Cloud Computing Architectures for Mobile Robotics

A.A. Dyumin, L.A. Puzikov, M.M. Rovnyagin, G.A. Urvanov, I.V. Chugunkov
National Research Nuclear University "MEPhI"
Moscow, Russian Federation
a.a.dyumin@ieee.org, puikov.lev@gmail.com, mmrovnyagin@mephi.ru, ivchugunkov@mephi.ru

***Abstract* — in last decade, classic IT-infrastructure in modern enterprises has been changed sufficiently by cloud computing. However, cloud approach can be used effectively in other applications. In this paper, we explore applicability of cloud paradigm in mobile robotics. First approach is using some classic cloud architectures, as IaaS, PaaS and SaaS, with heterogeneous distributed mobile robotics systems (MRS) – for example, its can be used to offload a time consuming computations from mobile platforms to remote nodes or services. Second approach is using some special "robotics" types of clouds, as Robot-as-a-Service and Function-as-a-Service to leverage abilities of a single robotics platform in MRS as services to MRS's users or other mobile platforms. We explore both approaches and give them some architectural examples.**

Keywords — mobile robotics, cloud computing, cloud robotics, software architectures

I. Introduction

Invention of cloud paradigm is rapidly changing IT-infrastructures in modern organizations. Dozens of vendors currently provide software and hardware systems that act in concern of one or several cloud models. According to [1], every cloud architecture must comply with five essential properties: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. However, this approach can be expanded beyond "classic" cloud (i.e. business) applications – one example is a cloud robotics.

Mobile robotics can obtain huge benefits from previously mentioned properties, most known of those benefits are offloading heavy computing task to the compute cloud, share knowledge about environment or even share actively learned skills between mobile robotics platforms through cloud-enabled service. It's worth to mention that this applications is not something brand new, for example, "remote brains" [2, 3] or RCIP (Remote Centre of Information Processing)[4] was introduced in much earlier works, but cloud approach can add valued properties to it leading to more effective and cheaper robotic systems. As expected, after first introduction of cloud robotics concept [5, 6] and one of the first implementation that we can call proof of concept [7] in 2010, many other robotic laboratories started active researches in this area.

In this paper, after analysis of related works we introduce some taxonomy for cloud robotic, and then show some steps to cloud robotic that we made in our laboratory.

The research supported in part by the Russian Foundation of Basic Researches (RFBR), grant #14-07-00843 – "Development and research of multi-channel information processing techniques for intelligent decision-making system"

978-1-4799-7306-4/15/$31.00 ©2015 IEEE

II. Related Works

Nowadays, we can see four major types of publications on cloud robotics.

First one, after several years of active researches in cloud robotic field, authors try to give some systematic view on the usage of the cloud concept in robotic. Most notable article is [8]. In the article authors outline some challenges and constraints in networked robotics, and then they give description of cloud robotic architectures, highlight a few cloud robotic applications and address some problems in its design and operating such as computation, communication and security challenges and need of optimization framework. Authors give some classification of robotic architectures based on robot-to-cloud computing models: peer-based model (robots and VM are equally treated as computing units), proxy-based-model (one robot treated by other networked robots as a proxy system for communication with a computing cloud) and clone-based model (every robot has corresponding clone on the cloud). Next good approach to classification and systematization of cloud robotic was made in [9]. In the paper authors evaluate cloud robotics definitions and introduce classification of cloud robotic architectures – they separate them in three groups: Cloud Networks (classic network robotic architectures), which they divide in three subgroups – Teleoperated robots (telepresence systems), Multi-robot systems and Sensors arrays; Cloud Computing (here they mainly refer to mentioned early [8]); Cloud Storage (robotic systems use remote storage for knowledge sharing and acquirement). Then they discuss some cloud robotic example, typical cloud robot dataflow, cloud robotic roadmap and limitation of cloud robotic systems.

