An Infrastructure for Robotic Applications as Cloud Computing Services

Carla Mouradian[1*], Fatima Zahra Errounda[2*], Fatna Belqasmi[1#], Roch Glitho[3*]
*Concordia University, Montreal, Canada
[#]Zayed University, Abu Dhabi, United Arab Emirates
[1*]ca_moura@encs.concordia.ca, [2*] f_errou@encs.concordia.ca, [1#]fatna.belqasmi@zu.ac.ae, [3*]glitho@ciise.concordia.ca

***Abstract*— Robotic applications are becoming ubiquitous. They are widely used in several areas (e.g., healthcare, disaster management, and manufacturing). However, their provisioning still faces several challenges such as cost and resource usage efficiency. Cloud computing is an emerging paradigm that may aid in tackling these challenges. It has three main facets: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). This paper focuses on the IaaS aspects of robotic applications as cloud computing services. It proposes an architecture that enables cost efficiency through virtualization and dynamic task delegation to robots, including robots that might belong to other clouds. Overlays and RESTful Web services are used as cornerstones. A prototype is built using LEGO Mindstorms NXT as the robotic platform, and JXTA as the overlay middleware. Related work is reviewed, the functional entities and interfaces of the architecture are described, and the prototype architecture is presented along with the implemented scenario.**

Keywords-component; Robotic applications, cloud computing, Infrastructure as a Service (IaaS), task delegation, task allocation

I. INTRODUCTION

According to ISO 8373 [1], a robot is an actuated programmable mechanism that can perform intended tasks by moving in its environment. Robots can be used in a plurality of applications (e.g., search and rescue operations in disaster management, surgery and logistics in healthcare). However, the cost efficient provisioning of these applications remains a big challenge, as robots' resources are still seldom used in an efficient manner.

Cloud computing is an emerging paradigm with inherent benefits such as cost efficiency [2]. It has three key facets: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The infrastructure is the actual dynamic pool of virtualized resources used by applications. Virtualization allows the abstraction of actual physical computing resources into logical units, enabling their efficient usage by multiple independent users [3]. Its role is key to resource efficiency. Virtualization can be performed at both node and network levels.

In this paper we define robot node level virtualization as the mechanisms that enable multiple applications to reside in and run concurrently on a single robot, analogous to the definition given in reference [4] for wireless sensor networks (WSN). On the other hand, we define robot network level virtualization as the dynamic formation of subsets of robot nodes, with each subset dedicated to a certain application at a given time. This is also analogous to the definitions used in the WSN world [5].

This paper focuses on the IaaS aspects of robotic applications as cloud computing services. We propose an architecture that enables cost efficient robotic application provisioning through virtualization and the dynamic delegation of tasks to robots, including robots that might belong to other clouds. Related work is reviewed in the next section, followed by the proposed architecture. The fourth section is devoted to the implementation. We conclude in the last section.

II. RELATED WORK

This section starts with the motivating scenarios and the requirements. The state of the art is then reviewed in the light of the requirements.

A. Motivating scenarios

The first scenario is wildfire suppression. We consider a wildfire suppression robotic application that detects and suppresses wildfires using a fleet of robots deployed in the forest. The robots have different capabilities. Some are equipped with cameras which allow them to supervise the fire area and send notification if the fire is growing. Other robots are equipped with arms that help them grab the extinguishers and suppress the fire using water and foam. Still others can detect obstacles and remove them. We assume that these robots do not necessarily belong to the same business entity. In a cloud environment, this means there are potentially several IaaSs owned by different business entities and that host robots with different capabilities. Using these robots in an efficient manner is of paramount importance.

