International Journal of Soft Computing and Engineering (IJSCE)
ISSN: 2231-2307, Volume-5 Issue-1, March 2015

A Cloud Robotics Architecture with Applications to Smart Homes

José M. Reyes Álamo, Aparicio Carranza

Abstract*—Cloud computing is a computational model in which interconnected computers over the Internet work together toward offering greater processing power and storage capabilities than stand-alone solutions. The use of the cloud has found application in a diversity of fields including robotics and mobile computing. This has resulted in the emergence of areas like cloud robotics, a paradigm where robots rely on the cloud to perform their heavy computations and for their storage needs while focusing on simpler computation tasks. The mix of mobile devices and the cloud has created the field of mobile cloud computing (MCC) where mobile devices like smartphones and tablets focus on data gathering and simple processing tasks while using the cloud for complex computations and greater storage. In this paper we review several mobile cloud robotics architectures that combine these concepts. We provide a background of the different technologies used to develop these solutions. We present a prototype implementation of one of the architectural models and also show some practical applications of it using a Smart Home environment as an example.

***Index Terms*— Cloud computing, cloud robotics, mobile cloud, smart home.**

I. INTRODUCTION

Cloud computing refers to a computational model where tasks are executed by several interconnected computers offering greater capabilities in terms of processing power and storage than stand-alone solutions. Researchers have applied the principles of cloud computing to the fields of robotics as well as mobile computing. This has resulted in the emergence of topics such as cloud robotics and the mobile cloud. Cloud robotics refers to the use of cloud computing for robotics and its applications [1], [2]. These applications involving robots usually require lots of computations and many times these are limited by the capability of the robot. By relying on the cloud applications have access to significantly more computational resources. This makes it possible for robotics applications to tackle more complex problems than before because of the extra capabilities that the cloud offers. Mobile and embedded devices have less computational capabilities, memory, and resources than stand-alone computers. Other limitations of these devices include the instructions set, battery life and overall performance.

Manuscript Received on February 2015.
Dr. José M. Reyes Álamo, Computer Engineering Technology, CUNY/ New York City College of Technology, Brooklyn, NY, USA.
Dr. Aparicio Carranza, Computer Engineering Technology, CUNY/ New York City College of Technology, Brooklyn, NY, USA.
Acknowledgement: Support for this project was provided by a PSC-CUNY Award, jointly funded by The Professional Staff Congress and The City University of New York.

The concept of mobile cloud computing (MCC) refers to the use of mobile devices such as smartphones, tablets, and embedded devices to detect phenomena, while depending on the cloud for performing computations and satisfying storage needs [3]. In this paper we review several mobile cloud robotics architectures that integrate concepts from cloud robotics and mobile cloud. We select one of the architectural models and developed a prototype implementation. We provide background information on some of the tools used such as the Lego NXT robot and the Raspberry Pi. This architecture allows the robots to communicate with each other and with a mobile server. This server also communicates with a cloud computing platform. We believe this architecture provides a platform that eases the development and deployment of applications and that it will be especially useful for research projects involving mobile applications and robotics. We also show some examples of the usability of the architecture using a Smart Home environment as an example. The rest of the paper is organized as follows: section II presents related work and background information of the different tools used in this project. Section III reviews the mobile cloud robotics architectures, while section IV provides the details of the prototype implementation of the architecture we believe better suits the needs of this project. Section V provides examples of applications of this architecture using a Smart Home environments as an example. Finally section VI provides our conclusions and future work.

II. RELATED WORK

In this paper we propose a mobile cloud robotics architecture that will facilitate the research, development, and testing of applications in fields such as mobile cloud and cloud robotics. In this work several technologies are being used including the Lego NXT robot, the Lego Java Operating System (leJOS), a Python interpreter, the Raspberry Pi, Phidgets, and the Seattle cloud computing platform. The next paragraphs provide background information about these.

A. Lego NXT Robot

The Lego NXT robot contains a programmable brick with a set of sensors and servo motors. The Lego NXT robot comes with a 32-bit ARM main processor, 256 kilobytes of flash memory for program storage, and 64 kilobytes of RAM for data storage during program execution. The Lego NXT robot contains a programmable brick, servo motors, light sensor, ultrasonic sensor, sound sensor, and touch sensors. The NXT brick contains three output ports that are used to power the servo motors. It also has four input ports that are used to connect the sensors. To get data from the sensors, an extra processor is included that has 4 kilobytes of flash memory and 512 bytes of RAM. For connectivity the Lego NXT robot supports the Bluetooth protocol for wireless, and USB for wired [4].