Scientific papers that can be assigned to the second type of works on cloud robotic field describe general or specialized cloud robotic frameworks. We can highlight several of them, for example at [10] authors proposed architecture of the senior companion robot system, some components of which can be characterized as cloud services – web-based user remote management service for remote robot control and the robotic multimodal interactive computation services for H2M (human-to-machine) interface such as speech recognition and sound source estimation, face recognition and text-to-speech services. They conduct evaluation of the proposed system using two lab robots and five senior users for different scenarios and obtain promised results. In [11] researchers introduced REALcloud – cloud platform to development robotic application with the ability to control robots over the network. They utilize early-

developed REALabs platform for networked robots and maintain several cloud infrastructure services as virtual machine (VM) management service and session validation service that grant access for VM-deployed applications to robotic resources. The core of proposed architecture is Workflow Management System; it provides layered service infrastructure and based on popular Grid framework. Paper [12] introduces 'Robot-Cloud' – framework for low cost robots that allow offloading resource intensive computation to the remote cloud nodes. The framework uses Service Oriented Architecture (SOA) to provide services of map building, path planning and object recognition; by utilization well-known robotic software packages, as ROS [13] and Webots, it provides development platform that also delivered as a service; furthermore, by utilization Map-Reduce it gives ability to offload from mobile platform to compute cluster some general algorithms. As a result, this framework supports three service delivery models used in modern clouds.

Different approach is used in [14]. In that paper, authors describe design and implementation of UNR-PF – platform for cloud networked robotic services. Essentially, this platform is middleware that gives to developer another level of abstraction from service robotic platform, and provides service robot (e.g., vacuum cleaning or grass-mower platforms) as a cloud service. Using this platform, developer can request some specification of robotic platform and the platform fulfill it if it is possible in a current situation. Authors conduct some experiments with the proposed architecture and give usage example of cleaning robotic service for elderly people. On the move of their research [15], authors describe integration of UNR-PF with the RoboEarth [16] knowledge base to achieve knowledge-enabled task execution in cloud robotic environment. They also describe implementation of example scenario for human-robot interaction in convenience store powered by knowledge-enabled cloud robotic architecture.

In work [17] authors proposed another cloud robotics framework with the main idea to increase survivability of multi-robot systems. They adopt two-tier model for cloud robotic: first tier is Robot-to-Cloud (R2C) tier that exploits "remote-brain" concept mentioned early; second-tier is Robot-to-Robot (R2R) tier, which allowed leveraging the computing and decision-making capability for each robot in local area by pooling all robotic resources on locality basis. They use energy evaluation models for estimation of effectiveness and survivability distributed robotic system. In work [18] authors designed cloud robotic sensor observation framework that incorporates cloud computational resources and mobile robotic platforms with mounted sensors. Main task of this framework is optimal placement sensors data processing tasks in the cloud or on the robotic node due satisfaction QoS constraints posed by dynamic network conditions. In addition, this paper introduced a mapcell-based robotic observation algorithm and gave some real life example of framework and algorithm usage.

Third type of publications on cloud robotics is papers investigated some theoretical aspects of building effective cloud robotics architectures that can be applied in vast majority of cloud frameworks. For example, in [19] authors proposed bandwidth allocation mechanism for cloud robots based on game theory – they postulated optimization problem as resource allocation task and described algorithmic modification of TCP network protocols to incorporate their proposal. This work was extended in [20], where authors investigate two-tiered robotic system with well-equipped leading robots and poor-equipped followers – they used leader's resources to solve co-localization problem. They utilized market-based scheduling for fair resource allocation. Follow-up work [21] from same authors introduce usage of hierarchical auction-based mechanism in cloud robotic systems.

For the last type of researches in cloud robotic field we can classify work that describe usage of common or specialized (but non-robotic) cloud infrastructures with networked robots. In [22], researchers used virtual cloud recourses accessed by the scouting robots to store navigational information. In [23], authors used cloud computational resources for offloading computational intensive data processing for vision acquisition system and formation controls algorithms. Integration of robotics platforms with cloud healthcare systems was depicted in [24].

Unfortunately, we cannot mention in this brief report all great efforts that scientific community had made in last five years, but highlighted works can give some notion about current trends in and diversity of cloud robotic.

III. Taxonomy for Cloud Robotic

Now we can introduce taxonomy for cloud robotic (Figure 1). We can divide cloud robotic architectures in two big families: in first family, robots utilize classic cloud infrastructures as compute and storage or general application-level services such as visual object or speech recognition (that can be accessed and may be useful not only by / for robots); in second family, robotic system provides some robotic services. Mixed environment where users take services from both worlds is also possible. First family partially related to M2C/M2M concept mentioned in earlier works [8, 9], more precise we can depict these architectures as C->R. Second family partially related to M2M/H2M models, and we can depict those architectures as R->U, where U (user) can be other robot (R) or H (human).

