A Distributed Semantic Knowledge Framework for Collaborative Robotics

Soumyadeep Choudhury, Sounak Dey, Arijit Mukherjee
TCS Research & Innovation
Kolkata, India
Email: {soumyadeep.choudhury|sounak.d|mukherjee.arijit}@tcs.com

***Abstract*—Collaborative work between robots are likely to be a keystone in Industry 4.0 which talks about large scale automation in industrial scenarios. The robots in warehouses, retail stores, manufacturing floors may be guided by a central system, but in unknown scenarios such as disaster management, or areas where connectivity may be intermittent, the robots should be able to coordinate the task(s) at hand among themselves without having to rely on a central coordinating entity. Further, there may be a likelihood of situations where the higher round-trip latency of cloud-based systems may be undesirable. In this work, we propose a distributed knowledge framework deployed modularly across multiple heterogeneous robots that enables them to perform tasks by real time knowledge sharing and collaboration. We also have performed extensive experiments to compare performance of such a distributed framework with a centralised one and reported the results here in detail.**

***Index Terms*—Collaborative Multi-robot System, Distributed Knowledge framework, Decentralized control, Automation.**

I. Introduction

Industry 4.0 [1] standard proposes autonomy of robotic agents that are capable of coordinating among themselves, do inferencing, and take decisions on their own with minimal human intervention. This kind of capability would be useful for a team of heterogeneous set of robots, drones, AGVs, etc. in scenarios like *disaster management, search & rescue* [2] etc. However this requires issues of interoperability (arising from heterogeneity in the team) to be resolved and capability of self reasoning to be imbibed in order to perform automated reasoning & decision making. Cloud based knowledge frameworks like KnowRob, Rapyuta, RoboEarth, etc. [3] [4] [5] [6], designed using semantic web technologies [7], are some works which cater to these issues [8]. Robots can submit queries to such cloud knowledge bases before starting or during the execution of their tasks. In these systems, all collaborative information exchange happens via cloud; robots do not communicate with each other directly.

Limitations. These cloud based systems, however, introduce a round trip latency that may grow in some real life scenarios where network connectivity is lossy and the bandwidth is low, resulting in non-real time performance of the multi-robot system. This limits the use of such cloud-based frameworks in outdoor emergency situations where connectivity cannot be guaranteed and tasks must be completed within reasonable time without manual intervention.

One answer to above problem can be a distributed framework, *without any cloud component*, where one robot in the team can carry the whole knowledge-base with it so that it is always within reach of the other teammates. But such system would be unreliable owing to single point of failure of that particular robot. Again, owing to the scarcity of computational resources of a robot, loading its memory with large knowledge bases can lead to serious performance issues.

Contributions. In this paper, we demonstrate a basic prototype of a distributed heterogeneous multi-robot system (refer Fig. 1), where knowledge base is distributed among robots in a modular fashion and the robots can handle a stream of incoming queries on their own. We have:

- shown how a distributed knowledge base amongst multiple robots can enable them to complete a task set on their own, and
- extensively studied the performance of this distributed system compared to its centralised counterpart in different work situations by varying four parameters: (i) network bandwidth, (ii) network loss for each such bandwidth, (iii) no. of robots, and (iv) no. of tasks. The results seem to favour distributed system in lossy and low bandwidth situation.

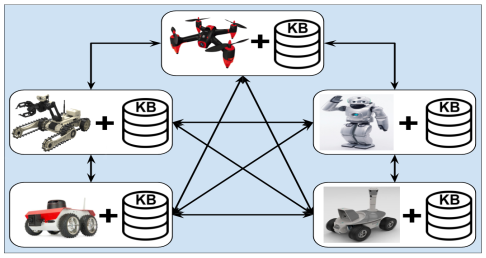


Fig. 1: Distributed knowledge based multi-robot system

In this work, typical robotic issues like mapping, path planning, collision detection, etc. were not in focus; and that is why, for experimentation, a simplified model of warehouse is considered instead of an unknown outdoor environment, where mapping and path planning would be more complex.

The paper is organised as follows: Section II below describes the prior arts relevant to our work while Section III explains the functional components of the system, description of knowledge bases, system workflow, etc. Section IV provides the details of experimental design and set up while Section V reports results of the experiment and their significance. We conclude in Section VI along with the future works.

