Mecanismos de invocación de métodos remotos en Android

N.D. Watkinson Medina, M.A. Piccini Méndez, L. Beltrán Rocha y A. Díaz Ramírez

Resumen: Las aplicaciones modernas requieren comúnmente de la ejecución de métodos o procedimientos que se encuentran en computadoras remotas. Este requerimiento es aún más importante en las aplicaciones para dispositivos móviles. En este artículo se estudian algunos de los mecanismos más importantes para la invocación de métodos remotos. Además, se presentan algunas de las opciones existentes para el sistema operativo *Android*, que ha sido diseñado para utilizarse en dispositivos móviles, y se presenta un comparativo de su desempeño. Las herramientas que se analizarán son los siguientes: *xml-rpc*, *ksoap2* y *json-rpc*.

Abstract: Modern applications require very often the invocation of remote methods. This requirement is even more important when applications execute on mobile devices. In this paper, we study some of the existing and most important mechanisms for remote method invocation. Moreover, we present some of the tools available for the Android operating system, that has been designed to be used on mobile devices, and we present a comparative analysis of their performance. The tools that are going to be analyzed are: *xml-rpc*, *ksoap2* and *jsonrpc*.

Introducción

Afiliación:

Neftalí David Watkinson Medina
Cetys Universidad, campus Ensenada.
Km1. Camino Microondas, Ensenada, Baja California, México
neftaliw@hotmail.com

Miguel Ángel Piccini Méndez
Cetys Universidad, campus Ensenada.
Km1. Camino Microondas, Ensenada, Baja California, México.
piccini_map@hotmail.com

Lucía Beltrán Rocha
Cetys Universidad, campus Ensenada.
Km1. Camino Microondas, Ensenada, Baja California, México
lucia.beltran@cetys.mx

Arnoldo Díaz Ramírez
Instituto Tecnológico de Mexicali.
Departamento de Sistemas y Computación.
Av, Tecnológico s/n, Col. Elías Calles, Mexicali, B.C., México.
adiaz@itmexicali.edu.mx

Las aplicaciones modernas que forman parte de un sistema distribuido, deben tener la capacidad comunicarse de manera efectiva con otras que se ejecuten en computadoras remotas. Las aplicaciones pueden estar desarrolladas en diferentes lenguajes de programación, arquitecturas de hardware y sistemas operativos, y aún así requerir ser integradas. Por otra parte, la proliferación de dispositivos móviles ha acentuado la necesidad de mecanismos que permitan la interacción de aplicaciones heterogéneas, debido en parte a la limitación de recursos a las que están sujetas.

En la actualidad, una gran cantidad de alternativas han sido propuestas. Una de las primeras son los *Llamados a Procedimientos Remotos* o *RPC* (*Remote Procedure Call*) [1]. Un RPC es una tecnología diseñada para la comunicación entre procesos, que permite que un programa computacional pueda invocar métodos o procedimientos que se encuentran en otras computadoras, de una manera similar en la que se invocan los métodos locales. Con este mecanismo es posible aprovechar de mejor manera los recursos computacionales disponibles en un sistema distribuido, así como permitir la cooperación entre diversas aplicaciones. El modelo de comunicación utilizado en los RPCs ha servido como base para el desarrollo de mejores y más modernos mecanismos de comunicación entre procesos. En este artículo se presentan algunos de los más importantes propuestos a la fecha.

Además, en este documento se presentan algunas de herramientas para la comunicación entre procesos remotos disponibles en el sistema operativo Android. La razón por la cual se eligió Android es porque es un sistema operativo novedoso y que ha sido desarrollado principalmente para dispositivos móviles de última generación, y cuenta con características que lo hacen atractivo para la implementación de aplicaciones ubicuas [2].

El artículo comienza con una breve descripción de los RPCs y presenta dos de los más utilizados estándares basados en ellos: CORBA y XML-RPC. Posteriormente se describe lo que son los servicios Web y se explican las principales diferencias entre servicios Web basados en SOAP y servicios Web RESTful. A continuación se analiza brevemente JSON RPC. El artículo continúa con una breve descripción de las características más importantes del sistema operativo Andorid, y presenta los resultados de un caso práctico en el que se discute el desempeño de tres herramientas

de invocación de métodos remotos disponibles para Android: XML-RPC, JSON RPC y kSOAP. Finalmente, se presentan las conclusiones y el trabajo futuro.