B. *Lego Java Operating System (leJOS)*

The Lego Java Operating System (leJOS) [5] is an open source firmware replacement for the Lego NXT robot. This OS provides an implementation of a Java Virtual Machine suited for the Lego Robot processor. leJOS supports having an alternative runtime environment that allows programming the robots using the popular programming language Java. leJOS includes a library of Java classes that implement the leJOS NXJ Application Programming Interface (API). This API consists of a library of classes for desktop programming that communicates with the brick via Bluetooth or USB. It includes as well a set of tools for debugging, firmware replacement, compiling, and uploading programs into the robot's brick. Some of the features supported by leJOS include: object oriented programming support, recursion, threads and synchronization, Java basic types such as float and String, most of the java.lang, java.util and java.io classes, arrays, exception handling, and a well-documented robotics API [6].

C. *Python*

Python is a high-level interpreted programming language that supports multiple programming paradigms and focuses on readability. In this project we used several tools to allow execution of Python code within the Lego NXT robot. These tools are Jython, PyMite, and NXT-Python. Jython is a Python interpreter written in Java [7]. PyMite is a lightweight Python interpreter especially designed to work on embedded devices [8]. NXT-Python is a Python library that provides commands to control the Lego NXT brick using Bluetooth or USB [9]. Executing Python code is important for using other tools including the Seattle cloud computing platform and communicating with the Raspberry Pi.

D. *Raspberry Pi*

The Raspberry Pi [10] is a credit card-sized single-board computer developed by the Raspberry Pi Foundation in the United Kingdom. The original intention of the Raspberry Pi was promoting the teaching of basic computer science in schools. The Raspberry Pi was designed to be an inexpensive yet useful computer. It is based on the Broadcom BCM2835 system on a chip (SoC) and comes with an ARM1176JZF-S 700 MHz processor and a VideoCore IV GPU. The latest models are known as Model B and Model B+. They come up with 512 MB of RAM, an improvement over the original Model A that has 256 MB of RAM. For storage, the Raspberry Pi supports SD or MicroSD cards. The operating system of choice is Linux. There are many distributions of Debian and Arch Linux ARM available for download, being Raspian the recommended OS. The Raspberry Pi has many tools and programming languages available such as Python, C, C++, Java, Perl and Ruby. All this for a small price, between $30 and $40 depending on the model at the time of this writing.

E. *Phidgets*

A phidget [11] is composed of two words: physical and widget. It is best described as a physical implementation of a GUI widget (e.g. an on-screen dial widget implemented physically as a knob). Phidgets rely on the use of USB ports to provide low-cost sensors and actuators. They were designed primarily for experimenting with alternative physical computer interaction systems. However because of their flexibility and ease of use, they have been widely used in fields like robotics and smart homes. Their name represents the attempt to build a physical analogue to the software of the widgets we find in GUI systems. This way complex physical systems can be constructed by combining a set of simpler components. These phidgets connect to a PC which controls them via the Phidgets Application Programming Interface (API). This API greatly simplifies the complexity of handling these physical devices so that developers can focus on the implementation of applications. Phidgets applications can be developed for a number of operating systems including the most popular versions of Windows, Linux, and Mac OS. There are also multiple programming languages supported including C, C++, C#, Java, and Python.

F. *Seattle Cloud Computing Platform*

Seattle is a very useful educational platform for teaching concepts such as cloud computing, distributed systems, and networking in general. Seattle is a community-driven effort supported by resources donated by its users and therefore free to use [12]. To use Seattle the user installs the software onto their computer. After installation the user must assigns a portion of the computer's resources to share it with the platform. The user will also receive resources shared from other computers around the world that are also using the platform. Seattle is available on different operating systems including Windows, Mac OS, Linux, and mobile devices such as Android phones and jail broken iPhones [13]. Programs running on the Seattle platform are sandboxed and securely isolated from the rest of the programs running on the same computer or device. By having hard resources guarantees it makes it more difficult for malicious or erroneous code to by-pass them. Seattle is widely deployed on hundreds of computers around the world, therefore users are able to test and run their programs on computers all over the Internet. To create Seattle application developers must use the programming language RePy which is a subset of Python. RePy is simple but expressive enough to allow developer to create interesting applications. Code written in RePy is portable across all supported platforms.