II. Relevant works

Systems like KnowRob [3], RoboEarth [5], etc. are cloud based robotic platforms which, as mentioned earlier, are dependent on cloud and may fail in lossy network situations. Smirnov et. al. [9] proposed another ontology based collaborative robotic system but that too is dependent on a central server to which robots always communicate during task execution. Another heterogeneous multi-robot system is proposed in [10] but it uses one robot as controller thus this system is prone to single point failure. AMiRALE [11] is a distributed robotic system but it does not involve semantic knowledge bases to autonomously execute tasks. SOIFRA [12] is an interesting distributed robotic system which targets to generalize platform-independent algorithms for unmanned aerial and ground robots but does not involve semantic knowledge bases to complete a set of tasks. [13] discusses coordinated multi-robot exploration but does not talk about knowledge sharing or autonomy. Most of the robots today are ROS-based [14] that internally uses TCP as communication protocol but TCP has certain operational overheads. Dey et. al. [15] have done a performance comparison between TCP and CoAP [16] for semantic data exchange between the robots. This is relevant to our work.

III. System Description, Implementation and workflow

In this section, we will discuss the functional modules within each robot, describe knowledge bases and finally will explain how the distributed framework works using one example task. Note, the terms ‘robot’ and ‘robotic agent’ are used interchangeably in this paper.

A. Functional Components of a Single Robotic Agent

In this distributed multi-robot system, each robotic agent has seven main modules (refer Fig. 2) as described below:

- **Knowledge Base (KB):** Each robot contains a knowledge base (in form of an ontology aka .owl file and related RDF triple store [17]), which is capable of answering queries pertaining to a particular domain. For example, in our case, one robot has Object.owl which contains feature-details such as weight, size, etc. of objects; another robot contains a Capability.owl which stores capability of all the robots in the team. Details of these KBs are discussed in subsection III-B. The terms *Knowledge base* and *KB* would be used interchangeably in this paper.
- **Configuration:** This module initializes the robotic agent by reading parameters (for eg. network ip, file paths, initial location of robot in map, etc.) from a configuration file. During run-time, these parameters are not changeable. It also keeps a *robot vs. KB* list that tells which robot has a particular KB. Once initialisation is done, it hands over control to the *Controller* module.
- **Controller:** This module syncs data and ensures execution of correct workflow between rest of the modules. Moreover, the *Controller* receives new tasks & identifies the unknown parameters of the task. It then informs the *Query Handler* module to raise required queries. Answers of these queries are send to the *Task Assigner* module and based on its decision, control is handed over to the *Task Execution* module.
- **Query Handler:** Based on the *Controller* feed, it raises a set of queries to find out task details, capabilities (required to do that task), etc. Depending on the availability of KBs, these queries are submitted either to self-KB of a robot or to KBs of other robot companions.
- **Task Assigner:** Once the queries about a task are resolved and required capabilities, present location of other robot are available, then this module does a cost calculation for each robot-task pair. It identifies the robot with minimum cost, in terms of battery consumption, so that the task can be assigned to that robot. This calculation is done in each robot so each robot knows to whom the task is assigned.
- **Task executor:** This module finally executes the task based on assignments. This module also plans the path for the robot, handles any real time issues during navigation. For path planning optimised Monte Carlo localisation technique [18] is used.
- **Communication:** This module enables a robot to communicate with its team-mates and external world via client/server based mechanism. It helps exchanging queries-answers, status of a robot, etc. within the team.

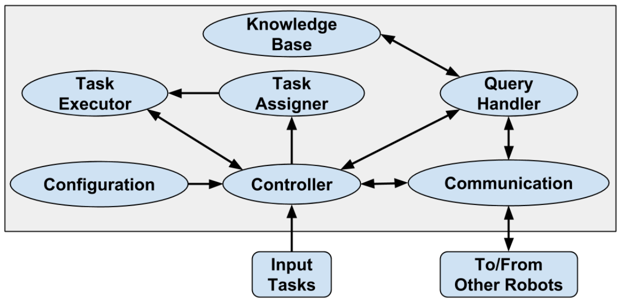


Fig. 2: Functional components in a single robot.

