Generación de un prototipo de comunicación sobre Android para app móviles reactivas

Susana Beatriz Chavez [1] Adriana Elizabeth Martín[2], Sergio Rafael Flores[3] ; A. Sara Zogbe [4], Ortiz, Alexis Rodrigo

Departamento e Instituto de Informática - F.C.E.F. y N. - U.N.S.J.
Complejo Islas Malvinas. Ignacio de la Roza y Meglioli.
C.P. 5402. Rivadavia. San Juan, 0264 4234129

[1]schavez@iinfo.unsj.edu.ar; [2] arianamartinsj@gmail.com;
[3]sergior@gmail.com; [4]sarazogbe@yahoo.com.ar;
rodriunsj5@gmail.com

Resumen

El gran avance de las comunicaciones ha cambiado drásticamente la forma en que las personas y las máquinas interactúan entre sí, permitiendo el acceso instantáneo a información y servicios en tiempo real. El objetivo de este trabajo es proponer un modelo de comunicación entre app móviles, que permita evaluar la disponibilidad de los canales de comunicación y, de esta manera, garantizar que un mensaje llegue a destino. Para ello, se propone trabajar sobre una plataforma con soporte a la programación reactiva. Esto conlleva analizar y entender qué propone este nuevo paradigma reactivo para que el desarrollo del software móvil sea una solución real y competitiva.

Para poder interactuar continuamente con su entorno, las apps reactivas deben poder adaptarse a la carga a la que se enfrentan, utilizando una mayor capacidad computacional cuando sea necesario. Esto significa que debe poder hacer un uso eficiente del hardware en un solo dispositivo (que puede tener uno o más núcleos), y también ser capaz de funcionar a través de varios nodos de cómputo a su disposición, dependiendo de la carga.

Para garantizar la conectividad requerida por las apps reactivas, se pretende desarrollar un componente que identifique todas las alternativas disponibles de comunicación en el hardware del dispositivo. Considerando dispositivo a cualquier equipo o máquina que sea capaz de generar y trasmitir información a otro dispositivo.

Palabras claves: Paradigma Reactivo, Dispositivos móviles, IoT, Java, Android SO.

Contexto

El presente trabajo se encuadra dentro del área I/D Innovación en Sistemas de Software y se enmarca dentro del proyecto de investigación: Modelo de Sistema de Comunicación en Programación Reactiva, que ha sido aprobado por CICITCA y está en desarrollo para el período 2020-2022. Si bien este grupo de investigación se ha formado para trabajar en conjunto a partir del año 2020, cada integrante viene realizando tareas en distintos grupos, algunos desde enero de 2000 en tecnologías asociadas a la Computación Distribuida.

Se incorporaron dos alumnos en instancia de trabajo final de tesis de grado.

Las unidades ejecutoras para dicho proyecto son el Departamento e Instituto de Informática de la FCEFyN de la UNSJ.

Introducción

Debido a que los procesadores multinúcleo se están convirtiendo en un estándar, se han creado múltiples niveles de abstracción para simplificar la concurrencia y permitir un desarrollo más simple.

Es importante entender conceptos claves, como son los modelos de concurrencia [1]

utilizados para modelar y manipular operaciones asincrónicas sin perder de vista la tolerancia a fallos.

Cada uno de estos modelos tiene un enfoque diferente:

