Communications design in multi-robotic systems

Manu Nair*, Sidney Givigi†
* Department of Electrical and Computer Engineering, Royal Military College of Canada, Kingston, ON, Canada
Email: manu.nair@rmc.ca
† School of Computing, Queen's University, Kingston, ON, Canada
Email: sidney.givigi@queensu.ca

***Abstract*—This paper describes the process for the design of communications in multi-robot systems using two approaches: centralized and decentralized architectures. Both approaches achieve joint collaboration through inter-robot coordination via exchange of sensor data and state information. As shown in a previous paper, depending on the tasks and the environment, the effects of explicit communications on centralized and decentralized control architectures for cooperative tasks are different. A comprehensive design process of the whole system for multi-robot task allocation will be discussed. Issues with the network design are introduced and possible solutions provided.**

I. Introduction

As discussed in [1], the communication among robots in multi-robot systems (MRS) needs to be taken into consideration prior to the design of the system as it may impact its performance. The reason is that MRS depend on how task allocations are performed, since robots interact with each other and coordinate tasks to complete a common goal [2].

In order to properly design MRS, many design factors have to be considered based on the requirements of the system. Some considerations include the robot team size, the operating environment and the composition of the robot teams, to name a few. Detailed considerations include what kind of task allocation and route planning algorithms to use, the control architecture of the system, and how the information exchange between system nodes is handled. Since MRS research is still in its infancy and a common design framework is not always easy to identify when designing the systems [3], approaches to MRS design were non-structured [4] [5], with most solutions being ad-hoc. In general, there was little standardization of how MRS problems were generally solved. One of the challenges with MRS design is how to optimally assign a set of robots to a set of tasks in order to optimize the overall system performance [6] [7] [8]. To address part of this issue, [4] proposed a taxonomy for classifying Multi-Robot Task Allocation (MRTA) problems.

Since then, other researchers have presented different ways of classifying MRS based on the application domains [9] and extending [4] taxonomy to include dependencies [7]. With available MRTA taxonomies, the task allocation component of MRS design is more streamlined with a possible set of solutions being made available. Given the abstract nature of these taxonomies, they do not fully capture all the design requirements for all MRS designs, and in particular, these taxonomies do not address real world, practical communications requirements that are often a bottleneck in MRS [10] [11]. In addition, there is a wide range of robotics applications where system performance is inherently tied to robot heterogeneity and environmental effects [12], yet it is also excluded from current taxonomies to allow them to be widely used.

This paper presents the communications design for two different approaches for a particular MRS problem. A centralized approach is derived and the issues of coordination due to the lack of bandwidth are discussed. Then, a decentralized approach is presented and coordination is further discussed.

This paper is divided as follows. Section II introduces the task for both the centralized and decentralized approaches. The simulation environment is discussed in section III. Section IV presents the design of the centralized architecture. Section V discusses the decentralized architecture approach. Section VI presents the results for some use cases. Section VII discusses network issues of the implementations, and the conclusions are presented in section VIII.

II. Task setup

Testing and validating any MRS remains a challenge. There are many variants of robotic systems that it becomes difficult to benchmark results and compare them with the research community. Most researchers find their own ad-hoc solution to testing research which becomes more difficult to benchmark against other works. In practice, most test and validation scenarios end up being ad-hoc to some degree, but researchers have attempted to standardize this. The basis for the test and validation testbed for this paper is based around the work by [13], that defined a realistic testbed to conduct multi-robot testing that included details on how the tests were configured and the metrics used to determine overall performance.

Their testbed is designed around ROS and MORSE simulator although it can run on similar simulators such as Gazebo. The authors do provide the full ROS packages to enable testing, though these packages have not been maintained with ROS upgrades over the years. That said, their testbed is a unified platform where only the experimenter's coordination strategies are tested, allowing for a benchmarking platform across the research community. A graphical representation of their testbed is shown in Figure 1.

