Flexible processing architecture for maintaining QoS in embedded systems applications

José Francisco Colom, Higinio Mora and David Gil [1]

Resumen— The growing available capacity on a single chip is leading to increasingly sophisticated applications in the field of embedded systems. In addition, the cloud computing paradigm, allows the extension of the capabilities of these systems using remote resources. Among the wide range of applications that can arise in this context, are those in which it is critical to meet certain quality of service (QoS) requirements, such as limited latency. In these cases, real-time operating systems (RTOS) provide a valid solution to guarantee predictability and response time using the resources of the embedded system. However, in applications where the elements to process can grow and decrease in a variable way, the load can exceed the capabilities of the embedded system, which is an important limitation. In this paper, a new architecture is proposed, aiming to take the most of remotely available resources only when the load temporarily exceeds the capabilities of the embedded system. The access to the remote resources is done by using cloud platforms maintaining an acceptable level of QoS for the application.

Palabras clave — Embedded system, Real-time operating system, Quality of service, Cloud computing

I. Introducción

EL desarrollo que los sistemas embebidos están experimentando en los últimos tiempos ha permitido su extension a nuevos campos de aplicación como la automocion, robotica, smart cities o healthcare. Sin embargo, su desarrollo en estos ámbitos requiere un salto cualitativo de diseño que tenga en cuenta las exigencias de rendimiento y tiempo de respuesta que estas aplicaciones requieren. En este sentido, la calidad de servicio (QoS—Quality of Service) es un aspecto fundamental para garantizar su buen funcionamiento.

El mantenimiento de una calidad de servicio en los tiempos de respuesta y calidad del resultado que los sistemas embebidos deben proporcionar en esas aplicaciones, los eleva a la categoría de sistemas de tiempo real [1]. En este tipo de sistemas, la validez de los resultados vendr dada no solo por su corrección sino también porque éstos estén a tiempo. Es decir, existen unas restricciones que condicionan su funcionamiento. El diseño y concepcion de estos sistemas debe proponer, por tanto, arquitecturas que contemplen los aspectos de correccion, adaptabilidad, predecibilidad, seguridad y tolerancia a fallos.

La evolucion tecnologica de los dispositivos embebidos los dota actualmente de unas prestaciones suficientes como para implementar sobre ellos estrategias de planificación complejas. Estas estrategias pasan por delegar en un sistema operativo de tiempo real embebido en los dispositivos la planificacion y ordenación de la ejecución de las tareas para cumplir con las restricciones impuestas por las aplicaciones [2], [3].

Aunque ese tipo de soluciones proporcionan importantes cotas de satisfacción de las restricciones, algunas aplicaciones pueden verse desbordadas temporalmente por las características de su ejecución y requerir unas prestaciones que exceden a su capacidad. En estos casos, los sistemas anteriores deberían rechazar la ejecución del exceso de tareas que excedan los tiempos de respuesta para garantizar el cumplimiento hard real-time de la planificación [4]. Sin embargo, este tipo de decisiones puede provocar in- terrupciones del servicio inasumibles en algunas apli- caciones críticas. Por ejemplo, sistemas e-health que supervisan y controlan variables biométricas de va- rios individuos simultaneamente, pueden experimen- tar un aumento de las necesidades de cómputo de- rivadas del incremento del conjunto de individuos a supervisar; o por ejemplo, un sistema de gestion de trafico de una smart city en el que cada vehículo re- coge y transmite información sobre su estado al res- to de vehículos y a la señalización del entorno puede verse saturado igualmente en escenarios densos con multitud de vehículos.

Una posible solución para paliar los problemas puestos de manifiesto en aplicaciones del tipo indicado, es la utilización auxiliar de redes de area amplia y recursos externos disponibles en la nube, por ejemplo mediante infraestructuras IaaS (Infrastructure as a Service), en la medida en que estas infraestructuras puedan proporcionar de manera escalable las capacidades necesarias. Sin embargo, también hay que tener en cuenta los costes económicos en los que se puede incurrir al utilizar estos recursos, en ocasiones proporcionales al tiempo de proceso o la cantidad de datos procesados, según distintos modelos de utility computing [5].