B. Description of Knowledge bases

As already mentioned above, we have considered a case of warehouse (for implementation and experimentation) where some objects are stored in racks and robots are supposed to pick them up by navigating via aisles. For this purpose, three KBs namely *Object.owl, Location.owl* and *Capability.owl* are created that are warehouse centric. *Object.owl* stores many features of the objects like color, size, weight, fragility, how it can be picked up (e.g. by grasping a handle or gripping its body), price, etc. The second KB namely, *Location.owl* stores information about the warehouse map i.e. number and location of aisles & racks, width and length of navigable paths, obstacle positions, location of objects stored in the warehouse, etc. It also contains real time locations of the robots that are roaming around in the warehouse. Finally, *Capability.owl* stores component and related capabilities (such as grasping, suction, having arms, wheels, total payload capacity, etc.) of each and every robot in the team. Fig. 3 below shows a subset of these concepts and their inter-relations as captured in owls.

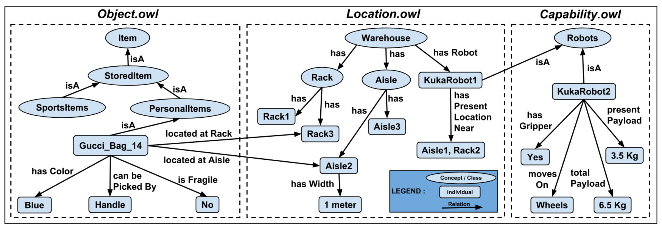


Fig. 3: Concepts & relations in modular knowledge bases.

C. Implementation

ROS is the standard robotic operation platform that internally uses TCP for robotic communications. TCP as a stack is heavy to be loaded and operated in robots with less computational resources. Moreover, it's overhead, owing to reliability, is a factor for short payload transfers between the robots. With this, not all robots or drones follow ROS standard. Thus, a message queue based lightweight open source library named ZeroMQ [19], that uses 'ZeroMQ Message Transfer Protocol' (ZMTP) internally, is used for internal communication between robots. Since pub/sub mode is asynchronous, it introduces unwanted delay and in some cases it incurs high data loss; thus client/server mode, supported by ZeroMQ, has been adopted for inter-robot communication. Each robot runs both server and client instances at the same time; the robot can answer queries via server instance while it can make queries via client instance. The knowledge bases are designed in OWL/RDF format using a well known tool called Protege[1]. A python based lightweight library name RDFLib [20] is used for making queries to the knowledge bases.

D. Workflow

To understand how the collaborative system works, we take an example task *"Pick up a Gucci_bag_14"* that needs to be collaboratively carried out by three robots namely Robot1, Robot2 and Robot3; also consider that three KBs namely Location.owl, Object.owl and Capability.owl are kept in those three robots respectively. A step-wise explanation of the workflow (refer Fig. 4) to execute this task is given below:

Step 1. As the task comes to the *Controllers* of each robot, they first need to know whether *Gucci_bag_14* exists in the Object.owl. The *Controller* sends this instruction to *Query handler* and it executes *Query_1* as shown below. As Robot2 has Object.owl, so it executes a self-query where other two robots send their queries to Robot2:

```
PREFIX ns_2: <http://www.tcs.org/2018/Object#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?object
WHERE
?object rdf:type ns_2:PersonalItems .
?object ns_2:Product_name 'Gucci_bag_14'.
```

[1]https://protege.stanford.edu/

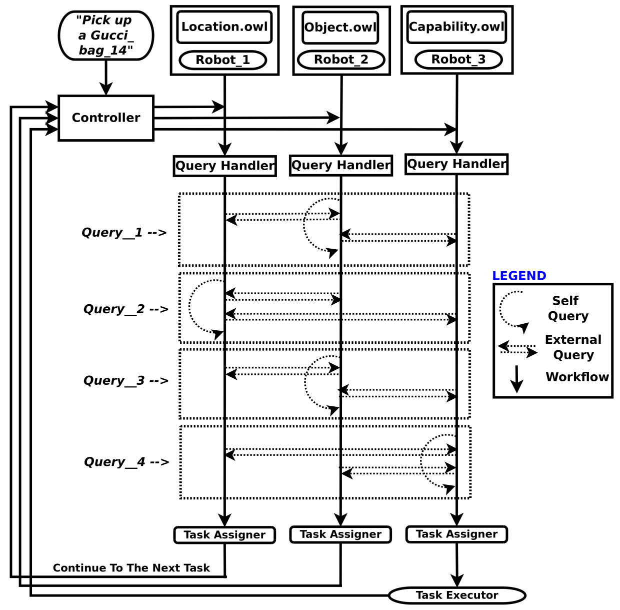


Fig. 4: A sample workflow to demonstrate collaborative task execution by three robots.