The testbed focuses on three key areas: Robot (the type, physical models and computing capabilities), Fleet (number of robots, homogeneity) and Environment (the terrain to be explored) [13]. While the type of robots and environments indirectly affect the communications, it is the number of robots

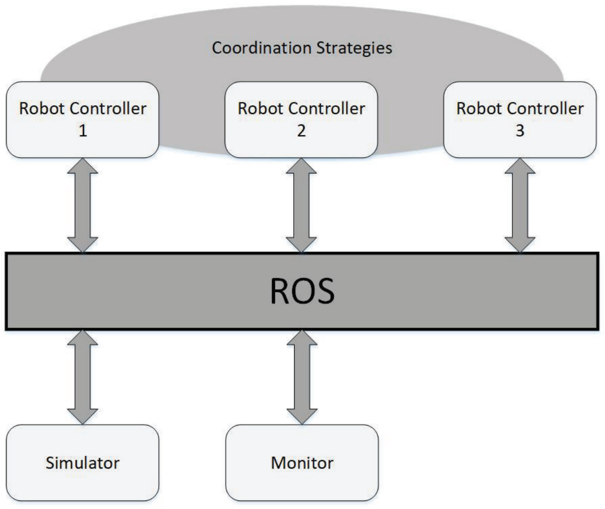


Fig. 1. MORSE testbed architecture

in the fleet that has the biggest impact on bandwidth and range [13].

However, there are challenges with using this testbed as defined. The packages and ROS distribution used to develop the testbeds are legacy and are not supported or tested with the current distributions of ROS. The Virtual Machine (VM) units used by the authors for each robot were based on Ubuntu 12.04.1 LTS and ROS Groovy. Therefore it is difficult to use this testbed as some newer packages are not backward compatible nor are the packages used in the testbed forward compatible. Without proper documentation, it is difficult for the community to maintain this particular testbed in future releases of ROS. Other limitations include requiring computer clusters or computers with lots of computing power.

Other considerations for not using this particular testbed as designed is the way communications modeling is handled. For bandwidth-heavy applications, [14] suggests adjusting the publish rates of various nodes in order to manage the network. While this will prevent simulation failure in large deployments, it may not necessarily be a true representation of the state of communications within the system.

Finally, the testbed also relies on a 1000 Mb/s (125 MB/s) Ethernet network, which may not be practical for common cases where only wireless networks are available. The authors' testbed also simulates the communication link based on calculating the distance between the robots in simulation and adjusting the communication costs as needed. This may not be a realistic representation of real world events and communication losses could be more complex.

III. Simulation Setup

For validation, it is imperative that communications components are tested using external wireless adapters to simulate real-world scenarios. While [13] can be used as a first approximation of a simulation setup, their specific software tools were not portable. Figure 2 shows the robot controller setup used with specific packages identified [1].

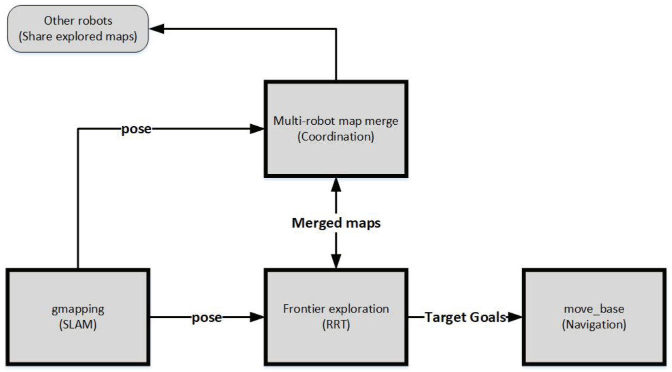

Other robots (Share explored maps)
pose
Multi-robot map merge (Coordination)
Merged maps
gmapping (SLAM)
pose
Frontier exploration (RRT)
Target Goals
move_base (Navigation)

Fig. 2. Software architecture

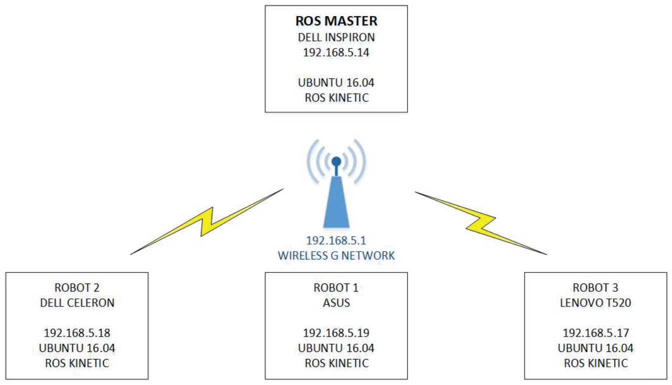


Fig. 3. Centralized Network Topology

One of the design goals for our testbed is the ability to switch from different control architectures, scalability and changing operating environments (worlds) as required. As with the approach used by [13], defining the metrics used and test automation is a necessity.