III. MOBILE CLOUD ROBOTICS ARCHITECTURE

In our previous work [14] we proposed several mobile cloud robotics architectures. In this paper we select one of the models, enhance it, and implement it. Also we present some of the applications within a Smart Home environment in which this architecture will be useful. What follows is a summary of the different architectural models we considered. The first proposed model was the Lego NXT Mobile Cloud Robotics Architecture. This first architecture consists of a Lego NXT robot with sound, ultrasonic, touch, and motion sensors. This robot also has several servo motors. The Lego NXT robot original operating system is replaced with leJOS. A Python interpreter such as Jython or PyMite is installed on the unit [5]. There is also a server that sits between the Lego NXT robot and the cloud. The Lego NXT robot communicates with the server via Bluetooth. This server is responsible for sending and receiving commands from the cloud and for performing some local computations. Among the advantages of this first model is that it will maximize the use of the Lego NXT robot by executing Java, Python, and RePy code inside the unit while relying on the cloud for complex computations and storage. However this might come at cost when it comes to performance as the processor of the Lego NXT robot is not very powerful, the unit has limited memory, and by relying on Bluetooth only the connectivity may be weak or lost possibly creating situations where applications get disrupted. The second proposed architectural model was an Android Powered Lego NXT Mobile Cloud Robotic Architecture with a similar configuration as the first one, but with the robot enhanced with an Android device attached to it. This Android device would be in charge of tasks such as execution of Python and RePy code and communicating with the server that sits between the Lego NXT robot and the cloud platform. The Lego NXT robot communicates with the Android device using Bluetooth, while the Android device communicates with the server using Bluetooth or WiFi as Android devices support both protocols. The server would be responsible for sending and receiving commands from the cloud as well as performing some local computations. This second model that enhances the Lego NXT robot by attaching an Android device at first seems like a good solution because it adds extra computational and communication capabilities. However this architecture adds another level of complexity by having two different devices. Experts in both Android and Lego NXT robots would be needed for development, testing, maintenance, and troubleshooting of applications. Also adding Android devices would increase the cost and the development time. The third proposed architecture is a hybrid model that combines elements of the previous two. Under this architecture only a subset of the Lego NXT robots would be attached to an Android device while the others will not. Under this hybrid architecture the Lego NXT robots would communicate with each other while the Android powered devices would have extra connectivity. Under the hybrid model there is no exclusive dependence on the Android device as the Lego NXT robots would be able to handle most of the communication and computation themselves, but those enhanced with an Android device attached to them would count with the extra capabilities. Another consideration is that Android devices could communicate with the cloud platform as well. So even though communication with the cloud computing platform would be the responsibility of the server, in case that the server cannot be reached, the Android devices might take over this task eliminating the potentially single point of failure found in the first model. Another important consideration is cost. The first model offers the lowest cost but also less connectivity and computational power, while the second model offers greater connectivity and computational power but at a greater cost and complexity as each Lego NXT robot has an Android device attached to it. Under the hybrid model only a subset of the Lego NXT robots would be powered by an Android device, resulting in improving connectivity and computational capabilities with respect to the first model while at the same time reducing the cost and complexity with respect to the second model. We believe that this hybrid model offers a good tradeoff between the simplicity and cost of the first model and the connectivity and computational capabilities of the second. In this work we preserve most elements of the hybrid model as we believe it offers a balance between performance and cost. One change we made in this project was to power the Lego NXT robots with Raspberry Pi devices instead of Android. This way we achieved similar results and at the same time reduced the cost. This architecture can be seen in *Figure 1* below.

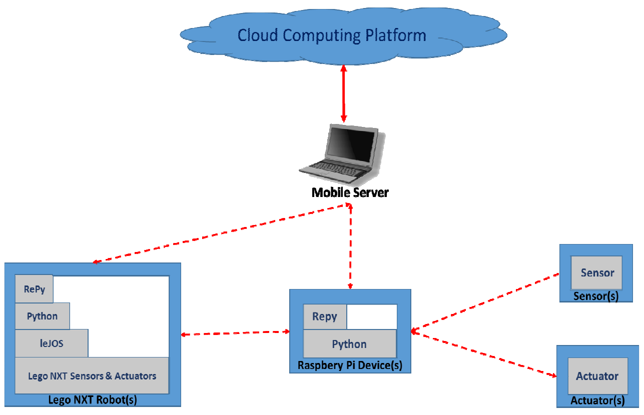

Figure 1: Hybrid Lego NXT Mobile Cloud Robotics Architecture

A network of interconnected units implementing this architecture is depicted in *Figure 2*, with the NXT robots powered with a Raspberry Pi identified with the π symbol.