Llamado a Procedimientos Remotos (RPC)

Los llamados a procedimientos remotos, o RPC, por sus siglas en inglés (*Remote Procedure Call*), son herramientas para la comunicación entre procesos que permiten que un programa computacional solicite operaciones o métodos desde una computadora externa. Por ejemplo, un programa desarrollado en algún lenguaje antiguo (*legacy*) puede requerir del acceso a información almacenada en una base de datos Oracle, para la cual no existen conectores definidos para dicho lenguaje. Sin embargo, es posible que el programa pueda acceder a los datos utilizando algún mecanismo RPC, ya que el código de acceso a la información puede ser procesado por un método que se ejecute en otra computadora y desarrollado en un lenguaje moderno. En este ejemplo, el RPC viene siendo el proveedor, que funciona de mediador entre los que generan el producto (Oracle) y quienes lo necesitan (programa *legacy*).
Los RPC logran su cometido estableciendo estándares para la comunicación entre procesos. Esto se hace generando formatos intermedios que traducen los procesos disponibles del servidor y los pedidos del cliente [1].
Existen una infinidad de RPCs para todo tipo de aplicaciones; la mayoría comunican procesos del mismo ambiente. Sin embargo, existen RPCs que comunican procesos que fueron desarrollados en ambientes y lenguajes diferentes. El ejemplo más claro es CORBA.

CORBA

CORBA es un acrónimo de *Common Object Request Broker Architecture*, y es un estándar definido por el *Object Management Group*. CORBA proporciona un mecanismo en software que normaliza el llamado de métodos entre procesos de distintos lenguajes [3]. Esto se consigue utilizando códigos *stub* tanto en el cliente como en el proveedor del método, de tal manera que puedan comunicarse al mismo tiempo que liberan al programador de muchos de los detalles de implementación. Con CORBA es posible que un proceso escrito en C, por ejemplo, pueda invocar a un método desarrollado en Java. Para facilitar esta interacción, el código generado por CORBA realiza la traducción necesaria para que ambos procesos puedan comunicarse, como puede observarse en la Fig. 1.

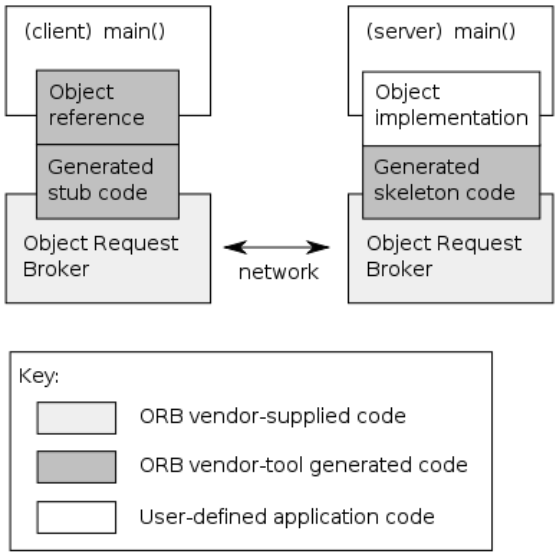

Fig. 1.- Diagrama del funcionamiento de CORBA

Este tipo de funcionamiento es utilizado por las implementaciones RPC existentes.

XML-RPC

XML-RPC es un protocolo de llamado a procedimientos remotos que fue desarrollado por Dave Winer y Microsoft en 1998.
El funcionamiento de XML-RPC consiste en enviar solicitudes dirigidos a la computadora que en la que se ejecuta el servicio o procedimiento. Una vez que la solicitud se recibe, se ejecuta y se envía al proceso solicitante el resultado. Las solicitudes y respuestas se envían en formato XML y utilizando el protocolo HTTP, como puede observarse en la Fig. 2. Se pueden transportar datos interrelacionados e incluso enviar objetos o estructuras como parámetros de entrada y salida.
Las ventajas de XML-RPC sobre otros mecanismos similares, como los basados en SOAP (descrito más adelante) es que permite solo un método de *serialización*, lo que lo hace mucho más sencillo de utilizar. Además, XML-RPC tiene un modelo de seguridad más simple y no requiere la creación de descripciones WSDL (Web Services Description Language) [5].

```
<?xml version="1.0"?>
<methodCall>

<methodName>examples.getStateName</metho
dName>
  <params>
    <param>
        <value><i4>40</i4></value>
    </param>
  </params>
</methodCall>

Respuesta:

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
        <value><string>South
Dakota</string></value>
    </param>
  </params>
</methodResponse>
```

