ClusterBots: A Modular Robotic architecture for shared Autonomy and Resources

Swarnabha Roy
swarnabha7@tamu.edu
Department of Electrical and Computer Engineering
Texas A&M University

Dezhen Song, and Shu-Hao Yeh
dzsong@tamu.edu, and ericex1015@tamu.edu
Department of Computer Science and Engineering
Texas A&M University

***Abstract* - The importance of coordination among multiple robots has grown significantly over the past few years. Researchers have so far focused on coordination between similar types of robots. Existing Modular Robotic Systems (MRS) consist of robots that are identical in shape, size, power requirements, and computational resources. Establishing proper coordination among different individual robots will enable more applications but it still needs a lot of refinement in terms of optimizing the algorithms and handling the computations to establish practical tasks that demand a higher level of complexity and coordination. We discuss the existing modular robotic architectures and compare them and, in the end, propose a new optimized architecture. MRS introduces reliability which is caused by information sharing and data fusion among the robots, and fault-tolerance which is provided by information redundancy. For example, multiple robots can localize themselves more efficiently in an unknown environment if they exchange their positional data whenever they sense each other in their vicinity. We can further modify the MRS to make it more modular, giving each of the robots' further capabilities in terms of computational performance, energy requirements, and mobility within a smart manufacturing or industrial environment. In this work, we propose a new modular robotic architecture called ClusterBots where various modular robotic clusters work together by sharing information via a Cloud Server. The central cloud server can be used for shared autonomy among the modular multi-robot units by sharing resources and sensor information.**

Index Terms - Modular Robotic System, Coordination, Network Security, Computation offloading

INTRODUCTION

The emergence of Cloud robotics provides an efficient alternative for offloading the physical computations from the robot to a cloud computing infrastructure. The creation of the World Wide Web has greatly advanced the possibility of connecting robots together to an external machine. Networked Robotics [1] has been the classical name given to robots connected over a network for functional collaboration. Naturally, this domain has seen a lot of paraphrasing with overlapping meanings. K. Goldberg et al. [2] were the first to connect a robot to the World Wide Web and control it through a web browser. They showed that the internet can be used as a means to control classrooms and other scientific lab equipment remotely. In 2009, people came up with 'RoboEarth' [2] which connected robots worldwide and discussed the key requirements for information sharing among robots, including performance optimization. Researchers highlight the fact that sharing data among robots can lead to faster learning. Robots with different hardware architecture can create a common database to share learnings. Cloud Robotics [4] is a platform that offloads the robot functions administered on a robot side to the cloud environment. The problem with it was that it was a completely different robotic technology of its own and the programmers needed to develop a new skill to make robot services. In [5], the researchers develop their high-level language 'CHARON' for the modular specification of interacting hybrid systems based on the notions of agents which have a finite set of behaviors or modes. Among the recent advancements in robotics, Smart Cloud [6] tries to mitigate this problem by using a JavaScript-based framework for application development. In [7], an open-source cloud robotics platform 'Rapyuta' has been discussed. It is an addition to the 'RoboEarth' technology where the robots add their learnings to a common repository. In this system, some of the robot's onboard computational processes are uploaded to a commercial data center. In 2017, a new protocol called RosLink [8] was proposed to overcome the limitations of ROS over a Wide Area Network (WAN). All this research so far has been solely based on a platform having some dependencies on language.

Our work expands on the current state of the art by developing a platform-independent system. We will use a

ROS master as a cloud server to deploy code in any language and communicate with any robot connected to the server network.

COMPARISON OF THE EXISTING ARCHITECTURES

The modular robotic architecture can be traced back to multi-robot systems which are divided into various subparts based on resource sharing, power management, and computational capabilities. There is a very thin line of demarcation between swarm and modular robotics. The swarm is a broad term coined by observing the behaviors of the natural organisms working together to achieve a task and interact with the environment. For example, a swarm of bees, ants, and even birds flying together. Modular robots are a type of multi-robot system where multiple dependent and independent units connect to achieve a task. They can be identical like individual units of a swarm as well as heterogeneous.