El objetivo de este trabajo es diseñar una arquitectura que proporcione un conjunto de servicios a aplicaciones en sistemas embebidos. Empleando estos servicios, las aplicaciones podran incorporar recursos de procesamiento y almacenamiento en la nube, en la medida en que estos sean necesarios para mantener una respuesta en tiempo real ante excesos puntuales de carga de trabajo. De esta forma, se aprovechan al máximo los recursos disponibles lo-

[1] Computer Technology Department, University of Alicante, e-mail: {jfcolom,hmora,dgil}@ua.es.

calmente en el propio sistema integrado, junto a las características de predecibilidad y fiabilidad del sistema operativo de tiempo real. Simultáneamente, se optimizan los costes derivados del uso de plataformas IaaS, recurriendo a ellas únicamente en caso necesario.

El resto del trabajo se ha organizado de la siguiente manera. En primer lugar, se presenta un estudio de los trabajos relacionados tanto con el aprovechamiento de recursos múltiples en la nube como con distintos enfoques para garantizar QoS en la ejecución de aplicaciones. A continuación se propone una arquitectura que da respuesta a las necesidades y retos planteados. Dicha arquitectura se ilustra posteriormente mediante una propuesta de caso de estudio. Finalmente, se presentan las conclusiones así como las futuras direcciones de esta investigación.

II. Trabajos relacionados

Un extremo en la configuración de los sistemas distribuidos lo constituyen los sistemas compuestos por dispositivos esencialmente sensores/actuadores que carecen de capacidad de procesamiento para tomar por sí solos las decisiones. Estos elementos, que hacen básicamente las funciones de transceptor, transmiten la información para que sea tratada por un *host* con capacidad suficiente [6], [7], [8]. Sin embargo, este planteamiento puede infrautilizar las capacidades de los propios dispositivos embebidos, resta agilidad en la respuesta y requiere de una infraestructura permanente adicional para realizar el procesamiento.

Por otro lado, se han propuesto sistemas multiprocesador como solución embebida para un amplio abanico de aplicaciones. En estos casos, no se requiere la configuración de ningún elemento no embebido, ya que los dispositivos empotrados cuentan con capacidad suficiente. En entornos en los que intervienen múltiples dispositivos se pueden establecer métodos de planificación que tengan en cuenta escenarios de multiprocesamiento en uno [9], [10] o varios elementos embebidos con características heterogéneas [11], [12]. Un paso más de esta estrategia lo constituyen los sistemas embebidos distribuidos que interactúan entre sí mediante una red de comunicaciones. Para estos casos, también se han realizado propuestas con el fin de mantener la calidad de servicio de las prestaciones [13], [14].

Entre las redes de sensores con escasa capacidad de procesamiento, y aquellas formadas por multiprocesadores, pueden darse configuraciones intermedias que repartan la carga de trabajo entre los dispositivos embebidos distribuidos y algún elemento no embebido. En esta línea, un campo con intensa producción científica, es el de la computación móvil en la nube (MCC—Mobile Cloud Computing). Su desarrollo constituye un paradigma denominado *offloading* [15] en el que la nube se emplea para realizar el procesamiento de dispositivos móviles con recursos limitados liberando de la carga de trabajo correspondiente del procesador del dispositivo móvil. Las propuestas se ordenan bajo dos tipos de enfoques [16]: por un lado los sistemas que tratan de adaptar las aplicaciones existentes identificando porciones de código externalizable [17], [18], [19], [20], y por otro, nuevas aplicaciones que en su concepción tienen en cuenta esta idea [21], [22].