Step 2. If the output of above *Query_1* is negative, then the task is skipped (as there is nothing to do) by all three robots and the next task in the pipeline is processed; otherwise the robots need to know the location details of the object (i.e. at which rack & aisle of the warehouse the bag is located ?). Thus all the robots executes *Query_2* below:

```
PREFIX ns_1: <http://www.tcs.org/2018/Location#>
SELECT ?loc ?warehouse ?aisle_no ?bin_no ?rack_no
WHERE
ns_1:Gucci_bag_14 ns_1:Available_At ?loc.
?loc ns_1:Location_name ?warehouse.
ns_1:'Gucci_bag_14' ns_1:Inventory_aisle_no ?aisle_no.
ns_1:'Gucci_bag_14' ns_1:Inventory_bin_no ?bin_no.
ns_1:'Gucci_bag_14' ns_1:Inventory_rack_no ?rack_no.
```

Here, Robot1 having Location.owl will generate a self-query where rest two will submit this query over network to Robot1.

Step 3. Once the coordinates of the *Gucci_bag_14* is found then the robots need to know the details about the features of that object such as how it can be picked up, what is its shape, whether it is fragile or not, etc. These information can be used for cost calculation later. Thus all the robots execute *Query_3* below:

```
PREFIX ns_2: <http://www.tcs.org/2018/Object#>
SELECT ?frag ?price ?color ?shape ?dim ?grip
WHERE
ns_2:'Gucci_bag_14' ns_2:Item_fragility ?frag.
ns_2:'Gucci_bag_14' ns_2:has_shape ?shape.
?shape ns_2:Item_dimension ?dim.
ns_2:'Gucci_bag_14' ns_2:can_be_picked_by ?grip.
```

As the Object.owl contains all these information, Robot2 will execute a self-query while Robot1 and Robot3 will submit the query to Robot1 over network via their *Communication* modules. The value of query variable `grip` can be `gripping`, `suction` etc. This value will be passed to next step. Say, the value is `gripping`.

Step 4. Next, the robots need to know which robot has the required capability of `gripping` (and has hardware component `gripper` as well) to pick up the bag with its handle. If more than one robot have same capability, then their present carrying capacity would be queried (as they may already be carrying some load as part of some older task). *Query_4* below finds out all these capability related information from Robot3, which contains capabilities knowledge base:

```
PREFIX ns_3: <http://www.tcs.org/2018/Capability#>
SELECT ?robots ?robo_name ?payload
WHERE
?robots ns_3:Can_Pick ns_3:gripping.
?robots ns_3:Robot_Id ?robo_id.
?robots ns_3:Robot_payload ?payload.
```

Say, *Query_4* returns `?robots`=Robot3. This means Robot3 has a gripper and it can pick the bag. With all these information in hand, the *Controller* of each robot invokes the *Task Assigner* module to initiate cost calculation which computes cost of doing the aforesaid picking task by each of the robots. As a result, Robot1 computes the cost for Robot2, Robot3 and itself. Same computation is performed by other two robots so that all three robots know which of them has minimum cost against this task. In this case, it will be Robot3; so, *Task Assigner* of Robot3 will kick-off its *Task executor* module so that task execution can start, while Robot1 and Robot3 will jump to next task in the queue because they know they are not going to do this task.

Decision of task assignment could be made in another way: let one robot calculate the cost and then communicate the cost value to others (*compute and communicate*). In our case, we chose *compute by all* strategy. Apparently these two strategies have same efficiency but if network bandwidth is weak, which is our focus area, then *compute and communicate* strategy may take more time, resulting into delay in kicking off task execution.

IV. Experimental Setup

Experiment design. As we want to compare the performance of the distributed knowledge system against a centralised one with respect to robotic tasks in poor network condition, we have designed the experiment in following fashion:

- Three groups of robots (R_1, R_2, R_3) are taken comprising of 3, 6 and 9 robots respectively.
- Five task-groups (T_1, T_2, T_3, T_4, T_5) have been created comprising of 5, 10, 20, 40 and 80 tasks of picking objects respectively.
- Each group of robots has performed the tasks in all task groups i.e. each R_i, $i \in \{1, 2, 3\}$ has performed T_j, $j \in \{1, 2, 3, 4, 5\}$.
- Each $\{R_i,\ T_j\}$ pair has been tested for four different network bandwidths B_k, $k \in \{1, 2, 3, 4\}$ having values 35 kbps, 128 kbps, 256 kbps, 512 kbps respectively.
- Each of $\{R_i,\ T_j,\ B_k\}$ has been tested under different network loss condition L_p, $p \in \{1, 2, 3, 4, 5\}$ having values 2%, 5%, 10%, 15%, 20% respectively.
- All these tests, as stated above, have been executed with all the varying parameters again with the whole knowledge base being stored in one remote cloud.
- A comparison between distributed and centralised version is recorded in terms of time to complete the task and total bytes exchanged over the network.

