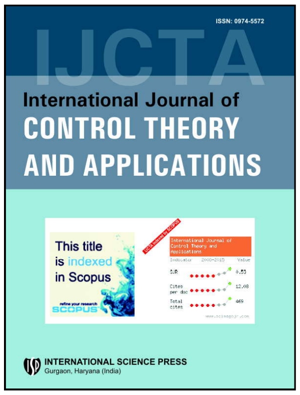


International Journal of Control Theory and Applications

ISSN : 0974-5572

© International Science Press

Volume 10 • Number 14 • 2017

Robo WS-A Map Building Web Service for Cloud based Robots

C. Nandhini[1] and Rajesh Doriya[1]

[1] *Department of Information Technology National Institute of Technology, Raipur, Chattisgarh, India, Emails: cn.nandhini@gmail.com,* rajeshdoriya.it@nitrr.ac.in

Abstract: Cloud Computing has made a tremendous impact in robotics. As a result of this, Cloud Robotics has emerged as one of the recent topics of interest. Cloud Robotics, is a promising solution for autonomous robots to perform various tasks in dynamic environments. With the growth of Cloud Robotics, robots are able to use the resources that are located outside of its onboard computer. Moreover, they rely on the cloud for massive storage, sharing of data, and parallel execution of computationally intensive tasks in real time. The Cloud Robotics architecture leverages the communication mechanism between the cloud and robots and also the communication between the team of robots in networked robots. This paper describes a novel communication framework which aids robots to offload their computational tasks to the cloud. We present the design of cloud robot interaction model using gSOAP and design of a standard Web Service using gSOAP which can be accessed by heterogeneous robots. We implemented RoboWS system which provides map building as a Web Service and validated our framework by providing the expensive map building process as a service in the cloud.

Keywords: Cloud Robotics, Robot map building, gSOAP, gMapping, Cloud robot interaction model, Robotic service, RoboWS System

1. INTRODUCTION

Over the past decades, industrial robots were programmed to do repetitive tasks and deployed in a controlled environment [1]. The robotics combines with the network technologies fosters the emergence of networked robotics which extends the robotics functionality. Network robotics can be classified as tele-operated robots where a human supervisor instructs the robots to perform various tasks or autonomous networked robots where multiple robots are co-ordinated on a single task over the network [2]. However, these robots use the control strategies which are specific to that robot so that the algorithms they use are not so complex and also the storage requirements are less. There is a strong demand for autonomous service robots that can perform human scale manipulations and can be used for various purposes where a single robot can clean the room, assists the elderly people and also serves the people in the restaurants [3-4]. In order to meet these demands, the algorithm to be used for vision processing, map building, navigation etc., will become more complex and also the onboard storage, computational power and battery life will not be sufficient enough. A current trend that can provide all these capabilities is the Cloud Computing. Cloud computing provides a ubiquitous, on demand resources based on the utilization needs with less maintenance effort. As per NIST definition "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [5]. Cloud Robotics is one of the fastest growing fields which utilizes the concept of cloud computing [6]. Cloud robotics can provide promising solutions for autonomous service robots to overcome various challenges in dynamically changing unstructured environments.

The growth of network technologies with high bandwidth internet access allows robots to offload their computationally intensive task to the cloud within no time. Moreover, the availability of large scale data centers allows robots to share their knowledge, data sets and execution of computationally intensive tasks efficiently with the pool of computing and storage resources.

Robots can access the computational and storage resources hosted in the cloud using various distributed computing architecture. Service Oriented Architecture (SOA) style in one of the distributed computing models in which all the functions are implemented as a service [7]. These services are exposed via the well defined interfaces so that the internals of the service is hidden to the outside world and can be used as a black box. It is a component based software development in which existing components can be reused for building faster systems. A service can use other services which reduce the risks and also the cost. Using SOA, we can build a loosely coupled interoperable machine to machine interaction over the network [7]. Moreover, SOA is considered as the enabling architecture for providing on-demand services and maximum resource utilization in the cloud environment. Web Services are the enabling technologies for implementing SOA. Web Services can be implemented in various ways like (i) RESTful (Representational State Transfer) Web Services are stateless and there is no standard for defining the interface. It is based on resource oriented architecture, lightweight and simpler to develop [8]. (ii) SOAP supports several protocols and technologies like XML for messaging and WSDL for defining the interfaces, the address and location of services. Moreover, SOAP is useful for handling asynchronous processing and invocation. Among these Web Service technologies SOAP suits our needs since the robotic services are deployed for use by the outside world so it needs very well defined standard interfaces and the services can be consumed by the application independent of its language and platform.