Gran parte de las proyectos relacionados con MCC están orientados fundamentalmente a alargar la vida de las baterías [23], [24]. En general, se persigue la liberación de la máxima cantidad de recursos computacionales locales. Sin embargo, el enfoque propuesto en este artículo pone el acento en el uso de recursos en la nube para optimizar un conjunto de parámetros de QoS, para lo cual puede ser necesario priorizar la ejecución de tareas en local frente al uso, por ejemplo, de servicios de infraestructura (IaaS).

Otro campo de intensa actividad investigadora relacionado con el mantenimiento de tiempos de respuesta y en general con QoS, es la gestión de las comunicaciones. En este área, se han realizado aportaciones relacionadas con el análisis adaptativo inteligente de los tiempos de servicio [25] y se han propuesto arquitecturas orientadas a cumplir con los requisitos de QoS [26], [27]. Estos trabajos no sólo tienen en cuenta parámetros de eficiencia energética sino también proponen estrategias para su cumplimiento con especificaciones de funcionamiento en tiempo real [28].

La utilización conjunta de las infraestructuras de computación distribuidas para garantizar la QoS es una opción que también está siendo ampliamente analizada [29], [30]. Los servicios que combinan los recursos de infraestructuras distribuidas de distinto tipo (*clusters*, *grids*, *cloud*, etc) son un mecanismo que refuerza los compromisos de QoS y permite, por tanto, especificar restricciones de *hard-RT* sobre elementos de computación en red. En esta línea, la preocupación por el mantenimiento de los tiempos de respuesta de modelos de computación en la nube está presente en numerosos trabajos en los que se han analizado y propuesto soluciones de QoS para sistemas *cloud computing* [31], [32], [33]. Aunque su enfoque está dirigido especialmente hacia aplicaciones multimedia (*online gaming*, vídeo bajo demanda) sus conclusiones pueden ser trasladadas a otros sectores (*business*, telemedicina, automoción, etc.) para la provisión de servicios con QoS [34].

En conclusión, los sistemas embebidos con necesidades de funcionamiento en tiempo real responden adecuadamente a sus consideraciones de diseño. Las mejoras en la tecnología y los sistemas multiprocesador contribuyen a este propósito manejados convenientemente por un RTOS. Sin embargo, estas nuevas capacidades no aportan mecanismos de flexibilidad que permitan eventualmente aumentar la carga de procesamiento por encima de un determinado nivel y por tanto, limitan su aplicación a situaciones acotadas de funcionamiento. Por otra parte, los recursos de computación en la nube pueden emplearse para ejecutar una parte del procesamiento con una cierta garantía de QoS a cambio de un cierto coste, pero en su concepción actual se requiere una cone-

xión permanente y no está planteada como apoyo para una utilización bajo demanda en función de las necesidades.

III. Arquitectura

Un sistema de tiempo real puede modelarse mediante un conjunto de tareas que se comunican de acuerdo a un cierto patrón o paradigma de computación paralela [35]. De entre los varios existentes, el paradigma *master-slave* es aplicable a gran cantidad de problemas. Según este paradigma, la carga de trabajo a procesar de divide entre un conjunto de tareas esclavas (*slave*). Una tarea principal (*master*), se encarga de distribuir el trabajo en las tareas *slave*. La comunicación se produce únicamente entre la tarea *master* y cada una de las tareas *slave*. Las tareas *slave* se mantienen independientes entre sí, esto es, no se comunican entre ellas [36].

La figura 1 ofrece una visión general de la arquitectura propuesta, aplicada a un sistema real en el que las tareas se comunican de acuerdo al mencionado paradigma *master-slave*. La arquitectura mostrada proporciona servicios de procesamiento flexible en la nube para aquellas aplicaciones que pueden descomponerse en un conjunto $\{s_1, s_2, \cdots, s_k, s_{k+1}, s_{k+2}, s_{k+n}\}$ de tareas *slave* independientes que se ejecutan a petición de una tarea *master*. Realmente, la arquitectura puede proporcionar sus servicios con independencia del paradigma de programación paralela concreto, y por tanto es aplicable también a otros patrones de comunicación entre tareas.