We have taken our network bandwidths to be in the range of cellular network bandwidths [21] like GPRS(35KBps), EDGE (135 KBps), etc. to ensure poor network conditions. High end cellular bandwidths like HSPA (1 Mbps) or WiFi are not considered because cloud connectivity would not be a problem in those cases. We also have assumed that:

- Tasks are sequentially triggered into the robotic team from an external source.
- Each robot knows what knowledge base is present with other team mates (thanks to the configuration file).
- For simplicity, task of *picking* is considered.

Setup. In typical robots, the computational resources like memory, processing power etc. are usually scarce. In our lab, we initially started executing our experiment using 3 custom-made R-Pi based robots which has 1 GB RAM each. We first checked the practical feasibility of the system using those custom-made R-Pi robots (as shown in Fig. 5) in a 3-robots-5-tasks scenario and it worked as per expectation.

But for practical complexity of experimenting with more than three robots, we had to simulate the robots using virtual machines (VM). Each VM has only 900MB RAM, keeping in mind the resource scarcity in robots. These VMs, representing the actual robots, communicate with each other via an internal network. NetEm [22] utility has been used for configuring different bandwidth and loss condition of this network. Wireshark [23] is used to capture network traffic and latency. Each VM has a running instance of robotic agent as per Fig. 2 and they communicate with each other via client/server mode.

Fig. 5: Experiment with three R-Pi based robots.

V. Experimental Results & Discussion

In this section we present the experimental results, their significance and related explanations. Fig. 6 shows a comparison between centralised and distributed system in terms of total no. of exchanged bytes at 35 Kbps bandwidth and for 5 tasks. The network loss (in %) has been varied and the no. of exchanged bytes over network is recorded for 3, 6 and 9 robots. The graph shows that the network traffic (i.e. bytes exchanged) for distributed system is nearly 42% less than that of centralised system for 2% network loss, and it does not change much even if the loss increases to 20%. Thus, the distributed system clearly wins. Similar behaviour is observed for other bandwidths but those results are not reported here in order to reduce redundancy.

Further, looking closely at Fig. 6, it is observed that for 35 Kbps scenario if the no. of robots increases (from 3 to 6 to 9) then the network traffic decreases approximately by 1% in each case. As shown in Fig. 7, this behaviour is same for all the other network bandwidths, i.e if the team size increases then network traffic decreases, keeping the no. of tasks and network loss unchanged. Fig. 7 however shows the plots for distributed case only. Similar plots will be there for centralised system too.

This result seems surprising but the reason is: in our experimental setup, we have not pushed N tasks sequentially to the group of robots; instead we have divided N tasks into two equal sets (in case of 6 robots) or into three equal sets (in case of 9 robots) and pushed each such task-set to a cluster of 3 robots for parallel processing. Thus in 9-robots case, N tasks are processed in parallel by 3 clusters of robots so that each cluster processes N/3 tasks effectively, while in 6-robots case, each cluster processes N/2 tasks and in 3-robots case, one single cluster processes all N tasks. As no. of bytes exchanged is directly proportional to the no. of queries made by robots, which is again proportional to the no. of task that a robot handles, this phenomena results into slight increase in network traffic as the no. of robots decreases.

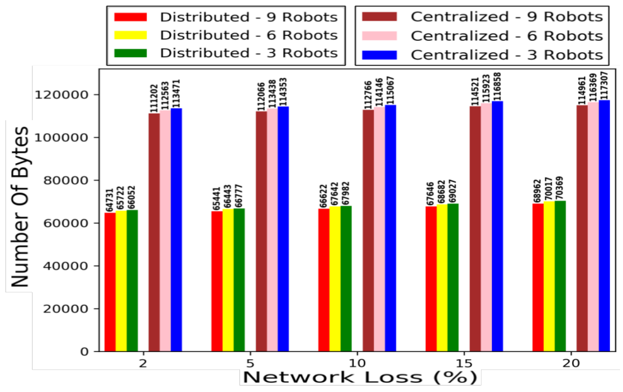


Fig. 6: Comparison of centralised vs. distributed system in terns of *no. of bytes exchanged* over network for different network loss % for 35 Kbps and 5 tasks.

