JAVA IN AN EMBEDDED ENVIRONMENT

POULES WARDA RAIHANA

MSc Thesis in
Electrical Engineering
DEC 2000

Ericsson AXE Research & Development.
Department of Teleinformatics, Royal Institute of Technology, Sweden.

Supervisors:
David Kassuja, Ericsson AXE Research & Development
Vladimir Vlassov, Department of Teleinformatics, Royal Institute of Technology
Examiner.
Björn Pehrson
Royal Institute of Technology

Table of Contents

Glossary

API	Applications Programming Interface
AU	Access Unit
DECT	Digital Enhanced Cordless Telecommunication
DHCP	Dynamic Host Control Protocol
DNS	Domain Name Server
IP	Internet Protocol
IPONAX	IP only access
IPTP	Internet Protocol Telephony Protocol
ISDN	Integrated Services Digital network
ISP	Internet Service Provider
JVM	Java Virtual Machine
LI	Line Interface
NFS	Network File System
POTS	Plain old telephone service
PSTN	Public switched Telephony Network
SS7	Signalling System 7
UG	User Gateway

CHAPTER 1

INTRODUCTION

The portability is an issue that is strongly recommended by the international community for standardization in order to move towards a globalise Data/Tele communication solutions.

Most leading companies are taking these issues in their consideration in the early stage of their new designed systems.

Today users are sitting in different platforms sending and receiving information to each other's without bothering about the type of the platform that the host computers are running on. The new challenge is application programs that are written to a certain platform cant be moved to a different platform without costing time and money. Java technology came in picture when the developers were thinking about a programming language that could solve this problem; the concept was to use a Java Virtual Machine. This machine isolates the operating system from the application program. That means that the developer will not think about on which platform his/her program is going to run on.

Problem definition

This thesis was performed at Ericsson AXE Research & Development. This thesis was started as Ericsson was considering using Java in a Network Terminal that will enable normal telephones (POTS) to access to a VOIP (Voice over IP) network. The purpose of the theoretical part of the thesis is to find out how far this can be possible using toady's Java technology e.g. Embedded Java, Java RTOS, Java telephony API's. The purpose of the practical part is to write a test program that can handle call control signalling in Java. The time differences between a request and response shall be measured.

1.2 Outline

The remainder of the thesis includes the following

Chapter 2 gives the background of this thesis. It starts with a description of the components of the IPONAX system and explanation to the IPTP protocol. Moreover, this chapter includes an introduction to the Java technology

Chapter 3 covers motivation, and implementation of the experiment. The achieved results will be presented analysed and discussed.

Chapter 4 contains references and bibliography

CHAPTER 2

2.1 Introduction

This chapter provides an overview of the material that I have studied in order to make a conclusion about whether the Java technology is capable of providing the network terminal functionality. In this chapter, the background of the thesis is introduced in the next section. The IPONAX system will be described**.** The construction and the main components of the IPONAX system will be described as well. The network terminal (AU) is briefly described. In section three, Java as programming language is introduced. The last two sections in this chapter are about the Java operating system and Java processors.

2.2 Background

Ericsson AXE Research & Development develops a system called IPONAX (IP only access). The concept behind IPONAX is using the Internet Protocol as a transparent bearer to all services that a distribution network can provide to the access networks. By having the AU (Access Unit) the user can access a variety of services for example making a telephone call using voice over IP, accessing the Internet. In this section, the components of the IPONAX will be described. More attention will be given to the AU, which represent the part of the system at which this thesis will deal with.

The IPONAX system [1] is divided into three main domains. In Figure 2.1, we can see the User Domain, the Access Network Domain and the Service Provider Domain. The User Domain represents users house or office .The Access Network Domain represent the access network that is owned by the access network provider. Service Provider Domain represents the different service providers for example the Internet Service provider and telephony service provider such as PSTN and ISDN.

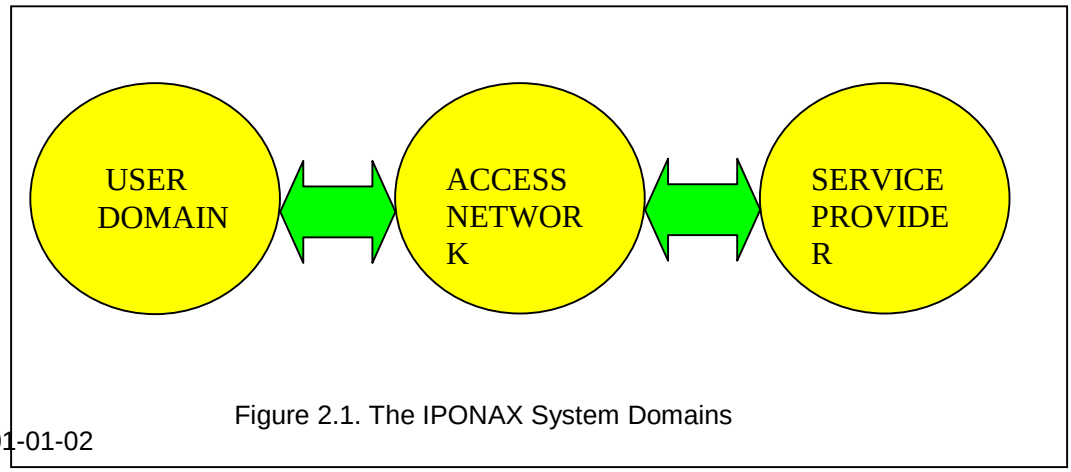

Figure 2.1. The IPONAX System Domains

In Figure 2.2, we can see more detailed information about the IPONAX. We can see the three domains that were described above and we can see the components that the IPONAX system is consists of.

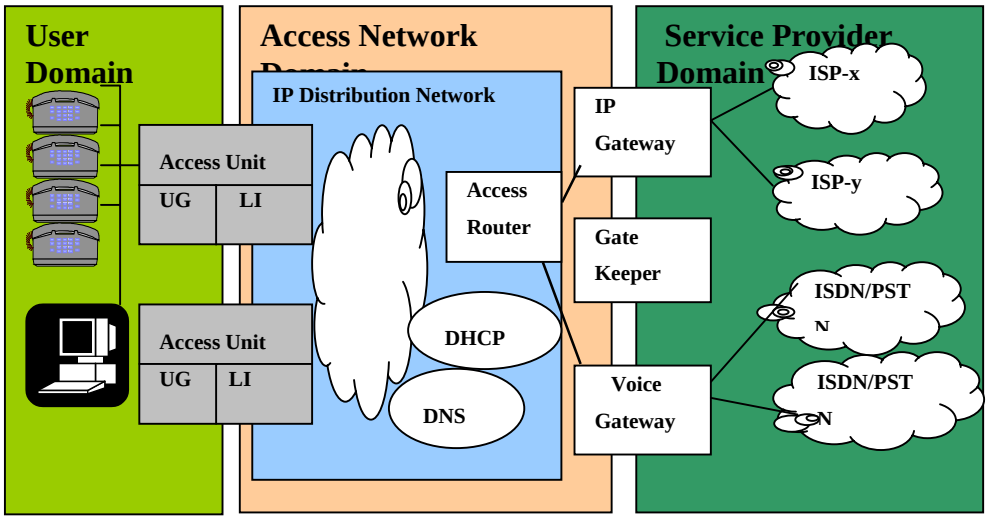

Figure 2.2. The IPONAX system

2.2.1 Access Unit (AU)

Figure 2.2, depicts the Access Unit, which includes two important parts. The first part is the User Gateway that performs all necessary protocol conversion between the devices that are connected to the AU and the IP based access network. This conversion is including the analogue POTS and digital DECT voice services to voice over IP. The second part is the line Interface (LI) that connects the AU to IP distribution network.

As mentioned above AU is the network terminal to which the user devices are connected. The AU contains a number of physical interfaces that are used to connect these devices. There are four telephones and one Ethernet connected to the AU.

2.2.2 IP Distribution Network

This component is used to transparently transport information between Access Units, Gateways and other components of the IP based access system. The functionality of this component is routing IP packets within the Access Network IP Domain, authenticate and allocate IP addresses to Access Units as well as binding them to service profile e.g. using DHCP.