Next level for specification is – what kind of services are used by robots: external (EC), mixed (MC) or internal (IC) toward robotic system. In EC->R, robotic platforms do not provide or use services to / from other robotic platform and only use external cloud recourses; in IC->R, robotic platform consume resources only from other platform in the robotic system. [12, 22, 23] are examples of EC->R systems and [20] is classic example for IC->R.

Last step is specifying service delivery model that can be for first family: IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service), SaaS (Service-as-a-Service) or ever XaaS (Everything-as-a-Service). Most or earlier mentioned researches exploited one or two delivery models – mainly SaaS [10] or IaaS [11] or both [17]. There a lack of papers where usage of common PaaS services is described, but in some work PaaS model used for robot-specific applications [11]. [12] is example of XaaS approach.

We can split second family in three delivery model: FaaS (Function-as-a-Service) or EaaS (Equipment-as-a-Service), but fist term has a broader notion, so we prefer FaaS), RaaS (Robot-as-a-Service) and RSaaS (Robotic-Service-as-a-Service).

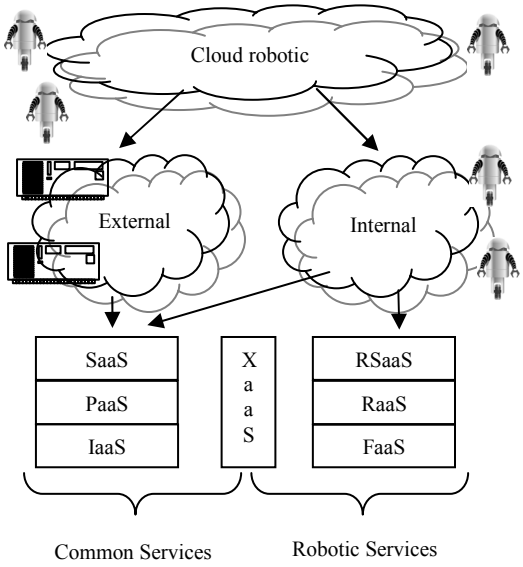

Fig. 1. Taxanomy for cloud robotics

On the one hand, FaaS is a low-level delivery model, by using this notion robotic system can give access to some robots resources as laser range finders, cameras or manipulators (EaaS); on the other hand, we can provide some higher level robotic tasks solvers as path planners, 2D/3D-map archives, SLAM frameworks, etc. Equipment and solver services can be shared between multiple task or / and users. For example, some particular robot can use video cameras from other platform to locate distant object of interest. [4] is example for FaaS, and it will be elaborated a little further in next part of the paper.

In RaaS model, user obtains access to some robotic platform as single entity, classic examples for this delivery model are teleoperated robots, where subscription to particular platform provides some functionality, some frameworks allow not only operate robot in manual or tactical mode, but also allow develop software using special API. [14] can be the good example for RaaS.

In RSaaS model, robotic system deliver high-order services to the end-users (mainly to the human-users), examples of this services are social robots – vacuum cleaning, drugs or supply delivery platforms, etc. Here users ask the framework about particular services with given constraints, but not ask about particular robotic platform. Ideas of RSaaS announced in some works, e.g. [10], but we have not seen the implementation of it in a mature framework yet.

IV. To the Cloud – FaaS Infrastructure

Using UCF [4] and control software decomposition by different layers, we can obtain flexible architecture for robotic system that natively supports notion of FaaS. The typical layer decomposition is depicted on the Figure 2.

Channel drivers (CD) are on the bottom layer. CD is a hardware-dependent and OS-dependent software that provides hardware interfaces (COM, USB, etc.) or network access with the unified program interface to the upper layers.

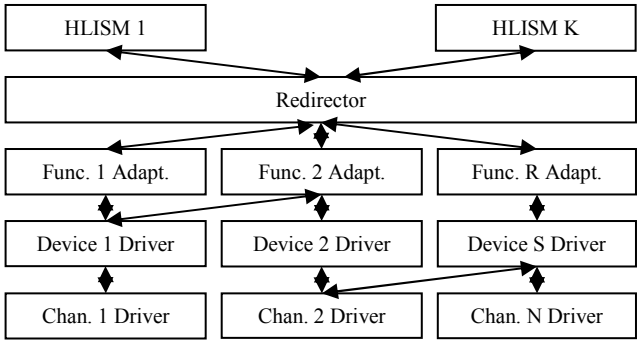

Fig. 2. Software layers in UCF

Device drivers (DD) are on the second layer. DD is a hardware-dependent and OS-independent software encapsulating all details (protocols, boundary values, etc.) for real devices that may use as mobile robot subsystems.

