Active network support for deployment of Java-based games on mobile platforms

Laurent Lefèvre, Awèni Saroukou
INRIA /LIP (UMR CNRS, INRIA, ENS, UCB)
Ecole Normale Supérieure de Lyon
46 allée d'Italie, 69364 Lyon Cedex 07, France
`laurent.lefevre@inria.fr`

Abstract

Active Networks are a promising way to develop new services for data transport. Active and programmable nodes can process "on the fly" streams of data to propose high level services. In this paper, we describe how to use an active network solution to perform deployment of java games on cellular mobiles and to face heterogeneity of platforms. We present the architecture of a transcoder service inside the Tamanoir Execution Environment.[1] *We describe the overall architecture of an active transcoding service (ActiveWapS) of JAD files and present first experimental results.*

Keywords: active network, mobile platforms, network emulation, programmable network

1 Introduction

Active Networks[1] allow user or applications to inject customized programs into the network nodes. The creation of new services is an original way to think about development and deployment of customized modules to perform computation within the network. This can lead to massive improvement of network functionalities.

New mobile phone generations integrate more and more a Java Virtual Machine. This JVM allows providers to propose applications and games working on

[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.

heterogeneous phones (without having to redo some specific development and to adapt them individually for specific features).

The Mobile Information Device Profile (MIDP) is specially designed for mobile phones. Applications written from the MIDP profile are called the MIDlets[2] and are similar with *applets* or *servlets*. These applications take account of the specific features of mobile phones and contain Java archive (JAR) and Java Application Descriptor (JAD).

The specificity of the API provided by the mobile phone constructors, limit the portability and force providers to create different versions of a same game, for each model of mobile. In order to efficiently exploit devices features, providers propose their own APIs dedicated to games which limit portability of applications. At the time of the downloading of a game by an user, providers add some specific information to the user in the JAD file. These information collected at the time of the player's registration, are stored on the data base.

In this paper, 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 adapt the files on the fly.;
- to reduce the usage of bandwidth and interactions between clients and games server;
- to efficiently support deployment of games without adding too much latency on real networks.

In this paper, we present the architecture of an active transcoding service (ActiveWapS) deployed inside the Tamanoir Execution Environment. This service transforms on the fly, parts of the games (JAD files) in order to adapt them to target mobile phones. We also present first experiments on a local platform with emulated wireless network.

This paper is organized as follows: Section 2 describes the standard deployment of games on mobile phones. Section 3 focuses on the architecture of an active transcoder of JAD streams in the active node. Section 4 presents our first experiments on a local platform. Section 5 presents conclusions and an outlook on future work.

2 Deployment of java games on mobile platforms

The development of an application 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 Microphone Edition[4]).

2.1 Java 2 Micro Edition

Java 2 Micro Edition (J2ME) is a Java platform for the embedded devices like cellulars and PDAs. Contrary to J2EE, J2SE and Java Card, J2ME is composed of a set of API standards defined by a group of experts including equipment providers, software developers and service suppliers.

The J2ME platform provides the power and advantages of the Java technology for mobile platforms. This one includes one supple user-interface, a robust security model, a palette of protocols networks integrated and a management of the local applications and network. This infrastructure allows the downloading of applications in the portable phone.

2.2 MIDP - 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 the *MIDlets[2]*. It is some similar applications by their structure to the *applets* or the *servlets*. These applications must take account of the configuration restricted of the mobile devices.

The Over The Air provisioning method (OTA [3]) is a part of the standard MIDP. It allows to recognize, to install, to actualize and to eliminate some MIDlets on the mobile.

OTA provisioning works as follows (fig. 1): 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 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.

2.3 Content data streams : JAD and JAR files

Every JAR file includes a *MANIFEST file* that provides information on classes that it contains. Manifests support nine attributes of which six must be defined for every MIDP application. These attributes are illustrated on table 1.

A JAD file provides textual information on JAR files (table 2).