2.2.3 Voice Gateway

This component converts the IP-based voice services between the Access Units and the voice networks. It connects to one or several service provider's PSTN/ISDN via standard protocols such as PRI or based on SS7.

2.2.4 H.323 Gate Keeper

This components translates between external addresses, such as telephone numbers, and IP-addresses within the access network IP domain used to locate the appropriate AU for incoming call or Voice Gateway for an outgoing call

2.2.5 IP Gateway

This component enables access from AU to other IP networks such as the Internet via a selected ISP or an Enterprise network.

2.3 Java

Java [2] is an object-oriented programming language with syntax derived from C and C++. Applets and application written in the Java language compile to a form that runs on the Java platform. The Java platform has two basic parts:

- Java Virtual Machine.
- Java Applications Programming Interface (API)

There is no full compatibility between Java and C/C++ because Java designers preferred to eliminate from these languages certain troublesome features. In particular, Java does not support enumerated constants, pointer arithmetic, traditional functions, structures and unions, multiple inheritance, go to statement, operator overloading. In their place, Java requires all the constant identifier and functions (methods) to be encapsulated within class declarations. Java provides standardized

support for multithreading and automatic garbage collection dynamically allocated memory.

There are many computer platforms today; there is for example Microsoft Windows, Macintosh, OS/2, UNIX platform; software must be compiled separately to run on each of these Platform. The Java platform is a new software platform but what sets the Java platform apart is that it sits on top of these other platforms, and compiles to *bytecodes,* which are not specific to any physical machine. Writing Java programs means writing to the Java platform and not to the underlying system.

2.3.1 Java Virtual Machine and Java Applications Programming Interface

The JVM is an abstract computing machine. Just like a real computing machines the Java Virtual Machine has an instruction set and uses various memory areas. The Java Virtual Machine is the key for the portability of the Java language as it represents the execution environment for Java code. The source code is compiled to bytecode and the Java virtual machine interprets and executes these bytecodes at run time. The JVM is implemented almost entirely in standard C code (a small part of JVM is written in assembly code) and all platform dependent code in the JVM is cleanly isolated in separate modules to make porting very simple.

Java API (Applications Programming Interface) is a standard set of libraries for writing Java programs. In 1995, these libraries were recognized to support Internet programming, and thus the Java API was created. The API contains a number of Packages and each Package includes a number of classes.

2.3.2 Java Compile and Runtime Environments

The Java language development environment is divided into two parts:

- Java compile-time environment.
- Java run-time environment.

Developers first write Java program source code and save it with (. java) extension file then compiles it to bytecodes. These bytecodes are instructions to the Java Virtual Machine.

Bytecodes are moved to the run-time environment. There the bytecodes will be first loaded to the memory by a *class loader*, and then the verifier will verify them, in order to insure that they are valid and they will not violate Java's security restrictions.

Afterwards they are interpreted by the *interpreter* one byte at a time. By interpretation, here means the translation of the bytecodes to a language that the host computers can understand.

2.3.3 Java Current State

One of Java2 platform group is called Micro Edition (J2ME) [4], this edition are targeted at the consumer electronics and embedded devices.

This J2ME architecture is designed to be modular and scalable. This modularity and scalability are defined by J2ME technology in a model with three layers of software (see Fig 2.3) built on a host operating system of the device:

Java virtual machine layer. This is an implementation of Java virtual machine that is customized for a special operating system and supports a particular J2ME configuration.

The configuration layer is invisible to the users but it is important to profile implementation. It is the minimum Java technology libraries and java virtual machine capabilities that an application developer can expect on implementing devices

Profile layer. It is the most visible layer to users and application providers. It represents a collection of APIs.

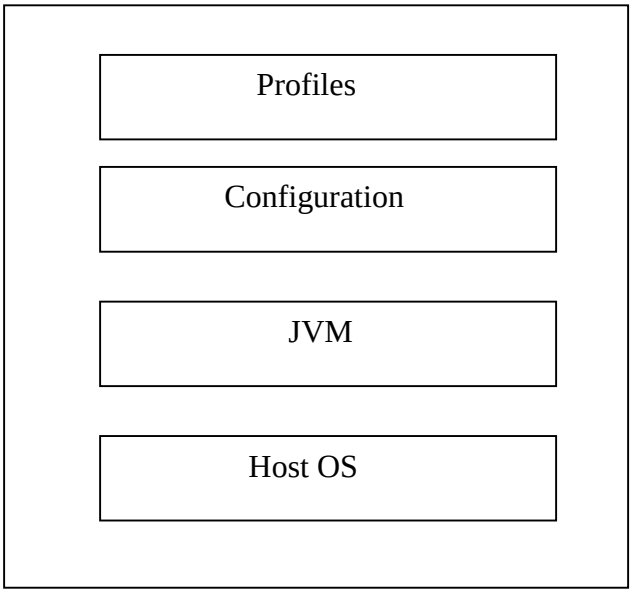

Figure 2.3 J2ME software layer stack

There are two configuration that are defined by the Java Community Process (JCP) for the J2ME architecture:

The ***Connected Device Configuration*** (CDC), which uses the classic Java virtual machine, a full feature VM that includes the all the functionality of a virtual machine residing on a desktop system. This configuration is intended for devices with at least a few megabytes of available memory. Example for devices of this configuration TV set top Boxes, Internet TV, Internet-enabled screen phones. A large range of user interface capabilities, memory size in the range of 2 to 16 megabytes, high-bandwidth network connections, most often using TCP/IP.

The ***Connected limited Device Configuration*** (CLDC) technology. This configuration is intended for devices with severely constrained memory environments such as wireless systems. Example for this category is cell phones, pagers and personal organizers. These devices have a very simple user interface, minimum memory size starting at 128 kilobyte, low bandwidth.

2.3.3.1 Connection limited Configuration (CLDC)

The CLDC configuration addresses the following areas:

Java language and Java Virtual Machine features, Core Java libraries (java.lang.*, java.util.*) , Input/output, Networking, Security and Internationalisation.

One goal for Java VM supporting CLDC is to be as compliant with Java language specification [5] as possible within the strict memory limits of the target. Accept that java will not support for floating point data types (float and double). Moreover Java will not support for finalization of class instances.

The Other goal for Java VM supporting CLDC is to be compliant to the JVM specification [6] as possible within memory constraints. Except the following

No support for floating point data types (float and double).

No support for JNI and thread groups or daemon threads.

No support for finalization of class instances.

The KVM is implemented in the C programming language, so it can be ported on various platforms for which C compiler (that is capable of compiling ANSI-compliant C files) is available.

The only non-ANSI feature in the source code is its use of 64-bit integer arithmetic.

2.3.3.1.1 CLCD libraries

The majority of the class libraries are a subset of the corresponding class in J2SE. The reason for that is to ensure upward compatibility and portability of application.
Only classes specified for wireless devices are specified by CDLC.
System Classes, Data type classes, Collection Classes, I/O classes and others are such example for subclasses see [7] for details.

2.4 Java Operating system

JavaOS implements the Java Platform for running Java applets and applications. As such, it implements the Java Virtual Machine and underlying functionality for windowing, networking and file system, without requiring the support of host operating system.
JavaOS is built from a combination of native code (instruction set and hardware platform specific) and Java code, which is platform independent.
JavaOS defines a platform as a CPU, physical memory, and any attached devices, buses, and slots. The platform independent component of the operating system is called runtime. The platform dependent portion of the OS as pictured in Fig2.4 is referred to as the JavaOS kernel for Java. It supports AWT and the networking and file-related I/O classes.
In order to support the Java Platform, JavaOS Supports the Java Virtual Machine using drivers for controlling a display, network interface, mouse and keyboard. JavaOS also supports the full API.