Another scenario could be a subway infrastructure that shows signs of degradation. Mobile robots are deployed in the subway to detect cracks and corrosion. The robots have different capabilities using different sensors and actuators. Some are equipped with more than one sensor, which allows them to perform more than one task. For example, a robot can provide video stream for the real-time video application, and it can cover an area requested by the patrolling unit using different sensors for each application. A group of robots can also be used by the patrolling application if the requested area

978-1-4799-3459-1/14/$31.00 ©2014 IEEE

is too large for one robot to cover. If real-time videos are requested at the same time that the robots are covering an area by patrolling, one set of the same robots may be used to provide the requested information. We again assume that all robots do not necessarily belong to the same business entity, which means that there are several IaaSs owned by different business entities that host robots with different capabilities. Here as well, it is critical to make the most efficient use of the robots.

B. Requirements

We divide the requirements into three groups, requirements of the IaaS, of the Robot and of the overall architecture.

1) Requirements of the IaaS:

The first requirement is that the IaaS should be able to delegate tasks to robots that belong to other clouds. This is very important, since the number and/or the capabilities of the local robots may not be sufficient for a given task. Second, the IaaS should support network level virtualization, since we are dealing with dynamic environments; we need to dynamically dedicate subset of robots to perform a specific task.

2) Requirements of the Robots:

Robots belonging to the IaaS should support node level virtualization. This allows the robot to execute more than one application at the same time without interference between these applications.

3) Requirements of the Overall Architecture.:

The first requirement for the overall architecture is the scalability, e.g., the overall solution should scale in terms of adding new robots. The second requirement is that the interaction interfaces of the proposed architecture should be based on standard technologies. The IaaS should also provide isolation, i.e., allow more than one robotic application to be run on the same IaaS and ensure that the execution of each application can be isolated from the execution of the other(s).

Another requirement is that the IaaS should be able to support heterogeneity, which means that the solution should be applicable to a wide variety of heterogeneous robots. The last requirement is extensibility; the overall architecture should be extensible in terms of adding new functionalities to the infrastructure (e.g., fault management).

C. The shortcomings of the state of the art

There are several studies that present the IaaS aspects of robotic applications as cloud computing services. The architecture proposed in [6] relies on Service Oriented Architecture (SOA) [7]. It decouples robots' sensing and actuating capabilities and the applications that use them by offering these capabilities as SOAP-based services. That architecture supports standard interfaces but does not address our requirements for the Robot and the Iaas, e.g., virtualization and heterogeneous robots support. Reference [8] improves the architecture presented in [6]; it supports network level virtualization by adding a mapping layer on top of the robots' infrastructure and it supports heterogeneous robots, but it does not meet our robot requirement, and delegating tasks to robots belonging to other clouds is not discussed.

Reference [9] provides a solution for node level virtualization by providing the possibility to run two operating systems on the same platform, but it does not address our requirements of the IaaS and of the overall architecture.

Reference [10] provides a framework that relies on SOA, and offloads computationally-intensive algorithms from the robots to the framework. Reference [11] provides a framework to support heterogeneous and low cost robots, and [12] proposes a distributed service framework to integrate various devices including robots with internet services. These three works all support standard interfaces for communication, but none of them discusses our robot or IaaS requirements. A partial exception is [11], which provides the possibility to have heterogeneous robots using Robotic Operating System [18], but it does not meet our other requirements of the IaaS, the robot, or of the overall architecture.

III. Proposed Architecture

This section presents our overall architecture and discusses the main interfaces and procedures. The architectural principles are discussed first, followed by the functional entities, interfaces, and procedures.

A. Architectural Priniciples

The first principle is the use of peer-to-peer (P2P) overlays for the communication between different IaaSs. We used P2P overlay because it provides distributed architectures, self-re-organization, and scalability [13].

The second principle is that the interaction interfaces of the IaaSs are REpresentational State Transfer (REST)-based. We selected REST because it is lightweight, standards-based, and can support multiple data representations (e.g., plain text, JSON, and XML). More information on REST can be found in [14].