The major aspect of our research involves the development of a standard Web Services that will allow heterogeneous robots to executes the computationally intense algorithms like map building and path planning as a service over the cloud. The remainder of this paper is as follows. Section 2 surveys the most relevant works on cloud robotics and discusses their contributions. Section 3 describes the components used for designing our Robo Web Service (RoboWS) system. In Section 4, we present the architecture of the RoboWS system, its design, and the implementation. Section 5 presents the experimentation and the results obtained from the system. Section 5 concludes the paper and discusses future works.

2. PREVIOUS WORK

Recent years several research groups have started investigating the usage of cloud technologies in robotic applications. This section explores the contribution of the researchers which made the cloud robotics to the present state of the art. Simultaneous Localization and Mapping (SLAM) is one of the computationally and data intensive tasks. As robots are designed with limited resources, C2TAM [9] proposed a framework for offloading the computation intensive SLAM task to the cloud. SLAM has a strong real time constraint, so moving all computation of the SLAM to the cloud would not be effective. C2TAM propose a method to partition a real time SLAM algorithm that allows part of the computation should be moved on to the cloud.

The DAvinCi [10] allows multiple robots to share their sensor information using clouds. The Grid based FastSLAM algorithm is implemented using Map Reduce Framework thus reducing the computation time by splitting the work across clusters. The Robot as a Service (RaaS) [11] aimed at designing and implementing a robot as an entire service oriented architecture. So that RaaS must contain a service provider, service broker and application client. RAAS should have the robotic functionalities implemented as a service; The RAAS service provider unit can provide multiple services; any client can deploy new service or remove any particular service.

In the complex and unstructured environment, the information generated by one robot should be reused by other robots in order to make the robots adjusts to the dynamic environment quickly. However, the data generated by robots with different hardware configuration are not same which prohibits the reuse of data. RoboEarth [12] focuses on how robots should store and share the knowledge among them which can speed up their learning process. Thus RoboEarth provides a platform for robots to collect, stores, reuse, and sharing of data independent of hardware over the network. RoboEarth provides standardization of the knowledge to be stored so that it's used by different types of robots.

The Rapyuta [13] is an open source which offers Platform as a Service (PaaS) framework for robotics application. In Rapyuta, the robots will get dynamically allocated computing environment for offloading their computation tasks. The computational environments are tightly interconnected which makes a suitable framework for multiple robot coordination. This interconnection allows robots to share their services and information with other robots. The computing environment has high bandwidth access to the knowledge repository so that the robots can benefit from other robots experience. Rapyuta Architecture consists of computing environments, communication protocols, set of core tasks and command data structure.

The SCMR framework [14] proposes a solution where robots can offload their computation to the cloud can handle the network connectivity issues in an effective way. Google Goggles [15] is a cloud based image recognition service for mobile users and used for robotic grasping. Limosani et al [16] provide a cloud based global navigation approach enables autonomous robots to navigate in an unknown environment. It divides environment into sub maps and allows mapping information to be stored with environment tags.

Finally, we capitalize on the development of a standard Web Services that will allow heterogeneous robots to executes the computationally intense algorithms like map building, path planning as a service over the cloud.

3. KEY COMPONENTS INVOLVED IN OUR FRAMEWORK

3.1. Robot Operating System (ROS)

The ROS [17] is an open source meta-operating system which runs alongside the traditional operating system which is used by robots. ROS provides service similar to an operating system with hardware abstraction to the user by hiding low-level device control. ROS uses a peer to peer network topology. ROS is a distributed framework of processes that are individually designed for specific purpose and can be loosely coupled at runtime. ROS provides a set of tools and libraries used to build robotic applications. The widely used robotic algorithms for motion planning, navigation and mapping are implemented, debugged are made into a standard package which can be used by other systems. In addition to that, the interface for message passing between the hardware and robotic software also available in package formats. ROS is designed to be thin so it can be integrated with any other robotic frameworks like player.