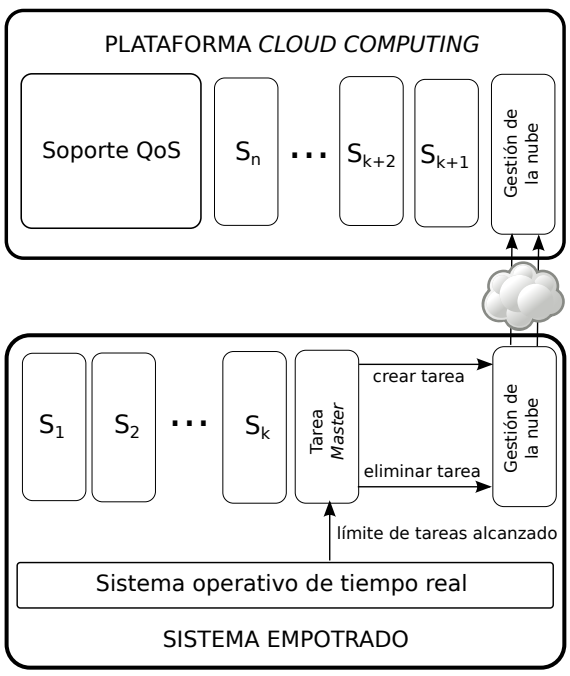

Fig. 1.Arquitectura propuesta para un sistema de tiempo real en el que las tareas se comunican de acuerdo al paradigma *master-slave*

El sistema operativo otorga a cada tarea una fracción del tiempo de uso de procesador, suficiente para la obtención de un resultado correcto, manteniendo el comportamiento del sistema predecible en todo momento en cuanto a latencia. La corrección del resultado no implica necesariamente precisión, ya que es posible el empleo de técnicas de computación imprecisa que permitan la obtención de resultados aceptables, pero mejorables en la medida en que haya más tiempo disponible.

Con un conjunto de recursos limitado, y teniendo en cuenta la predecibilidad del sistema, un intento de creación de más de k tareas, dará lugar a una respuesta de excepción por parte del sistema operativo, y deberá ser capturada por la tarea *master*. Es en este punto cuando entra en funcionamiento el servicio de creación de tareas en la nube proporcionado por la arquitectura (gestión de la nube, en el diagrama de la figura 1). También se requiere un servicio de eliminación de tareas, que será utilizado cuando el proceso *master* así lo determine a partir de las necesidades del entorno.

La figura 2 resume la actividad de la arquitectura. En primer lugar, responde a los eventos de creación y eliminación de tareas, dando lugar a la ejecución de los correspondientes servicios. La primera tarea extra requiere una comprobación previa de la disponibilidad de las diferentes plataformas en la nube, previamente configuradas. Por ejemplo, las plataformas utilizables pueden ser públicas, accesibles a través de Internet, pero también privadas y accesibles únicamente a través de redes específicas. Una vez determinada la lista de plataformas disponibles, será necesario realizar una selección de acuerdo a un sistema de prioridades que puede tomar en consideración parámetros de calidad de servicio o coste.