Let's first discuss the basic traditional architecture where each module is identical to each other, for example, the GZ-I module discussed in [9]. They have their embedded processing capabilities with an independent onboard controller. Each module has got four connecting faces which gives the modules the ability to form a structure that can change its shape in two dimensions. Each module has one rotational degree of freedom and the movements are assisted by RC servos. The modules can be controlled individually or collectively for motion control through the I2C bus based on task requirements. This is a type of distributed architecture with no central control agent such that all the robots are equal concerning control and are completely autonomous in the decision-making process. Each of the modules acts as an independent complete unit, so we cannot introduce a high variation of tasks assigned to each of the modules. Although such modules are programmable, it is difficult to program the modules separately as the architecture complexity increases. For several applications, creating a monolithic module that can cater to all aspects of a problem can be very expensive and complex [14]. That is why creating multiple and more specialized entities that can share the workload with an external entity offers the possibility of reducing the complexity of the individual modules. Each of the units has an onboard power system which is not always adequate based on the duration and type of work. Also, the software is optimized to imitate a natural organism to perform a task that may or may not be suitable for industry standards.

There is a second class of architecture in modular robotics in which there might not be a central core structure. They might be clustered into different groups and still communicate with one another [10][13]. Decentralized architectures can be further categorized into distributed architectures and hierarchical architectures. Distributed architectures are good in terms of longer span but there is a limitation on the amount of computation and power requirements that each module can handle. Distributed intelligent systems may require more communication bandwidth to coordinate with the other entities in the system as they must act without complete knowledge of the states of the other entities [14]. This typically increases the uncertainty about the state of the system as a whole. It is difficult to create a Markov Chain [11] to switch between different states of operation and may require rigorous computations. Thus creating a fallback state in case of a network connection failure or power breakdown is quite challenging.

When operating in unstructured or dynamic environments with many different sources of uncertainty, it is very difficult if not impossible to design architecture that will be scalable and reconfigurable. An efficient modular robotic system should have efficient mobility. It should have an efficient inter-module communication system that has enough bandwidth for information and power-sharing between the modules. Other desirables include fault-tolerant systems and genderless robust connectors with specialized sensors (like cameras or ultrasounds) and linkers. Cost is also a factor. Using several simple robots can be cheaper to construct, easier to program than using a single powerful robot that is complex and expensive since it is designed specifically for a task [13].

THE PROPOSED ARCHITECTURE

The hierarchical architecture is a hybrid architecture, which is an intermediate between a centralized architecture and a distributed architecture [13]. There exist one or more local central control agents which organize robots into clusters. In contrast to a centralized architecture, a decentralized architecture can respond better to unknown or dynamic environments and usually has better robustness, reliability, adaptability, and flexibility. This architecture is communication between multiple masters from different clusters. Thus, information transfer between the local masters is the major bottleneck. There is a similar structure for distributed modular robots called Molecubes [12] that was built with the emphasis on the expandability of the architecture at reasonable costs. The molecubes is a broad term for a set of modules with similar connection interfaces. The modules can be a controller, gripper, wheel, camera, and other passive modules. This architecture has a lot of potential in terms of scalability and robustness but is still in the primitive stage of development. This is a kind of centralized heterogeneous architecture that was solely built to successfully replicate a robotic behavior with smaller robotic modules working together. There is no substantial application that can, beyond doubt, prove modular technology superior to traditional robotics. Hence, we incorporate the Cloud computing infrastructure to further enhance the capabilities of the architecture. We call it the "ClusterBots". We expand on the previous work where we propose an architecture with modular components that can connect to a cluster to serve a particular use case. There may be more than one similar