3.2. gSOAP

Nowadays, there are various APIs available that provides SOAP client and Web Service development. Examples include SOAP Lite, Apache Axis, .Net framework and gSOAP. gSOAP [18] is an open source Software development kit that provides platform independent development environment for C/C++ Web Services. gSOAP offers a stub/skeleton compiler which makes the creation and deployment of C/C++ Web Service easily. It comes up with the XML generator which converts the C/C++ data types to equivalent XML data types. It also provides the WSDL importer which will automatically generate the WSDL files given the service functions declaration. Moreover, the system designed using gSOAP will be more efficient than the web service implementation in another language due to the C/C++ implementation of SOAP client/Web service [18]. gSOAP has the following design characteristics [18] which make us in deciding gSOAP for our Robot Client interaction model.

a. Pre-Compiled Marshalling Routines: Allows the overhead of generating the routines at runtime thus lowers the response time.

b. Data types are defined in Native language: The user defined data types are defined in C/C++ and they are serialized and de-serialized using pre-compiled marshalling routines which provide the following advantage.

   - Minimize memory copy operations
   - Reduces memory latency - The restructuring compiler techniques can be used to compile and optimize the marshalling routines with kernel routines

c. On Demand XML Parsing: Allows the Efficient XML parsing without the overhead of keeping XML Document in memory.

d. Minimize memory use: The executables has less than 150 K memory size which enables them to deploy in memory constraint devices.

e. Platform Independence: The compiler generates the stub and skeleton in C/C++ which can be independent of any platform. So that they can run on Linux, Windows or embedded systems.\end{enumerate}

3.3. gMapping

Map Building is one of the primary tasks of the autonomous robots in an unknown environment. It is commonly referred to as the Simultaneous localization and mapping (SLAM) problem. It is considered to be the complex problem and requires searching for the solution in a high dimensional space. Most of the mapping problems use probability models rely on Bayes rule. Fortunately, there are many open source implementation available for mapping includes gMapping, CoreSlam, HectorSlam etc. We are using gMapping for providing map building service. gMapping is the laser based SLAM algorithm which uses Rao-Blackwellized particle filter for generating 2D occupancy grid map. It is the most widely used map building package available in ROS. The reason for selecting gMapping for our implementation of map building service is multi-fold.

a. gMapping is having better performance with lesser error values both in simulation and real world scenarios.

b. It also uses fewer resources so that it can be used in real time [19].

c. It is an easily available open source package on ROS.

d. Finally, the nature of the algorithm allows parallel implementation as each particle is independent of other particles and there is no data sharing between the particles [20].

4. PROPOSED FRAMEWORK

We aim at designing a web based cloud robot interaction model which will be used by the robots for offloading its computational tasks. In addition, it provides the map building as a service over the cloud.

4.1. Proposed Architecture

We have designed a robot cloud interaction model using gSOAP toolkit [18], which provides SOAP/XML based web service. We developed a simple implementation of web applications using which the robot can interact with