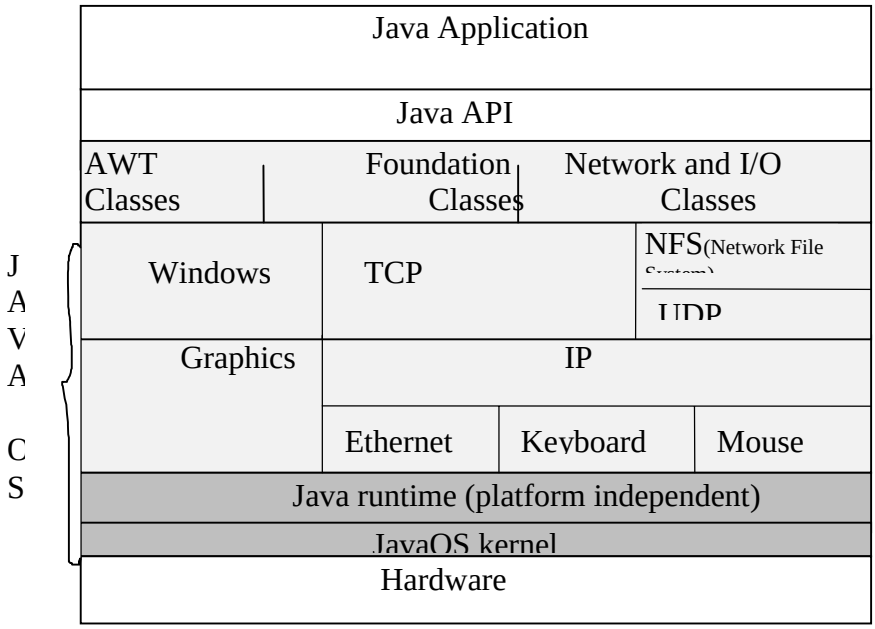

Figure 2.4 Java Platform running on JavaOS

All device drivers in JavaOS are written in the Java programming language. This is important for portability. There is two support classes written in C: The *Memory class* enables drivers to access and modify specific bytes and words of storage. The *interrupt class* handles interrupt dispatching. The methods of these classes are not made available to any Java application. JavaOS includes a suite of network protocols all written in Java programming language. These protocols include the basic transport and routing mechanism specified by TCP, UDP, IP, and ICMP standards. JavaOS uses both DNS and NIS for looking up host-names and supplying user names and passwords used during login. JavaOS supports both Reverse ARP and DHCP for discovering the network address of a device.

2.4.1 JavaOS Performance

JavaOS has not used Just -In-Time compiler to translate bytecodes into machine code, and makes minimal use of native methods. Therefore one might expect JavaOS to perform poorly, but this is not case.

Memory needed to support JavaOS is about 4MB of ROM and 4MB of RAM. In the ROM would be all code for JavaOS itself, including the kernel code, drivers, and Java

virtual Machine, standard classes, plus the JavaOS windows, graphics, and networking components plus the code for HotJava. Systems built using JavaOS that do not require windowing and HotJava code could run less than 2MB.

2.4.2 The target systems

Embedded devices are a target system for JavaOS. Some devices have only 1MB or 2MB of ROM, and possibly no graphical display. In such case, it is possible to remove the window and graphic codes from the JavaOS. Similarly if a device does not need certain network protocols, they could be eliminated. In order to meet some embedded devices it still needs to tune the Java Virtual Machine and garbage collection to support some real-time capabilities.

Sun is working with software tool vendors to build a rich software development environment for JavaOS .The memory footprints for JavaOS in its smallest possible configuration will be about 128K of RAM and 512 of ROM. Note that this memory is needed for JavaOS only, and more memory will be needed for the applications.

2.5 Java processors

The Java processor family consists of three production lines of microprocessors – PicoJava, microJava and UltraJava. These chips can execute Java bytecodes as their native machine language, as a result of that it will be not necessary to interpret or compile the bytecodes into some other CPU's machine language.

There are nine companies that are working on Java chips- SUN, NEC, IBM, Fujitsu, LG semicon, Rockwell, Siemens, Patriot Scientific and International Meta Systems.

Sun says that, PicoJava microprocessor cores are designed to natively execute bytecodes as defined by Java Virtual Machine, while they can also execute C/C++ codes as efficiently as comparable RISC CPU. The target markets are:

- Digital set-top boxes
- Internet TVs
- Screen phones

- PDA
- Automotive

MicroJava is another type of Java processor that Sun has and it is based on picoJava. It is a

General-purpose microprocessor that can be used in the following typical applications:

Embedded thin clients Automation

Network Computing

Terminal replacement

Consumer NCSA

Industrial control

Telecommunications

Office Automation

UltraJava processors will target advanced 3D graphics and other multimedia-intensive applications, which will be enabled through the Java Media API's. Potential devices that might take advantage of UltraJava processors would include media-oriented personal computers, high-end network computers, and intelligent televisions, advanced function set-top boxes.

Sun is designing UltraJava for 1999 or later.

So far, nobody has shipped actual products with Java chips (Byte May 1998). Marc Tremblay, a chip architect at Sun predicts that low-end Java chips based on PicoJava core will run Java about 20 times faster than interpreters running on Pentium at the same clock frequency and the chip will deliver about five times as much as Just-In-Time compiler running on Pentium.

CHAPTER 3

The Implementation and Experimentation

3.1 Introduction

This chapter includes the practical part of the thesis. The next section is about the specification of the practical part. The method applied in solving the problem is described in section three.

Ericssons aim was to implement this experiment in an embedded environment. It was planned that the Java Virtual Machine would be ported on OSE delta (a real time operating system, which used in embedded devices). This plan was not succeeded because of the delay in the delivery time for embedded Java. For this reason experiment was running on Windows NT platform.

3.2 The specification of the practical part of the thesis

The aim of this thesis is to write a test program in Java that implements the call control part of a telephony application. The application will be used to measure the time elapsed to sending a request and receiving a response. In response to that, this experiment was made.

The expectation from this experiment is to give a picture about Java performance.

3.3 Experiment illustration and network configuration

The experiment configuration is shown in Figure 3.1. As we can see, there are three computers, which are involved in implementing the experiment. The site keeper (will be discussed later in this section), a voice gateway to which the network terminal (Known as AU (Access Unit) in Ericsson) is connected, and finally a computer that hosts the test program.

From figure 3.1, we can distinguish the following:

- Voice gateway1 represents the machine that the test program is running on. This site will function as a call originator endpoint (caller side).
- The site keeper.
- Voice gateway2 represents the machine at which a real AU was connected. This site will function as a call terminator endpoint (called side).

The three machines were connected together by a hub. The physical layer used is an Ethernet cable. Each machine was configured to have an IP address. The site keeper must be configured to be aware of the existence of the two other machines. This configuration is a part of the IPTP (Internet Protocol Telephony Protocol) and without it, no registration request from the voice gateways will be confirmed.

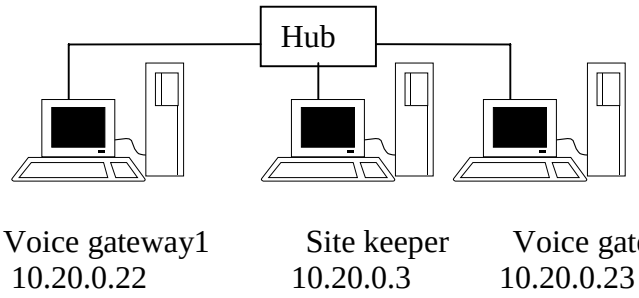

Figure 3.1. The configuration of the network

3.4 The method used

There are many approaches to measure Java performance. There are for example JDK profiling approach and installing own instrumentation approach. The first approach keeps track of the time spent at each routine and writes the information to a file. The flag used is **-prof** . This option is invoked with the commando java_g –prof myclass. The JDK profiles have uneven performance and some JDK versions have various instability [3]. The second approach is the one that is used in time measurement is achieved by inserting explicit timing as the following example.

```
Long start = System.currentTimeMillis ();
// operation to be timed goes here
long  stop = System.currentTimeMillis();
```

The elapsed time is then measured easy by

Long ElapsedTime = stop-start;

It is important to mention in this stage that the method **System.currentTimeMillis()** returns time in 1/1000ths of a second. Since that some systems (Windows for example) has time resolution less than 1millisecond this operation should be repeated **n** time and take the mean of results. This will give a more accurate measurement.

3.5 Motivation

The called voice gateway and the site keeper are the same for both parts A and B, see Figure 3.2. This means that the time elapsed in the shadowed area are constant. In other words, it is independent for our calculations. What are actually matters are the time points (T1, T2), which are pointed out in figure 3.4. The difference between T1 and T2 will give us the elapsed time for the application written in Java. This can be comparable to the time elapsed when application written in C++ is used.