Fig. 6 above does not capture effect of increasing *no. of tasks* on network traffic. Fig. 8 reports that. It shows a comparison between distributed and centralised system with respect to *total no. of bytes exchanged* over network for different sets of task and different network loss conditions keeping bandwidth fixed at 35 kbps and *no. of robots* being 3 only. Here we see that: as the *no. of tasks* increases exponentially (i.e. 5, 10, 20, 40, 80) so does the no. of related queries to the knowledge bases of each robots and this results into more network traffic. But we see that distributed system performs better for each task-set. Same comparison were done for other three bandwidths and similar trend is observed (related graphs are not shown here in order to save space).

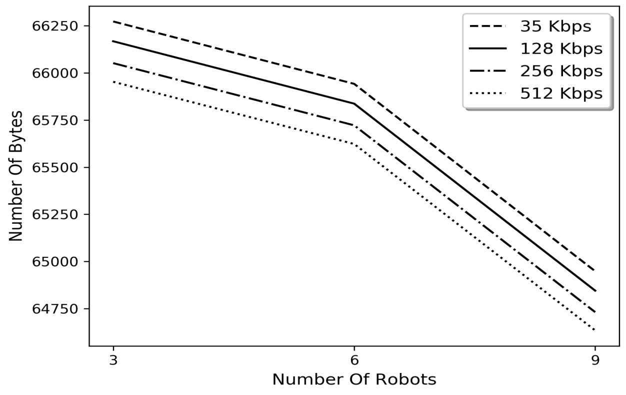


Fig. 7: Trend of network traffic (in *No. of bytes exchanged*) for different sets of robots for 5 tasks over network bandwidth of 35 Kbps.

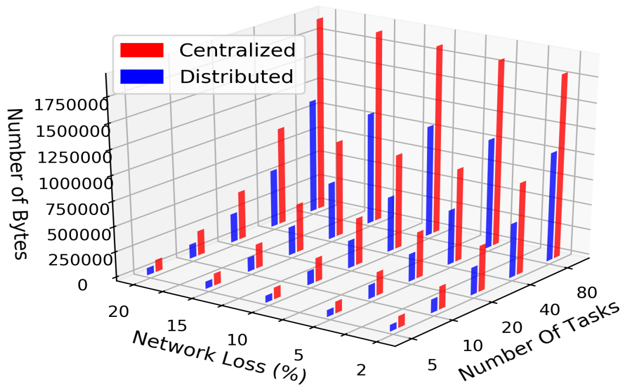


Fig. 8: A comparison of *network traffic (in terms of no. of bytes exchanged)* to complete the different sets of task by 3 robots under different network loss conditions at 35 Kbps bandwidth.

Though Fig. 8 shows centralised system performs worse than distributed one, but the trend of worsening is not very clear. Here in Fig. 9 we report that by showing the trend of *difference of no. of bytes exchanged* over network between centralised and distributed system by varying the no. of tasks. It shows that for a given network condition (35 Kbps bandwidth and 2% loss), bytes exchanged for the centralised system grows exponentially for increasing no. of tasks i.e. as more tasks are coming in, centralised system consumes more network bytes compared to distributed system. The reason is: if each task generates Q queries and of them say 2 queries (on average) are resolved internally (in case of distributed system) then for N tasks, there are N*(Q-2) queries are going in network. But for centralised system, N*Q queries are going in network. This difference (2N) is increasing as N increases resulting into an increase in difference in network traffic. On the other hand, for a given set of tasks, there is not much change in network traffic even if we increase no. of robots.

Task completion time is a very important metric to measure performance of a robotic system. Fig. 10a shows comparison between distributed and centralised system with respect to *task*

