Proyecto de Grado 2008

Anexo IV

Dispositivos Móviles - IP4JVM

Autores:

Leandro Scasso

Marcos Techera

Tutor:

Ariel Sabiguero

Tribunal:

Andrés Aguirre

Eduardo Grampín

Carlos Martínez

Tabla de contenidos

1.Introducción

En este capítulo se presentan los objetivos buscados en esta etapa del proyecto. También se realiza una descripción de la organización de este documento. Durante la vida del proyecto IP4JVM se ha buscado que el mismo funcione con las ultimas versiones de la máquina virtual Java y haciendo uso de las ultimas funcionalidades de las mismas como puede ser el uso de Generics, la utilización de properties, entre otras cosas. Es por eso que se planteo la duda, ¿Qué tan lejos estamos de Java Micro Edition[JME]?

1.1.Objetivos

Dentro de los objetivos iniciales del proyecto se encontraba el de poder dilucidar si el proyecto podía correr en un dispositivo móvil. Al avanzar el proyecto e ir obteniendo conocimiento sobre el mismo este objetivo dio lugar a nuevos puntos a plantearnos para poder responder al mismo. Estos puntos son:

- Que soporte brinda **Java Micro Edition** en comparación con **J2SE**.
- Que **sistemas operativos** están disponibles en los **dispositivos móviles**.
- Que alternativa al **Pcap** [PCAP] existe dentro de los sistemas operativos de los **dispositivos móviles.**

1.2.Organización de este documento

El presente documento se encuentra organizado en capítulos, cada uno de los cuales presenta un aspecto funcional de esta etapa del proyecto.

Capítulo 1, Introducción: en la introducción se define los términos generales del problema a tratar en el presente documento, los objetivos que se desean alcanzar y se describe la organización del documento.

Capítulo 2, Estado del arte: en este capítulo se detalla el estudio del estado del arte realizado, esto incluye Java Micro Edition, Sistemas Operativos de dispositivos Móviles y protocolos de redes disponibles en dispositivos móviles.

Capítulo 3, Conclusiones: en este capitulo es en donde se presentan los resultados obtenidos luego de la investigación realizada.

2.Estado del Arte

En este capítulo se presentan los resultados obtenidos durante la proceso de investigación de esta etapa del proyecto. En primer lugar se presenta y se realiza un estudio de la Java Micro Edition, y a continuación se presentan y se realiza un estudio sobre los sistemas operativos disponibles en los dispositivos móviles.

2.1.Java Micro Edition

Java Micro Edition(Java ME)[JME] nace por la necesidad de poder manejar las restricciones asociadas a las aplicaciones para dispositivos pequeños. Con estos motivos es que Sun define las bases de la tecnología Java ME de forma que sea posible crear y correr aplicaciones Java en dispositivos pequeños con limitaciones de memoria, pantalla y energía.

Java ME es una colección de tecnologías y especificaciones que pueden ser combinadas para construir un Java runtime especifico de forma que cumpla con los requerimientos de un dispositivo o mercado en particular.

La tecnología Java ME está basada en tres elementos:

•una configuración que provee de un conjunto básico de bibliotecas y capacidades de la máquina virtual para una amplia gama de dispositivos,

•un perfil, que es un conjunto de APIs que soportan una amplia variedad de dispositivos, y

•paquetes opcionales, los cuales son un conjunto de APIs especificas de una tecnología

2.1.1.Configuraciones de Java ME

Con el tiempo la plataforma Java ME se ha dividido en dos configuraciones base, una que se ajusta a los dispositivos móviles más pequeños y otra que está dirigida a los dispositivos móviles más capaces como pueden ser los teléfonos inteligentes (smart-phones) o los PDAs.

La configuración para dispositivos móviles pequeños llamada **"Connected Limited Device Configuration"** (CLDC)[CLDC] y para los dispositivos más capaces **"Connected Device Configuration"** (CDC)[CDC].

En la siguiente ilustración se puede apreciar una visión general de los componentes de la tecnología Java ME y como están relacionados estos con las demás tecnologías Java.