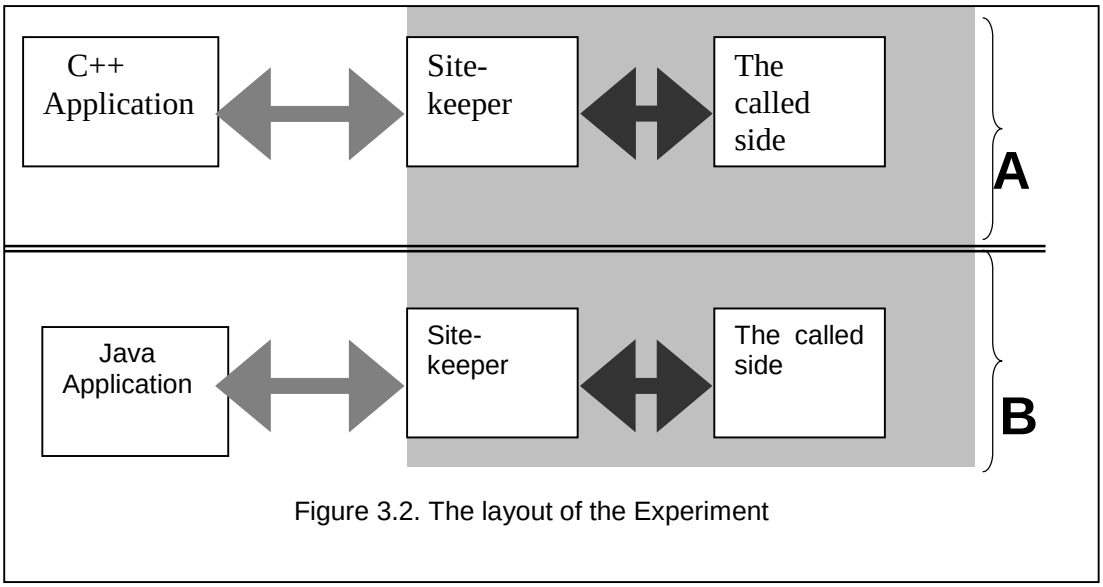

Figure 3.2. The layout of the Experiment

3.6 Test program

The test program includes implementation for some parts of the IPTP (Internet telephony Protocol) that are used in the current telephony application, which is written in C++. The test program consists of two components the registration program component and the set up component.

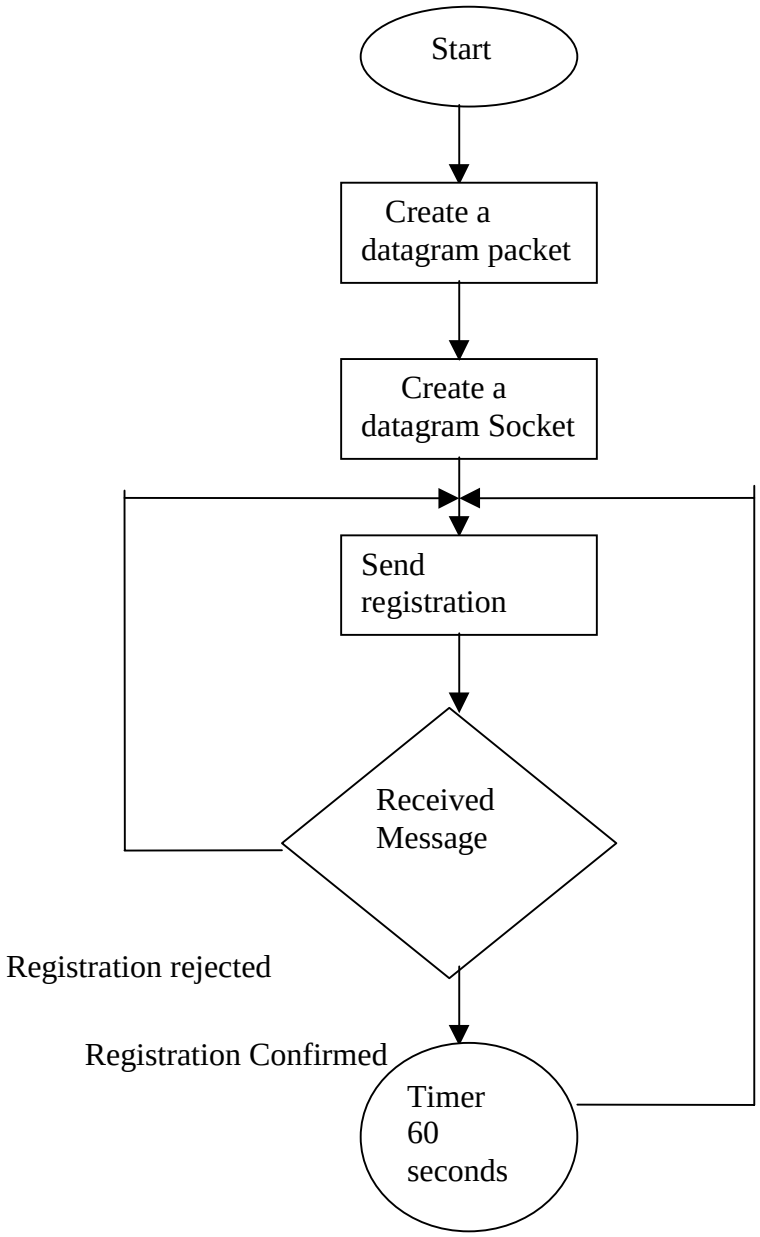

Figure 3.3 the flowchart of the registration

3.7 The registration and call control scenario

Fig 3.3 is a flowchart that will help to describe the Registration process. Both sides (the caller and the called) must be registered at the site keeper.

The registration process starts by creating a datagram packet then it creates a datagram socket. This socket is used to send/receive the datagram packets to/from the site keeper. A registration request will be packets into a datagrampacket and then sends to the site keeper. The registration will go into a loop waiting for datagrams that the site keeper will send. If the message received from the site keeper Registration rejected (for some reasons that are not in the scope of the test program) then the request will be send again. If the message received from the site keeper is registration confirmed then the registration process will wait one minute and send the registration request again.

Figure 3.4 shows the sequence diagram for the call control process. When the caller side receives registration confirmed message from the site keeper then a new process called call control process will start. In this process the caller and the site keeper will communicate through a communication channel called Socket.

This process starts by sending a setup messages from the caller side to the site keeper. The site keeper in his turn response to this message by sending back a call proceeding message and forward the setup message to the called side. At the same time the site keeper sends open logical channel. Then the caller side sends Open logical channel ACK. T1 and T2 will be registered and then the elapsed time will be calculated. The interval (T2-T1) we call it a Setup time .see in section 3.11 for the result of measurement.

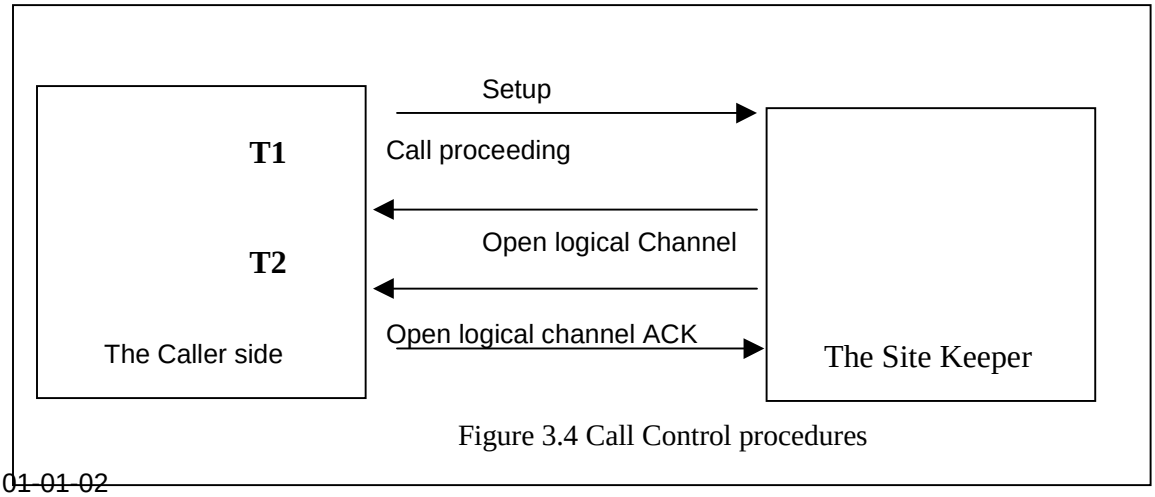

Figure 3.4 Call Control procedures

3.8 IPTP Messages

All the messages that are used to transfer data between the voice gateways and the site keeper have the format shown in table 1. From table1, we can see that the first two parameters, the Tag and the version parameter are common for all messages. Each message that has not these headers will be ignored by endpoints. Although the number of parameters in the parameter list varies of from message to message each parameter should follow the format that is shown in table 2.