- Futures: son la base de la programación asíncrona y reactiva en Scala. Permiten manipular los resultados de una operación que aún no ha sucedido y lidiar efectivamente con la falla de dichas operaciones, permitiendo un uso más eficiente de los recursos computacionales.
- Memoria transaccional de software (STM): simplifica enormemente la comprensión conceptual de los programas multiproceso y ayuda a que los programas sean más fáciles de mantener al trabajar en armonía con las abstracciones de alto nivel existentes, como los objetos y los módulos. Usado por Clojure.
- Stream Reactivos: proporciona una abstracción para aplicaciones asíncronas altamente concurrentes con soporte para el procesamiento de flujo asíncrono con backpressure sin bloqueo. Ej.: RxJava, Reactor, etc.
- Modelo basado en Actores: es un modelo concurrente de cómputo para crear sistemas altamente concurrentes y paralelizables en un entorno distribuido. Popularizado por el lenguaje de programación Erlang y es implementado por Akka sobre JVM [2]. Con este modelo los actores no invocan a métodos, sino que se envían mensajes entre ellos (ver Figura 1). El envío de un mensaje no transfiere el hilo de ejecución al receptor. Un actor puede enviar un mensaje y continuar sin quedar bloqueado. Por lo tanto, puede lograr más en la misma cantidad de tiempo. A diferencia del comportamiento de los *objetos*, que libera el control de su hilo de ejecución cuando el método invocado termina (return). Sin embargo, los actores se comportan como objetos, cuando reaccionan a los mensajes y devuelven la ejecución al terminar de procesar el mensaje actual. De esta manera, los actores logran realmente la ejecución que siempre se imaginó para los objetos.

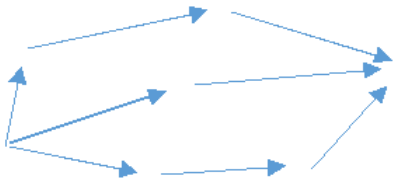

Figura1 . Interacción entre actores

La abstracción del modelo de actor permite pensar en el código en términos de comunicación.

No todos los modelos tienen un verdadero manejo de errores y recuperación de fallas. Algunos están diseñados para un alto rendimiento, pero no para escalar horizontal y/o verticalmente. Las ventajas que ofrecen cada uno de estos modelos pueden ser una "buena idea" en algunos contextos, pero no en otros. Es preciso cuestionar todo y asegurarse de que realmente se comprenden las fortalezas y debilidades de todas sus opciones de diseño, ya sea que se esté construyendo o no un sistema reactivo.

Protocolos de comunicación

Los dispositivos IoT (Internet of Things) utilizados tanto en entornos industriales, como en el entorno doméstico, pueden compartir aspectos de kernel (Linux, FreeRTOS, Windows Embedded, siendo el primero de ellos el más utilizado) y servicios de bajo nivel (*Real-Time Management, Context Discovery Management*), pero en términos de comunicación son muy diferentes. A continuación, se explican algunos de los protocolos utilizados a nivel doméstico e industrial.

Entorno doméstico

Cada día son más los hogares que disponen de uno o varios dispositivos IoT. como se muestra en la Figura 2. Existen multitud de

empresas que fabrican estos dispositivos. Cada uno tiene su propio protocolo de comunicación y aunque no es mucha la información que suelen facilitar, sobre todo si no son protocolos de código abierto, los más conocidos son: AllJoyn**,** HomePlug y HomeGrid**,** MFi (*Made For iPhone/iPod/iPad*), **OCF** (*Open Connectivity Foundation*), Thread (*network protocol*), etc.

Figura 2. Ejemplo IoT

Entorno industrial

Con la aparición de los dispositivos IoT, surge el concepto de industria 4.0 [10]. Podemos definirla como la digitalización completa a través de la integración de tecnologías de procesamiento de datos, software inteligente y sensores, desde los proveedores hasta los clientes. Pero para que esto sea posible, los dispositivos deben poder comunicarse, tanto entre sí, como hacia el exterior, como se muestra en la Figura 3. Algunos de los protocolos de comunicación existentes en la industria son: AMQP (*Advanced Message Queuing Protocol*), CoAP (*Constrained Application Protocol*), HTTP (REST/JSON) *(Hypertext Transfer Protocol*), XMPP (*Extensible Messaging and Presence Protocol*), etc.

Figura3. Ejemplo IoT en la industria

Cualquier dispositivo móvil está preparado para acceder a internet de manera instantánea a información y servicios en tiempo real. Las especificaciones de los dispositivos Android, en la actualidad, son lo suficientemente competentes, con CPU de 4 a 10 núcleos de 3 a 4 GB de memoria Se pueden usar hasta alrededor de 512 MB de memoria por máquina virtual (VM) con una configuración largeHeap = true.