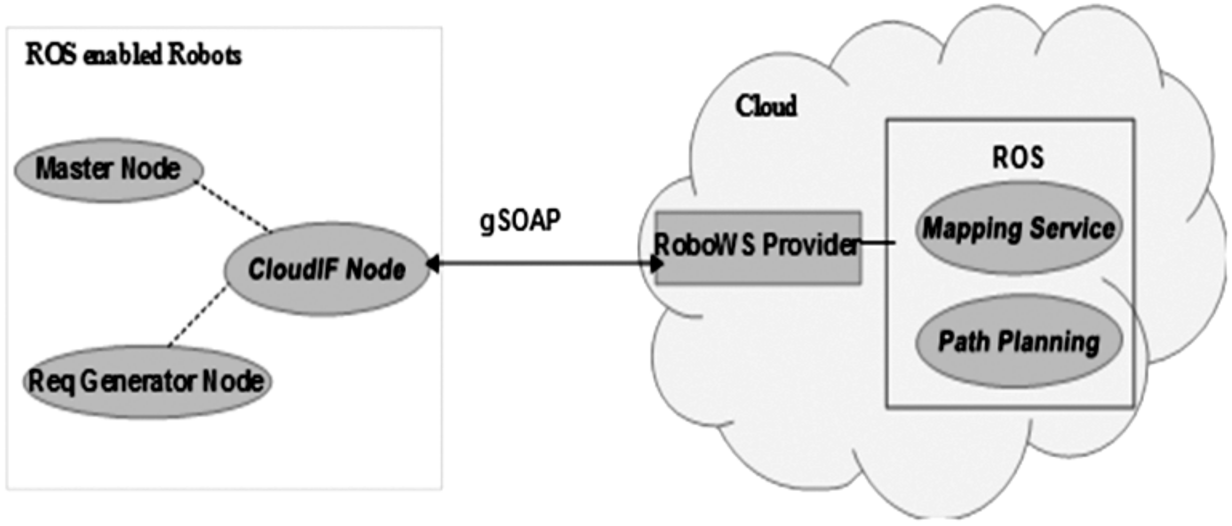


Figure 1: RoboWS System overview

the cloud for offloading its computational intensive tasks to the cloud. Figure 1 presents the overall system design. On the robot side the Cloud Interface (Cloud IF) node is a ROS node which acts as a proxy for communicating with cloud. The other ROS node which needs cloud service sends the request to the Cloud IF node using ROS communication mechanism. The Cloud IF node connects to the cloud using gSOAP protocol. On the server side the RoboWS Service Provider Process which is implemented as a standalone gSOAP server listens for the client request. It uses a separate thread for each client connection and processes the request using ROS. The RoboWS Service Provider uses remote RPC communication mechanism for servicing the client request.

4.2. Implementation of Robot - Cloud Interaction model using gSOAP

As far our knowledge there is no literature that provides Robot cloud interaction using gSOAP. Rosbridge [21] provides the communication mechanism for Non ROS users to communicate with ROS. Rapyuta [13], Robot Web Tools (RWT) [22] and SCMR [14] use Web Sockets and Twisted framework for cloud robots interaction. Srinivasan et. al [23] presents the analysis of using Web sockets for Robot tele-operation. DAvinCi [10] mentioned HTTP messages wrapped in XML and did not specify any specific framework.

For integrating gSOAP with ROS we have created a simple implementation using gsoap_ros package which consists two ROS Nodes.

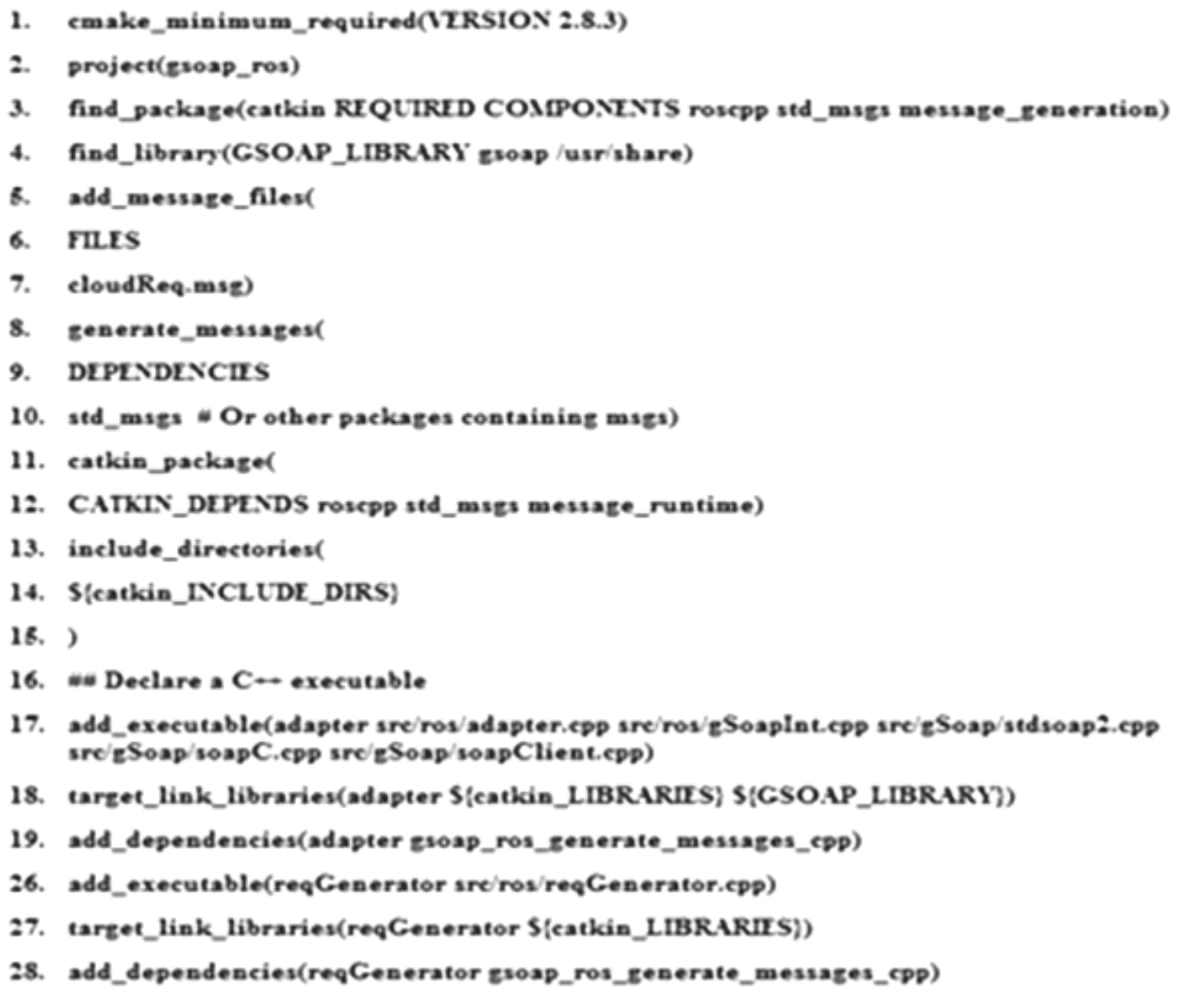
- *The Req Generator node* whenever it wants to access the Web Service deployed in the cloud it will publish the Cloud Request Message.
- *The Cloud IF node* which is subscribed for the Cloud Request Message from other ROS nodes of the robots. On receiving the cloud request, based on the cloud Request id it can access the different Web Service deployed in the RoboWS Service Provider node in the cloud as shown in Figure 1.

