Rapid prototyping of Industry 4.0 use cases for warehouse automation using a Network-Robot Co-Simulation Framework*

Srikrishna Acharya[#1], Parv Maheshwari[#2], Mukunda Bharatheesha[3], Bharadwaj Amrutur[4] and Yogesh Simmhan[5]

***Abstract*— With an efficient network of taskable robot agents, Multi-Robot Systems (MRS) have brought value addition in diverse domains like logistics, cargo management, and agriculture. In this work, we propose the use of a modular network-robot co-simulation framework CORNET 2.0, which is agnostic to the application middleware and hence can support diverse automation solutions in simulation. We showcase the benefits of such a co-simulation framework to evaluate a network-aware proactive collision avoidance approach in a multi-robot navigation scenario in which a robot's footprint is relayed over a network to all the other robots in the ego robot's network visibility. This allows all robots to proactively avoid each other, resulting in a higher average task execution efficiency and minimizing collisions under favourable network conditions.**

I. Introduction

The fourth industrial revolution, the Industry 4.0 [1] initiative, is poised to happen globally, connecting modern control systems via the Internet of things (IoT). The convergence of network technology and operational technology enables new ways of production, value creation, automation, and real-time optimization. On top of that, the COVID-19 pandemic has accelerated the growth of the e-commerce industry unprecedentedly, creating a massive logistical challenge for warehouse and supply chain management. The growing demand for higher warehouse throughput and a smaller workforce has expedited the requirement of robot operations. The projected growth in the number of robotic warehousing units in 2021 is $15\times$ that reported in 2016 [2].

As illustrated in Fig. 1, a typical warehouse facility using multiple robots requires a reliable and robust system for communication and cooperation. Development and real-world deployment of *Multi-Robot Systems (MRS)* poses several challenging problems, including but not limited to task allocation [3], task execution [4], path planning [5], coordination [6], localization [7], and swarming [8]. In addition, some of the application-specific solutions and algorithms for MRS, such as patrolling algorithms [9], map-merging [10], and flocking [11], have attracted much interest from research communities. In all these problems, coordination using communication channels forms an essential backbone. This requirement is the primary motivation for the contributions in this paper.

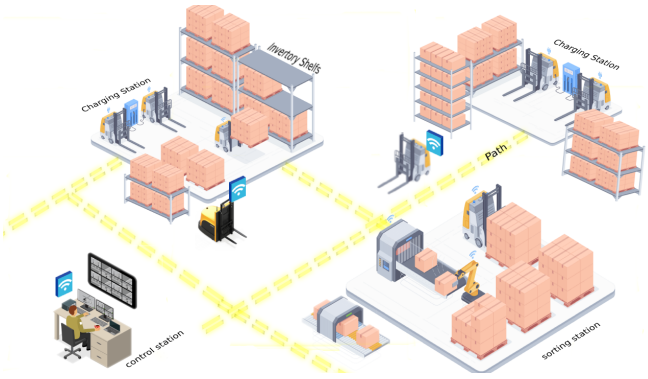


Fig. 1: Networked Warehouse Automation

When benchmarking MRS, realistic simulations have many advantages over experimental trials with actual robots. The simulations are reproducible and faster to deploy, enabling a broader range of evaluations using different settings, e.g., fleet sizes, environments, and robot types. However, the simulation must take into account both the physical environment and the network infrastructure as both affect planning operations. In this paper, we showcase a network and robot co-simulation framework (CORNET 2.0). The co-simulation framework enables planning and restructuring of the networking and the automation infrastructure iteratively in simulation. Once the desired automation is realized and thoroughly tested in simulation, a real-world deployment of the best-performing configurations can be considered, which will be highly cost-effective. To demonstrate this framework, we evaluate network enabled collision avoidance approach for MRS operations for a warehouse use-case.

The main contributions of this paper are as follows:

1) We demonstrate the evaluation of a multi-robot task execution approach with proactive collision avoidance.
2) We showcase the use of CORNET 2.0, a multi-agent network-robot co-simulation framework agnostic to robot's middleware, for prototyping and validating the algorithms.
3) We evaluate the effects of proactive collision avoidance using CORNET 2.0 and report results.

