

17<sup>th</sup> International Conference in Knowledge Based and Intelligent Information and  
Engineering Systems - KES2013

## Scalable adaptive group communication for collaboration framework of cloud-enabled robots

Romeo Mark A. Mateo<sup>a,\*</sup>

<sup>a</sup>*Research and Development Team, Nexus Community, 15F Ace Highend Tower 2, Guro 3 dong Guro-gu, Seoul 152-724, South Korea*

---

### Abstract

Recently, researchers have been exploring the idea of robots that rely on cloud-computing infrastructure to take advantage of processing power and access vast amounts of data. This approach allows a robot to delegate compute-intensive tasks like image recognition, graphical mapping systems and etc. However, the efficient information sharing of cloud-enabled robots is not addressed which is necessary in disseminating real time information. In this paper, a collaboration framework is presented which is designed for information sharing of cloud-enabled robots. Based on attributes, cloud-enabled robots form logical groups to perform information sharing through the Internet. To provide a faster way of disseminating messages, a scalable adaptive group communication (SAGC) is proposed. The logical groups are processed using fuzzy system to employ scalable grouping, and then, the logical links are adjusted by a node link weight function based on Brownian agent to direct the queries to robots with relevant information. The performance evaluation showed that the SAGC had the fastest response to queries compared to other group communication methods.

© 2013 The Authors. Published by Elsevier B.V.  
Selection and peer-review under responsibility of KES International

**Keywords:** Cloud computing, cloud robotics, group communications

---

### 1. Introduction

The cloud-enabled robot, or cloud robotics which is a term for some developers and researchers, is a recent topic in Cloud computing. Cloud-enabled robots are mainly designed to access services in the Web to provide smart functions to robots as emphasized by James Kuffner [1]. Google inc. introduced cloud robotics in [2] to enable cloud services to be used by robots. They emphasized that web services can be used by robots through the Cloud and extend its functions. A highly complex computation needed to process a task will quickly drain the battery of a mobile robot. However, if processed in a computer with a stable power source, the computational time will be faster and the energy will be conserved. Commonly, service robots are built and configured for a specific domain which is classified in [3]. In spite of the differences in domains, with common components, e.g. sensors, arms, wheels, etc., robots can perform each others functions by sharing its intelligence and thus there is no need to configure or train a particular robot. Providing a common format of message for communication is necessary to process the shared intelligence. Also, because of the need for real time process in such applications

---

\*Corresponding author. Tel.: +82-6420-2580 ; fax: +82-6420-2588.  
E-mail address: [mark@nexus.co.kr](mailto:mark@nexus.co.kr).

[4], information and intelligence sharing must consider responsiveness from the system. To address the need of efficient information sharing for cloud-enabled robots, a cloud collaboration framework is presented in this paper. The next subsections discuss the background of this study.

### *1.1. Networked robots and cloud-enabled robots*

Network robotics has grown around the use of network communication channels, including the Internet, as a means to remotely control robots by humans, as well as to support inter-robot interactions. OROCOS [5] and ORiN [6] were among the first studies involving visible-types networked robots. These researches explained a detailed infrastructure for networked robots to perform sharing of information and collaboration. Several studies like in [7] used the network capabilities to coordinate mobile robots. The network infrastructure in [8] provided robots with an efficient way of sharing data and collaboration. In these previous researches, the use of networked robots in different applications was tackled but intelligent functions are still limited because of the limitations of processing power and data storage. The research challenges from complex analysis and optimization methods identified by Schilling [4] are possible by using Cloud. Extending the functions of networked robots to cloud technologies will enhance the capability to process information and extend the functionalities by sharing information among robots. Some efforts took the advantage of cloud computing environment in parallelizing the computation of mapping. Chen [9] proposed a Robot-as-a-Software (RaaS) where the architecture of sharing drivers for robots using Microsoft Distributed Software Services was discussed. The web services like in [9] can be used as intelligence for robots. To implement this, the hardware components of a robot and interfacing should be easy to configure and accessible. Robots should be integrated in the Cloud to access cloud services. The processing module or brain of a real robot should be programmed in a computer with a stable power. This paper addresses these concerns in the components of the proposed framework.