We have to make changes in the *CMakeLists* file for integrating gSOAP with ROS which is shown in Figure 2. The *CloudIF node* has to be compiled along with *gSOAPInt.cpp* which contains the implementation of making SOAP Web Service Request and the gSOAP generated stub files as shown in line 17 of Figure 2. For linking the installation path of gSOAP Library should be mentioned and used as shown in line 4 and 18 of Figure 2.

4.3. Map Building Web Service Implementation

The RoboWS service Provider is implemented as a gSOAP stand alone server. It uses a separate thread for handling each client connections. Development of Web Service program using gSOAP requires the creation of WSDL file or header file which contains the service function with input and output parameters with its data type. The gSOAP compiler compiles the header file and generates the skeleton routine to implement the remote method as Web Service. The skeleton routines should be compiled with the application routine that implements the map building function to expose that function as a Web Service.

Moreover, ROS master is running on the server side for providing various robotic services. Currently, we are having a simple service function for providing 2D map building service. Later this will be enhanced by providing various services like path planning, 3D mapping, etc.



```
cmake_minimum_required(VERSION 2.8.3)
project(gsoap_ros)
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs message_generation)
find_library(GSOAP_LIBRARY gsoap /usr/share)
add_message_files(
FILES
cloudReq.msg)
generate_messages(
DEPENDENCIES
std_msgs # Or other packages containing msgs)
catkin_package(
CATKIN_DEPENDS roscpp std_msgs message_runtime)
include_directories(
${catkin_INCLUDE_DIRS}
)
## Declare a C++ executable
add_executable(adapter src/ros/adapter.cpp src/ros/gSoapInt.cpp src/gSoap/stdsoap2.cpp
src/gSoap/soapC.cpp src/gSoap/soapClient.cpp)
target_link_libraries(adapter ${catkin_LIBRARIES} ${GSOAP_LIBRARY})
add_dependencies(adapter gsoap_ros_generate_messages_cpp)
add_executable(reqGenerator src/ros/reqGenerator.cpp)
target_link_libraries(reqGenerator ${catkin_LIBRARIES})
add_dependencies(reqGenerator gsoap_ros_generate_messages_cpp)
```

Figure 2: Code Changes in CMakeLists file for gSOAP-ROS Integration

4.4. Client Implementation

The *CloudIF ROS node* subscribes for the Cloud Request message. On receiving the Cloud Request the *CloudIF node* will call the remote method based on the request Id. At the client side given the header file containing the definition of interface methods, the stub routines will be generated. The input to gSOAP compiler is the header file, which is then processed to generate the stub routine for the client application. The primary stub's responsibility is to marshal the input data send the request to the designed SOAP server over the network, wait for the response and once the response arrives demarshal the output data. The client application invokes the remote methods in the exactly similar way as it invokes any local method.