Esta conectividad de los dispositivos con internet ha llevado a la creación de nuevos conceptos como Internet de las Cosas (IoT) que ha ganado popularidad en los últimos años [3].

Varias aplicaciones dependen de la potencia informática y otros recursos proporcionados por la nube para su correcto funcionamiento. Los dispositivos como sensores y dispositivos portátiles utilizan los servicios en la nube para procesar los datos que generan.

El IoT, tiene como objetivo contar con sensores u objetos dispersos para que generen información desde cualquier sitio accesible o bien en el interior de una máquina [4][5][6]. Esto llevará a un futuro en el que no sólo sea usado para la comunicación entre personas, sino, entre humano y máquina, e incluso, entre diferentes máquinas (M2M) [7]. Por ello, cobran también importancia los Smart Objects: objetos físicos con un sistema embebido que le permite procesar información y comunicarse con otros dispositivos y realizar acciones con base en una acción o evento determinado [8][9].

No obstante, todos estos sistemas complejos presentan un problema al momento de interconectarlos debido a las diferencias entre software y hardware utilizado en los diferentes procesos y canales de comunicación.

Los desafíos no solo involucran elementos como el almacenamiento de datos o el poder de la computación, sino que también exigen soluciones de software que puedan administrar y procesar *la comunicación entre aplicaciones*.

Este hecho pone de manifiesto la necesidad de nuevas formas de manejar la comunicación de datos que se originan en estos dispositivos.

Líneas de investigación, Desarrollo e Innovación

Para poder interactuar continuamente con su entorno, las apps reactivas deben poder adaptarse a la carga a la que se enfrentan, utilizando una mayor capacidad computacional cuando sea necesario. Esto significa que debe poder hacer un uso eficiente del hardware en un solo dispositivo (que puede tener uno o más núcleos), y también ser capaz de funcionar a través de varios nodos de cómputo a su disposición, dependiendo de la carga.

En pos de garantizar la conectividad requerida por las apps reactivas, se pretende desarrollar un componente de comunicación que identifique todas las alternativas disponibles de comunicación en el hardware del dispositivo. Se considera dispositivo a cualquier equipo o máquina que sea capaz de generar información y pueda transmitirla a otro dispositivo.

Las alternativas que cobran importancia se enfocan en las tecnologías inalámbricas como 4G, GSM y UMTS, Bluetooth y otras actualmente en desarrollo, particularmente las relativas a redes inalámbricas de área personal (WPANs). A estas, se suma el sistema de localización GPS, las medidas de tiempo de ultrasonido, UWB (Ultra- Wide Band), radiobalizas (por ej. vecinos lectores RFID con coordenadas conocidas o estaciones base WLAN) y tecnologías ópticas.

El objetivo que se persigue es lograr con éxito enviar un mensaje de un nodo a otro, siguiendo el *modelo de actores*. Pasar mensajes entre nodos, es la norma ahora, ya sea a través de dispositivos o computadoras en red.

En este modelo de concurrencia, ya no hay memoria real compartida, los diferentes núcleos de un dispositivo se envían entre sí, fragmentos de datos explícitamente, tal como lo hacen las computadoras conectadas en una red. En lugar de ocultar el aspecto del paso del mensaje a través de variables marcadas como compartidas o utilizando estructuras de datos atómicas, un enfoque más disciplinado y basado en principios consiste en mantener el estado local a una entidad concurrente y propagar datos o eventos entre entidades concurrentes explícitamente a través de mensajes.

Los protocolos de IoT, tanto para el sector Doméstico como para el Industrial, están resueltos siempre y cuando haya conexión real de internet (datos o wifi). Cuando estas conexiones no están disponibles, no es posible mantener la comunicación, solo queda esperar a que se restablezca la misma, con las consecuencias que esto pudiera acarrear. Por este motivo se propone el diseño de un driver que se encargue de guiar en forma automática la comunicación de acuerdo a un esquema de jerarquía y en forma transparente al usuario.

En la Figura 4 se muestra cómo funciona el sistema de comunicación en forma normal.