### *1.2. Group communication*

Group communication is an important building block for distributed systems which is considerably a more efficient method to convey information to an entire group than separately to each member of the group. Researchers also implemented group communication techniques into networked robots. In [10], a group communication is used by mobile robots where a mesh construction and pruning are integrated to improve the multicast group communication. In [11], the robot framework took the advantage of using cloud computing in parallelizing the computation of mapping. DAVinci is proposed to use the scalability and parallelism advantages of cloud computing for service robots, but collaboration of robots is not considered. In [12], a social robot framework is proposed to group robots based on their functions and structures where grouped robots are used to solve tasks by collaboration. However, the group method only creates in-stance of robots and there is no method to discriminate the attributes of robots to form multiple groups. By discriminating data from robots, it is possible to classify and form relevant groupings for efficient communication. In this paper logical links of robots or robot grouping is defined as an activity to find robots with similar attributes in the network. In performing query, group communication or multicast operation is used because it is more appropriate model for communication from one process to a group of other processes in different computers to enhance availability and query more robots for intelligence sharing. The group communication tries to query all peer robots in the logical network to find the right answer. However, one of the concerns in group communications is that it generates high overheads and eventually causes network traffic which should be controlled [13]. The proposed algorithm considers the information changes over time where robots with more information have high probability to answer queries. Therefore, there is a necessity to adjust the logical links from robots to direct the queries to the significant robots with more knowledge. In implementing the adaptive communication, a node link weight function is used to determine the significant logical peer links. Logical links of robots are adjusted to direct the queries to other robots with relevant information.

## **2. Collaboration Framework for Cloud-Enabled Robots**

The interactions of software service providers, computers, sensor devices, and robots in the cloud environment are abstracted by a layered design. Figure 1 shows the over-all design of integrating the proposed system for cloud-enabled robots to an existing cloud system shown in three (3) layers: cloud service layer, communication and

collaboration layer, and robot control layer. The service provisioning system in [14] is used for the cloud service layer. Application services in the cloud service layer provide interfaces to be accessed by the cloud-enabled robots. In the communication and collaboration layer, the cloud collaboration framework acts as a middleware which uses a message-based mechanism to manage the communication between cloud services and cloud-enabled robots. Service components in the framework are used to coordinate tasks and information sharing. In the robot layer, the functions and hardware capability of a robot are considered. A *robot agent* is a program in a computer node representing the real robot shown in right of Figure 1. The logic engine is the working brain of a proxy robot which consists of database, inference engine and rule base.

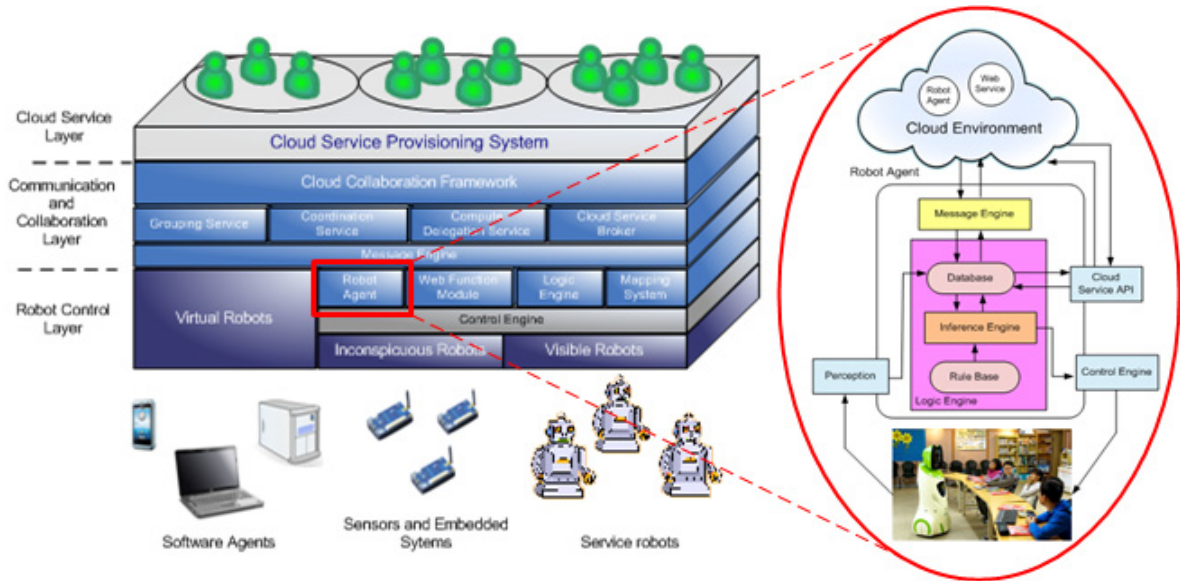


Fig. 1. Integration of the proposed cloud-enabled robot system in the cloud computing.

## 2.1. Robot agent

A robot agent has three main components which are the message engine, logic engine and control engine. Message engine pre-processes the received message before it can be processed to logic engine and formats the message to be sent as request to a cloud service or other cloud-enabled robots. The logic engine analyzes its environment by processing information from perceptions using a rule base system and then executes the specified response or action. A message-based command scheme processes XML formatted messages in the message engine. It translates the commands of logic engine to primitive message commands to be sent to control engine. The control engine translates and then executes the commands imposed by the logic engine. The robot agent processes data received from sensors in the logic engine which decides the commands based on action rules to perform robot controls. Also, it can extract and store information from the Web by the Web Function Module. On the right of Figure 1, the architecture of a robot agent is shown which is used to communicate with other robots. The architecture is based on a rule base system where a list of conditions and its associated actions are stored in a database. At runtime, these conditions are retrieved which are used to map perceptions and then maps a certain action rule. Within the logic engine, the interactions of its subcomponents to process an input from a sensor or a message from other robots are shown in Figure 1. The database stores a list of conditions and interpretations of sensor values. For example, a given temperature sensor value of 28 degrees Celsius is mapped in the database with its interpreted categorical value like *HOT*, *WARM* or *COLD*. These values are configured manually by the owner/developer of robots. The perception module always checks the database for matched conditions. If there is a match then it checks the action rules in the rule base which is processed in the inference engine. After it found a matching action rule then it calls the necessary commands to the control engine. However, if there are