Our design and evaluation are based on a warehouse

*This work is supported by Nokia fellowship

# Equally contributing authors

[1]Srikrishna Acharya is with Robert Bosch Centre for Cyber Physical Systems (RBCCPS), IISc, India bacharya@iisc.ac.in

[2]Parv Maheshwari is a student at Indian Institute of Technology, Kharagpur. parvmaheshwari2002@iitkgp.ac.in

[3]Mukunda Bharatheesha is with AI & Robotics Technology Park (ARTPARK)/RBCCPS, IISc, India mukunda@arkpark.in, mukundab@iisc.ac.in

[4] Bharadwaj Amrutur is with depts of ECE/RBCCPS, IISc, and , ARTPARK, IISc, India amrutur@iisc.ac.in

[5]Yogesh Simmhan is Associate Professor, CDS and Associate Faculty, RBCCPS, IISc, India simmhan@iisc.ac.in

automation use-case with a series of pickup and delivery navigation tasks using a fleet of mobile robots.

II. Related Work

In recent years there has been much work in MRS to leverage the network infrastructure in domains like task allocation [12]–[15], state estimation and data fusion [16], localization [17], and data sharing among multiple robots [18]. As regards network-aware path planning, [19] aims to solve the coordinated sensor fusion problem where mutual information between multi robots is coordinated over packet erasure channels. And path planning is used to increase the efficiency in terms of information collected. On the other hand, [20] addresses the offline path planning problem to maximize the communication quality.

In the domain of optimising path planning algorithm leveraging communication, most of the research focus is on path planning under communication constrained [21], [22] and unreliable networks [23], [24]. Well-structured and semi-structured domains such as logistics and cargo management will have a reliable and high-performance network infrastructure at their disposal. In this context, Luna and Bekris [25] leverage a communication network to optimize the path planning algorithms for MRS. Their work formulates the problem of computing paths for the robots within the communication range but this requires very high computation (Parallel computing cluster made of Sun Fire X4100 M2 Nodes, and 25 to 64 such nodes are required). We evaluate a simpler decentralized networked collision avoidance approach in Section IV-B. We also evaluate this new paradigm of network-aware algorithms using a multi-agent network-enabled robot prototyping platform.

The fundamental scope of *robot simulators* like Gazebo [26], AirSim [27], Carla [28], and open-source frameworks like SUMO [29] alongside middlewares like ROS1 and ROS2, is to model the robot's physical dynamics. On the other hand, *network simulators*, NS-3 [30], mininet-WiFi [31], and OMNET++ [32] are driving the innovations in mobile *ad hoc* networks, providing insights into large networks. Besides the physical interaction of the robots with the environment, information sharing is key to best exploit a multi-robot system's capabilities. However, both of these classes of simulators, by design, ignore or simplify the other component and hence suffer the repercussions of ill-representing reality.

As a naïve approach, we can capture traces from a network simulation and introduce them into the robot simulations. But for a realistic simulation of multi-robot systems, we need to close the perception–action loop over a communication channel, and this warrants that the simulation has to be performed in conjunction. There have been recent efforts on joint network-robot simulation [33]–[36]. These are designed to be application-specific and suffer from a siloed approach. In this context, other frameworks [37], [38] provide a more generalized approach. [37] does not offer a proper packet capture mechanism, and [38] has close coupling with ROS middleware for inter-robot communication.