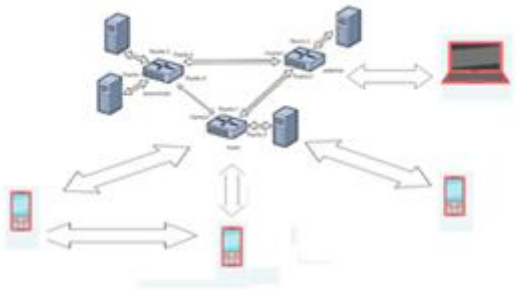

Figura 4. Comunicación normal

Cuando la comunicación se interrumpe debido a diversos factores, como por ejemplo fuera de cobertura, falta de crédito, o por cualquier otro motivo. Existen otras formas de comunicación que siguen vigentes sin ser diseñadas para este propósito, como por ejemplo el sistema de mensajería de texto o las llamadas de voz, incluso algunas

compañías proveen de servicio gratuito al sistema de comunicación WhatsApp, que si bien están destinados a un fin específico, se podrían utilizar para proveer de conectividad de manera temporal al móvil. De esa manera un móvil puede hacer de anfitrión para proveer la comunicación, por ejemplo: como se muestra en la Figura 5 el móvil M1 tiene comunicación con un sistema que a su vez se comunica con el móvil M3, si se rompe la comunicación de M1, el driver busca un anfitrión como por ejemplo el móvil M2 y accede a través de él al sistema

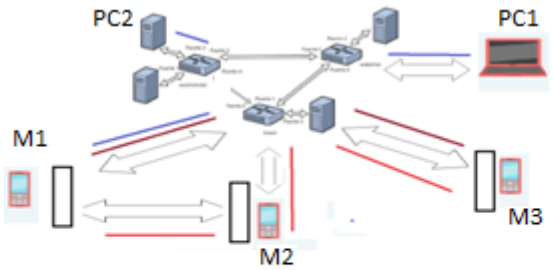

Figura 5. Comunicación a través de un anfitrión

Para ello es necesario la implementación de un driver que se comunique directamente con el sistema operativo y pueda discernir el mejor camino cuando se pierda la conexión natural de datos o wifi. La figura 6 muestra un esquema general de las funciones del driver.

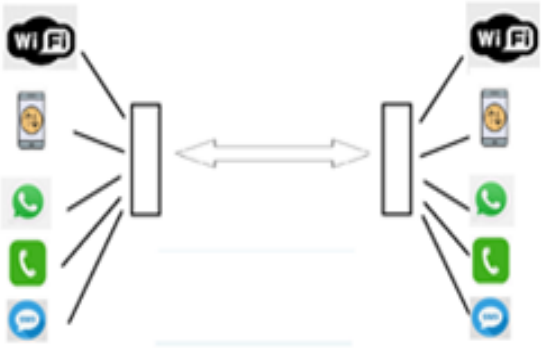

Figura 6. Funciones del Driver propuesto

Resultados y Objetivos

Resultados Obtenidos

Si bien, este equipo de investigadores se ha formado para trabajar en conjunto a partir del año 2020, cada integrante viene realizando tareas en distintos grupos, algunos desde enero de 2000 en tecnologías asociadas a la Computación Distribuida.

Con esta propuesta de trabajo se espera contribuir a la profundización y consolidación del conocimiento en esta área temática por parte de cada uno de los integrantes de este proyecto

Durante el periodo 2020-2021 transcurrido se realizaron diferentes publicaciones a fines a la propuesta.

A pesar de ser un periodo atípico debido a la pandemia el prototipo propuesto se encuentra en un estado avanzado de desarrollo e implementación.

Actualmente se está trabajando en establecer la comunicación punto a punto de los distintos sistemas de comunicación sin hacer eco con el dispositivo, es decir, generar mensajes internos, enviarlos, recibirlos en forma transparente al usuario, de modo que no los visualice y tampoco intervenga. El entorno de desarrollo es Android Studio, con el lenguaje de programación nativo java.