no action rules found then it queries other robot agents to ask about the current perception. In the collaboration, the perception query received from robot agents or cloud services is processed in the message engine and this is processed also in the logic engine by mapping the conditions. If a perception requires the use of a cloud service, which is defined in the database, then it executes the cloud service through its API. The result from a cloud service will be processed in the logic engine by matching the conditions based on the outputs to check if there is a necessary action to perform.

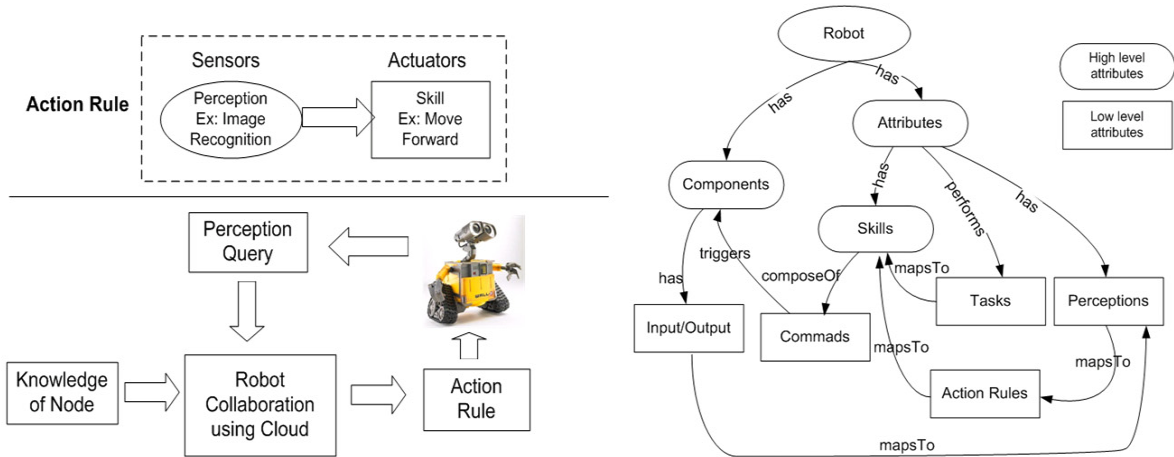


Fig. 2. Processing of perception query using action rule (left) and robot ontology (right)

The collaboration among robots uses the perception querying system shown on the left of Figure 2. An *intelligence sharing* is defined as the exchange of intelligence in the form of perception, skills and action rules. A perception is used by a robot to understand its reading from sensors which is defined by a developer. E. g., temperature sensor values of 0 10 degrees Celsius is COLD, 11 20 degrees Celsius is WARM and 21 80 degrees Celsius is HOT. A *skill* is a set of robot functions associated with actuators to execute a certain action. An action rule is composed of values from a sensor perception and a description of the perception with an option of setting an activation of skills. An *action rule* is composed of perceptions and skills. These are shared within a virtual group of robots when requested. If there is no action rule associated to the perception then it queries to its virtual group about the associated rule for the perception. A *perception query* is a query message used by a robot agent to find an action rule in a virtual group about a perceived input. A robot agent sends a perception query message and this is received by other robot agents within the virtual group. The query process of a robot in performing collaboration is shown on the left of Figure 2. After a robot agent received the perception query message, it will process this using its message engine and logic engine. If the robot agent has similar input/perception parameters in its database then it tries mapping the associated action rule and replies back, else, it replies with an empty rule.

## 2.2. Robot ontology and cloud collaboration framework

To perform collaboration in the Cloud, the attributes and functions of a robot are needed to be shared using a high-level understanding method where the ontology shown on the right of Figure 2 is used. The attributes, skills and device components of a robot are defined in the robot ontology to group similar robots and enable sharing of intelligence. The ontology in Figure 2 has main classes which are *Components*, *Skills*, and *Attributes* and these are shared within the web using Web Ontology Language (OWL). The *Components* contains device information of a robot, e.g., motorcontroller=model description, dual motor controller, voltage rating, wirelessensormodule=RF module description, microprocessor description, sensors components=(light, temperature, humidity). *Attributes* refers to the application purpose of a robot while *Skills* is a set of commands for the device components. The *Skills* is also abstracted by *Attributes* where it associates the current skills to *Attributes* so that a user can look further on the skill sets of a robot after retrieving the *Attributes*. The main classes are high-level attributes for easy