Fig. 2 Ejemplo del llamado a un método en XML RPC

Como se puede observar en el ejemplo de la Fig. 3, se invoca al método ***getStateName*** utilizando la etiqueta *MethodName*. La respuesta será enviada también utilizando XML con la etiqueta *methodResponse*, que en ejemplo es una variable del tipo *string*.

La principal crítica a este protocolo es que no se considera que agregue mucho a los mecanismos ya existentes para RPCs, como por ejemplo, DCE/RPC [7] o CORBA, a excepción de la utilización de XML. Muchos de los procedimientos son solo modificaciones de elementos que ya contiene el lenguaje original [6].

SOAP

El protocolo de acceso simple a objetos (SOAP por sus siglas en inglés) especifica el intercambio de información estructurada en las redes de computadoras. Es el sucesor de XML-RPC, aunque SOAP en sí no es un RPC, sino un protocolo de comunicación basado en XML que puede ser utilizado junto con el HTTP para la transmisión de mensajes que contengan parámetros y resultados [7].
Un mensaje SOAP consiste de tres partes: un sobre (*envelope*) que especifica qué tipo de mensaje es y como procesarlo; una serie de reglas para codificar la expresión de variables definidas por la aplicación (*header*); y una convención para representar métodos y llamado de procedimientos (*body*), como se muestra en la Fig. 3.

SOAP mantiene muchas ventajas sobre su predecesor, siendo la principal su versatilidad y la posibilidad de adaptarse a cortafuegos (*firewalls*) por el uso del modelo de respuesta de HTTP [8].

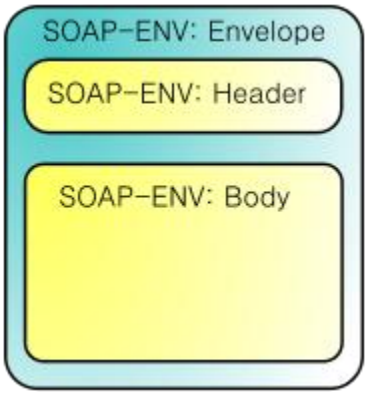

Fig. 3.- Estructura de un mensaje bajo SOAP

SOAP también presenta algunas desventajas. Por ejemplo, debido que uno de sus objetivos es el de permitir la interacción entre aplicaciones con bajo acoplamiento, es menos sencillo de utilizar. Por otra parte, la utilización de XML introduce una capa adicional de procesamiento, que a pesar de ser transparente al usuario, provoca una mayor sobrecarga en comparación con otras herramientas similares como CORBA. Además, el uso inapropiado de HTTP saturando las vías de comunicación o viajando a través de cortafuegos puede comprometer la seguridad.

Servicios Web

Un servicio Web es una aplicación independiente, disponible a través de una red, que puede ser descubierta e invocada automáticamente, utilizando protocolos basados en XML [8].
Android está orientado a servicios Web, los cuales normalmente son implementados por medio de alguna Interfaz de Programación de Aplicación o API (*Application Program Interface*), que permiten la creación de servicios a los que se puede acceder a través del protocolo HTTP [4].
Generalmente, los servicios Web pueden clasificarse en dos tipos. Los primeros son conocidos como *Big Web Services*, los cuales utilizan SOAP como protocolo de comunicación. Estos servicios requieren que exista una descripción de los servicios disponibles. La descripción de los servicios se hace utilizando WSDL, que es un lenguaje basado en XML y que permite describir, además de los servicios disponibles, los detalles técnicos para utilizarlos. Por lo tanto existe una relación

triangular entre el que hace la petición del servicio, el proveedor y un mediador, como se muestra en la Fig. 4.
El segundo tipo de servicios Web son los que utilizan el conjunto de restricciones conocido como REST (Estado Representado de Transferencia) [9], el cual se introduce como una alternativa a los servicios basados en SOAP. La utilización de esta arquitectura permite la combinación de muchos servicios Web en una misma aplicación, además de que no requiere definiciones XML, SOAP o WSDL. Si una implementación de servicios Web cumple con las restricciones REST, se dice que es un servicio Web de tipo RESTful.
Al utilizar REST, es común que las respuestas a las solicitudes de servicio se envíen utilizando XML, o la Notación de Objetos de *JavaScript* (JSON), o algo similar a XML.