Function adapters (FA) are on the next layer. FA is a hardware-independent and OS-independent software that encapsulates methods and algorithms for the class of devices (e.g. a laser measuring system, a general locomotion, etc.). After publication in service catalog, it can provide services for local and remote robotic users.

The redirector is a central software module in this layer hierarchy. It serves as linker between bottom-level and upper-level software modules; also, its responsibility is construction UCF software stack on particular platform and services publication. In addition, redirector serves as proxy for remote services, for every available FaaS in local robotic domain it creates entry in services catalog. In heterogeneous ad-hoc networks, redirector replicates his own entries with his neighbor.

High-level intelligent software modules (HLISM) are on the top layer. HLISM is special software that uses redirector to provide some robotic functionality as map building or objects tracking; it can be shared as service and / or it can use other cloud robotic services to perform its task.

Similar software hierarchy is used in many other projects, e.g. [25, 26], to build control software with hardware independency.

V. To the Cloud – Actor-based Infrastructure

The next step to the cloud is implementation of fault tolerant control and data processing system distributed between multiple robotic nodes. This approach can be extended to concept of “durable” robotic system – where computing nodes are distributed across robotic body and loss of body part do not cause loss of robots. It can be used in unmanned aerial vehicles (UAV), in robotic systems for hazard environments or mission-critical applications; for example, robot could still return to

base after taking damage during task execution or even continue task performance.

One approach, to develop that kind of systems is using Actor-based computational model [27]. An actor is an autonomous object that has a state, communicate with the other actors through message passing interface ("asynchronous mailbox") and message processing executed in separate thread. This approach is used in control and model software in robotic systems, for example in [28] or to develop software for cloud computing environment [29].

Nowadays, many frameworks support actor computational model [30]. Akka [31] is the most complete, mature and frequently updated framework amongst all actor model frameworks. It is developed by Typesafe Inc that specializes in field of reactive programming for many years. This framework provides rich actor model, SSL protected connection between the nodes, actor fault tolerance and error-handling fine tunable by a programmer. Akka supports usage of many computing nodes transparently for a programmer. It means that it has single namespace for all actors no matter on which node they are running. Other features of the framework are migration of actor from one node to another, actor's lifecycle monitoring, detailed log of system events and errors. Detailed documentation and usage examples are available at the Akka's site. Such solid approach to framework development makes it fast and easy to create and debug powerful distributed software.

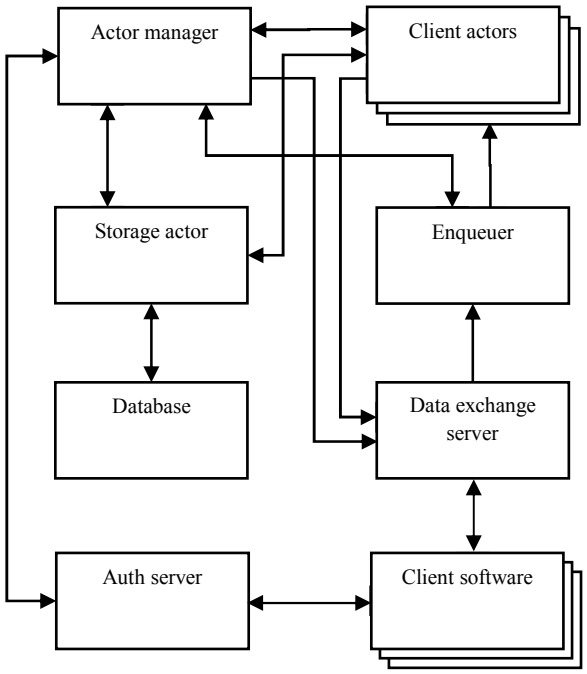

Fig. 3. Actor-based arcitecture of the robotic node software

Computing node has the following components:

- Actor manager – main actor that manages creation and deletion of other components.
- Authentication server – network server based on Netty [32] framework. It listens for authentication requests, checks authentication data and sends results of this check back to clients
- Data exchange server – also network server based on Netty framework. It accepts connection from already authenticated client and exchanges user messages with them. This actor is a system access point for client software
- Enqueuer – actor that receives user messages from data exchange servers and forwards them to corresponding client actor for processing.
- Data storage – actor that on the one hand is a client to database and on the other hand receives request for storing or reading data from other components
- Client actors – actors that are started and stopped according to connections and disconnections of client software. These actors are created on a per user basis.