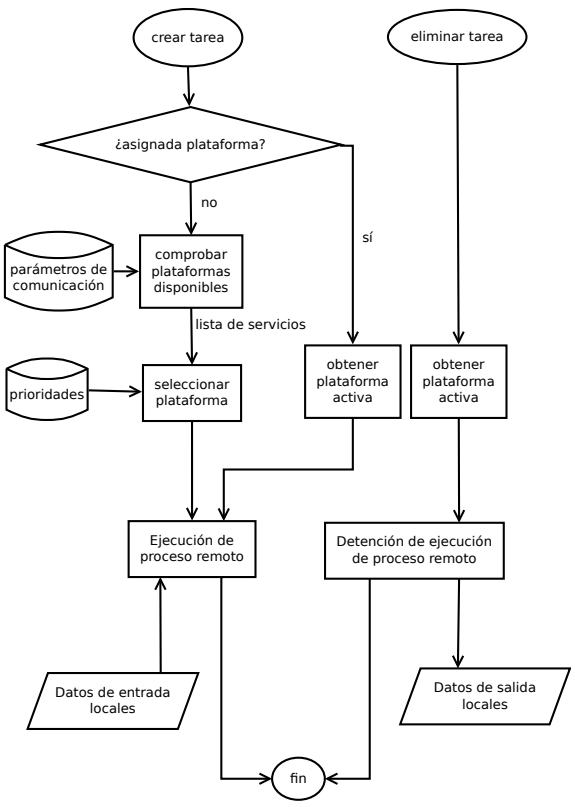

Fig. 2. Descripción de la actividad de la arquitectura

La ejecución de una tarea en la nube puede requerir el empleo de determinados datos de entrada disponibles únicamente mediante los dispositivos accesibles por el sistema empotrado. Estos datos deben enviarse a la plataforma en la nube para su procesamiento por parte de la tarea correspondiente. Las latencias debidas a las transmisiones de datos deben ser tenidas en cuenta por el sistema operativo a la hora de realizar la planificación del proceso correspondiente a la propia arquitectura. De manera similar, las tareas en la nube puede generar información que deba visualizarse en dispositivos dependientes del sistema empotrado.

En un sistema empotrado basado en multiprocesadores con memoria compartida, las tareas $s_i, i \leq k$ pueden ser implementadas como *threads* del proceso *master*. Las tareas $s_i, i > k$, serán ejecutadas en la nube, y por tanto se llevarán a cabo mediante procesos de acuerdo a un modelo de procesamiento (generalmente multicomputador) propio de la plataforma. El cumplimiento de los requerimientos de calidad de servicio proporcionados por los servicios en red, garantiza la corrección de los resultados en el sistema de tiempo real.

IV. Caso de estudio

Como ejemplo de aplicación de la arquitectura propuesta se propone un sistema de búsqueda y localización de objetos móviles en tiempo real, en un entorno *smart city*. Un posible objetivo del sistema puede ser la localización de personas dependientes (niños o enfermos), o vehículos robados en movimiento. Por tanto, la aplicación puede estar orientada como apoyo a los servicios de seguridad.

El personal de seguridad dispondrá de un sistema empotrado en un dispositivo móvil o integrado en vehículo de seguridad. El sistema analizará en tiempo real la información procedente de un número indeterminado de videocámaras situadas en un espacio geográfico amplio (polígono, barrio o ciudad). El número de fuentes que el sistema empotrado puede procesar en tiempo real es evidentemente limitado, por lo que existirán zonas geográficas en las que la disponibilidad de cámaras supera las capacidades del sistema empotrado. En estos casos, la arquitectura propuesta permite el aprovechamiento de plataformas en la nube para el procesamiento de información procedente de cámaras extra.

La aplicación que se propone como ejemplo debe analizar múltiples *streams* de vídeo simultáneamente. Para ello, consta de un proceso *master* (figura 3) que, a partir de las coordenadas de posicionamiento obtiene una lista de direcciones de *streaming* correspondientes a las cámaras de seguridad disponibles en el entorno de dicha posición. Para cada una de las fuentes localizadas, el proceso *master* creará un *thread* para procesar el *stream* correspondiente empleando los recursos disponibles en el propio dispositivo empotrado. El sistema operativo de tiempo real, planificará la ejecución de los *threads* creados, pero cuando éstos superen en número un cierto valor k, generará un evento al proceso *master*. En respuesta a este evento, el proceso *master* utilizará el servicio de creación de tareas en la nube proporcionado por la arquitectura. Como resultado de este servicio, se ejecutará un proceso en la nube por cada cámara extra, cada uno de los cuales recibirá la dirección correspondiente al *stream* a procesar.