There are two different configurations for simulation. One for centralized, and the other for decentralized. In both setups, it is assumed that all the robots will operate in a common environment where communications coverage and mode remained constant for comparison.

IV. Centralized Architecture Design

Usually, centralized design is simpler than a decentralized one. However, when communication is considered, centralized approaches may very soon generate issues, especially with the availability of bandwidth.

The initial attempt at setting up a centralized system led to interesting technical problems which will be briefly discussed. In the original setup, there were three robots connected via a local, dedicated wireless network as shown in Figure 3. This is a common topology used in centralized network setups. However, this topology's performance will degrade as more robots are added.

A centralized approach will generally work for a small group of robots, but extending it to a larger set will be a strain on communications [15]. However, in this setup, even a small

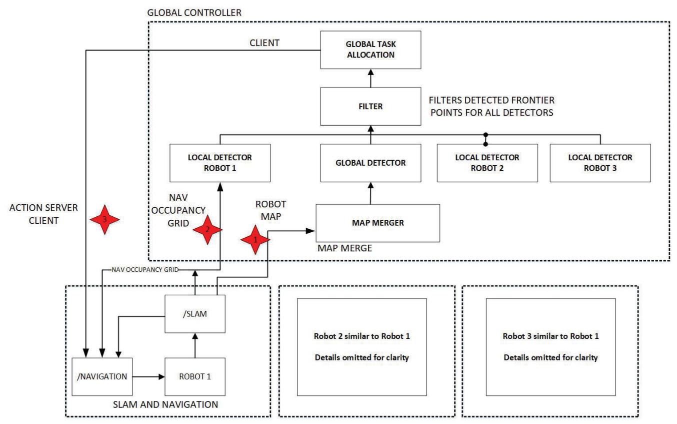


Fig. 4. Initial Centralized Node Layout

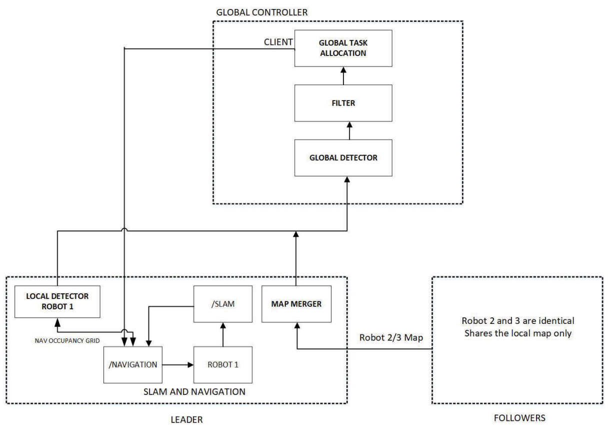


Fig. 5. Final Centralized Node Layout

group of three robots proved to be too network bandwidth heavy for the system to work using the software packages for exploration. Figure 4 shows the software road-map for the initial design.

During initial testing, all robots reported several errors which included warning messages from the process node responsible for navigation indicating that packet losses occurred (marked as red in Figure 4). Specifically, there are two issues: one is that some navigation goals are being dropped and the other issue is packets arriving at their respective robot computers with mismatched timestamps. The latter is caused by the built-in time function that applications use to time stamp each message sent in ROS. To elaborate, the message filter packages uses timestamps to ensure that only relevant data is sent and received. Specifically, the *TimeSynchronizer* and *TimeSequencer* filters ensure that messages are synchronized and temporally ordered. In this case, the messages sent by the ROS master to the robots had mismatched timestamps, making the robots think that either the data they received is in the past or it had to extrapolate into the future.

In centralized systems, the global planner has access to robot state information at varying levels (Limited, Partial or Full) [1] and makes decisions based on that. In this setup, the ROS packages required full information to make optimal decisions but at a high bandwidth cost. In this case, full information to the global planner caused the network to fail as the data required exceeded the channel capacity.