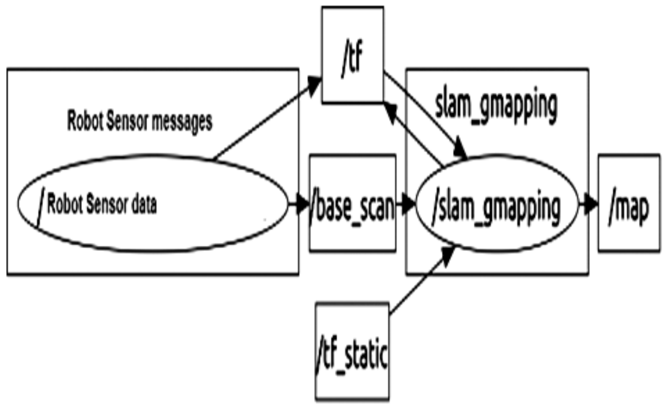
5. EXPERIMENTATION AND RESULTS

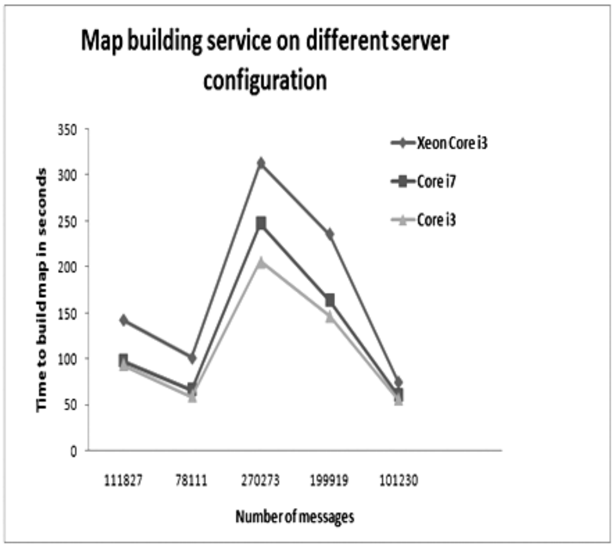
We have created and tested the RoboWS system. The RoboWS Service Provider process we made it run in Intel Xeon Processor, Intel i7 processor and Intel i3 processor with the different configuration. The PC configuration used in our experiment is shown in Table 1.

Table 1
Specification of Compared Machine

Item	*Xeon Core i3*	*Core i7*	*Core i3*
CPU Frequency (GHz)	1.3	2.20	2.53
Number of Cores	8	4	2
Memory (GB)	2	4	2

The slam_gmapping node receives the base_scan messages and odometry messages from the sensor and builds the map using improved version of Blackwellized Rao Particle filter approximation. The message flow graph generated using rqt_graph tool is as shown in Figure 3(a).





(a) **(b)**

Figure 3: (a) Message Flow graph using rqt_graph tool
(b) Time taken by Map building service

We measure the time taken by our Web Service to build the map using the bag files which contains the recorded sensor data. Moreover, we also tried with the bag files obtained from heterogeneous robots like PR2 robot from MIT Stata center dataset and Turtlebot robot from RTAB-Map site. We implement this framework in Ubuntu 14.04 LTS operating system. We used Jade version of ROS, gSOAP 2.8.3 and gMapping package available with ROS. The time to build the map is represented using the graph as shown in Figure 3(b). The sample output of 2D map build using our web service is shown in Figure 4. The details of the bag files used in our experiment are listed in Table 2.