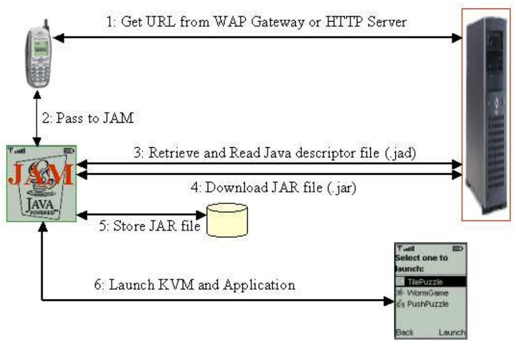
1: Get URL from WAP Gateway or HTTP Server
2: Pass to JAM
3: Retrieve and Read Java descriptor file (.jad)
4: Download JAR file (.jar)
5: Store JAR file
6: Launch KVM and Application
JAM
JAVA
Select one to launch:
TilePuzzle
WormGame
PushPuzzle
Back
Launch

Figure 1: Games deployment with OTA

MIDlet-Name	Name of the continuation
MIDlet - Version	Number of version of the continuation
MIDlet-Vendor	Developer of the continuation
MIDlet-x	An 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
MicroEdition - Profile	profiles It thereafter requisite
MicroEdition - Configuration	The requisite configuration thereafter

Table 1: Parameters for JAR (manifests)

3 How to dynamically support Java games deployment ?

Game providers personalize their applications for each of their customers (logos integation, adaptation to mobile features...).

The specificity of the API provided by cellulars designers, limit the portability and constrained providers various versions of the same game for different mobile platforms. During the download of a game by an user, the, providers add some specific information to the user in the JAD file and store these information in a data base.

The JAD files and JAR that constitute the MIDlets suites, are therefore specific to each game and each type of mobile. They are generated by the game server. When a user downloads a game, a directory *User/TypeMobile* is created and the JAD and JAR files are copied inside. This is the easiest way to guarantee the relation between the user, its mobile phone and the version of requested game.

The JAD file mainly describes the content of the JAR file, but one can also add

Information	Explanation
MIDlet-Name:	Example Name of the MIDlet suite
MIDlet-Vendor:	Provider of the MIDlet
MIDlet-Version:	1.0 Version number
MIDlet-Jar-Size:	25798 Size of the MIDlet
MIDlet-Jar-URL:	http://domain/directory/Example.jar Location of the JAR file
MIDlet-1: Example, , Example	Name of the MIDlet, Icon, main class

Table 2: JAD Descriptor Structure

there information, that will be able to to be called by a program, once the game opened out on the mobile phone. For each client the JAD file is thus, specific. It contains the customer's personal data that has been collected during his registration (phone number, date, of birth,...),.

Once the game is installed on the mobile, its management is very difficult. Each modification causes new transfers of files (updating application, modifying or addition announces advertising,...) which can become expensive in bandwidth.

To avoid these static time and bandwidth consuming adaptations, we propose to process JAD files of the fly inside the network between games server and terminal clients. This approach will be designed through the implement of a JAD file content transcoder active service.

3.1 Static deployment and activation of a game on a mobile platform

We support 2 kinds of deployment :

1. The user connects to the site of the game server through its mobile phone;
2. The user subscribes to a game operation. He receives a SMS integrating the option of downloading and installing the game.

The user should verify that the mobile phone on which he wishes to download the game is compatible with this service (Fig. 2). The URL indicated by the SMS corresponds in path access to the JAD file of the game asked, and is the format *http://GameSserver /User /MobileType /GameName.JAD*. Once the JAD is downloaded, the mobile phone downloads the JAR file, with the URL provided by the field *MIDlet - JAR - URL* of the JAD file (See tab. 2).