Aside from providing full state information to the global planner, the network will also require additional bandwidth to support the measurement tools that are being used. Measurement-induced errors and failures have been seen in other experiments [16].

Due to all these issues, another design, a hierarchical centralized architecture [1], was necessary. In this design, the leader (robot 1) concentrates all the information from the other robots. Therefore, te change to the hierarchical model minimizes network traffic as full state information is not needed any longer. The disadvantage is a reduction in performance. Nevertheless, this is the practical way to implement the communications in a centralized architecture. Figure 5 shows the software layout for the final design [1].

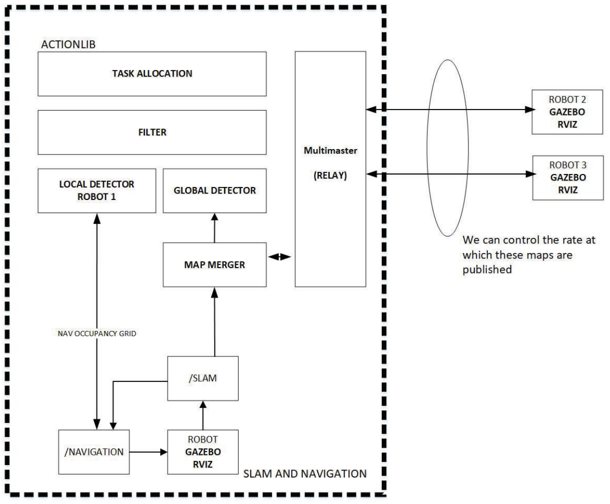


Fig. 6. Decentralized Node Layout

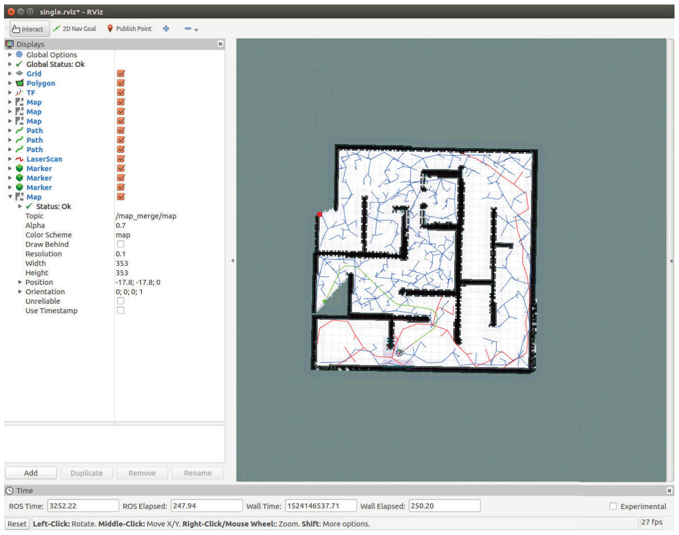
V. Decentralized Architecture Design

Counterintuitively, the decentralized design was easier to define than the centralized one, even though, it was more complicated to implement. The main reason for this design in the context of this paper is that by using cooperative algorithms, efficiency of communication is improved [17] by removing the need of information exchange with the centralized entity [15].

The decentralized architecture is shown in Fig. 6 [1].

In the figure, one can see that each node in the network has all the processes necessary for the robot's behaviour. The task allocation is done separately and the robots merge the maps only when data is received from other robots. Therefore, communications requirements are dropped substantially from the centralized architecture described in Section IV.

Both solutions look at solving a cooperative mapping problem. RVIZ (Robot Visualization) is a powerful tool that allows the user to visualize the data generated in the system. It is not part of the simulation package, but instead is another node that



Topic	/map_merge/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.1
Width	353
Height	353
Position	-17.8; -17.8; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐

ROS Time: 3252.22 ROS Elapsed: 247.94 Wall Time: 1524146537.71 Wall Elapsed: 250.20 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 27 fps

Fig. 7. View of Simulated World

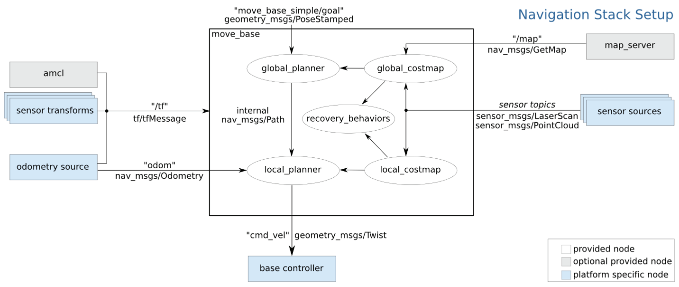


Fig. 8. Overview of the ROS Navigation Stack

subscribes to the data and displays it. RVIZ can also be used to publish data, to aid in troubleshooting or to provide system input. Figure 7 shows the environment on which the robots operate. It also shows how a robot map topic is visualized, by reading and displaying the sensor data.