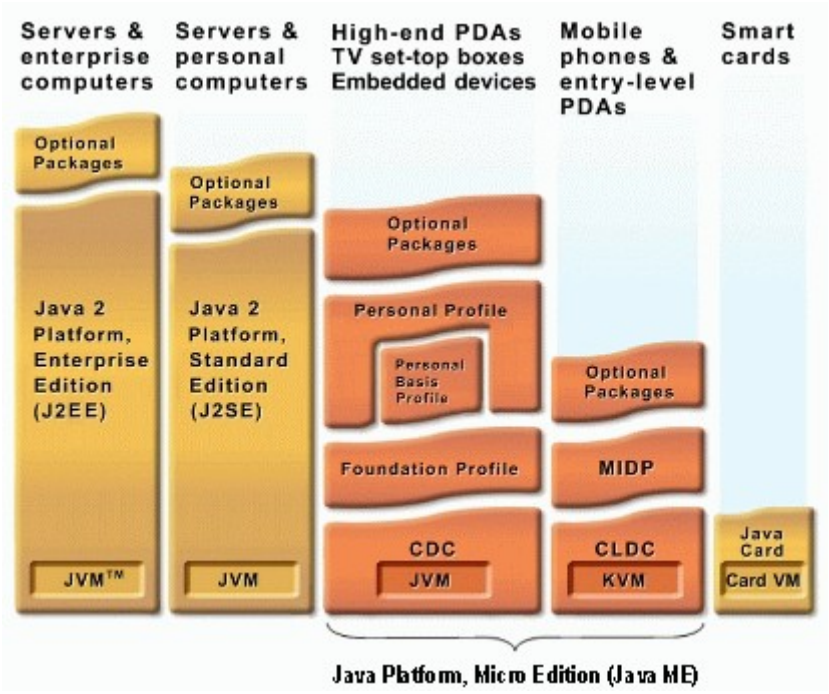

Ilustración 1: Tecnologías Java

2.1.2.KVM

KVM[KVM] también conocida como **K Virtual Machine**, es una máquina virtual Java compacta programada desde cero en C. Fue creada con la intención de correr en dispositivos con restricciones de recursos como ser teléfonos celulares, organizadores personales, etc.

El objetivo principal del diseño de la KVM fue el de crear una máquina virtual Java lo más completa y pequeña posible que mantuviera los aspectos centrales del lenguaje de programación Java y que sin embargo fuera capaz de correr en un dispositivo restringido con sólo unas pocas decenas o centenas de KBs de memoria disponibles (debido a esto es la K de KVM).

Entrando más en detalle la KVM fue diseñada para:

•Ser pequeña, el core de la máquina virtual debería estar entre unos cuarenta a ochenta KBs (dependiendo de la plataforma destino y de las opciones de configuración)

•Limpia y altamente portable

•Modular y costumizable

•Tan completa y rápida como sea posible, siempre y cuando no se tenga que sacrificar ninguno de los objetivos anteriores.

Si la aplicación Java que se realizó es para correr en una máquina de escritorio esta puede correr desde una línea de comando al igual que una aplicación para J2SE. En dispositivos en donde la interfaz permite iniciar aplicaciones nativas, como los palms OS, el KVM puede ser configurado para correr de esta forma. Para dispositivos que no tienen una interfaz para iniciar aplicaciones nativas KVM provee de una una facilidad denominada Java Application Manager (JAM), el cual cumple la función de ser una interfaz entre el sistema operativo y la máquina virtual. El JAM asume que que las aplicaciones están disponibles para ser descargadas como un jar desde una red o un storage.

KVM esta muy relacionada a CLDC ya que es la única configuración soportada por KVM.

2.1.3.Connected Limited Device Configuration

Como ya se dijo **CLDC** y **KVM** están muy relacionados y no sólo por que KVM sólo soporte esta configuración si no porque KVM es la única implementación de una máquina virtual Java que cumple con las especificaciones de CLDC.

Esta configuración está orientada a dispositivos con restricciones de recursos como pueden ser los teléfonos celulares. Esta especialmente diseñada para cumplir con las necesidades de una plataforma Java para correr en dispositivos con limitaciones de memoria, gráficas y de energía.

Los objetivos principales de esta configuración son:

•Definir una plataforma Java estándar para dispositivos con restricciones de recursos.

•Permitir instalaciones dinámicas de aplicaciones Java y contenido para los dispositivos contemplados en la primer meta.

•Posibilitar que desarrolladores externos puedan crear de forma sencilla aplicaciones y contenido para ser puesto en funcionamiento en los dispositivos ya mencionados.

Como requerimiento primordial se tiene la implementación de CLDC (máquina virtual más bibliotecas) no puede superar los 128 KB. Además la especificación CLDC asume que la aplicación puede correr con 32 KB de Java heap.

La CLDC no tiene soporte para:

•Manejo del ciclo de vida de una aplicación (instalación, ejecución, borrado)

•Interfaz de usuario

•Manejo de eventos

•Modelo de aplicaciones de alto nivel (la interacción del usuario con la aplicación)

Si bien CLDC no soporta los punto mencionados más arriba existen perfiles que implementan algunas de estas limitaciones, estos perfiles dan lugar a un conjunto de APIs de alto nivel. Uno de estos perfiles, el cual es altamente usado, surge de combinar la CLDC con la "Mobile Information Device Profile" (MIDP) para proveer un entorno Java completo para teléfonos celulares y otros dispositivos similares.

Las clases que se mencionan a continuación fueron heredadas directamente desde J2SE, aunque cada una de ellas es un subconjunto de la clase correspondiente en J2SE. sólo los campos y métodos que son apropiados para dispositivos con recursos limitados son especificados en CLDC.