Operations (figure 2)

- 1a: Enrollment

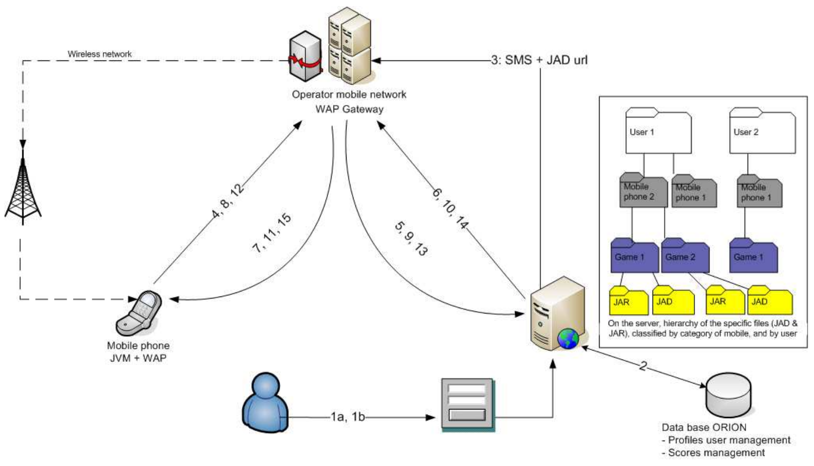

Figure 2: Game deployment without active network support

- 1b: Downloading game
- 2: Creation of the temporary repertories (user/mobile/game) and copy of the JAD and JAR files of requested game
- 3: Send SMS via mobile operator
- 4,8,12 : Request for WML, JAD, JAR files
- 5,9,13 : Request for WML, JAD, JAR files
- 6,10,14 : WML, JAD, and JAR files
- 7,11,15 : WML, JAD, and JAR files

3.2 Dynamic deployment of a game on a mobile platform with on the fly content transcoding

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 game by the user. An active node located on the stream path between the mobile terminal client and the game server will be able to play the role of content transcoder.

The proposed approach is to be able to reuse the same application software for all game deployment step and for main kinds of heterogeneous mobile platforms, without having to redesign programs. The stream adaptation can be applied on various parts of the data (personalized JAD file, logos, sounds, announcements, animations). It is necessary to allow a real identification of the user.

We develop an active transcoding service deployed on a Tamanoir node and localized between the games server and the customer with his mobile phone (fig. 5).

3.2.1 Tamanoir Active Node

The RESO team of the INRIA developed the TAMANOIR [8] execution environment [2], which allows to open out new services on the network.

A TAMANOIR node is composed of two main components (fig. 3), 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 send packets while processing tagged data packets with personnalized 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.

Two strategies of deployment are available (figure 4) :

1 - Service Broker: is a centralized solution, in which a Web server is dedicated to the distribution of active services. Services are localized by an URL.

2 - Deployment from node to node: the service is downloaded from active node to active node in parallel with stream data path.

3.2.2 Transcoding service

By deploying a Tamanoir Active Node near the terminal clients, we can efficiently support game deployment (fig. 5).

Operations (figure 5)

- 1a: Registration, 1b: Downloading game, 1c:Registration profile
- 2: Send SMS via operator mobile + 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