COMMON PARAMETERS	4BYTES	TAG
	4bytes	Version
Message Headers	4bytes	Length
	4bytes	Request ID
	4bytes	Error Code
	4bytes	Message type
	4bytes	Message ID
Parameter List	Message Dependent	Parameter
		……..
		Parameter

Table1 The overall format of a message I PTP

4 BYTES	LENGTH
4 bytes	Parameter ID
Variable	Data

Table 2 the parameter format in IPTP

PARAMETER NAME	VALUE	NOTES
Tag	Iptp	Protocol name
Version	1	The version of the protocol
Length	104	Length =(total length-(Tag +version))
Request ID	0	You can decide the request ID.
Error code	0	0→no error 1→ erroried
Message type	1	0→ indication 1→ request 2→ response
Message ID	200	200→Registration request. 201→Registration confirmed 202→Registration rejected
Length	24	The parameter length

Parameter ID	102	
Data	10.20.0.232	The IP address of the Endpoint
Data		
Data		
Data		
Length	12	
Parameter ID	406	Endpoint type
Data	1	1→ Gateway
Length	12	
Parameter ID	417	Registration Type
Data	0	If 0→registration If 1→reregistration
Length	36	
Parameter ID	433	Endpoint State
Data	1	Version
Data		IP Address
Data	0	Alarms 0→No Alarms 1→Alarms
Data		Nominal Capacity
Data		Used capacity
Data		Degraded Capacity
Data		Registration state 0→unknown 1→Registered

Table 3 the Registration Message implementation in IPTP

Once the registration is confirmed the endpoint should sent a re-registration message each 60 second. This is implemented by sending the registration request again after changing the registration type to be re_ registration instead of registration. Once the endpoint is registered a TCP connection could be opened for call control messaging. This will take us to the next section.

3.9 The implementation of the registration

Writing the registration part of the test program is an implementation for a server/client interaction. In order to be able to communicate with the site keeper we should know the port on which it is listening.

From the documentation of the IPTP system I found out that the site keeper was listening on port 12080.

This means that clients (in this case the caller side) who want to communicate with the site keeper have to establish a UDP connection. First the application creates a DatagramPacket by using the constructor used in the DatagramPacket class. The constructor used takes the following arguments.

SendPacket=new

DatagramPacket(m1,m2.length,InetAddress.getByName("10.20.0.3"),*12080);*

"10.20.0.3" is the Internet address of the site keeper and 12080 is the port number on which the site keeper is listening.

The next step is to create a DatagramSocket by using a DatagramSocket constructor. The constructor takes the host's port number as an argument.

SendreceiveSocket = new DatagramSocket(12090);

Where 12090 represents the port number on which the application will send/receive its DatagramPackets. What is remarkable here that the application puts the address of the destination machine and the port number in the datagram packet and send it through the network. The next step is to send the registration request to the site keeper. This is implemented by using the class method send.

SendreceiveSocket.send (SendPacket);

This method takes the created DatagramPacket as an argument. Once the registration request is sent the application enters in a loop waiting for a response from the site keeper.

The response will be either a registration confirmed or registration rejected.

In case of registration confirmed the program inters in a loop and sends registration message once a minute. In case of registration rejected the program will send the registration request again.

3.10 The call Setup Message

The implementation of the call control part starts by creating a TCP connection. As mentioned in section 3.7 once the endpoint is registered a TCP connection will be opened to take care of call control part of the telephony application. In this part we are going to measure the time elapsed from sending a request and receiving a response.

When an end point wants to set a telephone call it will send a set up message to the site keeper. The site keeper will forward this message to the called party. The implementation starts by creating a Socket of type client.

Socket client=null;

client =new Socket(InetAddress.getByName("10.20.0.3"),12081);

The Socket constructer takes two arguments. The first argument is server's IP address and the second argument is server's port number. In this case "10.20.0.3" is the site keeper IP address and 12081 is site keepers port number.

Once the client socket is created we need two data streams for communication the Input Data Stream and the Output Data Stream.

DataInputStream in=new DataInputStream (client.getInputStream ());

DataInputStream takes BufferedInputStream as argument. This is obtained by calling socket's class method getInputStream ().

DataOutputStream out=new DataOutputStream (client.getOutputStream ());

DataOutputStream takes BufferedOutputStream as argument. This is obtained by calling socket's class method getOutputStream ().

The following are used to create a setup message m1.

Message message= new Message ();

byte[] m1 = new byte[Message.setupMessage.length];

m1=Message.setupMessage;

To send the setup message we gust use the DataOutputStream method write().which takes message as an argument.

out. write(m1);

To register the time T1 a system call function called currentTimeMillis() .This function takes no arguments and return the time in Millisecond.

T1=(long)System.currentTimeMillis();

Then the call control goes into a loop reading from the DataInputStream and

```
while (true)
        {
        .
        .
```

```
.
in.read(inbuf,0,500);
}
```

To read from the input stream a DataInputStream method called read() is used this method takes three argument.

The first argument is an array of type Byte in which the inputstream will be stored .The second argument is the place in the array where the data starts and the third argument is the length of the of the bytes that it is read from the DataInputStream. In order to specify the Message Type to know if it is call proceeding we check the byte number 27 in the inbuf array.

```
switch (0xff&inbuf[27])
{
case 0x65:  //******************Receiving callproceding  message*****//
{
T2=System.currentTimeMillis();
System.out.println("a Call proceding signal is received at");
System.out.println("T2= "+T2);
System.out.println("T = "+(T2-T1));
}
……
……
……
default:
System.out.println("Strange Signal check if you are expecting something else");
...
...
...
}
```

3.11 Measuring of the call setup time

As we mentioned in section 3.7 this section presents the result of measuring the call setup time. See fig 3.4 in which T1 and T2 are depicted.

The results of the ras.java program is written to results.txt file during the execution time by giving the following command:

Java ras>result.txt (See Appendix B for details)

The measure points T1 and T2 (see Appendix A) are measured by using two methods of System Class called println() and currentTimeMillis().

Consider the time T is elapsed time from sending call request and receiving call proceeding

Then T = T2-T1.

To measure T1 , the following lines are added to the code of the client program:

T1=System.currentTimeMillis();

System.out.println("T1="+T1);

The system time accuracy in Windows NT where our experiment was running on is ± 10 milliseconds. All measurements below 1 millisecond are rounded to zero.

In the same way the measure point T2 is measured:

T2=System.currentTimeMillis();

System.out.println("T2="+T2);

Statistical calculation

As shown in the table the results are between 0 and 10 milliseconds. This is du to the accuracy in Windows NT Operative System. The average value for the above results is:

$$T' = \sum_{i=1}^{i=n} T_i / n$$

Where i = index, n= number of readings, T = measured value and T' is the average value.