understanding and integration to the cloud collaboration framework and the web. On the other hand, sub-classes are low-level attributes which are specific internal functions and skills of robots. The sub-classes are *Commands*, *Input/Output*, *Tasks*, *Perceptions* and *Action Rules* which are mapped by the high-level attributes. A *Tasks* is an activity of a robot based on a certain goal defined by a user and this is triggered by outside elements like human control or interaction from another robot. *Perceptions* contains a list of recognized inputs from sensors which is acquired in the *Input/Output* class. *Input/Output* maps the input and output values of component devices. The *Perception* is mapped to action rule to perform the skill based on the understanding of a robot from current sensor inputs. *Commands* contains values of components in performing the actual controls of robot devices. The *Commands* is used by *Skills* and Action rules, e. g. (*Skills*) Move Forward- $\zeta$  (*Commands*) primitivecommand:Forward value:100.

The collaboration of robots is performed within the virtual groups. A virtual group is consisted of robots with similar attributes which is further processed by a grouping technique like clustering. When it is the first time of a robot agent to connect to the network, it sends broadcast message to the group managers from different virtual groups. Grouping service provides a list of node addresses of group managers. A group manager will allow a robot agent to join the group after it identifies similar attributes to other grouped robot agent. Grouping the robots with similar attributes provides a responsive system in exchanging messages using the logical links when intelligence sharing is performed.

### 2.3. Scalable adaptive group communication

We refer a robot agent as a *peer agent* (*pa*) after it joined a group. Robot agents in each group are used for specific application domain, e. g., healthcare, rescue operations, etc. Each group manager establishes its communication to other group managers in able to forward query messages if a request perception query is not found within the group. A robot agent is provided with the list of members and these node links are used for query. Throughout this section, sample attribute inputs in Figures 3 and 4 are given for easy understanding. The values from  $P$  are attributes of a robot agent associated with the knowledge or characteristic from its ontology. E.g., *robot 1* in Figure 3 has attributes, fired armed, image recognition, fire fighting and surveillance which are labeled with A, F, H, G, respectively, where *robot 1* or  $pa_1$  has  $P = A, F, H, G$ . Robots are tagged with its attributes based on its function which are set by the owner of robots. The robot attributes with 1=exist or 0=not exist will be mapped based on a specific domain. The attributes are then selected based on the requirements of an application domain shown in the left of Figure 4 where these are accessed on an attribute repository. A selected group manager gathers the data from all group managers to construct the fuzzy system for grouping. The attributes are mapped and then scores the membership of each robot agent shown in the table list of Figure 4. These scores are then processed in the scalable grouping. A set of attributes is defined as  $P_x$  in a collection of  $P$  data points, ( $P_x = p_1, p_2, \dots, p_i, \dots, p_n$  for each  $P$ ), and each  $p_i$  is a numerical value from domain mapping. The  $P_x$  is sent to a group manager to assign the groups. Scalable grouping in [16] is used which is processed into two steps; (1) initialization of fuzzy system for scalable grouping, and (2) classification of robot agent to the appropriate groups using the fuzzy system.

A robot agent sends  $P_x$  to a group manager by  $sendreg(P_x)$  interface and then the group manager processes the robot agents attributes through the fuzzy system in deciding its group. A threshold value for membership, represented by  $\Phi_m$ , controls the membership of robot agents, where  $\Phi_m \leq 1$ . Equation 1 decides if a robot agent belongs to a group  $n$  where a value of 1 indicates that a robot agent is allowed to join group  $n$ .  $P_x$  is the attribute of  $pa_x$  which is changed to  $k$  and this is processed in the membership function  $q_n(k)$  of the group  $n$ .

$$Group_n(n_x \rightarrow P_x) = \{ 1, if q_n > \Phi_m, else 0 \} \quad (1)$$

### 2.4. Adaptive communication based on Brownian agent

The communications among peer agents use the weighted node links (emphw) which is based on the Brownian agent approach [17] to identify the nodes with significant knowledge and have high probability to answer queries. Brownian agent approach combines the features of reactive and reflexive agent concepts. A set of state variables for the Brownian agent is described by  $u^{(k)}_i$  where index  $i = 1, \dots, N$  refers to the agent  $i$ , while  $k$  indicates the different variables used to measure the significant node links. In this paper, the number of action rules of a robot represents the external variable of Brownian agent. The external variable, represented by  $u^{(1)}_i$ , is the ratio value