Paso seguido es comunicar las apps a través del canal de voz, es decir por medio de una llamada de voz. En este caso se deben realizar los siguientes pasos:

a) Establecer la llamada con el móvil destino, se realiza la llamada el móvil destino responde
b) Sincronizar la comunicación con el móvil destino, es decir, que sepa que no es una llamada de voz común, sino que se trata de una comunicación entre máquinas, se puede realizar esto por medio de una escucha de las notificaciones de llamadas, si es el número programado capturar la

llamada, enviar un ok para sincronizar (en otra frecuencia).

c) El texto a enviar se debe modular con una frecuencia de 42KHz, luego tomar los datos pasarlo a voz y modularlos.
d) Una vez sincronizado enviar la señal modulada.
e) En el dispositivo destino tomar la señal modulada con un formato de voz y aplicar el proceso de demodulación
f) El dato estaría disponible

La variedad de dispositivos con capacidades diferentes, es enorme y está al alcance de la mano. Resta configurar, adaptar, implementar y poner a punto un prototipo del modelo SiCo, por parte de este equipo de trabajo. De esta manera se contribuirá a la profundización y consolidación del conocimiento de esta área temática.

En cuanto a la movilidad de los dispositivos, los recursos pueden varían sin depender de los mismo, sino de los mecanismos de comunicación desde 4G pasando por GSM, solo WhatsApp, solo mensajería de texto o simplemente llamada de voz, es por ello que se necesita disponer de un mecanismo de selección automática del mejor sistema de comunicación disponible.

Objetivos

El objetivo del grupo de investigación es evaluar todas alternativas disponibles para manipular la comunicación entre aplicaciones que se originan en los distintos dispositivos, y construir un prototipo basado en una arquitectura reactiva donde la resiliencia y la comunicación asíncrona permitan que los sistemas estén exentos de errores.

Formación de Recursos Humanos

El equipo de trabajo está compuesto por 4 docentes-investigadores de la línea de investigación presentada que figuran en este trabajo y Alumnos avanzados de las Carreras de Licenciatura en Sistemas de Información, Y Licenciatura en Ciencias de la Computación en estado de tesis; pertenecientes a la Universidad Nacional de San Juan.

Se está trabajando con dos tesinas en el ámbito de la programación para dispositivos móviles. Se espera iniciar otra tesina de grado en el área motivo de la presente propuesta de investigación.

Referencias

1. Adam L. Davis: Reactive Streams in Java_ Concurrency with RxJava, Reactor, and Akka Streams-Apress (2018)
2. How the Actor Model Meets the Needs of Modern, Distributed Systems: https://doc.akka.io/docs/akka/current/typed/guide/ac tors-intro.html
3. José I. Rodriguez M. Tesis Doctoral “Metamodelo para la integración del Internet de las cosas y Redes sociales” Universidad Oviedo 2017
4. González García Cristian, “MIDGAR: Plataforma para la generación dinámica de aplicaciones distribuidas basadas en la integración de redes de sensores y dispositivos electrónicos IoT,” UNIVERSIDAD DE VIEDO, 2013.
5. L. Atzori, A. Iera, G. Morabito, and M. Nitti, “The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization,” Comput. Networks, vol. 56, no. 16, pp. 3594–3608, Nov. 2012.
6. C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, and G. Cueva-Fernandez, “Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios,” Comput. Networks, vol. 64, pp. 143– 158, May 2014.
7. R. Roman, J. Zhou, and J. Lopez, “On the features and challenges of security and privacy in distributed internet of things,” Comput. Networks, vol. 57, no. 10, pp. 2266–2279, Jul. 2013.

8. J. Pascual Espada, O. Sanjuán Martínez, B. C. Pelayo GBustelo, and J. M. Cueva Lovelle, “Virtual Objects on the Internet of Things,” Int. J. Interact. Multimed. Artif. Intell., vol. 1, no. 4, p. 23, 2011.
9. B. Xu, L. Da Xu, H. Cai, C. Xie, J. Hu, and F. Bu, “Ubiquitous Data Accessing Method in IoT-based Information System for Emergency Medical Services,” IEEE Trans. Ind. Informatics, vol. 3203, no. c, pp. 1–1, 2014. 100
10. La Ciberseguridad en la Industria 4.0 https://www.incibe-cert.es/blog/ciberseguridad-industria-4-0