The diagram of system structure is showed on a figure 3.

A. Actor manager

On system start, actor manager creates and start authentication server, data exchange server, enqueuer and storage actor then sends them messages with initialization data. On system stop, actor manager stops authentication server first then client actors and then other system actors.

Actor manager stores information about IP addresses and ports of authentication and data exchange servers, started client actors and number of client connections to each client actor. At runtime, it receives messages from authentication server about clients that have confirmed their identity, creates and initializes client actors if needed. If client actor have initialized successfully manager generates token that is used by client to access data exchange server and sends it to enqueuer. If there were errors during client actor initialization, manager does not generate a token and sends information about this error to authentication server.

B. Authentication server

Authentication server listens for client requests directed to the system. It checks if received credentials are allowed to access the system and if they are sends message containing client name to actor manager. Credentials are stored in database available through storage actor in from of login, password hash, message handler class name. After sending client name to actor manager, authentication server waits for message indicating either success or failure of system preparations to accept client connection. After successful preparation authentication server sends client message with token, IP address and port of data exchange server. In other case, it sends message indicating failure and cause of this failure.

C. Enqueuer

Enqueuer receives client messages from data exchange server, searches its map of client channels and actor references and forwards client message to client's handler.

In addition, enqueuer stores valid tokens with associated client. Token is valid one second after client authentication and only for IP address of that client. If token was not used during this time it is deleted from token list and client must request authentication again. If token was claimed then enqueuer gets client name associated with this token and client channel id from message with token and puts them in inner map.

One last type of message enqueuer receives are messages with information about client actors' starts and stops. On receiving such type of message enqueuer adds or deletes association "client name – actor reference" in inner map.

D. Data exchange server

Data exchange server receives messages from clients over a network and sends messages from client actor to clients. To establish a connection with server client must send access token in first message.

For each client connection data exchange server stores channel id, which is used on enqueuer to identify an actor to forward message to. After accepting connection with a client all its messages are forwarded to enqueuer.

To protect itself from DOS and DDOS attacks data exchange server stores to lists of IP addresses. The first one is white list – list of addresses allowed to connect. This list is filled with addresses provided by actor manager with access tokens. The other one is black list – list of addresses not allowed connecting. Addresses are put to this list after receiving five consecutive wrong tokens and are kept in this list for 5 minutes.

E. Storage actor

Storage actor receives and executes requests to database from any actor in the system. For each request actor sends result of operation back to actor that made a request. Storage subsystem is built in such a way that actors use unified API independent of used database type. This unified API is translated to API provided by database developer before execution of request. This approach allows to easily implementing support to nearly any database that has a java driver.

F. Client actors

Client actor is created after client first connection on a per client basis. A message is sent by actor manager to client actor at the start contains class name of a message handler provided by programmer that uses the system. Client actor tries to dynamically load and instantiate this class. If it succeeds, it send to actor manager client that it have been initialized successfully, if not then message with initialization error message. Client actor also provides programmer with API to access different system function from within message handler such as storage request, sending message back to client, sending message to other client's actor, and creating child actors for parallel computing purposes.

VI. Discussion and Future works

This paper introduced solid framework for using in further cloud robotic applications development. Using UCF in robotic environment gives us the level of abstraction for interaction between robotic platforms and remote control and compute systems. We effectively use UCF for several years to interact with some of ours lab robots by utilization FaaS delivery model.

Newly introduced actor-based framework allows development of more resilient mobile robotics software. Now we are in the process of testing and evaluating of this approach. We already developed the basic video processing system for our distributed robotic system using this actor model, and we are currently evaluating its performance and reliability. Some promising results were obtained already, so we believe that this computing model can be used as infrastructure for future XaaS robotic clouds.

However, some of cloud aspects were not implemented yet. We are planning integration of our robotic environment with external computing resources based on Map-Reduce computing model and GPU-computations support [33]. Another direction in our researches is integration with open and commercially available cloud services to obtain general compute and storage resources by using automation API and frameworks as OpenStack, Eucalyptus, etc [34]. A delivery of the classic robotic application as SLAM, navigation, surveillance, etc, as service in our robotic environment is our current task also.