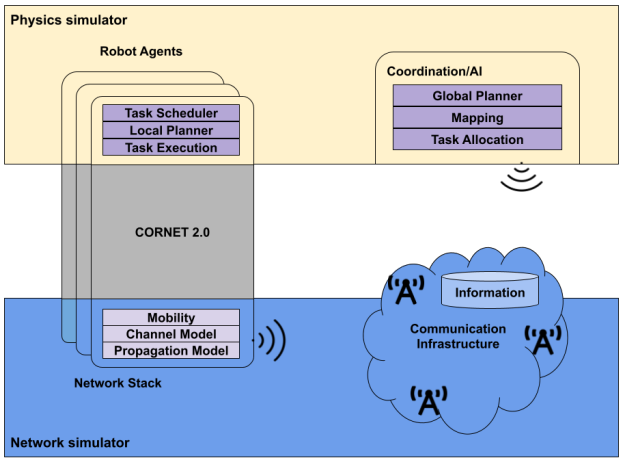


Fig. 2: CORNET 2.0: Architecture overview

III. Background: CORNET 2.0

Fig. 2 provides an overview of the CORNET 2.0 framework we use in our evaluation. It uses mininet-WiFi with Containernet support [31] for modeling network interactions, and Gazebo [26] simulation engine for realising the physical interactions. This middleware addresses our key requirement of seamlessly interconnecting the two simulation platforms to ensure the correctness of the simulation.

CORNET 2.0 consists of two components:

- A veth based end-to-end data-plane
- A position and time synchronization control plane

a) End-to-end data-plane: Each robot runs a full-featured IP-aware networked operating system, as shown in Fig. 2. The primary function of the data-plane is to create an end-to-end data path using a virtual network (e.g., mininet-WiFi) between the robot nodes simulated in the physics simulator (e.g., Gazebo), irrespective of the network topology. For every robot instance in the Gazebo simulation, a corresponding network host is created, and mininet-WiFi places the host process in different Linux OS network namespaces, connecting them through virtual Ethernet pairs and creating a virtual network topology. The host interfaces allow the configuration of parameters for controlling rate, delay, latency, jitter and loss to represent higher fidelity. The application nodes bind sockets to these virtual interfaces and communicate over them. This traffic gets redirected through the virtual network created in the mininet-WiFi and is released to the appropriate node based on the network parameters. This process is entirely transparent to the choice of the application middleware. For the purpose of our experiments, we use ROS2 as the application middleware.

b) Mobility and time synchronization: The co-simulation control plane captures the state and time information of the robots in a synchronized manner, ensuring the simulation's correctness. Each robot's state information is captured regardless of the choice of the user middleware application and updates the mobility of the corresponding nodes in the network simulator.

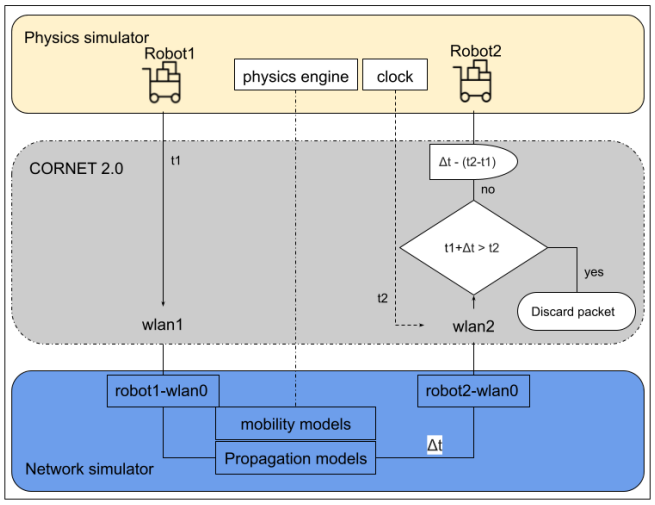


Fig. 3: CORNET 2.0: Design Specifics

By design, physics simulators are modeled as continuous-time dynamics systems. In contrast, network simulators employ the discrete-event model, where the state of the simulation switches based on events such as packet transmission and reception without a constant time-step. It is crucial to bridge the border that separates the cyber and physical parts of the system for a faithful representation of the co-simulation.

We have used Gazebo simulation time as the reference clock. As shown in Fig. 3, when the robot entity in the physics simulator releases the packet to CORNET 2.0 middleware, the message is time-stamped ($t1$) and passed on to the network layer. The network simulator simulates network artifacts, induces a network delay (Δt), and releases the packet to the CORNET 2.0 which keeps track of current physics simulator time ($t2$).

