows to emulate throughput, latency and packet loss on a link.

For classical mobile java applications, JAD files and JAR are on average, respectively 0.5KB and 45KB. The time of downloading a JAR file of 45Kb, on a GSM (9,6 Kb/s) network is usually about 1 minute.

4.1 Experimental platform

In order to experiment and validate our active network support, we deploy it on a local platform with network emulation environment (NistNet). This platform allows us to perfectly manage network features (bandwidth, latency, packet loss...).

A specific node emulates a large number of mobile clients. This node forwards client requests to a Tamanoir node, which creates dynamically the games by connecting to the company application server and downloading the resources files (class, graphics, music files...), if not in the cache of Tamanoir.

4.2 Adapted data streams

The first task was to verify the quality of the JAR files being created by the active nodes. Results were convincing, with a compression ratio for the generated archives similar in size with those created by the "jar" command line tool. On a classical mobile application, for instance, we were even able to spare few hundred bytes relatively to the original archive file.

This factor is important, since the delay to transmit the information must not be too long, thus the JAR file generated must be kept small. Since a normal application is about 45 KB, and the transmission time slow on GSM, sparing few hundred bytes is valuable, both for the client who pays

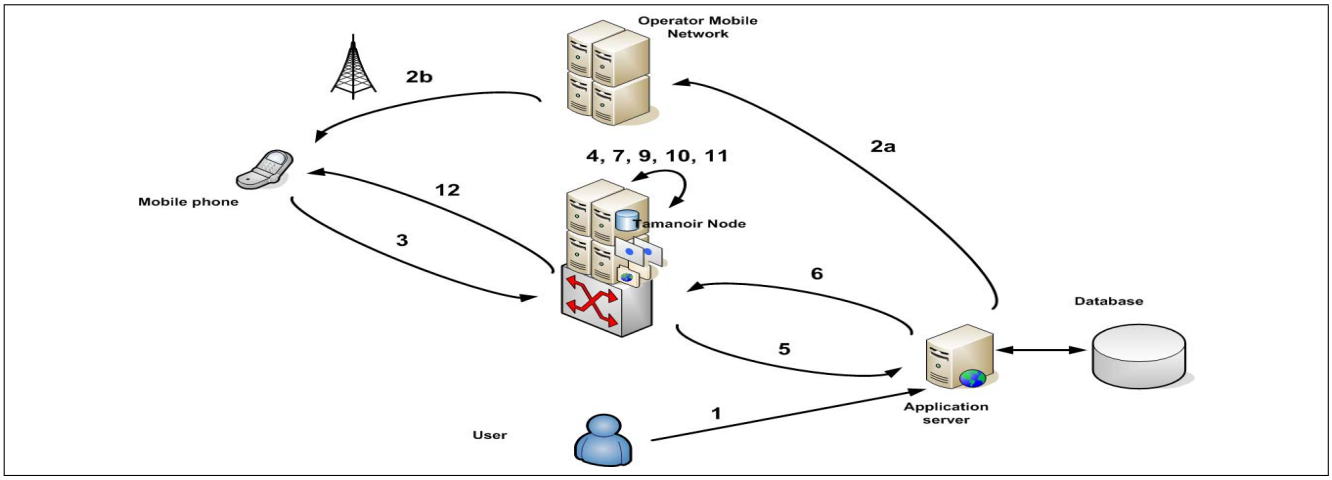

Figure 3. Logical operations during application deployment with active network support

less, but also for the scalability of Tamanoir, when hundreds of clients share the connection.

4.3 Experiments with network emulation

With NistNet, we emulate a GSM network by forcing the throughput to 10Kb/s between client machines (*C1 and C2*) and the Tamanoir Active Node (*T1*). We also reduce the bandwidth to 100Kb/s between *T1* and the application server (*A1*). By default the bandwidth emulated by the NistNet node (*N1*) is shared between all connections. For emulating a dedicated throughput between the mobile clients and application server, we generate connections with Tamanoir on different ports.

We wanted to exhibit the time devoted to the creation of the Jar files in the whole process. The total time includes the time to process the WML/HTML (negligible), the time to create the JAR and the time to create the JAD. The results show that most of the time is spent in the processing of the JAD file, not surprisingly (Fig. 4). Indeed, this file needs the size of the JAR file associated. This information can only be obtained after the JAR file is created, thus when all the java classes have been downloaded from the server and the compression finished. We can also see the impact of the cache management since the JAR processing tends to decrease along the time (all the files are in the cache, thus the time to process them decreases). These experiments tend to show that the Tamanoir platform performs well when a large community of mobile phones users want to download simultaneously the application files (with the limit that the actual configuration of the Tamanoir node can not support more than 1200 simultaneous requests, due to memory leak).

We have also measured the time spent in Tamanoir for the download of a file, thus the impact of Tamanoir in the process : This time is very small, never more than 3% of the total time for downloading a file over a GSM connection.

5 Discussion and current trends

The prototype presented here on mobile games is under test at the company side for mobile games deployment, as explained in this paper. For the moment, we are collecting usage traces to better understand the behavior of users and waiting times in a real world. We want to follow experimental validations on our platform by emulating next generation wireless networks (GPRS, UMTS) in order to evaluate the deployment impact on these networks.

The development of one application using this platform is being reduced, typically down to ten days, given the extreme easiness and portability of the platform. Usually, much time is spent in the actual deployment of the applications on the mobile phones, which is facilitated using our framework.