For the task at hand, the ROS package parameters for *multirobot_map_merge* were modified to publish the merged map data at different frequencies [1].This is done to control de amount of data that is sent over the communication channels. The base controller is shown as an output from the navigation stack in Fig. 8. The costmaps reflect the area that has been uncovered by the robots: locally and globally.

Robots can navigate an environment using a preexisting map or create one as it explores an unknown environment. One method of exploring is by using frontier detection. Frontiers are defined as regions between known space and unknown space [18]. Figure 9 shows an example of the boundary between free and unknown space, with the frontier detected at the edge of the contrasting spaces. By assigning a robot to the frontier, the robot's sensors can map the unknown space and advance the frontier.

In order to navigate and map an unknown space, a frontier exploration (ROS) package is required. There are many ROS packages available for frontier detection using different

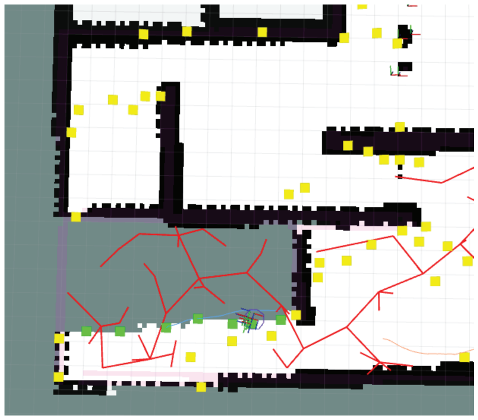


Fig. 9. Frontier Location

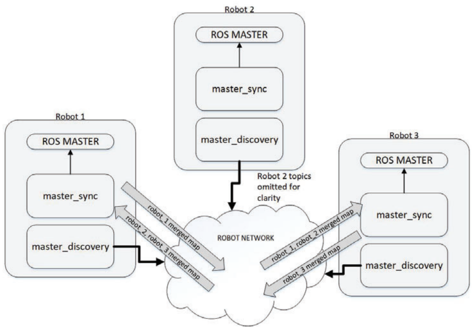


Fig. 10. Multi-master Topology

algorithms but the package selected for this research is based on a RRT algorithm [18], as it contained a built-in task allocator. This algorithm works by building a link (tree) from the location of the robot outwards to unexplored areas to find new frontiers. The tree then expands, contouring to detected obstacles, until it reaches a frontier edge and then assigns the robot a new target. The task is complete when the cells in the whole map are mapped and the environment is bounded.

The communications are done over a multi-master setup. The topics that are required to be shared need to be identified and the package handles the synchronization. In this configuration, the *multimaster_fkie* is setup using multicast protocol and is comprised of two nodes, a *master_discovery* and a *master_sync* node. The *master_discovery* is responsible for actively searching for other masters within the communications range and once identified, registers new host topics as defined. The *master_sync* node connects the discovered masters to allow for exchange of information. Figure 10 shows an overview of how the robots are connected and the topics that are shared using *multimaster_fkie*.

VI. Results

We will present the results for centralized merging rates of 0.01 Hz and 0.10 Hz as for higher frequencies, the exploration time does not change [1]. We will then present some network time synchronization issues that need to be considered in future designs

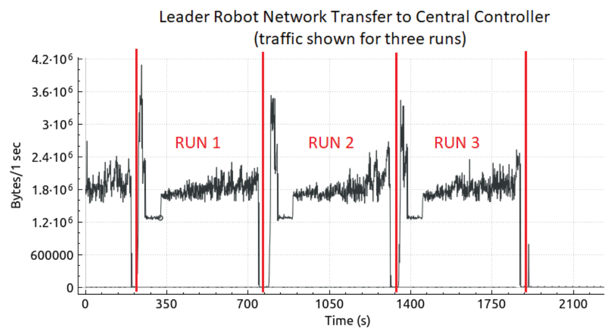


Fig. 11. Network Usage at 0.01 Hz by Leader robot

A. 0.01 Hz Merging Rate

For this test, the merging rate is set to 0.01 Hz. Using a similar setup as the baseline single robot test, 20 iterations of the experiments are run and the results are comparatively worse than the baseline single robot test. The total explore time is approximately 49 seconds longer than the single robot test.