Clases de Sistema (de java.lang):

•Object, Class, Runtime, System, Thread, Runnable, String, StringBuffer, Throwable

Clases DataType (de java.lang):

•Boolean, Byte, Short, Integer, Long, Character

Clases de colecciones (de java.util):

•Vector, Stack, HashTable, Enumeration

Clases de calendario y tiempo (de java.util):

•Calendar, Date, TimeZone

CLDC no soporta java.util.Properties, el cual es parte de la especificación J2SE, sin embargo un conjunto limitado de propiedades que comiencen con la palabra clave "microedition" pueden ser accedidas invocando el método System.getProperty(String Key).

Las bibliotecas de J2SE y J2EE proveen una gran variedad de funcionalidades para el acceso a storages y redes por medio de java.io y java.net. Sin embargo era muy difícil hacer que todas estas funcionalidades entraran en un dispositivo pequeño con sólo unos pocos KBs de memoria. Esto llevo a una generalización de las clases de red e I/O para J2ME.

Todas las conexiones son creadas usando un único método estático de una clase del sistema llamada javax.microedition.Connector. El método toma una parámetro que tiene la siguiente forma:

Connector.open("<protocol>:<address>;<parameters>");

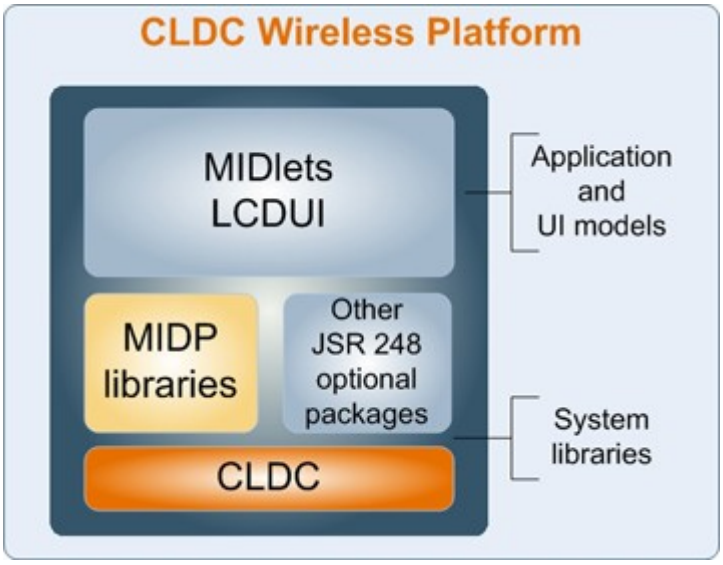

Ilustración 2: Ejemplo de un profile para CLDC

Como ya se mencionó en este documento la primera generación de tecnología Java para dispositivos móviles estaba basada en la KVM, un diseño referencia que demostró como podía ser implementada la especificación CLDC y fue la base del CLDC's Technology Compatibility Kit (TCK). Luego de esto Sun introdujo la **CLDC HotSpot Implementation**, la cual se centraba en la performance. Esta implementación no sólo es más performante que la KVM, cerca de un orden de magnitud en las mismas condiciones, si no que además es más robusta.

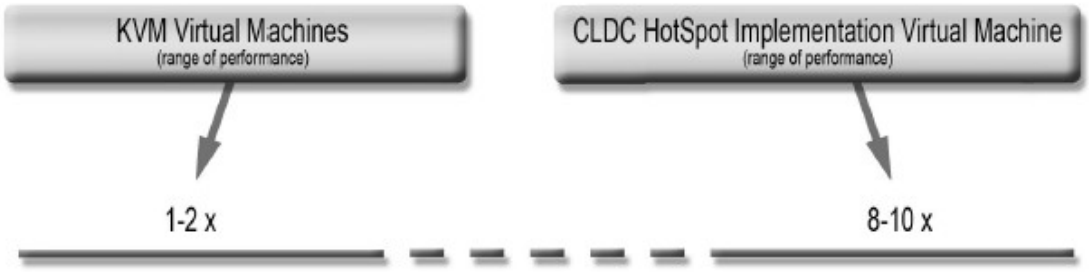

Ilustración 3: Comparación de performance entre KVM y CLDC HotSpot Implementation

Si bien la implementación HotSpot es más performante y robusta esta pensada para la segunda generación de dispositivos móviles, debido a esto es que algunos de los requerimientos que se piden para poder correr esta implementación cambian con respecto a los pedidos para correr la KVM.

Algunos de estos requerimientos son:

Ítem	Mínimo	Típico
Tipo de CPU	Comúnmente ARM	Comúnmente ARM
CPU Speed	50 MHz	50 a 200 MHz
RAM	300 KB (incluyendo MIDP)	> 600 KB (incluyendo MDIP)
ROM/Flash	1Mb	> 1.5 MB

Ilustración 4: Requerimientos KVM

Si bien la implementación puede correr con los requerimientos mínimos es necesaria la optimización de la misma para que lo haga de una forma aceptable.