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

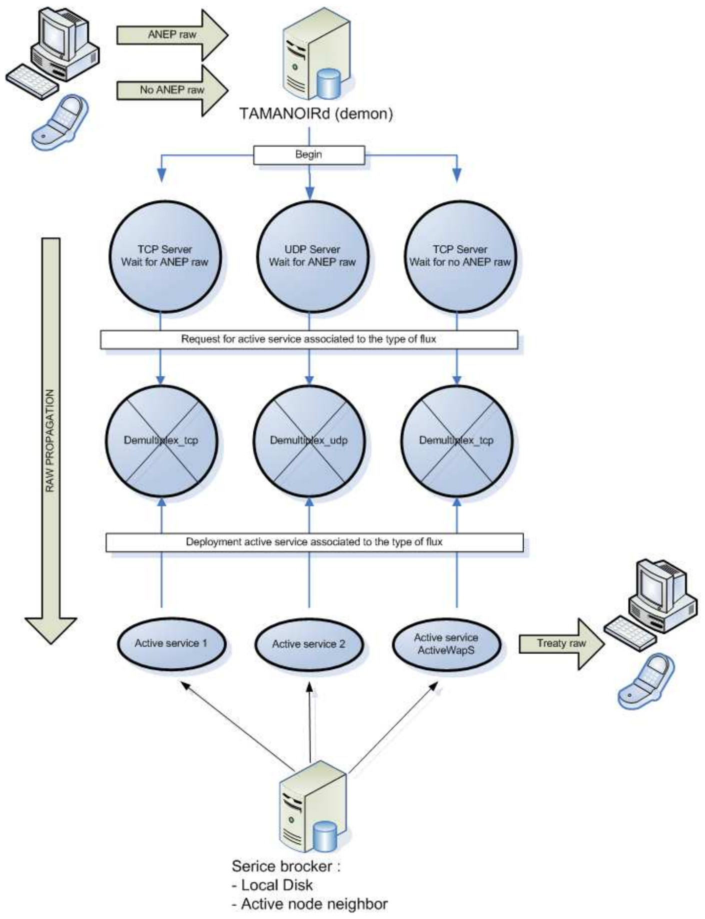
ANEP raw
No ANEP raw
TAMANOIRd (demon)
Begin
TCP Server Wait for ANEP raw
UDP Server Wait for ANEP raw
TCP Server Wait for no ANEP raw
Request for active service associated to the type of flux
RAW PROPAGATION
Demultiplex_tcp
Demultiplex_udp
Demultiplex_tcp
Deployment active service associated to the type of flux
Active service 1
Active service 2
Active service ActiveWapS
Treaty raw
Serice brocker :
- Local Disk
- Active node neighbor

Figure 3: Architecture of the TAMANOIR active node

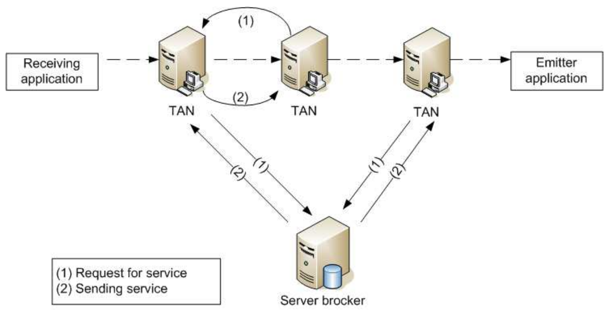

Figure 4: Active service deployment.

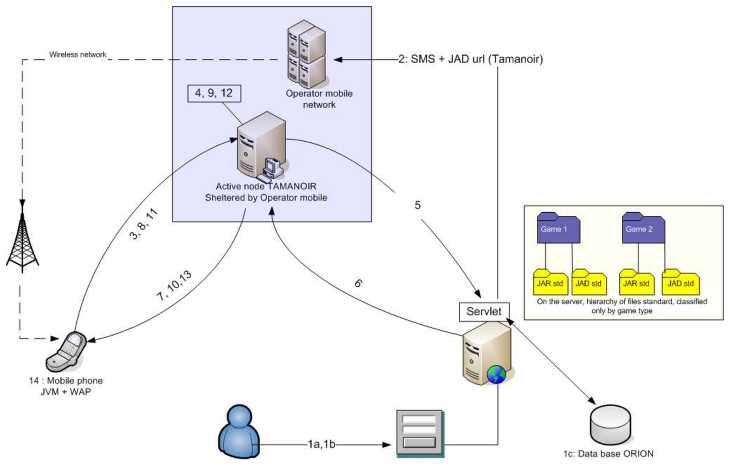
Wireless network
2: SMS + JAD url (Tamanoir)
4, 9, 12
Operator mobile network
Active node TAMANOIR Sheltered by Operator mobile
5
3, 8, 11
7, 10,13
6
Game 1
Game 2
JAR std
JAD std
JAR std
JAD std
On the server, hierarchy of files standard, classified only by game type
Servlet
14 : Mobile phone JVM + WAP
1a,1b
1c: Data base ORION