B. Overall Architecture and Functional Entities

The overall architecture is depicted in Fig. 1. Our architecture is mainly comprised of a P2P overlay and a robotic cloud, which includes the IaaS and the gateway, as shown in Fig. 2. P2P overlay is used as the interaction network between different IaaSs. The IaaS interacts with the PaaS to receive a task. The IaaS also selects robots in a cost efficient manner, divides the task into sub-tasks, and assigns the sub-tasks to the robots. Sub-tasks are assigned to specific robots by IaaS via gateways. It should be noted that some of the robots might belong to other clouds. The IaaS discovers them through the overlay and assigns them tasks via their IaaS through the same overlay. The gateway caters to heterogeneity by mediating between the standard interface supported by the IaaS and the proprietary interfaces supported by the robots.

In the overlay we have only one type of node, the Virtual Robotic Cloud (VRC). Each VRC represents one robotic cloud in the overlay. A VRC communicates with the robotic cloud that pertains to its corresponding cloud, and with the other nodes in the overlay network. We defined a protocol for the overlay that uses the messages summarized in Table I between the overlay nodes. When a VRC receives a Discover message, it asks the corresponding robotic cloud for the idle robots that

belong to it, and sends a response message back to the requesting VRC. When a VRC receives a task assignment request, it sends it on to its robotic cloud with the robot id that was selected. When a VRC receives a Notification message, it notifies the corresponding cloud that the robots belonging to other clouds have finished their sub-task. Whenever a VRC receives a message it sends back an acknowledgment to ensure the receipt of the message; it is a request/response protocol.

The IaaS includes six entities: the request handler that handles the request coming from the PaaS, the virtualization engine which performs network level virtualization by using an appropriate algorithm for coalition formation in Multi-Robot systems [15] in order to choose the most suitable group of robots for a given task, the robot discovery engine which is responsible for discovering local and external idle robots, the task delegator engine which sends task assignment requests to other clouds, the robot information repository, which holds a list of robots belonging to the IaaS, as well as their information (e.g., capabilities, states), and the robot monitoring engine which monitors the robots and updates the repository when a robot finishes or fails to finish its sub-task.

TABLE I Messages Exchanged in the Overlay

Message Name	Message Description	Message Address
Discover	Discovers idle robots in other clouds. Sent by a VRC to the overlay after receiving a request from a Robotic Cloud.	Broadcast
Task Assignment Request	Sends task assignment requests to robots belonging to other clouds and implicitly subscribes for event notification. Sent by a VRC to the specified VRCs.	Unicast
Notification	Sends notification when robots belonging to the subscribed node finish their job. Sent by a VRC to the VRC that subscribed for this event notification.	Unicast
Response	Sends a response to another node. VRCX sends a response to VRCY when it receives a Discover message from VRCY.	Unicast

C. Interfaces

The interface from the IaaS to the PaaS (R1) and between the IaaS and the Gateway (R2) are REpresentational State Transfer (REST)-based. The interfaces between the gateway and the robots (R3) are proprietary interfaces supported by the robots, varying with each robot provider.

Table II summarizes the proposed REST interface for the communication between the IaaS and the Gateway. It defines the resources on the gateway side and those on the IaaS side.