- If $(t2-t1) < \Delta t$, the network simulator event processing is faster than the physic simulator. So the CORNET middleware holds the message for $(\Delta t - (t2 - t1))$ interval before releasing packet to the physics simulator.
- If $(t2-t1) > \Delta t$, the network simulator event processing is slower than physics simulation. In such cases, we discard the packet as it has missed the deadline.

Due to space restrictions, we refer the reader to the supplementary materials for this paper [39] for additional details and performance analysis of CORNET 2.0.

IV. Network Aware Multi-Robot Task Execution

Using our framework we study a Multi-Robot Task Execution activity, which consists of two steps – Non-Networked Multi-Robot Task Allocation (MRTA) and Multi-Robot Path Planning (MRPP) using network information.

A. Multi-Robot Task Allocation

We consider the class of problems characterized by tasks that a single robot can execute at a time (SR), and robots that can execute only a single task at the time (ST). We also consider instantaneous assignment (IA) of tasks to robots, as the arrival of further tasks is not available and unpredictable [40], [41]. We have leveraged a market auction-based MRTA algorithm inspired by [3]. In our implementation, each allocated task can be decomposed into a single or several goal locations, unlike [3], where they have used a fixed definition of task that comprises three goals. This allows flexibility in terms of types of tasks that can be executed. We define the cost function for the robot traveling between two goals g_i and g_f as $G(g_i, g_f)$.

Let t_i and t_j be two tasks defined as $t_i = [g_{i0}, g_{i1}, g_{i2}, ..., g_{iM}]$ and $t_f = [g_{f0}, g_{f1}, g_{f2}, ..., g_{fN}]$. Therefore we modify the own cost function ($S(t_i)$) of a task t_i and the associated cost function ($R(t_i, t_j)$) between t_i and t_j as defined in [3], as:

$$S(t_i) = \sum_{k=1}^{M-1} G(g_{ik}, g_{i(k+1)})$$
$$R(t_i, t_f) = G(g_{iM}, g_{f0}) \quad (1)$$

For ease of discussion, we model the warehouse site as a rasterized rectangle but with the freedom to move diagonally; hence eight directions of movement are allowed, which helps us model the robots' movements efficiently. In contrast to using the Manhattan distance [3], we instead use $G(g_i, g_f)$ as the Octile Distance (Manhattan distance extended to allow diagonal moves) between the two goals in free space, which serves as the notional cost of the actual path.

B. Multi-Robot Path Planning (MRPP)

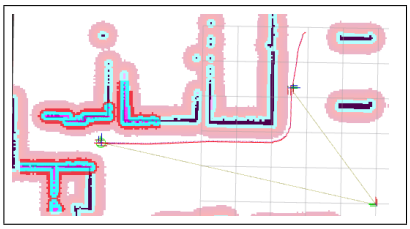
In reactive planning, the robots navigate and avoid dynamic obstacles based only on their sensor data, which directly implies a shorter time horizon available for avoiding the obstacles. On the other hand, proactive planning is where the robots plan well ahead in their path, considering the static and dynamic obstacles.

A high performance network will allow for realizing a distributed planner, which can in principle scale much better in terms of the number of robots. Hence we study a decentralized planning approach that provides proactive collision avoidance by utilising state information passed from one robot to another over the network infrastructure. This approach consists of two components - a global planner, and costmap updates using state information of other robots. For the global planner, we have used Navfn planner in NAV2 [42] which acts as a decentralised single agent path planner.

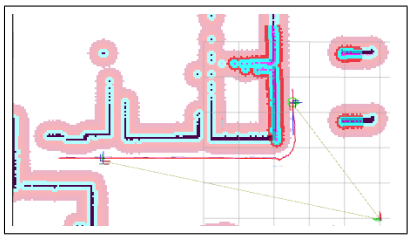
For updating the robot's costmap, we have designed a method to relay the robot's footprint over the network to all the other robots in the ego robot's neighbourhood[1]. These footprints can also act as corrective measures for errors in measurements by low-range obstacle detecting sensors.