Otro cambio importante con respecto a la implementación KVM es que esta implementación no tiene limite en la cantidad de clases que se cargan así como en el tamaño del heap.

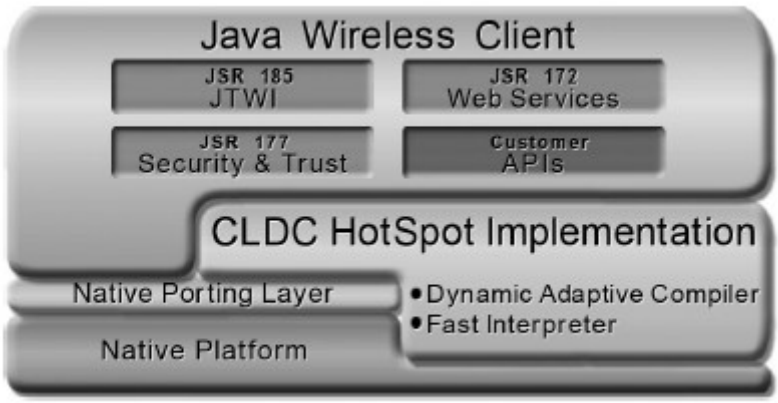

Ilustración 5: CLDC HotSpot Implementation con Sun Java Wireless Client 1.1.2

2.1.4.Connected Device Configuration

La Connected Device Configuration (CDC) esta pensada para dispositivos móviles con más capacidades como pueden ser los PDAs o los Smart Phones.

CDC soporta la especificación completa de la máquina Virtual Java. A nivel de las bibliotecas de clases CDC hace uso de bibliotecas J2SE cuyas interfaces han sido modificadas para de forma que estén optimizadas para dispositivos con poca memoria. El resultado de esto es un runtime Java flexible que puede correr sin problemas con 2 MB de RAM y 2 MB de ROM.

Al igual que para CLDC existen perfiles que extienden la funcionalidades de la especificación. Algunos de los perfiles más relevantes disponibles para CDC son:

- **Fundation Profile:** Es el perfil más básico de CDC, junto con la implementación de CDC provee soporte básico para redes y Entrada/Salida. No provee soporte para gráficos y GUI.

- **Personal Basis Profile:** Provee soporte para construir componentes livianos basados en un conjunto de herramientas GUI limitado basado en Abstract Window Toolkit (AWT)[AWT]. También tiene soporte de runtime para JavaBeans e incluye todas las APIs provistas en el Fundation Profile.

- **Personal Profile:** Brinda un soporte para AWT completo y limitado para beans. También incluye todas las APIs del Personal Basis Profile.

Aparte de los perfiles CDC permite incluir paquetes opcionales, lo cuales agregan más funcionalidades. Algunos de los paquetes opcionales son los siguientes:

- **RMI:** Subconjunto de Java SE RMI.

- **JDBC:** Subconjunto de la versión 3.0 de la API de JDBC.

- **AGUI:** Provee una implementación modificada de Swing con el propósito de proveer rich GUIs y tener soporte para Java 2D. Este paquete está basado en los perfiles Personal Basis Profile y en el Personal Profile.

- **Security:** Brinda un framework basado en Java SE que incluye SSL, criptografía, autenticación y autorización.

- **Web Services:** Da la posibilidad de que clientes Java ME accedan a Web Services.

El runtime de CDC esta compuesto por una unión de todo lo explicado en este punto, o sea una configuración, un perfil y un número de paquetes opcionales.

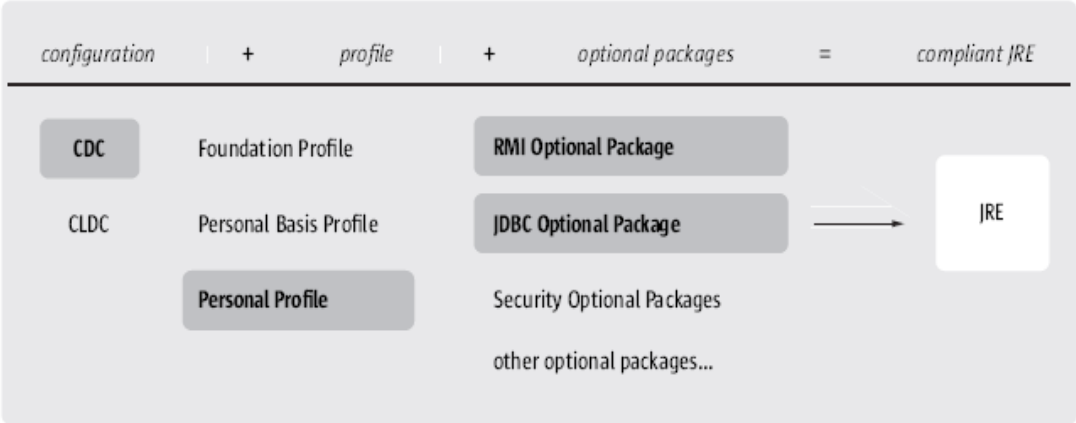

Ilustración 6: Ejemplo de runtime CDC

Al igual que para CLDC se cuenta con una implementación HotSpot la cual esta optimizada y es más performante.