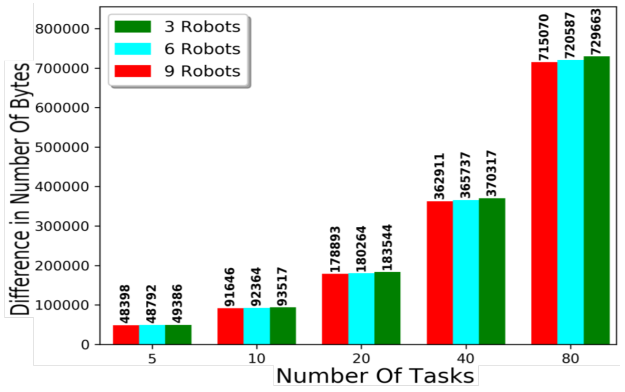
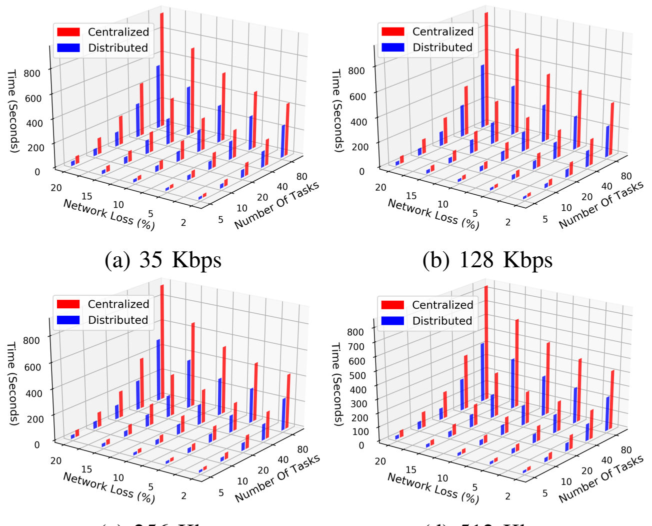


Fig. 9: Trend of difference in *no. of bytes exchanged* between centralised and distributed system for varying set of tasks under constant network condition.

completion time for different sets of task and different network loss conditions, for 35 kbps bandwidth and for 3 robots. Fig. 10b, Fig. 10c and Fig. 10d show similar comparison for other network bandwidths. Here also we see that distributed system performs better than centralised system in each situation. Moreover, as the network loss increases, *task completion time* also increases for a given task set. As the network goes from low to high bandwidth, we see task completion time is decreasing as expected.



(a) 35 Kbps (b) 128 Kbps

(c) 256 Kbps (d) 512 Kbps

Fig. 10: A comparison of *Task completion time* between distributed and centralised system for different sets of task by 3 robots under different network conditions

It is not very clear from Fig. 10 that how the *task completion time* varies between centralised and distributed system for different sets of task. Hence, Fig. 11 below shows that the behaviour over different task sets done by different sets of robots under 35 Kbps bandwidth and 2% network loss. It shows that for all sets of robots, the difference is not huge for smaller number of tasks; but as the no. of tasks increases the difference grows. This means: for larger no. of tasks performance of centralised system worsens. Fig. 12a shows

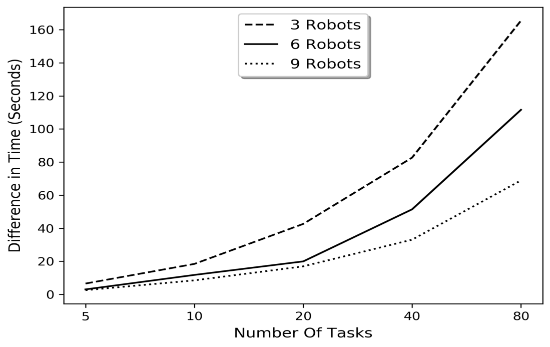
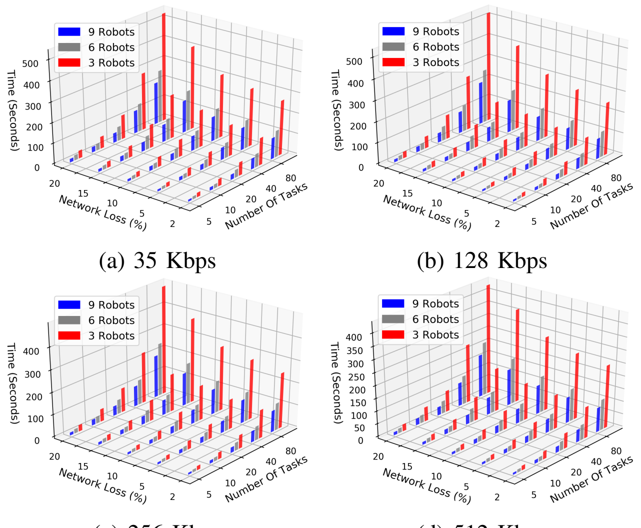


Fig. 11: Trend of difference in *task completion time* between centralised and distributed system for varying sets of tasks under constant network condition.

below a comparison of *task completion time* between different sets of robots (i.e. 3, 6, 9) for different number of tasks under different network loss conditions in distributed sytem for 35 Kbps network bandwidth. Fig. 12b, Fig. 12c, Fig. 12d shows the same comparison for the other three bandwidths. We observe that for all four bandwidth cases, as the no. of tasks increases so does the task completion time, which is expected. But if no. of robots is increased for a given set of task then task completion time reduces. Reason for this is already explained in second paragraph of this section. Moreover, for a given no. of tasks, if the network loss increase then *task completion time* will also increase slowly. This is again as per expectation for obvious reasons.