VII. Conclusion

In this paper, we explore the concept of cloud robotics and the current state of researches in this modern field. Cloud concept became very popular in mobile robotics as it allows to increase resiliency of robotic software and to decrease cost of robotic platforms without service quality loss. There are many approaches and frameworks using cloud service delivery model in robotic applications. Therefore, we introduced some basic taxonomy for cloud robotics and gave some examples for different classes.

In second part of this paper, we described ours past and current efforts in "cloudization" process in our robotic laboratory. By using UCF we deliver access for remote application to equipment on the robotic platforms and some basic robotic abilities as a service. Actor-based framework developed in the recent year can sufficiently increase robotic systems reliability without huge efforts in software development process.

Acknowledgment

The authors express their sincere gratitude to the Head of Computer Systems and Technologies Department of the National Research Nuclear University "MEPhI" Professor Ivanov M.A. for his help and support in conducting the research, and also to the Head of the interdepartmental

laboratory “Hybrid computing systems” Associate Professor Vasilyev N.P. and Head of the faculty research laboratory “Robotics” Associate Professor Chepin E.V. for providing the equipment and for consulting. Also authors thanks our current and former students Sergey Chernih, Andrey Ozeretskovskiy, Alexander Gridnev and Konstantin Suminov for their participation in development of robotic software referenced in this work.

REFERENCES

[1] T. Grace and P.Mell, “The NIST Definition of Cloud Computing”, National Institute of Standards and Technology, NIST Special Publication 800-145, September 2011.

[2] M. Inaba; S. Kagami, K.Sakaki, F. Kanehiro, H. Inoue, “Vision-based multisensor integration in remote-brained robots”, IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, 1994.

[3] S. Kagami. Y. Tamiya, M. Inaba, H. Inoue, “Design of real-time large scale robot software platform and its implementation in the remote-brained robot project”, in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS ’96), 1996

[4] A.A. Dyumin. “Architecture of reconfigurable software for mobile robotic systems”, in Proc. of the 11th International Workshop on Computer Science and Information Technologies (CSIT'2009), Crete, Greece, 2009.

[5] J. J. Kuffner, “Cloud-Enabled Robots”, in Proc. of IEEE International Conference of Humanoid Robotics, Nashville, 2010

[6] Erico Guizzo, “Robots With Their Heads in the Clouds”, IEEE Spectrum, vol. 48, pp. 16-18, March, 2011,

[7] R. Arumugam, V.R. Enti, Liu Bingbing, Wu Xiaojun, K. Baskaran, Foong Foo Kong, A.S. Kumar, Kang Dee Meng, Goh Wai Kit, “DAvinCi: A cloud computing framework for service robots”, in proc. of IEEE International Conference on Robotics and Automation (ICRA), Anchorage, USA, 2010

[8] Guoqiang Hu, Wee Peng Tay, Yonggang Wen, ”Cloud Robotics: Architectire, Challenges and Applications”, IEEE Network, May/June 2012

[9] S. Jordan, T. Haidegger, L. Kovacs, I. Felde, I. Rudas, ”The rising prospects of cloud robotic applications”, in Proc. of IEEE 9th International Conference on Computational Cybernetics (ICCC), 2013

[10] Yan-You Chen, Jhing-Fa Wang, Po-Chuan Lin, Po-Yi Shih, Hsin-Chun Tsai, Da-Yu Kwan, “Human-robot interaction based on cloud computing infrastructure for senior companion”, in Proc. of IEEE Region 10 Conference TENCON 2011, 2011

[11] L. Agostinho, L. Olivi, G. Feliciano, F. Paolieri; D. Rodrigues, E. Cardozo, E. Guimaraes, “A Cloud Computing Environment for Supporting Networked Robotics Applications”, in Proc. of IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (DASC), 2011

[12] R. Doriya, P. Chakraborty; G.C. Nandi, “‘Robot-Cloud’: A framework to assist heterogeneous low cost robots”, in Proc. of International Conference on Communication, Information & Computing Technology (ICCICT), 2012

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng, “ROS: an open-source Robot Operating System”, ICRA workshop on open source software, 2009

[14] J. Furrer, K. Kamei, C. Sharma, T. Miyashita, N. Hagita, ”UNR-PF: An open-source platform for cloud networked robotic services”. In Proc. of IEEE/SICE International Symposium on System Integration (SII), 2012

[15] M. Tenorth, K. Kamei, S. Satake, T. Miyashita, N. Hagita, “Building knowledge-enabled cloud robotics applications using the ubiquitous network robot platform”, in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013

[16] M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Galvez-Lopez, K. Haussermann, J. Montiel, A. Perzylo, B. SchieÃŸle, O. Zweigle, and R. van de Molengraft, "RoboEarth-A World Wide Web for Robots," Robotics &Automation Magazine, vol. 18, no. 2, pp. 69-82, 2011.