T'= 180/60 = 3 milliseconds

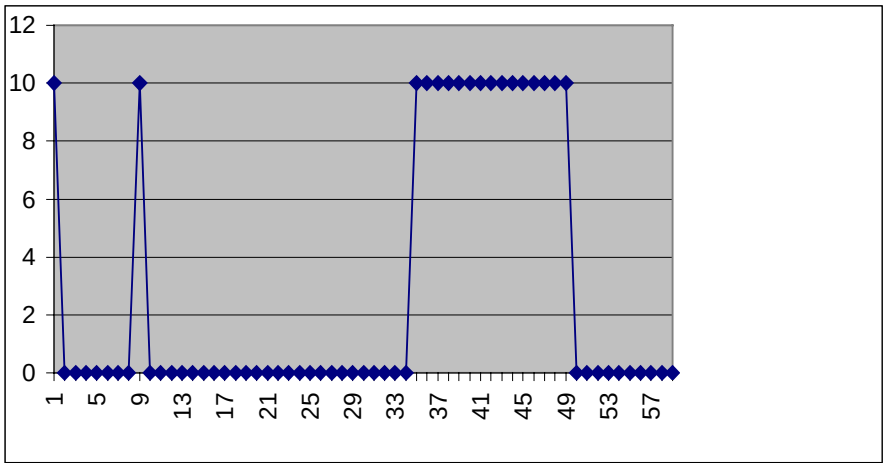

Fig 3.5 shows the time elapsed from sending a call setup message and receiving a call proceeding message.

CHAPTER 4

Discussion

The aim of this thesis as mentioned before is to investigate what can Java contribute to the network terminal.

4.1 Embedded Applications Concerns

There are three aspects that embedded devices are concerned with; speed, size and response times. These three aspects must be taken in consideration when we implement a Java solution that can offers program efficiency, which is equivalent to that offered by C/C++.

4.2 The advantage of using Java in the network terminal.

Maybe one of the most advantages of Java is its portability. This as we mentioned before means, that Java application can run on any platform that supports Java.

Portability

As mentioned in section 2.3 when Java Virtual Machine was introduced that the JVM is the key to the portability that Java Platform has. Java provides a standard set of class libraries to make the nature of the underlying platform invisible.

Because the Java platform is good at isolating the program from underlying platform dependencies, the platform used to prototype a new application is unimportant.

This means AU application can prototyped and developed on a workstation running Unix or Windows 95.There may be changes required to account for different resource availability such as less memory or non existence display. However, it is likely that a large amount of the application code will run unchanged.

Reliable and secure

Embedded applications typically need to be reliable to avoid failure at runtime. When an application uses entrusted code like third party extensions, they need to be secure against hostile attack or unintentional harmful consequences from executing that code.

The Java designer where aware of this problem and provided the language with features to increase the reliability as much as possible, for example elimination of pointers and all forms of direct memory access from the language. Primitive data is

always accessed and passed by value and classes are always accessed and passed by reference.

Dynamic Scalability

The Java language includes a dynamic loader that can load new Java code at run time. An application written in Java can load Java classes as and when they are needed. More Java classes may be loaded from remote storage devices like a remote computer via network connection.

The dynamic nature of Java also means upgrading software. For example if the embedded application is already loading classes remotely at start up then all installed instances of application can be upgraded simply by changing the remote class files.

Upgrading is also one of the troubles that we have today in the AU.

4.3 The disadvantage of using Java in the network terminal

Java performance

When it comes to the telephony application that are used in network terminal, these application has a real-time requirements from the underlying operating system

The Java Virtual machine relies on the interpretation of bytecodes at runtime. The JVM is simply simulating a virtual Java CPU at run time. Clearly, this will not result in Java programs with execution speeds comparable to those using C and C++ code. The interpretation solution is simply slow and there are no ways around it.

If a sacrifice in performance is not acceptable for our applications, The alternative is to use dynamic or Just-In-Time compiler. This will improve the performance of the Java code running on that client, but this will expands the size of the binary to three or four times and more memory will be needed for the compiler it self. Moreover, it will need an extra RAM for working space.

For embedded devices, this will be a problem as this costs money and increases complexity.

The Garbage Collection

The problem with garbage collection algorithm used by JVM is that the garbage collector must run uninterrupted until they are completed. This means that there can be no determinacy about scheduling of other Java threads once garbage collector daemon thread has been scheduled. A solution to go around this problem is to run JVM without the garbage collection daemon-thread.

Doing this the developer can guarantee that garbage collection will occur in only two situations namely, when the system class method gc is requested, or when it is unavoidable to avoid doing so i.e. du to the failure of a memory allocation.

The application can even include it own garbage collection daemon thread that invokes the garbage collection. This way it will make it possible to disable the thread while time critical section code is running.

The object-oriented paradigm is widely supported by analysis and design methods, which can help to speed up the development of the application.

The JavaOS

There are advantages of using JavaOS, including:

The JavaOS may be stored on ROM, enabling simple, low-cost systems that boot quickly.

JavaOS achieves the goal of eliminating the overhead of host operating system. Because JavaOS contains no extraneous features found in other operating systems, it allows smaller and simpler devices to build and that execute Java programs more efficiently than other systems. Yet we have not seen it in the market. Using it in the AU will dramatically decrease the price of the product and make the AU more price-concurrent.

Summary

With the Java technology that is available today, it is avoidable to use Java when performance is an important issue for the application. The Java platform for the current time is not ideally suited for hard real-time application. There are some factors that affects the language namely the speed and garbage collection As the performance constitute an important issue in some parts of the telephony application it is not recommended to use Java in those parts On the other hand it is quite comfortable to use Java on other parts of the telephony application that performance is not an important. The graphical user interface is such an example.

References

[1] System Design Specification IPONAX by Jan Ulander and Bengt Werdin ; EUS/SL-98:XXX

[2] Java virtual machine *http://java.sun.com/docs/books/vmspec/html/Introduction.doc.html*

[3] *Thinking in Java* by Bruce Eckel.

[4] Java 2 Micro Edition http://java.sun.com/j2me.

[5] The Java Language specification by James Goslin, Bill Joy and Guy L.Steele.

Addison Wesley, 1996, ISBN 0-201-63451-1.

[6] Java Virtual Machine by Tim Lindholm and Frank Yellin.

Addison-Wesley, 1996, ISBN 0-201-63452-x.

[7] Java 2 platform Edition (J2ME) technology for creating Mobile Devices May 19,2000 ;white paper .by Sun Micro Electronics, Inc.

APPENDIX A: THE MEASURING DATA

index	T1	T2	T2-T1	index	T1	T2	T2-T1
1	921238982078	921238982088	10	31	921240842654	921240842654	0
2	921239042165	921239042165	0	32	921240902670	921240902670	0
3	921239102181	921239102181	0	33	921240962686	921240962686	0
4	921239162197	921239162197	0	34	921241022703	921241022703	0
5	921239222214	921239222214	0	35	921241103659	921241103669	10
6	921239282230	921239282230	0	36	921241163746	921241163756	10
7	921239342246	921239342246	0	37	921241223842	921241223852	10
8	921239402263	921239402263	0	38	921241283928	921241283938	10
9	921239462279	921239462289	10	39	921241344015	921241344025	10
10	921239522295	921239522295	0	40	921241404101	921241404111	10
11	921239582312	921239582312	0	41	921241464188	921241464198	10
12	921239642328	921239642328	0	42	921241524274	921241524284	10
13	921239702344	921239702344	0	43	921241584360	921241584370	10
14	921239762360	921239762360	0	43	921241644447	921241644457	10
15	921239822377	921239822377	0	45	921241704543	921241704553	10
16	921239882393	921239882393	0	46	921241764630	921241764640	10
17	921239942409	921239942409	0	47	921241824716	921241824726	10
18	921240002426	921240002426	0	48	921241884802	921241884812	10
19	921240062442	921240062442	0	49	921241944889	921241944899	10
20	921240122458	921240122458	0	50	921242004975	921242004985	10
21	921240242491	921240242491	0	51	921242065052	921242065052	0
22	921240302507	921240302507	0	52	921242125068	921242125068	0
23	921240362523	921240362523	0	53	921242185084	921242185084	0
24	921240422540	921240422540	0	54	921242245100	921242245100	0
25	921240482556	921240482556	0	55	921242305117	921242305117	0
26	921240542572	921240542572	0	56	921242365133	921242365133	0
27	921240602589	921240602589	0	57	921242425149	921242425149	0
28	921240662605	921240662605	0	58	921242485166	921242485166	0
29	921240722621	921240722621	0	59	921242545182	921242545182	0
30	921240782638	921240782638	0	60	921242605198	921242605198	0