2.2. Sistemas operativos en dispositivos móviles

Luego de realizar algunas búsquedas en la web y basándonos en la experiencia laboral de los integrantes del proyecto se decidió que los sistemas operativos que se estudiaran serían los siguientes:

- **Symbian OS [SOS]**
- **Android [AND]**
- **IPhone OS [IOS]**
- **Windows Mobile [WMOB]**

Lo que respalda esta selección es que estos sistemas operativos cubren la gran mayoría de dispositivos móviles y abarcan productos que han ido evolucionando desde un largo tiempo, como también productos nuevos e innovadores tal es el caso de Android.

Algunos de los datos que respaldan estas decisiones se pueden apreciar en las siguientes ilustraciones, pudiéndose acceder a las fuentes de esta información mediante la referencia [MMRK] que se encuentra en el capítulo final del documento principal.

Worldwide smart phone market
Market shares Q3 2008, Q3 2007

OS vendor	Q3 2008 shipments	% share	Q3 2007 shipments	% share	Growth Q3'08/Q3'07
Total	**39,850,100**	**100.0%**	**31,156,240**	**100.0%**	**27.9%**
Symbian	18,583,060	46.6%	21,219,390	68.1%	-12.4%
Apple	6,899,010	17.3%	1,107,460	3.6%	523.0%
RIM	6,051,730	15.2%	3,298,090	10.6%	83.5%
Microsoft	5,425,470	13.6%	3,797,360	12.2%	42.9%
Linux	2,028,490	5.1%	1,361,810	4.4%	49.0%
Others	862,340	2.2%	372,130	1.2%	131.7%

Source: Canalys estimates, © canalys.com ltd. 2008

Ilustración 7: Distribución de los Sistemas operativos en los terminales móviles

Ilustración 8: Distribución de los Sistemas operativos según el acceso a internet desde terminales móviles

La ilustración 7 fue tomada de la página web http://www.canalys.com y corresponde a la comparación de las ventas correspondientes a los periodos Julio-Septiembre 2007 y Julio-Septiembre 2008, mientras que la ilustración 8 fue tomada de la web http://marketshare.hitslink.com y muestra el porcentaje que posee cada sistema operativo en el acceso a la web desde dispositivos móviles.

También se realizó un estudio sobre los modelos de celulares que se encuentran en el mercado a partir información que fue extraída de la página WURFL(Wireless Universal Resource File) [WURFL]. En esta página se puede encontrar un archivo xml el cual contiene las especificaciones de los distintos modelos de celulares. De aquí se pudo extraer que la mayoría de lo modelos de celulares hacen uso de Symbian OS (50% aproximadamente) seguido muy de lejos por Windows Mobile OS (en el entorno del 10%).

Algunos de los sistemas que no se contemplaron fueron: RIM [RIM] (usado en dispositivos BlackBerry), PALM OS[PALMOS], BREW[BREW] (usado en Skypephones) y SavaJE[SAJE] (basado en JavaME).

2.2.1.Symbian OS

Symbian es la compañía que se encuentra detrás del desarrollo de Symbian OS el cual es usado por 14 empresas que desarrollan dispositivos móviles. Muchas de estas empresas son acreedoras de Symbian pero es Nokia quien a partir de Diciembre de 2008 se hace el acreedor mayoritario con el fin de crear una organización sin fines de lucros denominada Symbian Fundation

Se espera que Symbian Fundation comience a funcionar durante la primer mitad del 2009. La plataforma Symbian Fundation estará disponible a los miembros de la misma mediante una licencia "royalty-free". La fundación estará a cargo de proporcionar, gestionar y unificar la plataforma para sus miembros. La fundación también se compromete a evolucionar la plataforma durante los próximos dos años con la intención de utilizar la licencia "Eclipse Public Licence"[EPL]. Esto hará que la plataforma quede a disposición de todos de forma gratuita y permita una participación más amplia de la comunidad.

Symbian OS está basado en Psion EPOC OS y fue diseñado sobre tres reglas fundamentales:

•La integridad y seguridad de los datos del usuario

•No se debe de malgastar el tiempo del usuario

•Todos los recursos son escasos.

Tanto las aplicaciones como el propio sistema sigue un diseño MVC aunque la presión del mercado, la introducción de un kernel con soporte para real time y la plataforma de seguridad hizo que este modelo se fuera diluyendo de a poco en las nuevas versiones del sistema operativo.