[17] V. Ramharuk, I. Osunmakinde, “Cloud Robotics: A Framework Towards Cloud-enabled Multi-robotics Survivability”, in Proc. the Southern African Institute for Computer Scientist and Information Technologists Annual Conference 2014 on SAICSIT 2014 Empowered by Technology, 2014

[18] F. Nordlund, M. Higashida, Y. Teranishi, S. Shimojo, M. Yokoyama, M. Shimomura, “Designing of a Network-Aware Cloud Robotic Sensor Observation Framework”, in Proc. of IEEE 38th International Computer Software and Applications Conference Workshops (COMPSACW), 2014

[19] Lujia Wang. M.Q.-H. Meng, “A game theoretical bandwidth allocation mechanism for cloud robotics”, in Proc. of 10th World Congress on Intelligent Control and Automation (WCICA), 2012

[20] Lujia Wang, Ming Liu, M.Q.-H. Meng, ”Towards cloud robotic system: A case study of online co-localization for fair resource competence”, in Proc. of IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012

[21] Lujia Wang, Ming Liu, M.Q.-H. Meng, “Hierarchical auction-based mechanism for real-time resource retrieval in cloud mobile robotic system”, in Proc. of IEEE International Conference on Robotics and Automation (ICRA), 2014

[22] S. Srivastava, A. Sarkar, B.S. Manoj, “Hazard control algorithms for heterogenous multi-agent cloud-enabled robotic network”, in Proc. of IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), 2013

[23] L. Turnbull, B. Samanta, “Cloud robotics: Formation control of a multi robot system utilizing cloud infrastructure”, in Proc. of IEEE Southeastcon, 2013

[24] Min Chen, Yujun Ma, Sana Ullah, Wei Cai, Enmin Song, “ROCHAS: robotics and cloud-assisted healthcare system for empty nester”, in Proc. of the 8th International Conference on Body Area Networks (BodyNets’13), 2013

[25] J. Kramer, J Magee, “Self-Managed System: an Architectural Challenge”, in Proc. of the International Conference on Software Engineering 2007 Future of Software Engineering (FOSE’07), 2007

[26] C. Santano, “An Erlang Framework for Autonomous Mobile Robots”, in Proc. of the ACM SIGPLAN workshop on Erlang, 2007

[27] G. Agha, I. Mason, S. Smith, and C. Talcott. A foundation for actor computation. Journal of Functional Programming, 7(1):1–72, 1997.

[28] E..Pereira, C. Potiron, C.M. Kirsch, R. Sengupta, “Modeling and controlling the structure of heterogeneous mobile robotic systems: A bigactor approach”, in Proc. of IEEE International Systems Conference (SysCon), 2013

[29] S. Bykov, A. Geller, G. Kliot, J. R. Larus, R. Pandya, J. Thelin, ” Orleans: cloud computing for everyone”, in Proc. of the 2nd ACM Symposium on Cloud Computing, 2011

[30] R. K. Karmani, A. Shali, G. Agha, “Actor frameworks for the JVM platform: a comparative analysis”, in Proc. of the 7th International Conference on Principles and Practice of Programming in Java, 2009

[31] J. Allen, “Effective Akka”, O'Reilly Media, Inc., 2013

[32] N.Maurer, “Netty in Action”, Chapter 1, Manning Publications Co, erly access edition

[33] N.P. Vasilyev, M.M. Rovnyagin, “Hybrid clusters for budget supercomputers and cloud computing”, Automation and Remote Control, vol. 75, issue 10, pp. 1869-1874, October 2014.

[34] G. von Laszewski, J. Diaz, Wang Fugang, G.C.Fox, “Compirison of Multiple Cloud Frameworks”, in Proc. of IEEE 5th International Conference on Cloud Computing (CLOUD), 2012