Figure 5: Game deployment with active network support

- 5: Request for file Standard JAD + Sending of user_agent,User_ID, Game_ID to the servlet
- 6: Sending of JAD and JAR files (standard)
- 9,10: Adaptation of the JAD content switch user ID and mobile type + Sending
- 11: Verification JAD information, if OK request for JAR file
- 12: Adaptation of JAR file
- 13: Sending file standard JAR
- 14: Verification + game installing

Our network architecture is composed of an ORION data base server in order to open out easily various ways of applications (Web, EJB, JSPs, JAR, servlets...).

The active service of JAD transcoding (called ActiveWapS), via the method, *getServletInfo ()*, requests to ORION data base the deployment of a servlet[9] ServletInfo.jsp " (See fig. 6).

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

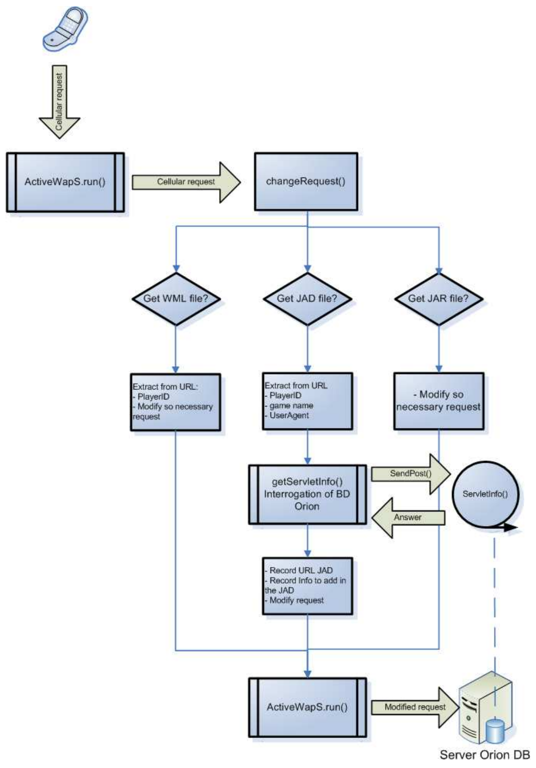

Figure 6: ActiveWapS service architecture.

node to support the deployment of games and some scalability issues. To validate this device several measures and tests have been conducted with a network emulator tool (NistNet [10]) which allows to emulate throughput, latency and packet loss on a link.

For classical mobile java games, 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 around 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…).

Figure 7 presents the experimental platform composed of 6 machines linked with a GigaEthernet network (bi-processor Intel Pentium III-S 1400 MHz,892 MB of RAM, 2 Giga Ethernet cards).

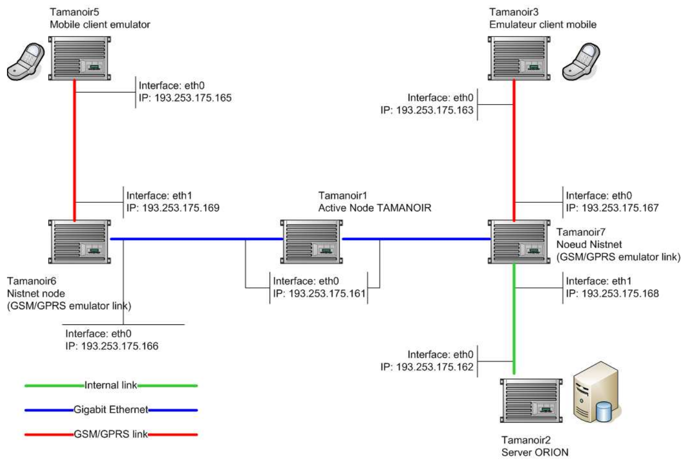

Figure 7: Architecture of the experimental platform

Involved machines have different roles. Machines 3 and 5 emulate large number of mobile applications and clients. Nodes 6 and 7 embed the NistNet emulator tool for emulating wireless networks. Node 7 also emulates distant connections to application server. One node(1) is dedicated to embed a Tamanoir active node and one node (2) runs the ORION application server.