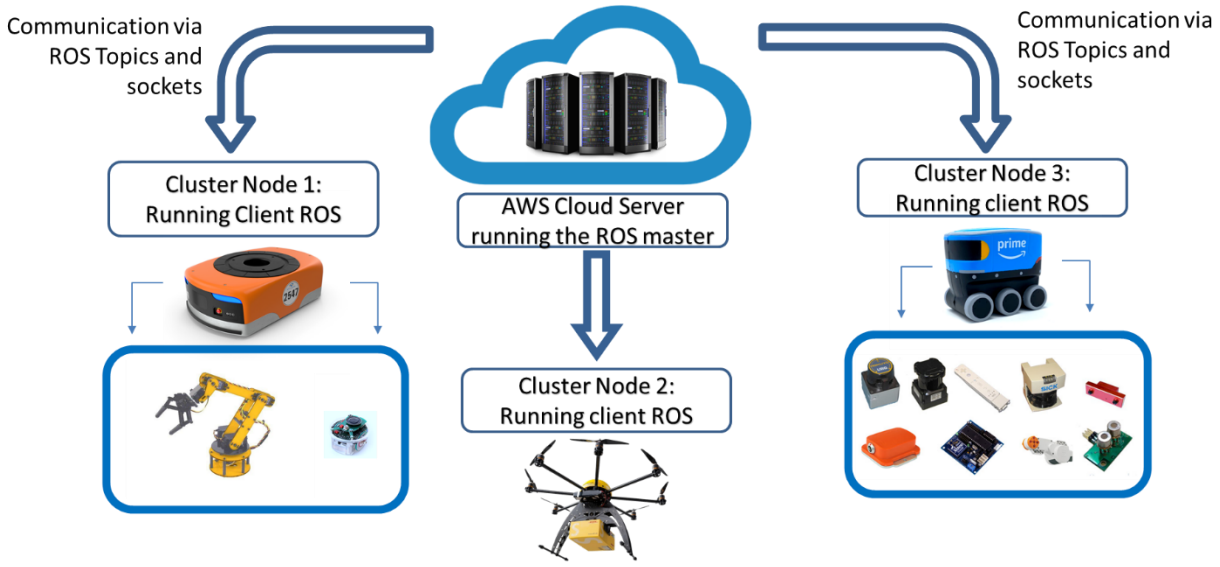

Figure 1: The proposed ClusterBot architecture

cluster that can work independently and share information through the Cloud Server. Fig. 1 shows a rough overview of the architecture as a whole. Making new independent modular clusters adds a new dynamic to the architecture. Not only it gives the overall structure expandability but also gives the clusters the ability to perform complex computations.

In this class project we try to utilize the collective information stored by individual clusters and try to map the room or environment. The main motivation of this architecture is mostly focused to aid industry smart manufacturing and farming. ClusterBots can help in seamless data collection from various sensors, drones, and cameras and can help in automating the production and/or farming.

We will be running the Robot Operating System (ROS) as the basic framework for information sharing and data processing. The Cloud Server will act as the ROS Master which will communicate with the Clusters through ROS Nodes. The Clusters will act as ROS Clients and will have a primary module or rover where all the other modules will connect to. Our job will be to test the performance of this architecture to the existing ones in terms of onboard CPU usage, latency, and security. We will compare the latency as well as the volume of data that can be handled simultaneously (throughput). The general idea will be to create microservices for Robots which are independent on their own where each module can act as a microservice. We can scale each microservice separately. We will also explore the fallback behavior of a module in case there is a break in the connection between the robot within the same network or the cloud server.

OUR IMPLEMENTATION

We have chosen an Amazon EC2 instance as the cloud server. The Cloud server is a Ubuntu 16 build with ROS Melodic. For the purpose of this class, we have chosen to show a three-way communication, transferring messages between two different ROS machines on two completely different networks. One of the machine is our lab computer which is running on Ubuntu 18.04 with ROS Melodic. And the other is a ROSBot 2.0 Pro running on ROS Melodic.