(a) 35 Kbps (b) 128 Kbps

(c) 256 Kbps (d) 512 Kbps

Fig. 12: A comparison of *task completion time* in distributed system only by different sets of robots under different network loss at different network bandwidths.

VI. Conclusion and Future Work

In this paper, we have introduced a distributed semantic knowledge based multi-robot system and compared its performance issues with a centralised system in different situations. As a next step, we have started testing the robustness of this system (in case of failure of one or multiple robots in the team) and are working on a related backup strategy. To improve on scalability and security issues, using Information Centric Network (ICN) [24] is one of the future plan.

References

[1] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in *System Sciences (HICSS), 2016 49th Hawaii International Conference on*. IEEE, 2016, pp. 3928–3937.

[2] "Imavs 2016 competition," http://www.imavs.org/2016/competition.html, [Online; accessed 22-Nov-2018].

[3] M. Tenorth and M. Beetz, "Knowrobknowledge processing for autonomous personal robots," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 4261–4266.

[4] M. Stenmark and J. Malec, "Knowledge-based instruction of manipulation tasks for industrial robotics," *Robotics and Computer-Integrated Manufacturing*, vol. 33, pp. 56–67, 2015.

[5] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The roboearth cloud engine." in *ICRA*. Citeseer, 2013, pp. 438–444.

[6] G. Mohanarajah, D. Hunziker, R. D'Andrea, and M. Waibel, "Rapyuta: A cloud robotics platform," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, 2015.

[7] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 34–43, 2001.

[8] G. s. Stephan, H. s. Pascal, and A. s. Andreas, *Knowledge representation and ontologies*. Springer, 2007.

[9] A. Smirnov, A. Kashevnik, S. Mikhailov, M. Mironov, and M. Petrov, "Ontology-based collaboration in multi-robot system: Approach and case study," in *System of Systems Engineering Conference (SoSE), 2016 11th*. IEEE, 2016, pp. 1–6.

[10] T. R. Wanasinghe, G. K. Mann, and R. G. Gosine, "Distributed collaborative localization for a heterogeneous multi-robot system," in *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on*. IEEE, 2014, pp. 1–6.

[11] V. Autefage, S. Chaumette, and D. Magoni, "Distributed collaborative system for heterogeneous swarms of autonomous mobile robots," in *Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS), 2015 International Conference on*. IEEE, 2015, pp. 1–7.

[12] W. A. Arokiasami, P. Vadakkepat, K. C. Tan, and D. Srinivasan, "Interoperable multi-agent framework for unmanned aerial/ground vehicles: towards robot autonomy," *Complex & Intelligent Systems*, vol. 2, no. 1, pp. 45–59, 2016.

[13] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[14] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[15] S. Dey, A. Bhattacharyya, and A. Mukherjee, "Semantic data exchange between collaborative robots in fog environment: Can coap be a choice?" in *2017 Global Internet of Things Summit (GIoTS)*, June 2017, pp. 1–6.

[16] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," *RFC 7252*, 2014.

[17] S. Bechhofer, "Owl: Web ontology language," in *Encyclopedia of database systems*. Springer, 2009, pp. 2008–2009.

[18] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2. IEEE, 1999, pp. 1322–1328.

[19] P. Hintjens, *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[20] R. Team, "Python library rdflib," 2013.

[21] "What are the actual speeds of gprs, edge, umts, hspa, etc," http://www.speedguide.net/faq/what-are-the-actual-speeds-of-gprs-edge-umts-hspa-366, 2014, [Online; accessed 22-Nov-2018].

[22] "Netem," http://man7.org/linux/man-pages/man8/tc-netem.8.html, [Online; accessed 22-Nov-2018].

[23] "wireshark," https://www.wireshark.org/, [Online; accessed 22-Nov-2018].

[24] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and B. Mathieu, "A survey of naming and routing in information-centric networks," *IEEE Communications Magazine*, vol. 50, no. 12, 2012.