When new versions of an application are distributed, the Tamanoir node must not use any more the cached files. Thus, we have a management facility for the application provider so that they can upload directly the modified java classes or resources to Tamanoir, in a proactive way. This process is mandatory for the system to be consistent over the time. This can be easily automated when the class files have been approved by the application designer (some tests have to be extensively done at the provider side before the application being deployed).

Another concern is the size of the cache. The experiments have be done with an ideal large cache size (never filled). In fact, given the small size of an application, it is feasible to have quite a lot of applications ready to be constructed in the Tamanoir cache. Otherwise, we would have to select some class files to be deleted, starting for those

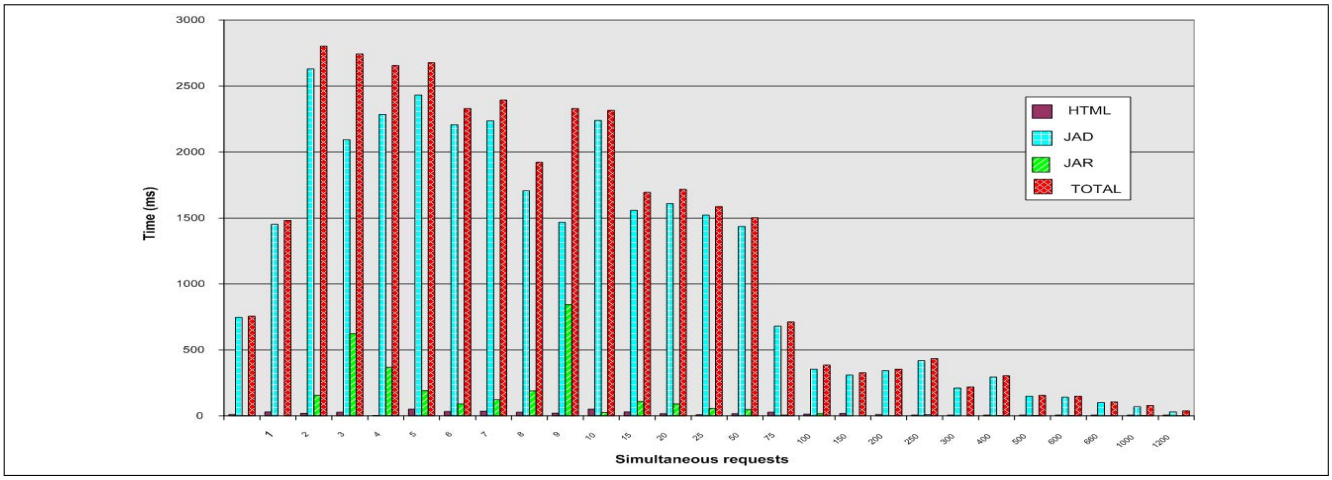

Figure 4. Processing times of WML/HTML, JAR and JAD

least accessed for instance, or any other cache policy strategies.

The Tamanoir platform is ideally suited to perform data adaptation on the way from the server to the client. We are currently working on a video adaptation strategy using Tamanoir. The overall idea is to perform the downloading of a generic video from the server, and to adapt it to the mobile phone characteristics on the Tamanoir node. The file is transcoded from MPEG2 to 3GP file formats, which is a common format understood by many available mobiles. Here the possible adaptations concern basic features like the size of the video or the number of colors, ... More sophisticated adaptations will be added in the future like for instance the language of the movie or of the subtitles, the length of the video, ... We have already some results in this direction [4] within a classical proxy approach.

6 Conclusion

This paper presents our ongoing work on operational support of java application on mobile platforms through active networks. Three goals were followed in doing this approach: to reduce application development time, to reduce required bandwidth between applications server and clients and to improve download performance by putting an active equipment closed to client terminals. We believe we reached these goals using the Tamanoir nodes in the core of the network. We have proposed an experimental platform based on network emulator and streams generation tools to emulate the whole architecture. Through the use of the Tamanoir execution environment, we can propose a scalable solution.

References

[1] Java 2 platform, micro edition (j2me). http://java.sun.com/j2me/.

[2] Ota. Over The Air User Initiated Provisioning: Recommended Practice for the Mobile Information Device Profile, Version 1.0, May 2001.

[3] Midlets. Settings for OTA Download of MIDlets Version 1.0, Document, http://www.forum.nokia.com, Sept. 2002.

[4] G. Berhe, L. Brunie, and J.-M. Pierson. Content adaptation in distributed multimedia systems. *Journal of Digital Information Management, special issue on Distributed Data Management*, 3(2), June 2005.

[5] M. Carson and D. Santay. Nist net: a linux-based network emulation tool. *SIGCOMM Computer Communication Review*, 33(3):111–126, July 2003.

[6] J.-P. Gelas, S. El Hadri, and L. Lefèvre. Towards the design of an high performance active node. *Parallel Processing Letters journal*, 13(2), June 2003.

[7] L. Lefèvre. Heavy and lightweight dynamic network services : challenges and experiments for designing intelligent solutions in evolvable next generation networks. In I. Society, editor, *Workshop on Autonomic Communication for Evolvable Next Generation Networks - The 7th International Symposium on Autonomous Decentralized Systems*, pages 738–743, Chengdu, Jiuzhaigou, China, Apr. 2005.

[8] L. Lefèvre and A. Saroukou. Active network support for deployment of java-based games on mobile platforms. In I. C. Society, editor, *The First International Conference on Distributed Frameworks for Multimedia Applications (DFMA'2005)*, pages 88–95, Besancon, France, Feb. 2005.

[9] D. Tennenhouse and D. Wetherall. Towards an active network architecture. *Computer Communications Review*, 26(2):5–18, April 1996.