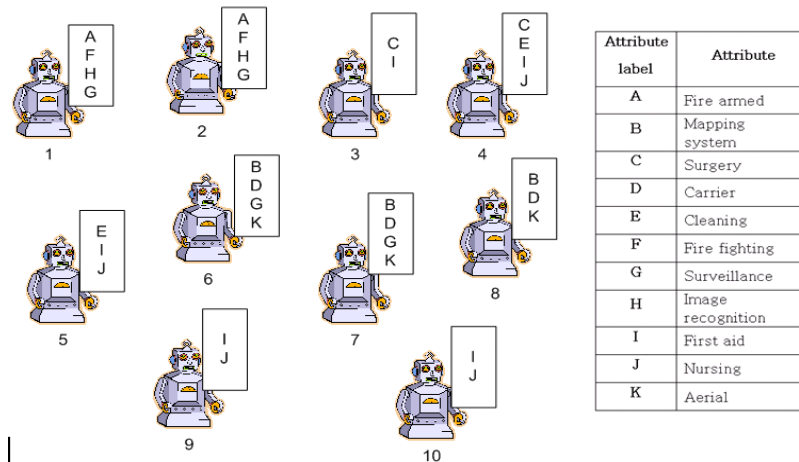


Fig. 3. Ten (10) sample robots with specified attributes to process the fuzzy system.

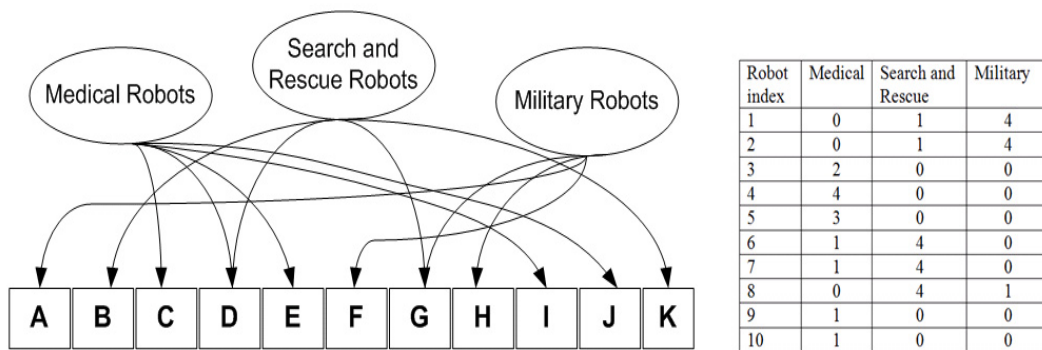


Fig. 4. Attribute mapping of a specified domain by a coordinator based on the requirements of an application domain.

of rule count of the peer agent  $i$  ( $a_i$ ) to the rule count of a peer with the highest rule count.  $a_i$  is the number of rules of peer agent  $i$  divided by  $\max_A$  which is the peer agent with the highest count of rules. The internal variable represented by  $u^{(2)}_i$ . The internal variable is calculated by the number of replies with correct action rule for a perception query of the robot.  $R$  is the number of requests and  $r_i$  is the score of a node link every time a correct rule was replied. An important continuous state variable in the context of Brownian agents is the internal energy depot  $u^{(3)}_i = e_i$ , which determines whether agent  $i$  may perform a certain action or not. This approach is used by the group communication to adapt its link by determining the significant nodes which have high probability to answer the queries.  $N$  is the number of peer agents in the group and  $n$  is the index of peer agent.  $w_i$  is the link value of a peer agent  $n$  to other peer agent  $i$  in its list of link in  $W = w_1, w_2, w_i$ . In our method,  $w$  represents the energy depot of a Brownian agent ( $w_i = e_i$ ) which decides if the query will be sent to the peer agent  $i$ . Peer agent  $n$  adjusts its node links depends on how much a member contributes to queries. The links are processed in a weight function in Equation 2 which uses two variables stated in the Brownian agent approach; external ( $\theta_1$ ) and internal ( $\theta_2$ ) variables. A peer agent  $n$  calculates its link to peer agent  $i$  by,

$$w_i = e_i = \frac{\theta_1 - \theta_2}{2} \quad (2)$$

$$W' = \sum_{i=1}^W \phi(w_i) \quad (3)$$

$$\phi = \{pa, \text{if } w_1 > \Phi_A, \text{else } 0, \text{if } w_1 > \Phi_A\} \quad (4)$$

In Equation 2, the external and internal variables are averaged represent the significant values from the number of rules and contribution to answer queries. Using a threshold value, a peer agent decides which node it will send the query. In Equation 3, the node links are collected after it was compared to a threshold value which is shown as the function of  $\phi$  in Equation 4. All peer agents that were collected in Equation 4 are sent with the perception query in multicast. The result of choosing only the significant links minimizes the message overheads generated in the group communication.