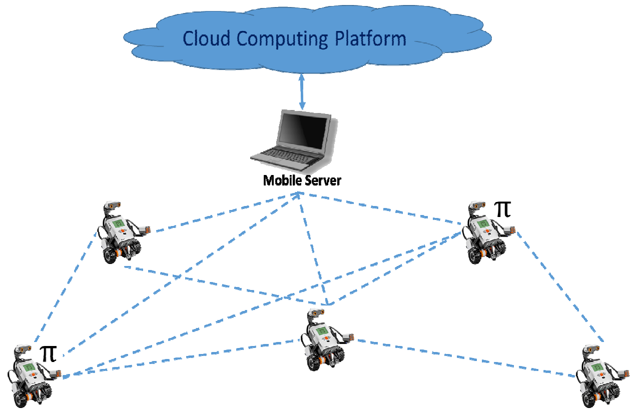

Figure 2: Network of Units under the Hybrid Lego NXT Mobile Cloud Robotics Architecture

IV. PROTOTYPE IMPLEMENTATION

To test the feasibility of our proposed architecture, we developed a prototype implementation. In our prototype implementation we included the following components: Lego NXT robot, raspberry pi, phidgets, a cloud server, and a laptop. There is a Lego NXT robot that is connected to a raspberry pi to enhance its capabilities. We call this particular robot NXTPi. As the raspberry pi is a more powerful device all computations and communications are carried by it. The NXT robot with the raspberry pi also has the phidgets devices attached to it. These phidgets provide several sensor and for this particular implementation we tested the rotation, temperature, and the light sensors. We also have several stand-alone NXT robots communicating with the NXTPi and the server. These robots use the Bluetooth protocol to communicate among each other when they are within range. Now we describe one of the test we performed in our prototype implementation. A python script was developed and ran in the main unit, the NXTPi. This unit reads a value from the phidgets rotation sensor via its USB interface and stores it into memory. After that the NXTPi connects to another NXT unit via its Bluetooth interface. The NXT unit provides a reading from its sound sensor. This value is stored as well in the NXTPi memory. Now with the values obtained from different devices, the NXTPi disseminates these values to other robots within range in order to reach the cloud server. In order to do that, a connection to a peer robot or to the mobile server is stablished. The mobile server has a script that waits for messages to be sent and expects the sensor readings from the NXTPi unit. The NXTPi then sends the phidgets reading and the sound sensor reading to the mobile server. The mobile server receives these values and sends and acknowledgement to the NXTPi. After receiving the values these must be sent into the cloud. The mobile sever then establishes a connection to the cloud server. After stablishing that connection the values are forwarded into the cloud server, processed, and stored. The cloud server also sends an acknowledgement to the mobile server that the values were successfully received. This way our prototype implementation shows that it is possible to enhance an NXT robot using a raspberry pi, connect another device to it such as a phidget in order to gather more context data, establish a connection to a peer around in order to reach the server, forward these values to the cloud for processing and storage, and finally send acknowledgments for successful communications.

V. Applications To Smart Homes

A Smart Home is a house that uses technology in order to enhance the quality of life of the resident and to assist in the activities of daily living [15]. The devices and programs mentioned previously are used in our Smart Home prototype to show the feasibility of them. A very important application in Smart Home Environments is context-awareness and detection of phenomena. Usually sensors are deployed at fixed strategic locations within a Smart Home. With our solution presented in this paper we have a set of robots carrying the sensors and in this way sensing is not limited to just certain areas of a house. Instead the robots are able reach other areas because of their mobility. We believe this solution can improve context-awareness and be use for more advanced detection mechanisms. Another application we are currently implementing is a coordinated detection mechanism. This detection mechanism consists of a robot that detects a phenomena and requests help from other robots in order to confirm it. By communicating with robots around it as well as sending a message to the server and the cloud, other robots can be instructed to check the environment at that particular location to confirm whether the phenomena is occurring. This can be useful especially for applications that detect hazardous conditions such as a smoke, fire or noise.