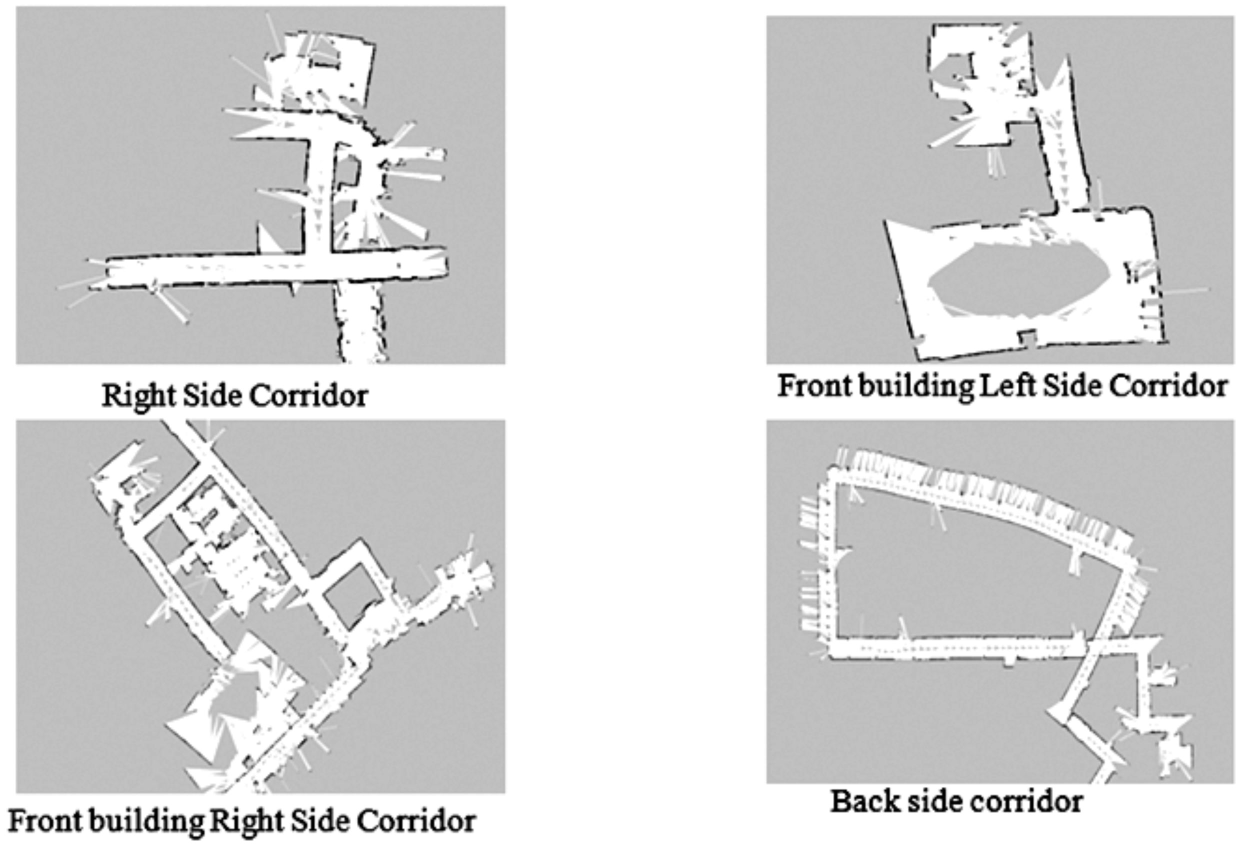

Right Side Corridor
Front building Left Side Corridor
Front building Right Side Corridor
Back side corridor

Figure 4: Sample 2D Map generated using our Map building Web Service.

Table 2
Details of the bag files used in our experiment

Bag Files	*File Size*	*Total Messages*	*Transmission Time (sec)*
Right Side Corridor	600.1MB	111827	3
Front building Left Side Corridor	387.8MB	78111	1
Front building Right Side Corridor	1.4 GB	270273	7
Back side corridor	1 GB	199919	5
MIT Stata center – 2nd Floor	4.3 GB	101230	15
MIT Stat center – 4th Floor	625.8 MB	380172	4

We used time streaming method for transferring sensor data from our client to the server. From our observation, the average time to transmit the sensor data is significantly less compared to map building process. So energy utilized to transmit the sensor data will be comparatively less compared to the energy utilized for processing. So it saves the battery life for the robot. Moreover, if we implement parallel processing of SLAM using Hadoop Clusters processing time will also be reduced.

6. CONCLUSION AND FUTURE WORK

With the higher computational capacity available in the cloud, robots can offload the computationally intensive tasks to the cloud. So that the robots can execute the tasks efficiently and also reduces its power consumption. In this paper, we have described the RoboWS system which allows the heterogeneous robots to access the Web

Service available in the cloud. We presented the design and implementation of the RoboWS system which uses the gSOAP as the communication mechanism for robot cloud interaction. Our proposed framework has been tested by implementing a Map Building Web Service using gSOAP. The implementation and results showed that communication using gSOAP is feasible for ROS enabled robots to communicate to the cloud.

In our framework we have managed to provide a list of Web Services that can be accessed by heterogeneous robots for their common task like vision processing, map building, path planning, etc. We are also working on improving our system to provide the parallel implementation of those algorithms which can be executed on the cluster of computers. We tested the system in our lab with Intel Xeon processor; we also want to test our system by deploying in the cloud environment and work with performance measures.

REFERENCES

[1] R.Ayres, S.Miller, “Industrial robots on the line”, The Journal of Epsilon Pi Tau, vol. 8, pp. 2–10, 1982.

[2] V. Kumar, D. Rus, G. S. Sukhatme, “Networked robots, in: Springer Handbook of Robotics”, Springer, pp. 943–958, 2008.