### 3. Implementation and Evaluation

We implemented the proposed framework in our cloud robot prototypes. The first prototype is connected to a computer using USB and serial connections. Shown on the left of Figure 5, the curious robot v1 performs a perception process using a web camera. It was designed to perceive images to recognized and perform actions based on its perceptions using the cloud collaboration framework (CCF). We used a standard web camera for its input perception and a serial motor controller, a Pololu product, to operate forward, backward, left and right movements to its motor. Inputs perceived from the web camera were processed in message engine and then a message will be sent using socket network to communicate with the robot agent. In the message engine, message commands were mapped into byte codes to control the motors. A simple function was tested for our simulation where the curious robot v1 was trained for character recognition. If the curious robot reads FORWARD written on a paper then it tries to understand it and performs a forward motion which is mapped through its list of action rules. The image perceived by the curious robot v1 processed in the logic engine of proxy robot. The second prototype shown on the right of Figure 5 is a robot controlled by a smart phone. We used a robot rover model from DFRobot which is mainly controlled by Arduino module. To communicate wirelessly, the bluetooth module of Arduino was used to connect to a smart phone and mobile application to send message to Arduino and vice versa. The mobile application polls to a server using Representational state transfer (REST) method. By using a web server, we can use any computer device to control the second prototype like iPad as a web service. Moreover, useful web services or web plug-ins can be integrated easily to the web server to provide additional services for your robot. In our case, we used the WebToolkit of chrome to test the speech to command actions where the service translates the speech into command to move the robot, forward, back, left or right.

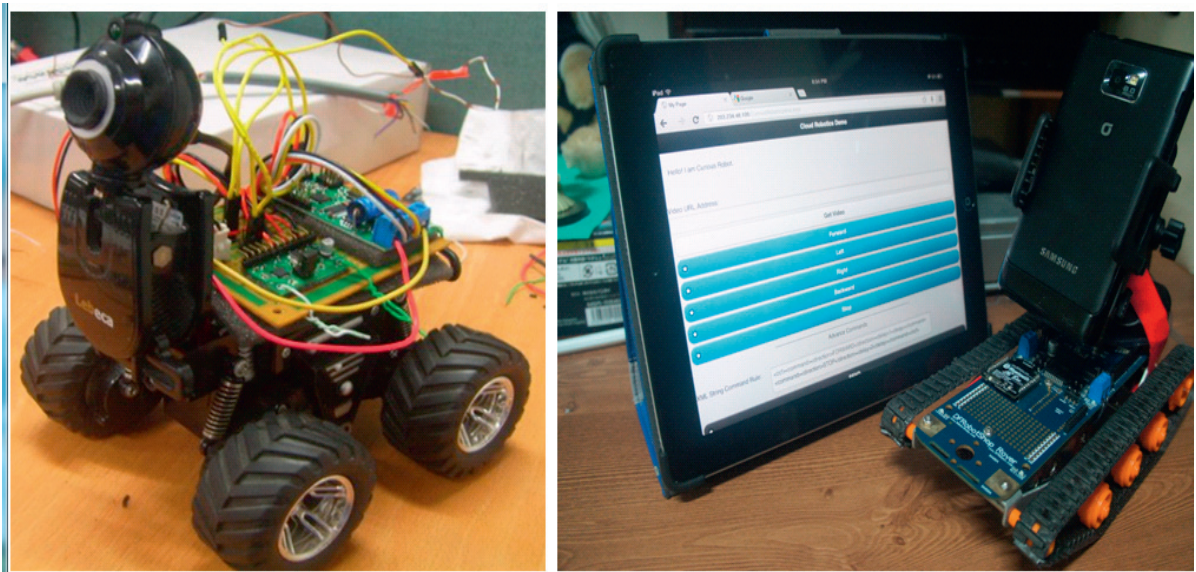


Fig. 5. Curious robot v1 (left) and curious robot v2 (right) were implemented with CCF.

```

Oct 9, 2011 9:17:51 PM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXP
Oct 9, 2011 9:17:51 PM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://Romeo-Win7:7778/acc
Oct 9, 2011 9:17:51 PM jade.core.AgentContainerImpl joinPlatform
INFO: -----
Agent container Main-Container@203.234.48.104 is ready.
-----
Robot-agent A is ready.
received perception, processing...
found a rule for the input perception...
received perception, processing...
found a rule for the input perception...
received perception, processing...
found a rule for the input perception...
received perception, processing...
found a rule for the input perception...
action for perception input not found, querying other robots...
received action rule from Robot-agent D for perception input, processing...

Oct 9, 2011 9:23:03 PM jade.core.mobility.AgentMobility initialized
Oct 9, 2011 9:23:03 PM jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
Oct 9, 2011 9:23:03 PM jade.core.messaging.MessagingService clear
INFO: Clearing cache
Oct 9, 2011 9:23:03 PM jade.mtp.http.HTTPServer <init>
INFO: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal
Oct 9, 2011 9:23:03 PM jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://Romeo-Win7:7778/acc
Oct 9, 2011 9:23:04 PM jade.core.AgentContainerImpl joinPlatform
INFO: -----
Agent container Main-Container@203.234.48.104 is ready.
-----
Robot-agent D is ready.
received query perception message from Robot-agent A, finding...
found a rule for the input perception, sending the action rule...

```

Fig. 6. Robot-agent A requests for an action rule associated with the perception to another robot agent which is logically connected to the same virtual group (left) and robot agent D shares the action rule to the robot agent A (right).