Table 1 the measuring of the setup time

APPENDIX B: THE FILE RESULT.TXT

The results founded on running the test program at the WindowsNT machine
which has the following specifications:

```
C:\java\bin>java ras

Waiting Response from the SiteKeeper
201
Registration Confirmed
A setup message is sent at:
T1= 921238982078
a Call proceding signal is received at
T2= 921238982088
T = 10
The OLCA was sent at
T3= 921238982109
a relese complete is received
T4= 921238982129msec
T3-T2=  21msec
T4-T3=  20msec
This is Reregistration number: 1 The System time is: 921239042165
A setup message is sent at:
T1= 921239042165
a Call proceding signal is received at
T2= 921239042165
T = 0
The OLCA was sent at
T3= 921239042175
a relese complete is received
T4= 921239042175msec
T3-T2=  10msec
T4-T3=  0msec
This is Reregistration number: 2 The System time is: 921239102181
A setup message is sent at:
T1= 921239102181
a Call proceding signal is received at
T2= 921239102181
T = 0
The OLCA was sent at
T3= 921239102181
a relese complete is received
T4= 921239102191msec
T3-T2=  0msec
T4-T3=  10msec
This is Reregistration number: 3 The System time is: 921239162197
A setup message is sent at:
T1= 921239162197
a Call proceding signal is received at
T2= 921239162197
T = 0
The OLCA was sent at
T3= 921239162197
a relese complete is received
T4= 921239162207msec
T3-T2=  0msec
T4-T3=  10msec
This is Reregistration number: 4 The System time is: 921239222214
A setup message is sent at:
T1= 921239222214
a Call proceding signal is received at
T2= 921239222214
T = 0
The OLCA was sent at
T3= 921239222214
a relese complete is received
T4= 921239222224msec
T3-T2=  0msec
T4-T3=  10msec
This is Reregistration number: 5 The System time is: 921239282230
A setup message is sent at:
```

T1= 921239282230
a Call proceding signal is received at
T2= 921239282230
T = 0
The OLCA was sent at
T3= 921239282230
a relese complete is received
T4= 921239282240msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 6 The System time is: 921239342246
A setup message is sent at:
T1= 921239342246
a Call proceding signal is received at
T2= 921239342246
T = 0
The OLCA was sent at
T3= 921239342246
a relese complete is received
T4= 921239342256msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 7 The System time is: 921239402263
A setup message is sent at:
T1= 921239402263
a Call proceding signal is received at
T2= 921239402263
T = 0
The OLCA was sent at
T3= 921239402273
a relese complete is received
T4= 921239402273msec
T3-T2= 10msec
T4-T3= 0msec
This is Reregistration number: 8 The System time is: 921239462279
A setup message is sent at:
T1= 921239462279
a Call proceding signal is received at
T2= 921239462289
T = 10
The OLCA was sent at
T3= 921239462289
a relese complete is received
T4= 921239462289msec
T3-T2= 0msec
T4-T3= 0msec
This is Reregistration number: 9 The System time is: 921239522295
A setup message is sent at:
T1= 921239522295
a Call proceding signal is received at
T2= 921239522295
T = 0
The OLCA was sent at
T3= 921239522295
a relese complete is received
T4= 921239522305msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 10 The System time is: 921239582312
A setup message is sent at:
T1= 921239582312
a Call proceding signal is received at
T2= 921239582312
T = 0
The OLCA was sent at
T3= 921239582312
a relese complete is received
T4= 921239582322msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 11 The System time is: 921239642328
A setup message is sent at:
T1= 921239642328

a Call proceding signal is received at
T2= 921239642328
T = 0
The OLCA was sent at
T3= 921239642328
a relese complete is received
T4= 921239642338msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 12 The System time is: 921239702344
A setup message is sent at:
T1= 921239702344
a Call proceding signal is received at
T2= 921239702344
T = 0
The OLCA was sent at
T3= 921239702344
a relese complete is received
T4= 921239702354msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 13 The System time is: 921239762360
A setup message is sent at:
T1= 921239762360
a Call proceding signal is received at
T2= 921239762360
T = 0
The OLCA was sent at
T3= 921239762360
a relese complete is received
T4= 921239762370msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 14 The System time is: 921239822377
A setup message is sent at:
T1= 921239822377
a Call proceding signal is received at
T2= 921239822377
T = 0
The OLCA was sent at
T3= 921239822387
a relese complete is received
T4= 921239822387msec
T3-T2= 10msec
T4-T3= 0msec
This is Reregistration number: 15 The System time is: 921239882393
A setup message is sent at:
T1= 921239882393
a Call proceding signal is received at
T2= 921239882393
T = 0
The OLCA was sent at
T3= 921239882393
a relese complete is received
T4= 921239882403msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 16 The System time is: 921239942409
A setup message is sent at:
T1= 921239942409
a Call proceding signal is received at
T2= 921239942409
T = 0
The OLCA was sent at
T3= 921239942409
a relese complete is received
T4= 921239942419msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 17 The System time is: 921240002426
A setup message is sent at:
T1= 921240002426
a Call proceding signal is received at

T2= 921240002426
T = 0
The OLCA was sent at
T3= 921240002426
a relese complete is received
T4= 921240002436msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 18 The System time is: 921240062442
A setup message is sent at:
T1= 921240062442
a Call proceding signal is received at
T2= 921240062442
T = 0
The OLCA was sent at
T3= 921240062442
a relese complete is received
T4= 921240062452msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 19 The System time is: 921240122458
A setup message is sent at:
T1= 921240122458
a Call proceding signal is received at
T2= 921240122458
T = 0
The OLCA was sent at
T3= 921240122458
a relese complete is received
T4= 921240122468msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 20 The System time is: 921240182475
A setup message is sent at:
T1= 921240182475
a Call proceding signal is received at
T2= 921240182475
T = 0
The OLCA was sent at
T3= 921240182475
a relese complete is received
T4= 921240182485msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 21 The System time is: 921240242491
A setup message is sent at:
T1= 921240242491
a Call proceding signal is received at
T2= 921240242491
T = 0
The OLCA was sent at
T3= 921240242491
a relese complete is received
T4= 921240242501msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 22 The System time is: 921240302507
A setup message is sent at:
T1= 921240302507
a Call proceding signal is received at
T2= 921240302507
T = 0
The OLCA was sent at
T3= 921240302507
a relese complete is received
T4= 921240302517msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 23 The System time is: 921240362523
A setup message is sent at:
T1= 921240362523
a Call proceding signal is received at
T2= 921240362523

T = 0
The OLCA was sent at
T3= 921240362523
a relese complete is received
T4= 921240362533msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 24 The System time is: 921240422540
A setup message is sent at:
T1= 921240422540
a Call proceding signal is received at
T2= 921240422540
T = 0
The OLCA was sent at
T3= 921240422540
a relese complete is received
T4= 921240422550msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 25 The System time is: 921240482556
A setup message is sent at:
T1= 921240482556
a Call proceding signal is received at
T2= 921240482556
T = 0
The OLCA was sent at
T3= 921240482556
a relese complete is received
T4= 921240482566msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 26 The System time is: 921240542572
A setup message is sent at:
T1= 921240542572
a Call proceding signal is received at
T2= 921240542572
T = 0
The OLCA was sent at
T3= 921240542572
a relese complete is received
T4= 921240542582msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 27 The System time is: 921240602589
A setup message is sent at:
T1= 921240602589
a Call proceding signal is received at
T2= 921240602589
T = 0
The OLCA was sent at
T3= 921240602599
a relese complete is received
T4= 921240602599msec
T3-T2= 10msec
T4-T3= 0msec
This is Reregistration number: 28 The System time is: 921240662605
A setup message is sent at:
T1= 921240662605
a Call proceding signal is received at
T2= 921240662605
T = 0
The OLCA was sent at
T3= 921240662605
a relese complete is received
T4= 921240662615msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 29 The System time is: 921240722621
A setup message is sent at:
T1= 921240722621
a Call proceding signal is received at
T2= 921240722621
T = 0