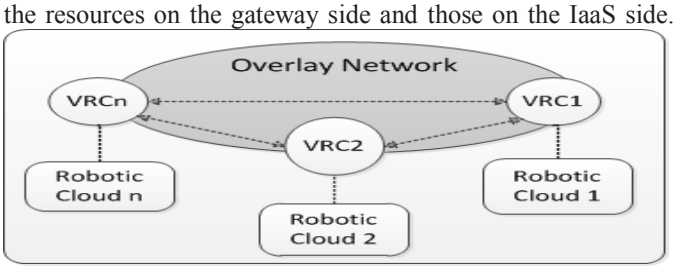

Fig. 1. Overall Architecture

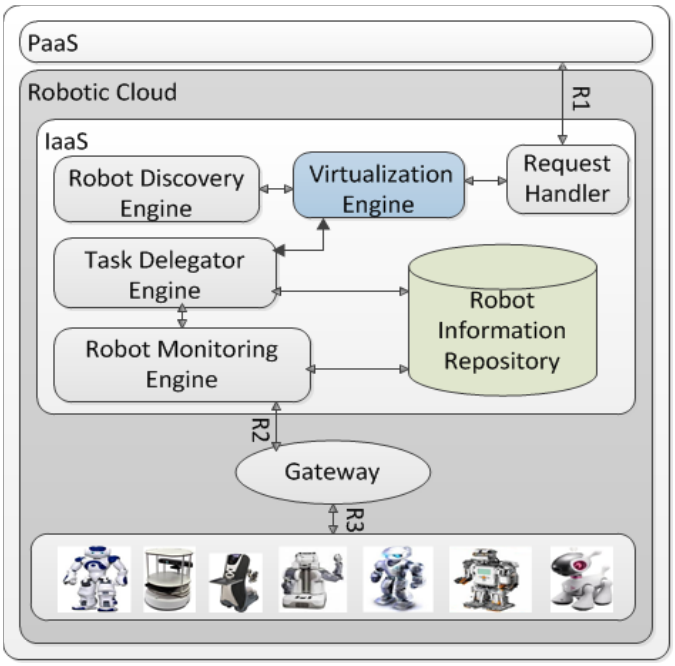

Fig. 2. Robotic Cloud

The gateway-side resources are used to reserve robot resources when adding new robots, and to reserve task resources when sending a task to a specific robot or group of robots, or to modify task resources for an ongoing task. These resources are also used to get the list of robots to update the repository and to get the state of an ongoing task. The IaaS-side resources allow the gateway to send notification to the IaaS when robots finish their job or when there is a failure in one of the robots.

TABLE II Resources between the IaaS and the Gateway

Resource	Operation	Http Action
List of Robots	Create: Add a robot	POST:/Robots
	Read: get a list of all robots to update the repository	GET:/Robots
List of Groups	Create: Add a group of robots	POST:/Robots/Groups
Specific Group	Read: get information about a specific group	GET:/Robots/Groups/{GroupId}
	Delete: delete the specific group after the task is finished	DELETE:/Robots/Groups/{GroupId}
List of Tasks for specific Robot	Create: Send the task to a specific robot	POST:/Robots/{RobotId}
Specific Task Specific Robot	Read: Get the state of the task from a specific robot	GET:/Robots/{RobotId}/{TaskId}
	Update: Change the status of an ongoing task on a specific robot	PUT:/Robots/{RobotId}/{TaskId}
List of Tasks for Specific Group	Create: Send the task to a specific group	POST:/Robots/Groups/{GroupId}
Specific Task Specific Group	Read: Get the state of the task from a specific group	GET:/Robots/Groups/{GroupId}/{TaskId}
	Update: Change the status of an ongoing task on a specific group	PUT:/Robots/Groups/{GroupId}/{TaskId}
Notification of failure	Create: Send notification when there is a failure in any robot	POST:/FailureNotification
Notification of State change	Create: Send notification when a robot finishes a task	POST:/TaskNotification