### 3.1. Performance evaluation

The proposed cloud collaboration framework was developed in Java 2 SDK included with other libraries like Jade Framework (JADE). The web server communicates with the CCF to perform the logical grouping of robots and implement the services from the framework. The library of Jade Framework was integrated in the cloud collaboration framework. Each node calls the jade library to support agent communication approach and interoperability among robot agents for intelligence sharing. The procedures of a robot agent to join virtual groups were implemented in the codes of Node class. The establishment of communication is initialized after booting Jade. Each agent uses grouping service to find group managers within the network. The robot agent sends its attributes from robot ontology to the group managers and these will be processed to consider the request to join. After joining a group, the robot agent stores the addresses of peer agents. In Figure 6, the interaction between robot agents in the same group is shown.

A simulation was prepared to evaluate the proposed scalable adaptive group communication (SAGC) where the environment used 1000 nodes for the network and assigned each node with a virtual robot. In the physical network, nodes were divided equally into 100 domains where each domain serves 10 nodes. In a domain, nodes were connected to a router with a latency of 10 milliseconds (ms) to send a message. Routers were connected in a ring topology having two router neighbors in each router with a latency of 100 ms in each connection. There were 1000 perceptions associated with 1000 action rules generated to process the proposed method. Action rules were randomly generated and distributed throughout the nodes and each node varies of numbers of action rules not exceeding of 50 rules. To process the virtual groups, the robot in a node used its attributes identified by its ontology and mapped the attributes to the associated classes. Four application domains of robots were identified which are medical, search and rescue, military, and disaster support. Moreover, a learning model was used where each simulation generated additional action rules to robots which were assigned randomly. Interest-based peer grouping is used in group communication to provide relevant links which are studied in [16]. The interest-based groupings which are multicast, ring, interest-based shortcuts, scalable grouping and k-means-based peer grouping are used to compare to SAGC. Multicast, ring and interest-based shortcuts are commonly used in group communication in P2P while scalable grouping and k-means use the property or attributes of peer nodes to form the P2P connections. Additional module like robot agent and learning module were added to perform the perception query. 100 to 1000 perception queries were generated where the source node of the requests and files queried were randomly selected. A message query is forwarded to peers until it reaches its maximum hops. The following are determined; 1) message overheads by the total message generated and, 2) turn-around time of a query to return to the source node.

Obviously, multicast produced the highest volume of messages generated while ring had the lowest volume



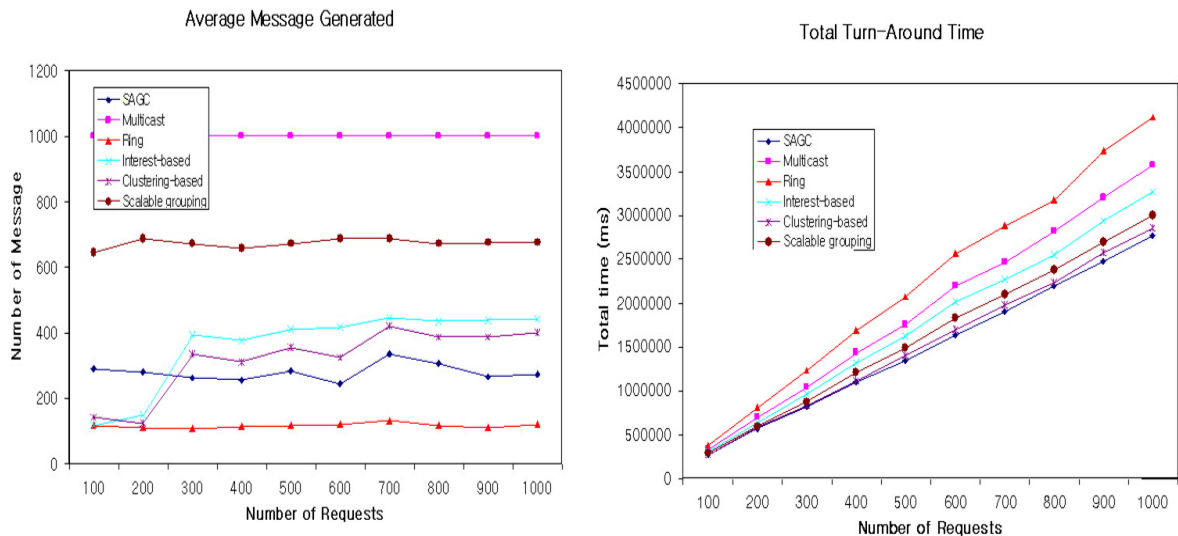


Fig. 7. Average message generated based on number of requests (left) and, total turn-around time of requests (right).

of message generated and better of almost 6 times than the multicast shown in left of Figure 7. Note that the multicast was constantly generating 999 messages in each query within the network. Multicast generated greater volume of messages which indicate that it tried to find the rule for perception query throughout the network of peer agents. In right of Figure 7, the throughput performance shows that the SAGC has the fastest response time to requests by having the lowest total turn-around time. This result is the most crucial performance result because of the need for real time response when performing the perception query. The clustering-based also produced a good result because of its adaptability property. While ring had the lowest message overhead shown in right of Figure 7, it was the slowest to response because of the ring behavior in passing the queries. Multicast produced a lot of network traffic because of high volume of messages generated every time a node performs a query which was the reason it was the second to the slowest to response.