Symbian OS v9 tiene el siguiente soporte para Java ME:

•CLDC 1.1 con compatibilidad para la especificación CLDC 1.0 y con algunas mejoras que permiten dar soporte a operaciones de punto flotante.

•MIDP 2.0 con compatibilidad para la versión MIDP 1.0 y con nuevas características como ser multimedia, funcionalidades para juegos, rich GUIs, conectividad extensa y un modelo de seguridad.

•Una gran variedad de APIs disponibles que permiten manejo de aplicaciones y configuraciones tales como WMA, Bluetooth, gráficos 3D y multimedia avanzada.

•Posibilidad de realizar debug en el dispositivo móvil.

2.2.2.Android

El grupo Open Handset Alliance, formado por más de 30 empresas vinculadas a la tecnología y móviles, ha desarrollado Android. El cual es la primera plataforma móvil completamente abierta y gratuita.

Android es un stack de software para dispositivos móviles que incluye un sistema operativo, middleware y aplicaciones claves.

Los desarrolladores pueden crear aplicaciones para la plataforma usando la SDK de Android, la cual esta en versión Beta, haciendo uso del lenguaje de programación Java. Las aplicaciones corren sobre Dalvik[DVIK], que es una máquina virtual diseñada para correr de forma embebida, la cual a su vez corre sobre un kernel de Linux versión 2.6.

Características principales de Android:

•Framework de aplicación, permite re-uso y remplazo de componentes.

•Máquina virtual Dalvik optimizada para dispositivos móviles

•Gráficos optimizados

•SQLite

•Soporte multimedia

•Telefonía GSM (dependiente del hardware)

•Bluetooth, EDGE, 3G y WiFi (dependiente del hardware)

La siguiente ilustración muestra la arquitectura de esta plataforma.

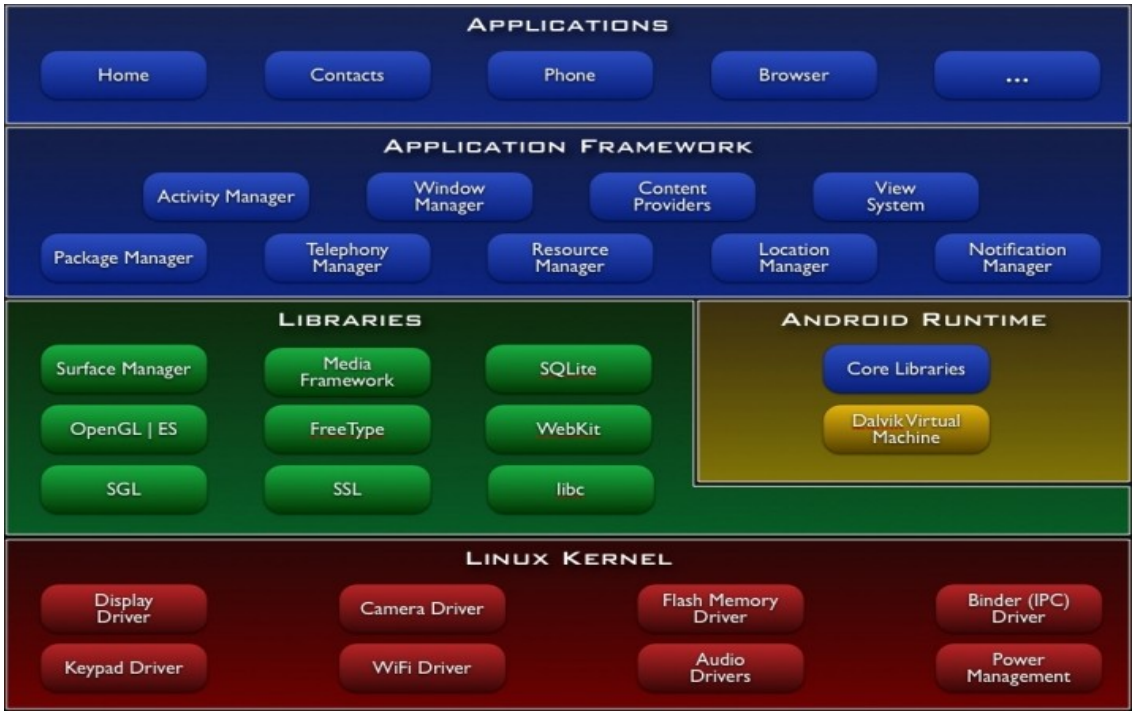

Ilustración 9: Arquitectura de Android

Android no tiene un soporte completo para JAVA ME ni para Java SE, tiene un subconjunto limitado de los principales paquetes de Java, por ejemplo los paquetes java.io, java.util y java.lang son incluidos. Uno de los paquetes más relevantes que no se incluyen son los relacionados Swing o AWT, en su lugar hay un paquete denominado GWT [GWT] el cual es un proyecto que es llevado acabo por Google.