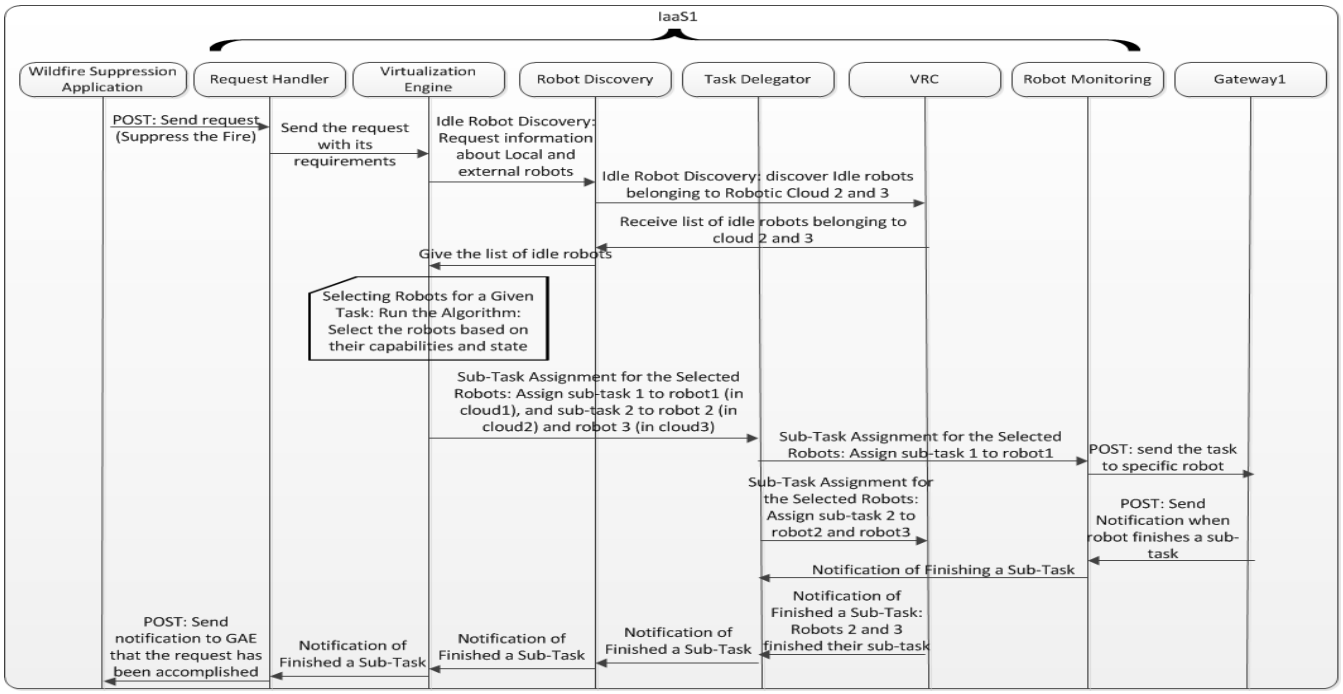

Fig. 3. Sequence Diagram

D. *Procedures*

This section discusses the four main procedures: idle robot discovery, selecting robots for a given task, sub-task assignment for the selected robots, and notification of a finished sub-task. Because of space limitations we will present each procedure very briefly.

Idle robot discovery is used by the robot discovery engine to discover idle robots belonging to other clouds. It utilizes the overlay to allow the virtualization engine to perform network level virtualization on all robots, including robots that belong to other clouds. The second procedure, selecting robots for a given task, is performed by the virtualization engine, which can run a coalition formation algorithm for multi-robot systems. It selects the robots in the most efficient manner by considering the robots capabilities and the task requirements.

The third procedure, sub-task assignment for the selected robots, is the procedure of dividing a task into sub-tasks and assigning each sub-task to a robot from the group of selected robots. The sub-task is assigned to the local robots through the gateway, and to robots belonging to other clouds via the overlay. The last procedure, notification of a finished sub-task, is sent by the robots, when they finish their sub-task, through the gateway to the robot monitoring engine, where the latter updates the robot information repository.

IV. IMPLEMENTATION

A. *Implemented scenario*

As a prototype, we implemented the scenario presented in section II.A, the wildfire suppression scenario. We used three robots with different capabilities. The first robot is equipped with arms that allow the robot to grab the extinguisher and thereby suppress the fire. We used plastic balls instead of real extinguishers. The second and third robots can detect obstacles. We used red colored objects as obstacles that can be detected by the robots' light sensors. These two robots can also remove the objects using the hands attached to their motors. We assume each of the three robots belong to different business entities. This leads to an implementation with three clouds, one robot per cloud. One of the clouds hosts the wildfire application in addition to the IaaS to which its robot belongs, while the two other clouds only host the IaaS to which their robots belong. It is important to note that all three clouds host a gateway in order to be able to interact with the actual robots. Fig. 3 shows the sequence diagram of the implemented scenario.