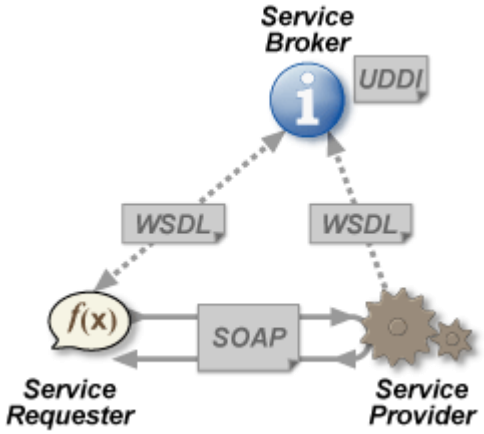

Fig. 4.- Estructura general de un servicio Web

Un problema común del uso de REST consiste en que cuando se invocan servicios Web complejos, esto es, servicios que a su vez invocan a otros servicios, el usuario no tiene control sobre éstos, lo que los hace vulnerables a fallar.
La implementación de un servicio Web variará por el mecanismo utilizado. Los tres principales mecanismos o estilos son RPC, SOAP y REST, como hemos visto. A continuación se describirán algunos elementos de su uso.

RPC

Los servicios Web que utilizan el estilo de los *llamados a procedimientos remotos* o RPCs están centrados en un documento en el que se definen los métodos o procedimientos que estarán disponibles en la red. Esta definición se hace utilizando algún lenguaje neutral, que en algunos casos se llama *Interface Definition Language* o IDL [7]. Con este documento de definición es posible generar el código de los *stubs* de clientes y proveedores, de tal manera que su construcción sea más sencilla y no tenga que incluir aspectos relacionados con la *serialización* de los datos o los detalles inherentes a la comunicación, como lo son *sockets*, *puertos*, entre otros. Ejemplos de algunas opciones parar utilizar este estilo de comunicación son CORBA, Java RMI o XML-RPC, que fueron utilizados para implementar los primeros servicios Web. Sin embargo, han sido criticados por su falta de flexibilidad y de presentar la clase y no solo sus métodos.

SOAP

Los servicios Web basados en SOAP han sido considerados como una alternativa para la implementación de la *Arquitectura Orientada a Servicios* (SOA), que a diferencia de los RPCs, se centra en el mensaje en lugar de la operación. Una característica que distingue a los servicios Web basados en SOAP es que definen de manera inequívoca un **contrato**, en el que se establecen los servicios disponibles y sus características, así como los detalles técnicos para invocarlos. El contrato se define utilizando el protocolo WSDL.
El diseño de servicios Web basados en SOAP presenta un par de variantes. Una de ellas es llamada *primero el código* (*code-first*), en el que se desarrolla primero el código de los servicios y posteriormente, con el uso de alguna herramienta, se genera de manera automática el contrato o documento WSDL. Por otra parte, si se utiliza el *método primero el contrato* (*contract-first*), se debe elaborar en primera instancia el documento WSDL y a partir de éste se crean automáticamente los esqueletos de las clases con una herramienta generadora de código.

REST

Los servicios Web que utilizan REST se benefician de la utilización de los métodos definidos para el protocolo HTTP, que son GET, POST, PUT y DELETE. Cada uno de estos métodos está relacionado con alguna de las operaciones CRUD: Crear, Obtener, Actualizar y Eliminar. De esta manera no es necesaria la existencia de un contrato en el que se especifiquen los métodos o servicios disponibles, pues los únicos servicios que pueden ser solicitados son los definidos en las operaciones CRUD.
Esto permite una rápida y más sencilla comunicación entre clientes y proveedores informáticos ya que el conjunto de operaciones disponibles se reduce al mínimo [10].

Comparación de los estilos.