2.2.3.IPhone OS

IPhone OS incluye el sistema operativo y las tecnologías que se utilizan para correr aplicaciones de forma nativa en el IPhone y en el IPod. Aunque el IPhone OS está basado en el Mac OS X, el kernel del IPhone OS está basado en una variante del kernel de Mac OS X, el primero fue diseñado para satisfacer las necesidades de un entorno móvil.

El sistema tiene 4 capas de abstracción, la capa de núcleo del sistema operativo, la capa de servicios principales, la capa de medios de comunicación y la capa Cocoa Touch.

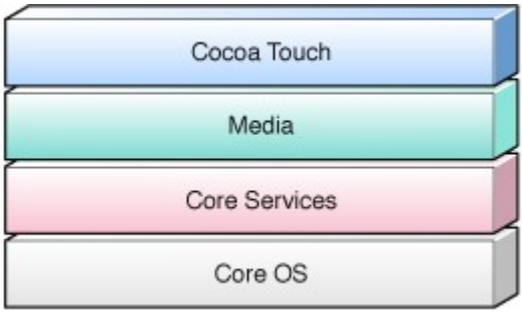

Ilustración 10: Capas del IPhone OS

Cuanto más arriba en las capas más avanzada es la tecnología que se puede encontrar.

El IPhone OS no viene con Java integrado y Sun esta desarrollando una JVM [JVPOS] para el mismo de forma que esta este disponible para ser descarga como software de terceros desde el Apple Store, aunque al momento de realizar este documento la misma aún no se puede encontrar en la tienda.

También se puede encontrar en la web una discusión [JVPOS] en donde se afirma que la máquina virtual sólo podrá estar disponible si Apple lo permite, ya que en el acuerdo de conformidad que se acepta a la hora de desarrollar aplicaciones para el IPhone establece que no se puede correr código de terceros desde una aplicación. O sea que una aplicación Java no podría correr código de la máquina virtual. En conclusión tecnológicamente puede ser posible incluir una JVM en un IPhone OS, pero realizar el deploy de una aplicación infringiría los acuerdos aceptados para realizar aplicaciones para el IPhone OS.

2.2.4.Windows Mobile

Por más que se realizó una búsqueda intensa para encontrar información la mayoría de esta es comercial. No se pudo encontrar una página oficial en donde se explicara como está conformada la arquitectura de este dispositivo o bien si está basada en alguna de las versiones de escritorio de Windows.

Si se encontró información sobre las herramientas de desarrollo disponibles y especificaciones de como desarrollar aplicaciones para este sistema operativo[MSDN]. Las cuales se estudiaron en busca de poder extraer información relevante para este estudio. De esta búsqueda es que surgió que Windows ME posee soporte para la especificación CLDC de Java ME.

Dentro de los blogs oficiales que se encuentran auspiciados por Microsoft y en donde escribe principalmente mucho de su personal se encontraron desarrollos que vinculan a Windows ME con la especificación CDC, pero siempre realizando recompilaciones del código.

2.3.Pcap en dispositivos móviles

Luego de ver los principales Sistemas Operativos para dispositivos móviles pasamos a investigar el soporte que estos tienen para poder tener acceso al stack de mensajes que intercambia el dispositivo con la red a la cual esta vinculado. En otras palabras ver si existía una alternativa a Pcap dentro de eso sistemas operativos.

Pcap existe como libpcap para sistemas operativos basados en Unix o Linux y como WinPcap para las distintas versiones de Windows.

Para Windows Mobile existe una versión experimental disponible para descargar en la página de WinPcap. Como toda versión experimental tiene sus limitantes, algunas de las cuales se enumeran a continuación:

- Solo se puede abrir un adaptador y sólo se puede usar una instancia del mismo en cualquier momento de tiempo.
- Solo soporta un request a la vez
- Los request no son encolados
- Es desarrollado por Compaq Pocket PC y HP Jornada, debido a esto WinPcap no da ningún tipo de soporte sobre este producto, por más que este disponible su descarga desde la página http://winpcap.org.

Symbian no está basado ni en Windows, Unix o Linux sin embargo el emulador puede hacer uso de WinPcap 3.0 para poder tener conexión a internet por medio de una Ethernet. Esto sólo sirve para el emulador y no se pudo encontrar información sobre realizar la misma tarea dentro del propio sistema operativo que está en el dispositivo móvil. Si existe la posibilidad de escribir módulos y plugins los cuales tienen acceso a los mensajes de la red del celular.

El sistema operativo IPhone OS puede hacer uso de la biblioteca libpcap, debido a que no difiere en nada de como es usada en cualquier sistema Unix o Linux. De todas formas existe información disponible dentro de sitio de desarrolladores de Apple.