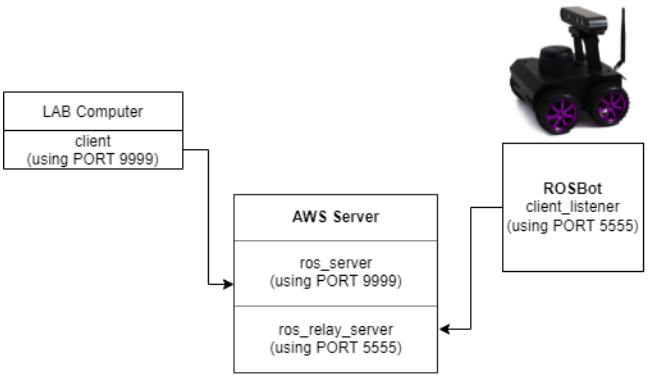

Figure 2: The implemented proptocol

ROS has a built-in multi-robot coordination functionality which allows us to define one ROS Master throughout the network. The other devices on the network have to define their ROS_MASTER_URI environment variable same as the ROS Master and all the ROS topics shared can be easily accessed throughout all the clients. This library is finicky and has a lot of glitches. It is not very reliable since none of the robots connected to the network has their own roscore running. They all connect to the master node via

ROS_MASTER_URI. If the connection is broken due to network failure, the system won't have a fallback behavior to fall upon. Also, all the robots will have to stay within the same WAN spawn by the Master.

To counter this issue, we have proposed a TCP/IP socket based ROS library and architecture that can not only connect to devices on different networks but at the same time have a fault tolerant behavior of their own. We are using different PORT numbers to establish different kinds of communication sockets. To subscribe to a particular message, the devices can connect through respective PORT and listen to messages via rostopics. The protocol implemented is shown in Fig. 2.

RESULTS AND FUTURE WORK

The ROSBot kinematics takes in information via rostopic /cmd_vel. We ran the client listener code on the ROSBot which subscribes to PORT 5555 where we published the /cmd_vel messages.

```
ubuntu@ip-172-31-11-116:~$ rostopic bw /cmd_vel
subscribed to [/cmd_vel]
average: 25.91B/s
        mean: 5.00B min: 5.00B max: 5.00B window: 2
average: 18.02B/s
        mean: 5.00B min: 5.00B max: 5.00B window: 5
average: 12.98B/s
        mean: 5.17B min: 5.00B max: 6.00B window: 6
average: 17.11B/s
        mean: 5.27B min: 5.00B max: 6.00B window: 11
average: 21.41B/s
        mean: 5.53B min: 5.00B max: 8.00B window: 17
average: 25.22B/s
        mean: 5.91B min: 5.00B max: 8.00B window: 23
average: 26.12B/s
        mean: 6.19B min: 5.00B max: 10.00B window: 27
average: 28.80B/s
        mean: 6.26B min: 5.00B max: 10.00B window: 34
average: 30.97B/s
        mean: 6.34B min: 5.00B max: 10.00B window: 41
average: 31.82B/s
        mean: 6.36B min: 5.00B max: 10.00B window: 47
```

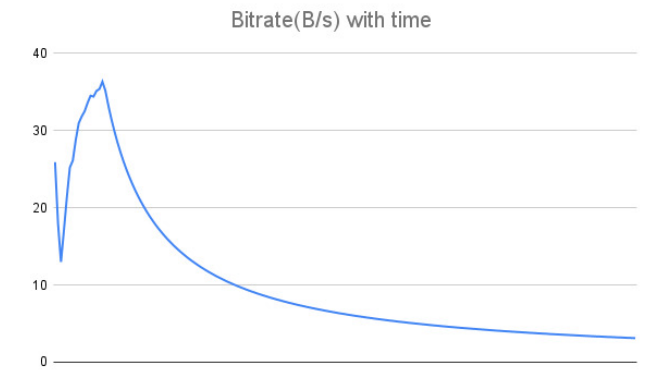

Figure 3: Bandwidth and Bitrate for the /cmd_ vel rostopic published

The data bandwidth and bit rate is significant enough to transfer a low resolution image within seconds as shown in figure 3. The bitrate spiked as we started to send data and gradually settled to zero on halt.