La principal crítica de la utilización de servicios Web basados en SOAP es que suelen ser muy complejos y se enfocan en las necesidades de las grandes corporaciones en lugar de las pequeñas implementaciones con estándares abiertos. Sin embargo, los servicios que utilizan REST tienen su debilidad en la sencillez de elaboración, ya que los vuelve poco flexibles a cambios y un servidor específico requerirá de un cliente específico, haciendo muy costoso el estar actualizando a grandes escalas [11]. Determinar cuál de las alternativas para la implementación de servicios Web es mejor depende en gran medida del tipo de aplicación. Es un hecho que existe una mayor cantidad de servicios Web disponibles basados en SOAP. Sin embargo, recientemente se ha observado un incremento en el uso de servicios Web del tipo RESTful.

JSON RPC

El protocolo JSON RPC está basado en el lenguaje JSON (*Java Script Object Notation*), el cual es un estándar ligero y basado en texto, diseñado para el intercambio de datos legibles por humanos. En su nombre se obvia la similitud que tiene con la sintaxis de Java además de las variables que utiliza. JSON es un lenguaje equivalente al XML.
El protocolo JSON RPC es muy simple y se asemeja al de XML-RPC. Sin embargo permite la comunicación bidireccional entre el servicio y el cliente tratándolos como iguales y permitiendo una mayor riqueza en la interactividad. También permite el uso de llamadas múltiples que pueden ser atendidas fuera de orden.
Además de utilizar el protocolo HTTP como medio de transporte, también puede usar *sockets* o conexiones TCP/IP lo que permite mayor flexibilidad. Sin embargo, el que la mayoría de los cortafuegos en Internet permitan el paso de mensajes HTTP lo convierte en la opción más utilizada.

Android

Android es el nombre que recibe el sistema operativo de código abierto de reciente creación y que es uno de los más utilizados en el mercado de la telefonía celular. Andorid compite con otros sistemas similares comerciales como *IPhone OS*, que es el líder del mercado, y *Symbian*, entre otros.
Android fue presentado en el año 2008 por la empresa con el mismo nombre. Actualmente pertenece a la *Open Handset Alliance* y sus principales impulsadores son *Google*, *HTC*, *Samsung* y *Texas Instruments*, por mencionar algunos [4].

Historia.

Andorid está desarrollado en lenguaje C y se distribuye bajo la licencia Apache 2.0. Es un sistema operativo de código abierto orientado a dispositivos móviles. Actualmente se encuentra en su versión 2.1, llamada *Eclair*.

Características.

El núcleo (*kernel*) de Android es una versión modificada de Linux, y como tal, es un sistema operativo de código abierto, gratuito y que puede ser distribuido y modificado libremente.
El sistema operativo está dirigido a soportar varias aplicaciones orientadas a servicios Web. Estas aplicaciones constan de dos partes: una programada en un lenguaje modificado de Java, y otra en XML.
Ya que Android trata cada aplicación en un ambiente virtual independiente de cualquier otra aplicación, cuenta con un RPC interno que permite la comunicación entre aplicaciones. Este trabaja con una interfaz llamada *IBinder*, la cual funciona como mediadora entre la aplicación cliente, y la aplicación servidor, como puede observarse en la Fig. 5.

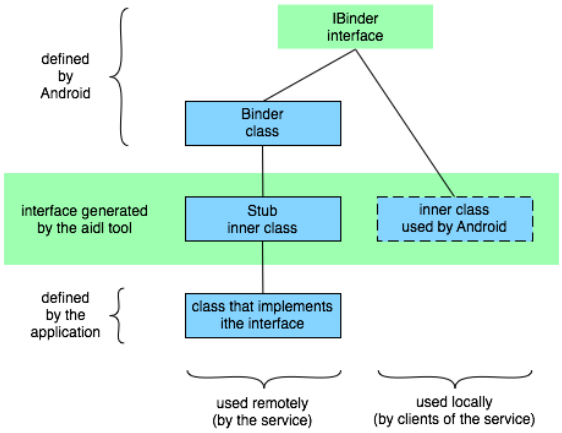

Fig. 5.- Funcionamiento de RPC y la clase *IBinder* en Android

Por medio de la herramienta *Android Interface Definition Language* (AIDL), que está incluida en Android, es posible generar el código *stub* que facilita la comunicación entre el cliente y el servidor, y define la forma en que ambos se van a comunicar, como se explicó anteriormente.