Analysis of the data was conducted to determine the cause of the increase in explore time over the single robot test. At the start of the experiment, all three robots are assigned the global map area to explore, which is the normal start sequence. However, the global assigner cannot start the exploration until it receives the initial merged map from the leader robot. In this case, the initial map could take upwards of 100 seconds to be published. Once the map has been received, the global assigner sends target exploration goals to the robots.

The global assigner determines the frontier points from several sources, including the global costmaps and the merged map. Since we are employing a leader/follower topology, the merged map and the leader robots global costmap are only available for the global assigner to use. With such a low frequency to publish the merged map, the assigner has to determine target goals based on the global costmap of the leader robot. The rate at which the three robots move overlaps the rate at which the global assigner can send it valid points. This causes the robots to inefficiently explore regions based on partial information until a merged map is received and more valid exploration points are determined.

The global assigner receives the updated map every 100 seconds and can calculate new frontier points based on this. Even if the whole map area has been physically covered by one of the three robots and collected by the leader robot, the experiment only concludes when the central controller has the full map and the global assigner determines that there are no further frontiers to explore. With this test frequency, this can only occur every 100 seconds and timing of when the merged map is received could add time to exploration time.

In terms of network usage, the 0.01 Hz test consumed the lowest network bandwidth as it publishes the data less frequently. Figure 11 shows the network traffic of three experiments at 0.01 Hz. From this plot, we can see that the network traffic increases slightly over the duration of the single test as the contents of the merged map increases.

B. 0.10 Hz Map Merging Rate

In [1], it was reported that 0.10 Hz is an inflection point frequency for the centralized architecture. At this frequency, the global assigner receives the updated map every 10 seconds and could adequately assign the robots to explore the environment. Every 10 seconds, the central controller is able to send the robots relevant exploration targets because the overall merged map would not have changed much due to the robot velocity. After this point, there overall explore times remains at an average of 259 seconds and a standard deviation of $\sigma = 14.9$.

VII. Network Issues

During the experiments, the network throughput and speeds are monitored across the system. While conducting the decentralized tests, no noticeable issues were found with the network connection (over WiFi).

However, during the centralized system setup, there were excessive errors and warnings indicating timing issues, specifically missing messages and extrapolation into future errors. The source of the problems are two-fold: one is the need to synchronize the clocks between all the computers and the other is the data capacity of the network.

A. Network Time Synchronization

In ROS, most common messages will be timestamped. Timestamps are used to tag each message with the time it was taken, and is based on the UNIX epoch time (computer time). In systems that have various latencies between computers, data will arrive at different times and consequently results in cases where data is required to be extrapolated into the future or past. In most cases, this appears as a warning, but the data packet is dropped, leading to challenges coordinating the robots. The causes of these types of errors are either CPU processing capabilities or out of sync clocks. Since the computers used had low CPU usage at the time, this was ruled out as a cause.

A workaround used is to employ a time server such as Chrony. The ROS master is also set up as a time server with all other computers being clients. The software ensures that all clocks are synchronized with very low latency.

B. Network Capacity

A second problem had to do with the capacity of the WiFi network. In this case, the network is built around an IEEE 802.11g wireless router. For this particular router, the maximum speed is listed as 54 Mb/s (6.75 MB/s) at 2.4 GHz. This amount is the physical layer rate and is the theoretical maximum assuming an isolated, optimal distanced and interference free setup. In the experimental setup, the router is placed within one meter from all four computers and had a dedicated network assigned. The actual speeds are also a lot lower once the protocol overheads are considered.