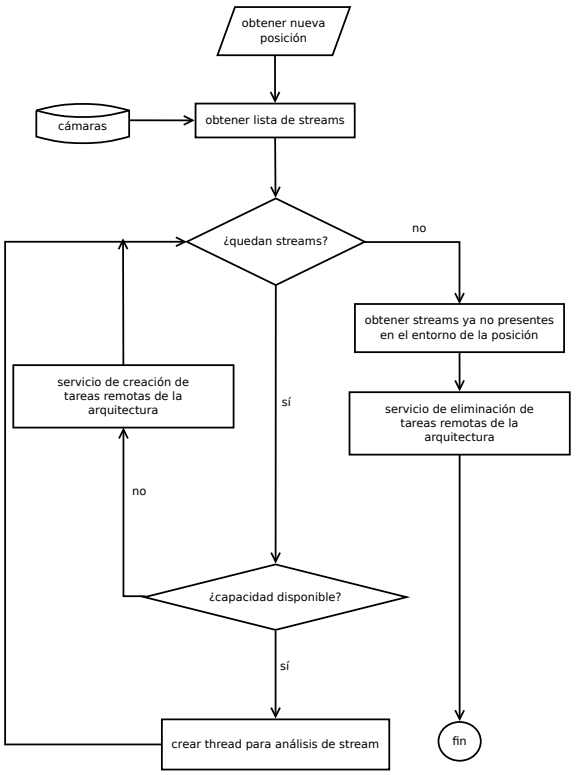

Fig. 3.Descripción del proceso *master* para la aplicación de búsqueda y localización de objetos

Es conveniente aclarar que los *streams* pueden obtenerse ya preprocesados de manera adecuada para reducir el ancho de banda y la complejidad de los algoritmos de localización de objetos. De esta forma, sería posible acomodar más tareas en el sistema empotrado, reduciendo la necesidad de recurrir a los recursos proporcionados en la nube.

V. Conclusión

Este trabajo propone una arquitectura que permite aprovechar los recursos disponibles en la nube en beneficio de aplicaciones con requerimientos de tiempo real y carga variable, que se ejecutan en sistemas embebidos con capacidades limitadas. La arquitectura aporta un conjunto de servicios que abstraen a la aplicación la complejidad relacionada con la integración de recursos de computación en la nube para el desempeño de sus objetivos, adaptándose de manera flexible a sus necesidades de carga cambiantes.

Un RTOS puede garantizar la ejecución en tiempo real de las tareas propias de la aplicación, siempre y cuando su número no exceda un determinado valor. El excedente de tareas que se puede producir en determinadas circunstancias es gestionado por la arquitectura propuesta, empleando plataformas en la nube que proporcionan sus servicios con una cierta

QoS a un coste determinado.

La arquitectura propuesta inicia una línea de trabajo en curso en el ambito del grupo de investigación en arquitecturas y tecnologías de computadores, y pone de manifiesto una serie de retos para ser abordados en trabajos futuros.

Uno de los principales retos es la integración de la arquitectura propuesta con modelos de computacion paralela en entornos heterogéneos, que tomen en consideracion las necesidades de comunicación y sincronizacion de threads y procesos. Estos modelos favorecen el intercambio de informacion entre tareas con independencia de su ubicacion en el sistema empotrado o en la nube, pero su utilizacion tiene implicaciones importantes para el mantenimiento de QoS. Por otro lado, algunas infraestructuras con soporte de QoS proporcionan servicios para ejecucion de procesos de acuerdo a un modelo concreto, orientado a un determinado tipo de aplicaciones. Tal es el caso de aplicaciones que analizan conjuntos complejos de datos (big data), para las que exiten plataformas IaaS específicas que soportan frameworks tales como MapReduce.

Otra posible ampliacion es la consideracion de aplicaciones en las que la computacion en la nube constituye una clara preferencia frente a la ejecución en el sistema empotrado. En estos casos la arquitectura puede ofrecer los recursos locales para aportar, por ejemplo, una solucion de alta disponibilidad en caso de incapacidad de las plataformas en la nube para proporcionar disponibilidad y QoS.