B. *Prototype Architecture*

Our prototype architecture is depicted in Fig. 4. We used JXSE 2.5, a Java-based implementation of JXTA [16], to implement the overlay network. JXTA is an open source peer-to-peer protocol specification. For each VRC we created a JXTA peer. JXTA peers can communicate in networks where P2P communication is allowed; we therefore connected the three laptops in the same Local Area Network (LAN).

We used Google App Engine to implement the wildfire suppression application in robotic cloud 1. The three IaaSs and their corresponding gateways are implemented as RESTful web services. The web service that implements the gateway communicates with the robot through Bluetooth. Google App Engine communicates with the IaaS via HTTP. Each IaaS, with its corresponding gateway and JXTA peer (VRC) runs on a different laptop. The REST interfaces on different components were implemented using the Restlet framework [17]. We used three LEGO Mindstorms NXT robots as the

robot fleet.

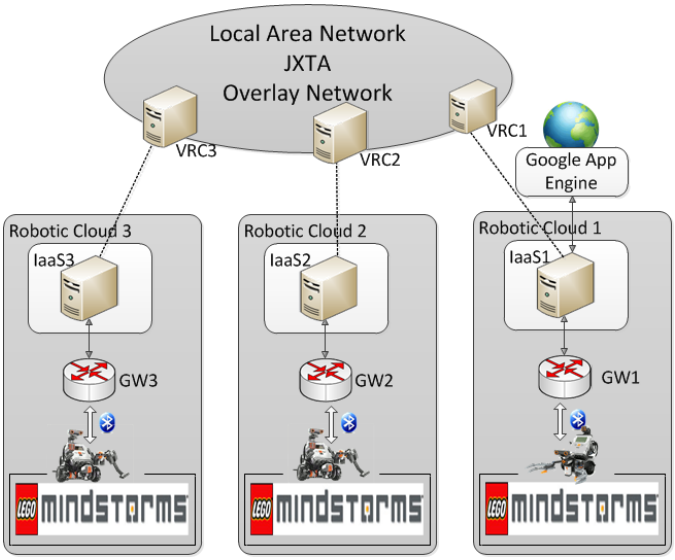

Fig. 4. Prototype Architecture

We implemented the four procedures described in section III.D: the idle robot discovery, selecting robots for a given task, sub-task assignment for the selected robots, and the notification of a finished sub-task.

We used a simple algorithm in our prototype for selecting robots for a given task. The algorithm runs on all the idle robots belonging to the three IaaSs. The task is suppressing the fire, which requires a set of sensing skills and actuating capabilities: {2 light sensors to detect the obstacles, 1 set of arms to grab the extinguisher, 2 kicking arms to remove the obstacles, and 3 movement motors}. This task cannot be performed by a single robot because of the limited capabilities of individual robots -- it requires the cooperation of several robots. Thus, the algorithm selects a group of three robots for the given task based on their capabilities: robot 1 with {1 set of arms, and 1 movement motor}, and robots 2 and 3 each with {1 light sensor, 1 kicking arm, and 1 movement motor}. The sum of the capabilities owned by this group of robots fulfills the task requirements. Finally, the task is assigned to the selected robots.

When the gateway receives the sub-task assignment for the selected robot (i.e., POST/Robots/{RobotID}), it sends the command to the robot. For the gateway that is monitoring robot 1, the command is to go to a location and grab a ball, for the gateways monitoring robots 2 and 3; the command is to detect the obstacle using the light sensor and to remove it using the kicking arm. We used leJOS NXJ API, a Java programming environment for the LEGO MINDSTORMS NXT robots. For example, for grabbing and releasing a ball we implemented two methods called grab( ) and release( ) in Java, which instructs the robot to grab and to release the ball:

```
public void grab( )
{
   Motor.B.stop( );
   Motor.C.stop( );
   Motor.A.setSpeed(200);
   Motor.A.forward( );
   Delay.msDelay(500);
   Motor.A.backward( );
   Delay.msDelay(1000);
   Thread.sleep(500);
}
public void release()
{
   Motor.A.forward();
   Delay.msDelay(500);
   Motor.A.stop();
}
```