Since ROS operates at the TCP layer, the actual speeds would be considerably lower as system overhead must be considered. Using the experimental setup, the maximum throughput is measured at approximately 8.5 MB/s. In a pure centralized setup, each computer (robot) will need approximately 3-4 MB/s depending on the configuration. It is clear that the current network setup will not be adequate for four computers (made up of three robots). WiFi networks will negotiate to the lowest adapter speed, so it is important to ensure all robots are using similar Ethernet adapters (which is the case here).

VIII. Conclusions and Future Work

This paper analyzed a use case for communications in MRS. A detailed description of the design process for a centralized and a decentralized architectures was provided.

A simulation environment that uses ROS was introduced. The steps of implementing both architectures in a multi-robot application were presented. Issues with the centralized architecture were discussed. Discussions of results with specific focus on the centralized architecture were provided. Some common network issues of the implementations were discussed with the hope that other researchers can avoid some issues.

In the future, the architectures could be tried with different communications technologies such as 5G to assess how newer approaches can benefit the design of robotic systems.

References

[1] M. Nair and S. Givigi, "The impact of communications considerations in multi-robot systems," in *2019 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pp. 1–6, March 2019.

[2] N. P. Lucas, A. K. Pandya, and R. D. Ellis, "Review of multi-robot taxonomy, trends, and applications for defense and space," *Unmanned Systems Technology XIV*, vol. 8387, pp. 1–10, 2012.

[3] A. Marino Advisor Ing Fabrizio Caccavale Co-Advisor Ing Gianluca Antonelli, "A Null-Space-based Behavioral Approach to Multi-Robot Patrolling,"

[4] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[5] B. P. Gerkey, "ON MULTI-ROBOT TASK ALLOCATION," *CRES Technical Report CRES*, pp. 3–12.

[6] A. M. Khamis, A. M. Elmogy, and F. O. Karray, "Complex task allocation in mobile surveillance systems," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 64, no. 1, pp. 33–55, 2011.

[7] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[8] B. Woosley and P. Dasgupta, "Multirobot Task Allocation with Real-Time Path Planning," *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference (FLAIRS-26 Conference), St. Pete Beach, FL, USA*, p. 574579.

[9] R. N. Darmanin and M. K. Bugeja, "A Review on Multi-Robot Systems Categorised by Application Domain *," *25th Mediterranean Conference on Control and Automation (MED)*, no. July, pp. 701–706, 2017.

[10] G. M. D. Araújo, a. R. Pinto, J. Kaiser, and L. B. Becker, "Cooperative Robots and Sensor Networks," *Studies in Computational Intelligence*, pp. 1–18.

[11] J. D. Labrado, B. A. Erol, J. Ortiz, P. Benavidez, M. Jamshidi, and B. Champion, "Proposed Testbed for the Modeling and Control of a System of Autonomous Vehicles*," *2016 11th System of Systems Engineering Conference (SoSE)*.

[12] L. Parker, "Task-oriented multi-robot learning in behavior-based systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, vol. 3, pp. 1478–1487, IEEE, 1996.

[13] Z. Yan, L. Fabresse, J. Laval, and N. Bouraqadi, "Building a ROS-Based Testbed for Realistic Multi-Robot Simulation: Taking the Exploration as an Example," *Robotics*, vol. 6, no. 3, p. 21, 2017.

[14] Z. Yan, N. Jouandeau, and A. A. Cherif, "A Survey and Analysis of Multi-Robot Coordination," *International Journal of Advanced Robotic Systems*, vol. 10, p. 399, dec 2013.

[15] L. Liu and D. A. Shell, "Optimal market-based multi-robot task allocation via strategic pricing," *Robotics: Science and Systems*, vol. 9, no. 1, pp. 33–40, 2013.

[16] R. Worst, H. Surmann, E. Zimmermann, T. Bagosi, T. Svoboda, M. Achtelik, and T. Consortium, "DR 6.1: Multiple asynchronous sorties to assess a large-scale static disaster area,"

[17] W. P. N. D. Reis and G. S. Bastos, "Multi-Robot Task Allocation Approach Using ROS," *Proceedings - 12th LARS Latin American Robotics Symposium and 3rd SBR Brazilian Robotics Symposium, LARS-SBR 2015 - Part of the Robotics Conferences 2015*, pp. 163–168, 2016.

[18] H. Umari and S. Mukhopadhyay, "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees," in *IEEE International Conference on Intelligent Robots and Systems*, 2017.