Referencias

[1] J. A. Stankovic, "Real-time and embedded systems," Acm Computing Surveys, vol. 28, no. 1, pp. 205–208, 1996.

[2] F. Pianegiani, "Qos-based dynamic allocation in embedded systems: a methodology and a framework," in IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2011, IEEE Instrumentation and Measurement Technology Conference, pp. 1035–1039.

[3] D. Matschulat, C. A. M. Marcon, and F. Hessel, "A qos scheduler for real-time embedded systems," 2008 9th International Symposium on Quality Electronic Design (ISQED '08), pp. 564–7, 2008.

[4] Higinio Mora Mora, Jerónimo Mora pascual, Juan Manuel García Chamizo, Antonio Jimeno Morenilla, "Real-Time Arithmetic Unit," Real-Time Systems Vol. 34 (1), 53-79, 2006.

[5] S. Penmatsa and A. T. Chronopoulos, "Cost minimization in utility computing systems," Concurrency and Computation-Practice & Experience, vol. 26, no. 1, pp. 287–307, 2014.

[6] B. Fenge, H. Lifeng, B. Li, and W. Junying, "Distributed system for transfusion supervision based on embedded system," 2008 International Conference on Computer Science and Software Engineering (CSSE 2008), pp. 206–10, 2008.

[7] Intel, "Computer-controlled farms change the game in urban agriculture," 2014.

[8] W. K. Edwards and R. E. Grinter, "At home with ubiquitous computing: Seven challenges," in Ubicomp 2001: Ubiquitous Computing, GregoryD. Abowd, Barry Brumitt, and Steven Shafer, Eds., vol. 2201 of Lecture Notes in Computer Science, pp. 256–272. Springer Berlin Heidelberg, 2001.

[9] C. Boneti, F. J. Cazorla, R. Gioiosa, and M. Valero, "Soft real-time scheduling on smt processors with explicit resource allocation," Architecture of Computing Systems - Arcs 2008, Proceedings, vol. 4934, pp. 173–187, 2008.

[10] F. J. Cazorla, A. Ramirez, M. Valero, P. M. W. Knijnenburg, R. Sakellariou, and E. Fernandez, "Qos for high-performance smt processors in embedded systems," Ieee Micro, vol. 24, no. 4, pp. 24–31, 2004.

[11] B. Andersson and G. Raravi, "Real-time scheduling with resource sharing on heterogeneous multiprocessors," Real-Time Systems, vol. 50, no. 2, pp. 270–314, 2014.

[12] M. L. Dertouzos and A. K. L. Mok, "Multiprocessor online scheduling of hard-real-time tasks," Ieee Transactions on Software Engineering, vol. 15, no. 12, pp. 1497–1506, 1989.

[13] N. D. Thai, "Real-time scheduling in distributed systems," Par Elec 2002: International Conference on Parallel Computing in Electrical Engineering, pp. 165–170, 2002.

[14] Z. Bin, W. Jun, and L. Haiqing, "Research of optimal task scheduling for distributed real-time embedded systems," 2008 International Conference on Embedded Software and Systems, pp. 77–84, 2008.

[15] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can wifi deliver?," Ieee-Acm Transactions on Networking, vol. 21, no. 2, pp. 536–550, 2013.

[16] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," Ieee Wireless Communications, vol. 20, no. 3, pp. 14–22, 2013.

[17] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: Making smartphones last longer with code offload," in Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, New York, NY, USA, 2010, MobiSys '10, pp. 49–62, ACM.

[18] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," Eurosys 11: Proceedings of the Eurosys 2011 Conference, pp. 301–314, 2011.

[19] H. Flores and S. Srirama, "Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning," in Proceeding of the Fourth ACM Workshop on Mobile Cloud Computing and Services, New York, NY, USA, 2013, MCS '13, pp. 9–16, ACM.