The OLCA was sent at
T3= 921240722631
a relese complete is received
T4= 921240722631msec
T3-T2= 10msec
T4-T3= 0msec
This is Reregistration number: 30 The System time is: 921240782638
A setup message is sent at:
T1= 921240782638
a Call proceding signal is received at
T2= 921240782638
T = 0
The OLCA was sent at
T3= 921240782638
a relese complete is received
T4= 921240782648msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 31 The System time is: 921240842654
A setup message is sent at:
T1= 921240842654
a Call proceding signal is received at
T2= 921240842654
T = 0
The OLCA was sent at
T3= 921240842654
a relese complete is received
T4= 921240842664msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 32 The System time is: 921240902670
A setup message is sent at:
T1= 921240902670
a Call proceding signal is received at
T2= 921240902670
T = 0
The OLCA was sent at
T3= 921240902670
a relese complete is received
T4= 921240902680msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 33 The System time is: 921240962686
A setup message is sent at:
T1= **921240962686**
a Call proceding signal is received at
T2= 921240962686
T = 0
The OLCA was sent at
T3= 921240962686
a relese complete is received
T4= 921240962696msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 34 The System time is: 921241022703
A setup message is sent at:
T1= 921241022703
a Call proceding signal is received at
T2= 921241022703
T = 0
The OLCA was sent at
T3= 921241022703
a relese complete is received
T4= 921241022713msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 35 The System time is: 921241082719
A setup message is sent at:
T1= 921241103659
a Call proceding signal is received at
T2= 921241103669
T = 10
The OLCA was sent at

T3= 921241103689
a relese complete is received
T4= 921241103709msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 36 The System time is: 921241163746
A setup message is sent at:
T1= 921241163746
a Call proceding signal is received at
T2= 921241163756
T = 10
The OLCA was sent at
T3= 921241163786
a relese complete is received
T4= 921241163806msec
T3-T2= 30msec
T4-T3= 20msec
This is Reregistration number: 37 The System time is: 921241223832
A setup message is sent at:
T1= 921241223842
a Call proceding signal is received at
T2= 921241223852
T = 10
The OLCA was sent at
T3= 921241223872
a relese complete is received
T4= 921241223892msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 38 The System time is: 921241283918
A setup message is sent at:
T1= 921241283928
a Call proceding signal is received at
T2= 921241283938
T = 10
The OLCA was sent at
T3= 921241283958
a relese complete is received
T4= 921241283978msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 39 The System time is: 921241344005
A setup message is sent at:
T1= 921241344015
a Call proceding signal is received at
T2= 921241344025
T = 10
The OLCA was sent at
T3= 921241344045
a relese complete is received
T4= 921241344065msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 40 The System time is: 921241404091
A setup message is sent at:
T1= 921241404101
a Call proceding signal is received at
T2= 921241404111
T = 10
The OLCA was sent at
T3= 921241404131
a relese complete is received
T4= 921241404161msec
T3-T2= 20msec
T4-T3= 30msec
This is Reregistration number: 41 The System time is: 921241464178
A setup message is sent at:
T1= 921241464188
a Call proceding signal is received at
T2= 921241464198
T = 10
The OLCA was sent at
T3= 921241464218

a relese complete is received
T4= 921241464238msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 42 The System time is: 921241524264
A setup message is sent at:
T1= 921241524274
a Call proceding signal is received at
T2= 921241524284
T = 10
The OLCA was sent at
T3= 921241524304
a relese complete is received
T4= 921241524324msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 43 The System time is: 921241584350
A setup message is sent at:
T1= 921241584360
a Call proceding signal is received at
T2= 921241584370
T = 10
The OLCA was sent at
T3= 921241584390
a relese complete is received
T4= 921241584410msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 44 The System time is: 921241644437
A setup message is sent at:
T1= 921241644447
a Call proceding signal is received at
T2= 921241644457
T = 10
The OLCA was sent at
T3= 921241644487
a relese complete is received
T4= 921241644497msec
T3-T2= 30msec
T4-T3= 10msec
This is Reregistration number: 45 The System time is: 921241704533
A setup message is sent at:
T1= 921241704543
a Call proceding signal is received at
T2= 921241704553
T = 10
The OLCA was sent at
T3= 921241704573
a relese complete is received
T4= 921241704593msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 46 The System time is: 921241764620
A setup message is sent at:
T1= 921241764630
a Call proceding signal is received at
T2= 921241764640
T = 10
The OLCA was sent at
T3= 921241764660
a relese complete is received
T4= 921241764680msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 47 The System time is: 921241824706
A setup message is sent at:
T1= 921241824716
a Call proceding signal is received at
T2= 921241824726
T = 10
The OLCA was sent at
T3= 921241824746
a relese complete is received

T4= 921241824766msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 48 The System time is: 921241884792
A setup message is sent at:
T1= 921241884802
a Call proceding signal is received at
T2= 921241884812
T = 10
The OLCA was sent at
T3= 921241884832
a relese complete is received
T4= 921241884852msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 49 The System time is: 921241944879
A setup message is sent at:
T1= 921241944889
a Call proceding signal is received at
T2= 921241944899
T = 10
The OLCA was sent at
T3= 921241944929
a relese complete is received
T4= 921241944939msec
T3-T2= 30msec
T4-T3= 10msec
This is Reregistration number: 50 The System time is: 921242004965
A setup message is sent at:
T1= 921242004975
a Call proceding signal is received at
T2= 921242004985
T = 10
The OLCA was sent at
T3= 921242005005
a relese complete is received
T4= 921242005025msec
T3-T2= 20msec
T4-T3= 20msec
This is Reregistration number: 51 The System time is: 921242065052
A setup message is sent at:
T1= 921242065052
a Call proceding signal is received at
T2= 921242065052
T = 0
The OLCA was sent at
T3= 921242065052
a relese complete is received
T4= 921242065062msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 52 The System time is: 921242125068
A setup message is sent at:
T1= 921242125068
a Call proceding signal is received at
T2= 921242125068
T = 0
The OLCA was sent at
T3= 921242125068
a relese complete is received
T4= 921242125078msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 53 The System time is: 921242185084
A setup message is sent at:
T1= 921242185084
a Call proceding signal is received at
T2= 921242185084
T = 0
The OLCA was sent at
T3= 921242185084
a relese complete is received
T4= 921242185094msec

T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 54 The System time is: 921242245100
A setup message is sent at:
T1= 921242245100
a Call proceding signal is received at
T2= 921242245100
T = 0
The OLCA was sent at
T3= 921242245100
a relese complete is received
T4= 921242245110msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 55 The System time is: 921242305117
A setup message is sent at:
T1= 921242305117
a Call proceding signal is received at
T2= 921242305117
T = 0
The OLCA was sent at
T3= 921242305117
a relese complete is received
T4= 921242305127msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 56 The System time is: 921242365133
A setup message is sent at:
T1= 921242365133
a Call proceding signal is received at
T2= 921242365133
T = 0
The OLCA was sent at
T3= 921242365133
a relese complete is received
T4= 921242365143msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 57 The System time is: 921242425149
A setup message is sent at:
T1= 921242425149
a Call proceding signal is received at
T2= 921242425149
T = 0
The OLCA was sent at
T3= 921242425149
a relese complete is received
T4= 921242425159msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 58 The System time is: 921242485166
A setup message is sent at:
T1= 921242485166
a Call proceding signal is received at
T2= 921242485166
T = 0
The OLCA was sent at
T3= 921242485166
a relese complete is received
T4= 921242485176msec
T3-T2= 0msec
T4-T3= 10msec
This is Reregistration number: 59 The System time is: 921242545182
A setup message is sent at:
T1= 921242545182
a Call proceding signal is received at
T2= 921242545182
T = 0
The OLCA was sent at
T3= 921242545182
a relese complete is received
T4= 921242545192msec
T3-T2= 0msec

T4-T3= 10msec
This is Reregistration number: 60 The System time is: 921242605198
A setup message is sent at:
T1= 921242605198
a Call proceding signal is received at
T2= 921242605198
T = 0
The OLCA was sent at
T3= 921242605198
a relese complete is received
T4= 921242605208msec
T3-T2= 0msec
T4-T3= 10msec