[3] J. Forlizzi, C. DiSalvo, “Service robots in the domestic environment: a study of the roomba vacuum in the home”, in: Proceedings of the 1st ACMSIGCHI/SIGARTconferenceonHuman-robotinteraction, ACM, pp.258–265, 2006.

[4] R. Schraft, C. Schaeûer, T. May, “Care-o-bot tm: The concept of a system for assisting elderly or disabled persons in home environments”, in: Industrial Electronics Society, IECON’98. Proceedings of the 24th Annual Conference of the IEEE,vol. 4,pp.2476–2481, 1998.

[5] P.Mell, T.Grance, “The NIST deûnition of cloud computing”, Communications of the ACM, vol. 53, pages 50, 2010.

[6] B.Kehoe, S.Patil, P.Abbeel, K.Goldberg, “A survey of research on cloud robotics and automation”, IEEE Transactions on Automation Science and Engineering, vol. 12, pp. 398–409, 2015.

[7] K. Channabasavaiah, K. Holley, E. Tuggle, “Migrating to a service-oriented architecture”, IBM Developer Works, vol. 16, 2003.

[8] K. Wagh, R. Thool, “A comparative study of soap vs rest web services provisioning techniques for mobile host”, Journal of Information Engineering andApplications, vol. 2, pp. 12–16, 2012.

[9] L.Riazuelo, J.Civera, J.Montiel, “C2tam: A cloud framework for cooperative tracking and mapping”, Robotics and Autonomous Systems, vol. 62, pp. 401–413, 2014.

[10] Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, G. W. Kit, “Davinci: A cloud computing framework for service robots”, in IEEE International Conference on Robotics and Automation (ICRA),pp.3084–3089, 2010.

[11] Y.Chen, Z.Du, M.Garc´ýa-Acosta, “Robot as a service in cloud computing”, in Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), pp.151–158, 2010.

[12] D. Hunziker, M. Gajamohan, M. Waibel, R. D’Andrea, “Rapyuta: The roboearth cloud engine”, in: IEEE International Conference on Robotics and Automation (ICRA), pp. 438–444, 2013.

[13] G. Mohanarajah, D. Hunziker, R. D’Andrea, M. Waibel, “Rapyuta: A cloud robotics platform,” IEEE Transactions on Automation Science andEngineering , vol. 12, pp. 481–493, 2015.

[14] I. Osunmakinde, V. Ramharuk, “Development of a survivable cloud multi-robot framework for heterogeneous environments”, International Journal of Advanced Robotic Systems, vol. 11, 2014.

[15] J.Raphael, “A hands-on tour: Google goggles visual search”, 2009.[Online;accessed09-November-2016].

[16] R.Limosani,A.Manzi, L.Fiorini, F.Cavallo, P.Dario, “Enabling global robot navigation based on a cloud robotics approach”, International Journal of Social Robotics, pp.1–10, 2016.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, “Ros: an open-source robot operating system, in: ICRA workshop on open source software, vol. 3, p.5, 2009.

[18] R. A. Van Engelen, K. A. Gallivan, "The gSOAP toolkit for web services and peer-to-peer computing networks", in: Cluster Computing and the Grid, 2nd IEEE/ACM InternationalSymposium , pp.128–128, 2002.

[19] J. M. Santos, D. Portugal, R. P. Rocha, "An evaluation of 2d slam techniques available in robot operating system", in: 13 IEEE International Symposium on Safety,Security,and Rescue Robotics(SSRR), pp.1–6, 2013.

[20] B. D. Gouveia, D. Portugal, D. C. Silva, L. Marques, "Computation sharing in distributed robotic systems: a case study on slam", IEEE Transactions on Automation Science and Engineering, vol.12, pp. 10–422, 2015.

[21] C. Crick, G. Jay, S. Osentoski, B. Pitzer, O. C. Jenkins, "Rosbridge: Ros for non-ros users", in: Proceedings of the 15th International Symposium on Robotics Research, 2011.

[22] B.Alexander, K.Hsiao, C.Jenkins, B.Suay, R.Toris, "Robot web tools [rostopics]", IEEE Robotics &Automation Magazine, vol. 19, pp. 20–23, 2012.

[23] Srinivasan, Lakshminarasimhan and Scharnagl, Julian and Schilling, Klaus, "Analysis of websockets as the new age protocol for remote robot tele-operation", in IFAC Proceedings, vol. 46, pp.83-88, 2013.