4.2 Active node performances

First experiments concern the behavior of TAMANOIR when it processes a large number of simultaneous connections and to validate times of execution of the service of JAD stream adaptation.

For each mobile clients connection 3 threads are created: for the WML file, JAD, and the JAR file. In this experiment, we focus on JAD processing through active services (database access via the servlet and JAD adaptation on the fly). Tamanoir requires 130ms on average to process a JAD file. One notes that it can execute a maximum of 474 threads simultaneously. With more than 200 threads, Tamanoir machine becomes overloaded but continues to support new requests.

4.3 Experiments without network emulation

We have deployed our local platform to have 100Kb/s between tamanoir1 (active node), tamanoir7 (router) and tamanoir3 (old client version) . We have 1Gb/s between all others nodes. In the static (old) version, mobile clients send their requests directly to the ORION server and files are directly downloaded from

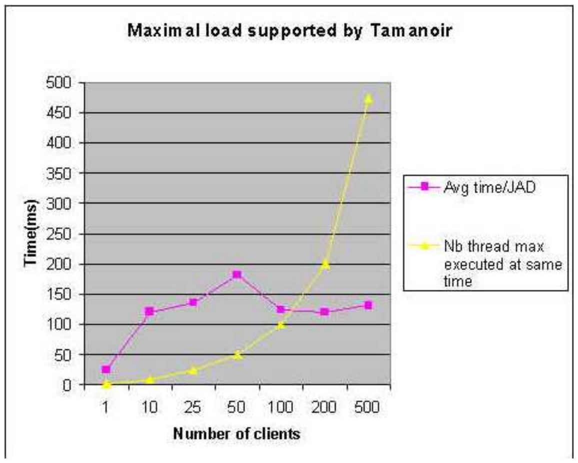

Figure 8: Maximal load supported by TAMANOIR.

Player/typeMobile/game repository.

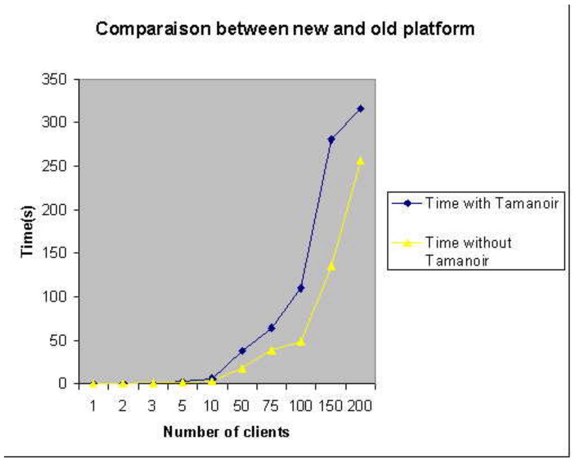

Figure 9: Comparison performance between static deployment and active network supported deployment

We can observe on figure 9 the latency added by the use of the Tamanoir Active Node. While the active transcoding in the network will allow to quickly create new games, the latency added for the deployment remains small for mobile clients. This experiment is too optimistic in terms of wireless client networks (shared network of 100 Mbits).

4.4 Experiments with Nistnet

With Nistnet, we emulate a GSM network by forcing the throughput to 10Kb/s (between tamanoir5 (client new version) and tamanoir6 (router)) and the internal network by applying a throughput of 100Kb/s (between tamanoir7 (router) and tamanoir2 (application server)). We keep 1Gb/s between all others links.