Android no tiene herramientas nativas para ninguno de los protocolos mencionados en este documento, a excepción del mecanismo RPC interno mencionado anteriormente. Tampoco existe suficiente

documentación relacionada con este sistema operativo, en parte debido a que ha sido creado recientemente. Sin embargo, a pesar de que no existen muchas herramientas para la invocación de métodos remotos, las existentes son interesantes y merecen ser revisadas. Entre éstas, destacan Android - JSON RPC, Android – XML-RPC y kSOAP (versión portátil de SOAP).

Caso práctico: Evaluación de resultados de JSON RPC, XML-RPC y kSOAP

Para el caso práctico se elaboró un programa en los que el cliente (Android) solicitaría a un proceso remoto que devolviera el mensaje "*Hola mundo*" y además de realizar la suma de dos números que se le envían. Antes de ejecutar los programas, se hace notar que JSON RPC es el que cuenta con mayor documentación de los tres, además de ser el que cuenta con mayor número de herramientas, como por ejemplo para conectarse a una base de datos MySQL. Por otra parte, es importante mencionar que kSOAP está orientado a .NET principalmente. En cuanto a la facilidad de comprensión del código, XML-RPC y JSON RPC son los más sencillos ya que su estructura es muy similar al lenguaje de programación Java, del cual está derivado el lenguaje de Android. Por otra parte, al comparar las herramientas de desarrollo o IDE (*Interface Development Environment*) encontramos que kSOAP es el más pesado de los tres y puede alargar el tiempo de arranque hasta el doble que aquellas utilizadas para desarrollar aplicaciones XML-RPC o JSON RPC. Sin embargo, una vez que el programa está en ejecución, ninguno de estos factores afecta su desempeño.
JSON RPC y XML-RPC dejan al servidor vulnerable a ataques externos ya que por la falta de un mecanismo de seguridad, ya que cualquiera puede saturar el servidor con un gran número de peticiones.
Por otra parte, la velocidad de respuesta fue muy similar en los tres métodos ya que el desempeño depende principalmente del servidor y variables externas al mecanismo de invocación remota, por lo que se acudió a la investigación para obtener datos de velocidad, latencia entre otros. En la Tabla 1 se observan los diferentes resultados en el formato de una *Pugh Matrix* (Tabla que en lugar de presentar datos directos, presenta resultados en comparación a una base). Para interpretar la tabla, nuestra base será XML-RPC, y se evaluarán parámetros tales como la velocidad de respuesta y seguridad, entre otros. Si uno de los otros mecanismos es más rápido que XML-RPC, entonces tendrá un símbolo de suma; si es más lento, entonces tendrá el símbolo de menos. Y si es igual o no se tiene el dato, entonces tendrá una S. Al final se suman todos los resultados (+ =+le +1, S=0 y -= -1) para encontrar la opción adecuada.

	XML RPC	kSOAP	JSON RPC
Peso (más ligero)	s	-	S
Tiempo de respuesta	s	-	+
Compatibilidad	s	-	S
Seguridad	s	+	S
Flexibilidad (Multilenguaje)	s	-	S
Total	**0**	**-3**	**1**

Tabla 1. Pugh Matrix con los resultados del análisis comparativo.

Una vez evaluados los resultados, se eligió JSON RPC para la elaboración de una aplicación para una Universidad, en la cual el usuario principal es el estudiante, que introducirá su matrícula y clave. Con esta información, el programa realiza una conexión al servidor haciendo una petición del servicio "consultar" con los parámetros otorgados. El servidor regresa una cadena que contiene la información de las materias y calificaciones del alumno con un identificador el cual indica al cliente donde comienza y termina un parámetro para pasarlo a una tabla la cual es legible por el usuario, como puede observarse en la Tabla 2**.** Por ejemplo, una cadena con las calificaciones de las materias de programación y matemáticas llegaría así: Juan Pérez; Programación; 10; Matemáticas; 9.

Nombre	Juan Pérez
Programación	10
Matemáticas	9

Tabla 2. Ejemplo utilizando JSON RPC

Una vez con la información se puede presentar de mejor forma con la interfaz gráfica. También se pueden enviar fotografías, direcciones de páginas de internet, entre otras cosas que se envían como cadena y es trabajo del cliente realizar la conversión del tipo de datos.