#### 4. Conclusion

The current problems in robotics are the complex computations to process its intelligence and data storage. Integrating the services of cloud computing to robots, with the proper design, can be a solution to these problems. In this paper, the efficient interaction among robots and integration of cloud services to robots were considered to improve the collaboration and intelligence sharing in the Cloud. The intelligence sharing was proposed to improve collaboration of cloud-enabled robots which was supported by the scalable adaptive grouping communication (SAGC). The SAGC uses the fuzzy system to perform the scalable grouping and after identifying the groups, the messaging method of cloud robots is adjusted based on the Brownian agent. The scalable search was used for more robots to query while the adaptive communication function minimizes message overheads by considering the significant logical links. A network topology was configured for the simulation where 1000 nodes were used, and then the performance of SAGC was evaluated in efficiency of the scalable search. The graph results from Figure 7 show that the SAGC is the second to the least number of messages generated within the network. The turn-around time of requests in Figure 7 shows that SAGC is the fastest to respond to queries. The results concludes that the integration of SAGC into the real cloud robots will improve the information sharing throughout the cloud environment with consideration of having responsive system by preventing unnecessary message overhead through the adaptive group communications.

## Acknowledgements

This paper is supported by the Research and Development Center of NEXUSCOMMUNITY. NEXUSCOMMUNITY, the author's current company, is a well-known CTI solution provider in South Korea. They also provide their solutions to overseas like Japan and China. NEXUSCUBE, which is their main product, is a client/server program for CTI system and their solutions are based on cloud environment which are CUBA, CAiRO and DOME. Refer to this URL for more details <http://support.nexus.co.kr/kr/nexus/index/index.php>

## References

- [1] Guizzo, E., Robots with their heads in the Cloud, *IEEE Spectrum Magazine* 2011;16-18
- [2] Google's android apps for cloud robotics, Retrieved November 2011 at <http://rj3sp.blogspot.com/2011/05/googles-android-apps-for-cloud-robotics.html>
- [3] Technical committee on service robots, Retrieved November 2011 from <http://www.service-robots.org/technologies.htm>
- [4] Schiling, K., *Networked robots: research challenges in ETSI: Networked Mobile Wireless Robotics Workshop* (2010)
- [5] OROCOS, March 2013 from Website: <http://www.orocos.org>
- [6] ORiN, March 2013 from ORiN Website: <http://en.wikipedia.org/wiki/ORiN>
- [7] Nayak, A. and Stojmenovic, I., Topology control in sensor, actuator, and mobile robot networks, *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, Wiley-IEEE Press 2010 ; 185 -207.
- [8] Tezuka, H., Katafuchi, N., Nakamura, Y., Machino, T., Nanjo, Y., Iwaki, S. and Shimokura, K.I., Robot Platform Architecture for Information Sharing, *Journal of Robotics and Mechatronics* 2006; **18**, 3: 325-332.
- [9] Chen, Y., Du, Z. and Garca-Acosta, M., Robot as a service in cloud computing, in *Proc. IEEE Symposium on Service Oriented System Engineering* 2010; 151-158.
- [10] Piao, S., Zhong, Q., Liu, Y and Li, Q. Research of group communication method on multi-robot system, *Communications in Computer and Information Science* 2011; **159**, 9: 457-461.
- [11] Arumugam, R., Enti, V.R., Bingbing, L., Xiaojun, W. Baskaran, K., Kong, F.F., Kumar, A. S., Meng, K.D. and Kit, G.W., DAvinCi: a cloud computing framework for service robots, in *Proc. IEEE International Conference on Robotics and Automation* 2010; 3084-3089.
- [12] Suh, J. and Woo, C.W., Design and development of a social robot framework for providing an intelligent service, in *Proc. World Congress on Engineering and Computer Science* 2009; **2**.
- [13] Khanna, S., Naor, J. S. and Raz, D., Control message aggregation in group communication protocols, in *Proc. Automata, Languages and Programming, LNCS 2380*: 135-146.
- [14] Mateo, R.M.A. and Lee, J., Dynamic service assignment based on proportional ordering for the adaptive resource management of cloud systems, *KSII Transactions on Internet and Information Systems* 2011; **5**, 2 : 2294-2314.
- [15] Mateo, R.M.A. and Lee, J., Scalable search based on fuzzy clustering for interest-based p2p networks *KSII Transactions on Internet and Information Systems* 2011; **5**, 1: 157-176.
- [16] Schweitzer, F., Active brownian particles: artificial agents in physics, *Stochastic Dynamics, Berlin: Springer, Lecture Notes in Physics* 1997; **484**: 358371.