[20] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," 2012 Proceedings Ieee Infocom, pp. 945–953, 2012.

[21] X. Zhang, A. Kunjithapatham, S. Jeong, and S. Gibbs, "Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing," Mobile Networks & Applications, vol. 16, no. 3, pp. 270–284, 2011.

[22] V. March, Y. Gu, E. Leonardi, G. Goh, M. Kirchberg, and B. Sung Lee, "$\hat{\mathbf{I}}\frac{1}{4}$ cloud: Towards a new paradigm of rich mobile applications," Procedia Computer Science, vol. 5, no. 0, pp. 618 – 624, 2011.

[23] A. Saarinen, M. Siekkinen, Y. Xiao, J. K. Nurminen, M. Kemppainen, and P. Hui, "Smartdiet: Offloading popular apps to save energy," Acm Sigcomm Computer Communication Review, vol. 42, no. 4, pp. 297–298, 2012.

[24] A. Khairy, H. H. Ammar, and R. Bahgat, "Smartphone energizer: Extending smartphone's battery life with smart offloading," 2013 9th International Wireless Communications and Mobile Computing Conference (Iwcmc), pp. 329–336, 2013.

[25] E. Gelenbe, R. Lent, and A. Nunez, "Self-aware networks and qos," Proceedings of the Ieee, vol. 92, no. 9, pp. 1478–1489, 2004.

[26] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. O. Foghlu, and W. Donnelly, "Towards autonomic management of communications networks," Ieee Communications Magazine, vol. 45, no. 10, pp. 112–121, 2007.

[27] T. Frantti and M. Majanen, "An expert system for real-time traffic management in wireless local area networks," Expert Systems with Applications, vol. 41, no. 10, pp. 4996–5008, 2014.

[28] Z. Hamid and F. B. Hussain, "Qos in wireless multimedia sensor networks: A layered and cross-layered approach," Wireless Personal Communications, vol. 75, no. 1, pp. 729–757, 2014.

[29] S. Delamare, G. Fedak, D. Kondo, and O. Lodygensky,

"Spequlos: a qos service for hybrid and elastic computing infrastructures," *Cluster Computing-the Journal of Networks Software Tools and Applications*, vol. 17, no. 1, pp. 79–100, 2014.

[30] S. Kianpisheh and N. M. Charkari, "A grid workflow quality-of-service estimation based on resource availability prediction," *Journal of Supercomputing*, vol. 67, no. 2, pp. 496–527, 2014.

[31] P. Nadanam and R. Rajmohan, "Qos evaluation for web services in cloud computing," *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT 2012)*, pp. 8 pp.–8 pp., 2012.

[32] J. M. Pedersen, M. T. Riaz, J. Celestino, B. Dubalski, D. Ledzinski, and A. Patel, "Assessing measurements of qos for global cloud computing services," *Proceedings of the 2011 IEEE 9th International Conference on Dependable, Autonomic and Secure Computing (DASC 2011)*, pp. 682–9, 2011.

[33] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, "Qos ranking prediction for cloud services," *Ieee Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1213–1222, 2013.

[34] L. Shou-Yu, T. Dongyang, C. Tingchao, and W. C. C. Chu, "A qos assurance middleware model for enterprise cloud computing," *Proceedings of the 2012 IEEE 36th IEEE Annual Computer Software and Applications Conference Workshops (COMPSACW)*, pp. 322–7, 2012.

[35] L. Yan, X. Cheng, and W. Lidong, "A generalized multiframe real-time task model upon heterogeneous multiprocessors system," *Journal of Theoretical and Applied Information Technology*, vol. 47, no. 1, pp. 318–25, 2013.

[36] F. Almeida, D. Gonzalez, and L. M. Moreno, "The master-slave paradigm on heterogeneous systems: A dynamic programming approach for the optimal mapping," *Journal of Systems Architecture*, vol. 52, no. 2, pp. 105–116, 2006.