Conclusiones y Trabajo Futuro

En este artículo se presentaron algunos de los mecanismos más utilizados para la invocación de métodos que se ejecutan en una computadora remota. Se estudiaron brevemente las características de los

Llamados a Procedimientos Remotos o RPC, así como su evolución hasta llegar a los servicios Web. Debido a que las aplicaciones que se ejecutan en dispositivos móviles se benefician en gran medida del uso de este tipo de mecanismos, en el artículo se presentaron las características del sistema operativo Android, que es un sistema operativo muy reciente y que ha sido desarrollado para ser utilizado en dispositivos móviles. Además, se estudiaron algunas de las herramientas para la invocación de métodos remotos disponibles en Android, y se presentaron los resultados de un comparativo entre ellas.

Como resultado de la evaluación de estas herramientas se concluye que el sistema operativo Android aún es muy joven y es difícil predecir cuál será el camino que la mayoría de las aplicaciones tomarán. En este caso, era muy importante que la aplicación fuera rápida y ligera además de que tuviera facilidad en la implementación y que se ajustara bien al ambiente del sistema. Probablemente en una aplicación más formal se requiera de un método como kSOAP, pero XML-RPC no muestra ninguna ventaja perceptible por encima de los otros dos métodos. Una vez que se sabe trabajar con JSON RPC, la codificación para comunicar el cliente con el servidor se torna algo sencilla y es similar a como se comunica con procesos locales, donde el desarrollador no tiene que entender todo lo que está ocurriendo detrás así como no se tiene que ser un mecánico para manejar un carro, lo cual es el objetivo principal de los mecanismos de invocación remota; esto es, facilitar el desarrollo de aplicaciones que requieran del uso de estos mecanismos, algo que JSON RPC consigue exitosamente.

Como trabajo futuro, se pretende una evaluación de las herramientas para la implementación de servicios Web disponibles en Android.

Referencias

[1] A. D. Birrell and B. C. Nelson, *Implementing Remote Procedure Calls*, ACM Transactions on Computers, Vol. 2, Issue 1, pp. 39 – 59, Feb 1984.

[2] Z.S. Hassan, *Ubiquitous Computing and Android*, In Proceedings of the IEEE 3rd International Conference on Digital Information Management, pp. 166 – 171, Nov 2008.

[3] S. Vinoski, *CORBA: Integrating Diverse Applications within Distributed Heterogeneous Environments*, IEEE Communications Magazine, Vo. 14, No. 2, Feb 1997.

[4] J. DiMarzio, *Android: A programmer's guide*, McGraw-Hill, ISBN 0071599886, 2008.

[5] B. McLaughlin, *Java & XML*, O'Reily 2001

[6] S. Laurent, J. Johnston, E. Dumbill, *Programming web services with XML RPC*, O'Reilly 2001.

[7] A. Vogel, B. Gray and K. Duddy, *Understanding Any IDL Lesson One: DCE and CORBA*, In Proceeding of the 3rd International Workshop on Services in Distributed and Networked Environments, pp. 114 – 121, Jun 1996.

[8] M. Papazoglou, *Web Services: Principles and Technology*, Prentice Hall, ISBN 0321155556, 2008.

[9] R. T. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, Ph D Thesis, University of California, Irvine, 2000.

[10] Richardson, Leonard Ruby, Sam (2007) *RESTful web services* O'Reilly Media, Inc. p. 299

[11] B. Marchal, "SOAP 1.2 and the GET request" in *Developer Works*, IBM, 2010.

Currículo corto de los autores

Neftalí David Watkinson Medina
Estudiante de la carrera de Ingeniería de Software.
Cetys Universidad, campus Ensenada.

Miguel Ángel Piccini Méndez
Estudiante de la carrera de Ingeniería de Software.
Cetys Universidad, campus Ensenada.

Lucía Beltrán Rocha
Coordinadora de la carrera de Ingeniería de Software.
Cetys Universidad, campus Ensenada.

Arnoldo Díaz Ramírez.
Profesor Investigador del Instituto Tecnológico de Mexicali.
Ingeniero en Ciencias Computacionales por el Cetys Universidad, Campus Mexicali.
Doctor en Informática por la Universidad Politécnica de Valencia, España.