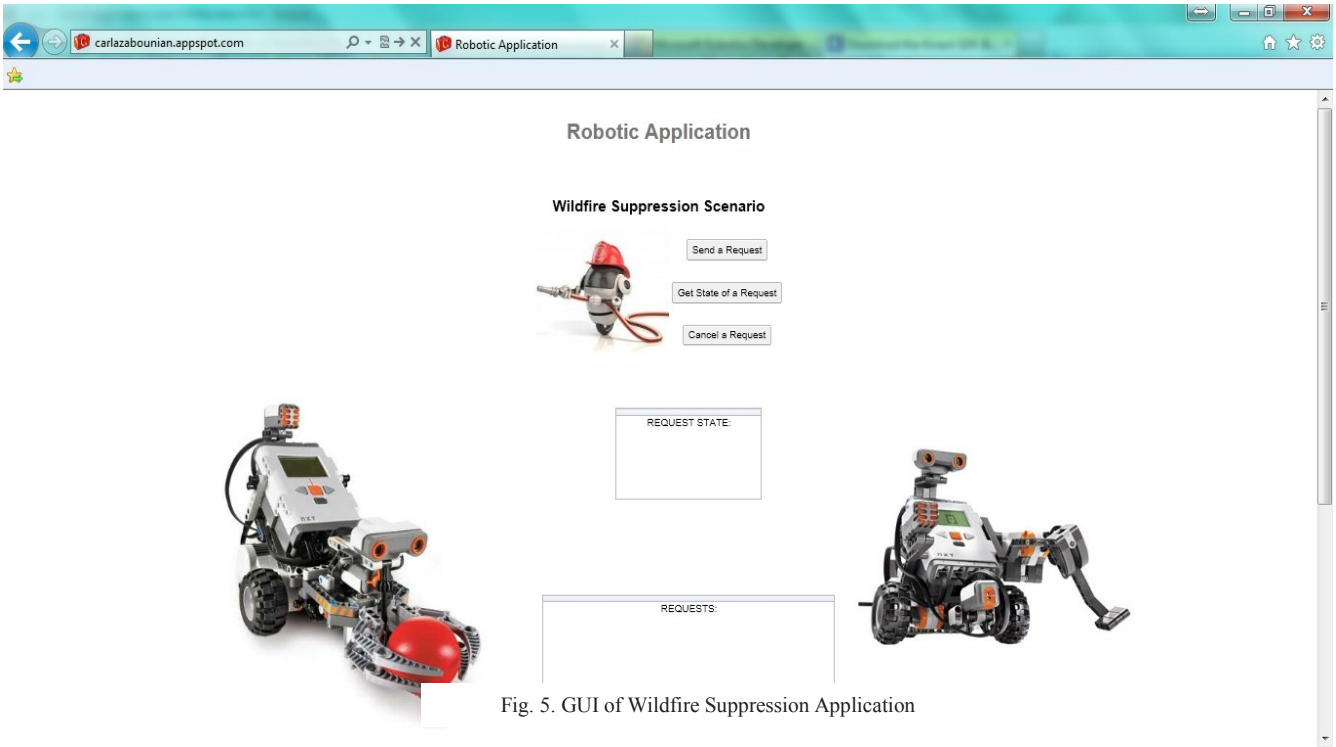

Fig. 5. GUI of Wildfire Suppression Application

Fig. 5 shows the GUI of the wildfire suppression application hosted on Google App Engine.