VI. Conclusions & Future Work

In this work we reviewed different proposed architectures for supporting a mobile cloud robotics architecture using components such as the Lego NXT robot and the raspberry pi. We summarized three different architectural models: the Lego NXT Mobile Cloud Robotics Architecture, the Android Powered Lego NXT Mobile Cloud Robotics Architecture, and the Hybrid Lego NXT Mobile Cloud Robotics Architecture. We argue that the Hybrid Lego NXT Mobile Cloud Robotics Architecture offers the best tradeoff between the simplicity and cost of the first model and the connectivity and computational capabilities of the second. We modified this architecture using a raspberry pi instead of an Android device. We implemented a prototype of this architecture and showed an example of its use. Also we presented how this platform can be use within Smart Home environments with an applications for context-awareness and detection of phenomena. Future work includes testing the performance of this architecture especially when a lot of data is being generated and to develop more applications that make use of it.

References

[1] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a Service in Cloud Computing," in *2010 Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE)*, 2010, pp. 151–158.

[2] G. Hu, W.-P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE Netw.*, vol. 26, no. 3, pp. 21–28, 2012.

[3] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wirel. Commun. Mob. Comput.*, p. n/a–n/a, 2011.

[4] M. W. Lew, T. B. Horton, and M. S. Sherriff, "Using LEGO MINDSTORMS NXT and LEJOS in an Advanced Software Engineering Course," in *2010 23rd IEEE Conference on Software Engineering Education and Training (CSEE T)*, 2010, pp. 121–128.

[5] R. U. Pedersen, J. Nørbjerg, and M. P. Scholz, "Embedded programming education with Lego Mindstorms NXT using Java (leJOS), Eclipse (XPairtise), and Python (PyMite)," in *Proceedings of the 2009 Workshop on Embedded Systems Education*, New York, NY, USA, 2009, pp. 50–55.

[6] "LeJOS, Java for Lego Mindstorms." [Online]. Available: http://www.lejos.org/. [Accessed: 22-Oct-2013].

[7] "The Jython Project," Nov-2013. [Online]. Available: http://www.jython.org/. [Accessed: 11-Nov-2013].

[8] "PyMite: python-on-a-chip," 2013. [Online]. Available: http://code.google.com/p/python-on-a-chip/. [Accessed: 11-Nov-2013].

[9] "nxt-python - A pure-python driver/interface/wrapper for the Lego Mindstorms NXT robot. - Google Project Hosting." [Online]. Available: http://code.google.com/p/nxt-python/. [Accessed: 11-Nov-2013].

[10] J. D. Brock and R. F. Bruce, "Sensing the World with a Raspberry Pi," *J Comput Sci Coll*, vol. 30, no. 2, pp. 174–175, Dec. 2014.

[11] S. Greenberg and C. Fitchett, "Phidgets: Easy Development of Physical Interfaces Through Physical Widgets," in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, New York, NY, USA, 2001, pp. 209–218.

[12] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: a platform for educational cloud computing," in *Proceedings of the 40th ACM technical symposium on Computer science education*, New York, NY, USA, 2009, pp. 111–115.

[13] "Seattle." [Online]. Available: https://seattle.poly.edu/html/. [Accessed: 23-Oct-2013].

[14] J. M. Reyes Álamo, M. Benito, and A. Carranza, "Towards An Architecture for Mobile Cloud Robotics," in *IHART*, Las Vegas, NV, 2013, vol. 31, pp. 391–398.

[15] N. Noury, G. Virone, P. Barralon, J. Ye, V. Rialle, and J. Demongeot, "New trends in health smart homes," in *Enterprise Networking and Computing in Healthcare Industry, 2003. Healthcom 2003. Proceedings. 5th International Workshop on*, 2003, pp. 118–127.

Dr. José M. Reyes Álamo is an Assistant Professor in the Department of Computer Engineering Technology at the New York City College of Technology of the City University of New York. His research interest include smart homes, embedded devices, cloud computing, and software engineering. He received his Ph.D. in Computer Science from Iowa State University and his B.S. in Computer Science from the University of Puerto Rico at Bayamón.

Dr. Aparicio Carranza is an Associate Professor in the Department of Computer Engineering Technology at the New York City College of Technology of the City University of New York. His research interest include virtualization & cloud computing and software defined networks. He received his Ph.D. in Electrical Engineering from the Graduate School of CUNY and both his M.S.E.E. and B.S.E.E. in Electrical Engineering from the City University of New York.