Como ya se mencionó Android se encuentra sobre un kernel Linux y dentro de las bibliotecas que se le pueden instalar a este se encuentra la biblioteca libpcap, con lo cual no es necesaria buscar una alternativa al Pcap ya que soporta al mismo.

3.Conclusiones

Luego de realizada la investigación sobre Java ME, los sistemas operativos para dispositivos móviles y las posibles alternativas a Pcap se debería de poder afirmar si el proyecto IP4JVM funciona o no dentro de un dispositivo móvil. Pero se esta lejos de responder esta pregunta con certeza, ya sea para decir NO o decir SI.

Porque no se puede responder con certeza se debe a que son muchos los puntos que hay que decir "si" o decir "no" para poder llegar a una afirmación final y en algunos de estos puntos no es posible responder de una forma u otra, si no que la respuesta más apropiada sería un quizás. En las siguientes lineas se presentarán estos puntos.

Lo primero y más relevante a responder es ¿Java ME soporta el proyecto IP4JVM? En este caso la respuesta es sencilla y es un SI, pero tiene una condicionante y es que no puede ser cualquier implementación de Java ME si no que debe de ser una implementación que soporte la especificación CDC ya que la especificación CLDC tiene demasiadas limitantes con respecto al proyecto. Algunas de estas limitaciones, las más relevantes, son: no tener la clase Timer, no poder hacer uso de archivos properties y un tamaño de stack limitado. Aun si se tuviera un timer y se pudiera hacer uso de los properties se tendría que realizar una recodificación de algunas clases debido al uso de colecciones tipadas y de que no todas las estructuras de datos están soportadas, si no que sólo las más simples.

Ahora que podemos decir que Java ME soporta a IP4JVM debemos respondernos si los sistemas operativos dentro de los dispositivos móviles soportan Java ME en su especificación CDC. Esta respuesta no puede ser respondida de forma absoluta ya que se realizó el estudio de sólo 4 de los muchos sistemas operativos móviles existentes pero si se puede decir indicar si la mayoría de los dispositivos móviles soportan la mencionada especificación. Para el sistema Symbian OS responder esta pregunta es sencillo ya que en la especificación del mismo esta muy claro que la versión de Java ME que soporta es la CLCD, con lo cual la respuesta seria un NO. Debido a que Symbian es el sistema operativo que se encuentra en la mayoría de los dispositivos móviles ya podemos responder que la mayoría de estos dispositivos no pueden correr el proyecto IP4JVM, pero aún falta responder si algún dispositivo móvil puede hacerlo. Al igual que para Symbian podemos decir que el IPhone OS tampoco soporta Java ME, es más no tiene soporte alguno para Java. También está Windows Mobile el cual trae de forma nativa soporte para Java ME CLDC pero puede ser recompilado para dar soporte a Java CDC con lo cual en este caso podríamos decir que existe la posibilidad de poder poner una versión del proyecto dentro de este sistema operativo. Por último nos queda Android un sistema operativo basado totalmente en Java, este sistema tiene un soporte para Java ME CDC pero no de forma completa ya que no incluye los paquetes de AWT y Swing. Debido a que estos paquetes no son usados por el

proyecto IP4JVM se puede decir que el mismo funcionaría dentro de la máquina virtual Java disponible en Android.

Por último faltaría responder si dentro de los sistemas operativos ya mencionados existe o no la posibilidad de interactuar con el stack de paquetes de red. En Symbian por más que ya se sabía que no era posible correr el proyecto se buscaron alternativas al Pcap pero no fueron encontradas. Lo mismo se hizo para el IPhone OS con resultados más auspiciosos que para el Symbian ya que como el primero esta construido sobre un sistema Unix el mismo puede hacer uso de la biblioteca libpcap. Lo mismo sucede para Android que se encuentra construido sobre un kernel de Linux y algo similar para Windows Mobile el cual puede hacer uso de una versión experimental del WinPcap.

En caso de que algún dispositivo móvil cumpliera con la necesidades tecnológicas este debería disponer de 1 MB de memoria para poder alojar el proyecto IP4JVM (este ocupa 968k) y de 14 MB de memoria RAM que es lo que ocupa este al entrar en funcionamiento.

Luego de analizadas y respondidas las preguntas que siguieron podemos concluir lo siguiente:

- Los dispositivos móviles cuyo sistema operativo sea Symbian no pueden correr el proyecto IP4JVM,
- Mientras no exista una JVM para el IPhone OS este tampoco podrá hacer uso del proyecto.
- Realizando algunas modificaciones sobre Windows Mobile y haciendo uso de herramientas experimentales se puede hacer correr el proyecto sobre dispositivos que cuenten con este sistema.
- Android es un proyecto opensource con lo cual si se pudiera modificar la implementación de la máquina virtual de forma similar a como se hizo con la OpenJDK se podría llegar a hacer funcionar el proyecto en dispositivos móviles con este sistema.

4.Índice de ilustraciones