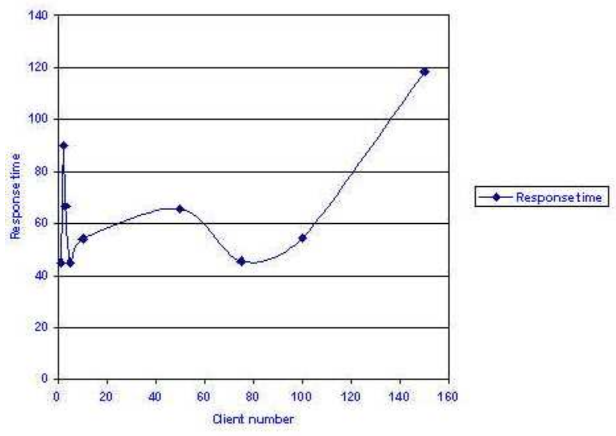

Figure 10: Tamanoir response time (seconds)

By default the bandwidth emulated by NistNet is shared between all connections. Then for emulating a dedicated throughput between the mobile clients and game server, we generate connections with Tamanoir on different ports. Thus we observe that on a emulated platform (a GSM network), active network support allows an intelligent usage of network resources and can transcode on the fly the data streams without noticeable latency (figure 10). The network is not shared, and therefore, the time of answer is on average 45s and remains constant with 100 simultaneous clients.

5 Conclusion and Future Work

This paper presents our first step for an operational support of java games on cellular through active networks. Three goals were followed in doing this approach: to reduce game development time, to reduce required bandwidth between games server and clients and to improve download performance by putting an active equipment closed to client terminals.

We have proposed an experimental platform based on network emulator and streams generation tools to emulate whole architecture. Through the use of the

Tamanoir execution environment, we can propose a scalable solution.

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. Additional experimental validations will be included in final version of the paper.

References

[1] David Tennenhouse and Wetherall, Towards an active network architecture *Computer communications Review*,26(2):5-18, April 1996

[2] *Settings for OTA Download of MIDlets* Version 1.0, Document published at Forum Nokia: http://www.forum.nokia.com /main.html, 27 Sept 02.

[3] *Over The Air User Initiated Provisioning: Recommended Practice for the Mobile Information Device Profile*, http: //jcp.org /jsr /detail /37.jsp. , Version 1.0, May 7, 2001.
Over-the-Air Provisioning with the J2ME Wireless Toolkit: http: //developers.sun.com /techtopics /mobility /midp /ttips /wtkota/
Introduction to OTA Application Provisioning: http: //developers.sun.com /techtopics /mobility /midp /articles /ota

[4] Mesud HASANOVIC, Lucas PRETRE, *MOBILE SERVICES AND MIDLETS* (Infrastructure de services sur terminaux mobiles), http:// www.lpretre.com/ travaux/ midlets/ mobserv_midlets.pdf, F'evrier 2003.

[5] *Deploying Wireless Java Applications:* http:// developers.sun.com/ techtopics/ mobility/ midp/ articles/ deploy/
Nokia Developer's Suite for J2ME, Version 2.0 for Linux: http://www.forum.nokia.com

[6] TAMANOIR *project home page*, available on http: //www.ens-lyon.fr /LIP/RESO/Tamanoir/

[7] JP. Gelas, L. Lefèvre, *TAMANOIR: A High Performance Active Network Framework, Active Middleware Services (AMS)*, Kluwer Academic Publishers, ISBN 0-7923-7973-X, August 2000.

[8] JP. Gelas, S. El Hadri and L. Lefèvre, *Parallel Processing Letters: Towards the Design of a High Performance Active Node*, World Scientific , Volume 13-Number 2-June 2003

[9] *RBL-JavaSide.com:* http://www.javaside.com/ - Java[tm] Servlet Technology - http://java.sun.com/products/servlet/archive.html

[10] National Institute of Standards and Technology (NISTNET), *Home page:* http://www.itl.nist.gov/div892/itg/carson/nistent/index.html