Hence, we successfully demonstrated the ClusterBot architecture using ROS and socket connections. We further plan to research more on the bandwidth of the ROS sockets and the number of rostopic pipelines we can open at the same time. Also our goal is to dynamically create rostopics and break large information or files into packets and transfer via different socket PORTs. In the future, we plan on exploring using IPv6 and create three different data socket pipelines based on the bandwidth requirement and urgency of the data: Ultra-fast, medium and high-definition. Ultra-fast pipeline will be the fastest with low bandwidth and high data rate and so on.We will use reinforcement learning based approach to categorize the data.

CONCLUSION

Multi-robot coordination and collective data sharing is a relatively young topic. Despite the advancements in swarm algorithms, the transition to competitive environments like agriculture and industry is still lagging, thereby lies an active field of opportunities to explore and improve. With the rapidly developing world and ever-increasing population, the demand for rapid production and generation of resources would grow. The main objective of this research is to promote automation with the sole motivation of speeding up the existing human processes within a limited space and time.

APPENDIX

Link to Github repository:
https://github.com/swarnabha13/CSCE643_Robotics_and_Spatial_Intelligence

REFERENCES

1. Haidegger, Tamás, Péter Galambos, and Imre J. Rudas. "Robotics 4.0–Are we there yet?." 2019 IEEE 23rd International Conference on Intelligent Engineering Systems (INES). IEEE, 2019.
2. K. Goldberg, M. Mascha, S. Gentner, N. Rothenberg, C. Sutter, and J. Wiegley, "Desktop teleoperation via the world wide web," in Proceedings of 1995 IEEE International Conference on Robotics and Automation, vol. 1, May 1995, pp. 654–659 vol.1
3. Waibel, Markus, Michael Beetz, Javier Civera, Raffaello d'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Häussermann et al. "Roboearth." IEEE Robotics & Automation Magazine 18, no. 2 (2011): 69-82.
4. J. Kuffner, "What's Next: Cloud-Enabled Humanoids?" in 10th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2010) Workshop, 2010.
5. Fierro, Rafael, Aveek Das, John Spletzer, Joel Esposito, Vijay Kumar, James P. Ostrowski, George Pappas et al. "A framework and architecture for multi-robot coordination." The International Journal of Robotics Research 21, no. 10-11 (2002): 977-995.
6. Penmetcha, Manoj, Shyam Sundar Kannan, and Byung-Cheol Min. "Smart Cloud: Scalable Cloud Robotic Architecture for Web-powered Multi-Robot Applications." arXiv preprint arXiv:1912.02927 (2019).

7. Mohanarajah, Gajamohan, Dominique Hunziker, Raffaello D'Andrea, and Markus Waibel. "Rapyuta: A cloud robotics platform." IEEE Transactions on Automation Science and Engineering 12, no. 2 (2014): 481-493.
8. Koubaa, Anis, Maram Alajlan, and Basit Qureshi. "ROSLink: bridging ROS with the internet-of-things for cloud robotics." Robot Operating System (ROS). Springer, Cham, 2017. 265-283.
9. Houxiang Zhang, Juan Gonzalez-Gomez, Zhizhu Me, Sheng Cheng and Jianwei Zhang, "Development of a low-cost flexible modular robot GZ-I," 2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 2008, pp. 223-228, doi: 10.1109/AIM.2008.4601663.
10. Simoens, Pieter, Mauro Dragone, and Alessandro Saffiotti. "The Internet of Robotic Things: A review of the concept, added value, and applications." International Journal of Advanced Robotic Systems 15, no. 1 (2018): 1729881418759424.
11. Geyer, Charles J. "Practical Markov chain monte carlo." Statistical science (1992): 473-483.
12. Zykov, Victor, Andrew Chan, and Hod Lipson. "Molecubes: An open-source modular robotics kit." IROS-2007 Self-Reconfigurable Robotics Workshop. 2007.
13. Yan, Zhi, Nicolas Jouandeau, and Arab Ali Cherif. "A survey and analysis of multi-robot coordination." International Journal of Advanced Robotic Systems 10.12 (2013): 399.
14. Parker, Lynne E. "Distributed Intelligence: Overview of the Field and its Application in Multi-Robot Systems." AAAI fall symposium: regarding the intelligence in distributed intelligent systems. 2007.