V. CONCLUSION

We have proposed an architecture that enables cost efficient robotic application provisioning through virtualization and the dynamic delegation of tasks to robots, including robots that might belong to other clouds. We derived the requirements and reviewed the state of the art. A prototype was successfully implemented as a proof of concept using a simple algorithm for robot selection.

REFERENCES

[1] Seungbin Moon, Soon-Geul Lee and Kwang-Ho Park, "Recent progress of robotic vocabulary standardization efforts in ISO," in SICE Annual Conference 2010, Proceedings of, 2010, pp. 266-268.

[2] L. M. Vaquero et al., "A Break in the Clouds: Towards a Cloud Definition", ACM SIGCOMM Computer Communication Review, Vol. 39, No1, January 2009

[3] S. Loveland et.al, "Leveraging virtualization to optimize highavailabilitysystem configurations", IBM Systems Journal, vol. 47, no.4, 2008, pp. 591-604

[4] Y. Yu et al., "Supporting concurrent applications in wireless sensornetworks", In proceedings of the 4th International Conference on Embedded Networked Sensor Systems, SenSys06, Boulder, Colorado, 2006, pp.139-152

[5] P. Jayasumana, Q. Han, and T. Illangasekare, "Virtual sensor networksa resource efficient approach for concurrent applications, " In Proc. ITNG 2007, Las Vegas, Apr. 2007

[6] Y. Chen, Z. Du and M. García-Acosta, "2010 fifth IEEE international symposium on service oriented system engineering; robot as a service in cloud computing" in 2010, pp. 151-158.

[7] M. P. Papazoglou and W. Heuvel, "Service oriented architectures: approaches, technologies and research issues " The VLDB Journal, vol. 16, pp. 389-415, 2007.

[8] Zhihui Du; Weiqiang Yang; Yinong Chen; Xin Sun; Xiaoying Wang; Chen Xu, "Design of a Robot Cloud Center," Autonomous Decentralized Systems (ISADS), 2011 10th International Symposium on , vol., no., pp.269,275, 23-27 March 2011

[9] Qiang Yu; Hongxing Wei; Miao Liu; Tianmiao Wang, "A novel multi-OS architecture for robot application," Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on , vol., no., pp.2301,2306, 7-11 Dec. 2011

[10] Doriya, R.; Chakraborty, P.; Nandi, G.C., "Robotic Services in Cloud Computing Paradigm," Cloud and Services Computing (ISCOS), 2012 International Symposium on , vol., no., pp.80,83, 17-18 Dec. 2012

[11] Doriya, R.; Chakraborty, P.; Nandi, G.C., "'Robot-Cloud': A framework to assist heterogeneous low cost robots, " Communication, Information & Computing Technology (ICCICT), 2012 International Conference on , vol., no., pp.1,5, 19-20 Oct. 2012.

[12] Nakagawa, S.; Igarashi, N.; Tsuchiya, Y.; Narita, M.; Kato, Y., "An implementation of a distributed service framework for cloud-based robot services," IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society , vol., no., pp.4148,4153, 25-28 Oct. 2012.

[13] Eng Keong Lua; Crowcroft, J.; Pias, M.; Sharma, R.; Lim, S., "A survey and comparison of peer-to-peer overlay network schemes," Communications Surveys & Tutorials, IEEE , vol.7, no.2, pp.72,93, Second Quarter 2005

[14] F. Belqasmi, R. Glitho, C. Fu, "RESTful web services for service provisioning in next-generation networks: a survey", Communications Magazine, IEEE, vol. 49, no. 12, pp. 66–73, 2011

[15] B. P. Gerkey and M. J. Mataric´, "A formal analysis and taxonomy of Task allocation in multi-robot systems," Int. J. Robot. Res., vol. 23, no.9, pp. 939–954, 2004

[16] Gong, Li. "JXTA: A network programming environment." Internet Computing, IEEE 5.3 (2001): 88-95.

[17] Restlet framework; available on web at: http://www.restlet.org

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: An open-source robot operating system," in ICRA Workshop on Open Source Software,2009.