Two robots are considered to be neighbours if they are connected to the same access point. However different neighbourhood functions can be realized by appropriately modifying the filter function in Fig. 5

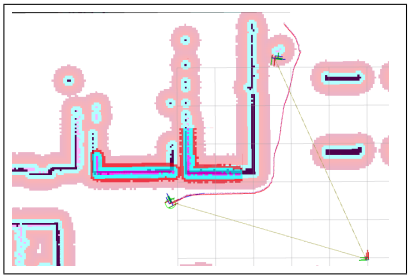
[1]The term *ego* is used for the current robot whose motion is being considered.



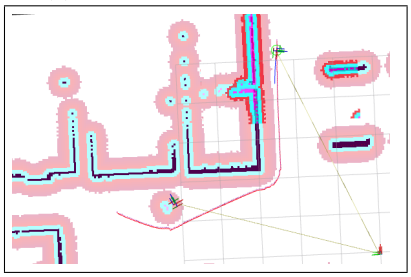
(a) Non-Network case - Robot 1



(b) Non-Network case - Robot 2



(c) Network case - Robot 1



(d) Network case - Robot 2

Fig. 4: In both the non-network and the network cases, the robots cannot detect each other solely using lidar measurements. So in the non-network case, two robots plan their global trajectory through each other. In the network case, they can proactively plan a path avoiding other robots based on the footprints of other robots received over the network.

The knowledge of other robots' footprints in the network neighborhood of the ego robot helps it avoid collisions proactively, and this increases the system's overall performance in terms of:

1) the average time the robots take to complete the assigned tasks
2) reducing the number of collisions, and probability of using the emergency obstacle avoidance maneuver
   - This is measured using the minimum non-collision distance (MNCD) metric. It is defined as the minimum distance in the case of non-colliding interactions between robots.
   - Interactions are defined as two robots coming close to each other, within a certain distance threshold.

   In our experiments, we have kept that threshold at $2m$ since that is the range of the lidar on-board our robots.

There are many applications of proactive planning, and it is beneficial in cases where reactive planning has a higher probability of failing:

1) In the case of low-range lidar which are cost-effective, robots rely only on reactive planning, which in turn means that the chances of collision are high.
2) In Fig. 4, the robots 1 and 2 have planned paths around the corner and are on a collision course. Here, reactive planning in Figs. 4a and 4b may not have enough time to avert the situation as the lidar cannot detect the other robot around the corner, while proactive planning in Figs. 4c and 4d is able to detect and avoid the other robot.
3) In reactive planning, the robot's threshold on the max velocity depends on the lidar range for safe navigation and collision avoidance. Proactive planning helps increase this threshold as the robot is already aware of the position of the other robots.

V. Architecture and Message Flow

This section elaborates on each component of our proactive planning approach along with the message flow. As shown in Fig. 5, the orders are dispatched to the task allocator module to convert them into robot tasks. Based on the MRTA algorithm discussed in Section IV-A once a robot is selected to execute this task, the corresponding task is transferred to the respective robot node. These robot nodes act as intermediary nodes between the task allocator and the navigation stack of a robot and have the following responsibilities:

1) keep track of the list of tasks allocated to a robot
2) sequentially send the current task's goals to the waypoint follower of the navigation stack
3) retrieve the feedback about the current task from the navigation stack, and update the local and central list of tasks allocated to that robot.

Using CORNET 2.0, a network topology is generated to have a coverage span of the entire warehouse, and the network node is in sync with the robot node in the Gazebo environment. In each robot's network node, we run a simple ping server that generates the ping status of all the other participating robots and publishes this ping status array from each robot to the footprint filter node.

The footprint filter subscribes to all the published footprints along with the ping array from the robot's ping server and creates a 2-D Boolean array whose i^{th} row shows the network visibility of other robots with respect to the i^{th} robot. Therefore, the i^{th} robot's footprint filter only publishes the footprint of the robots which are in the network visibility of the i^{th} robot.

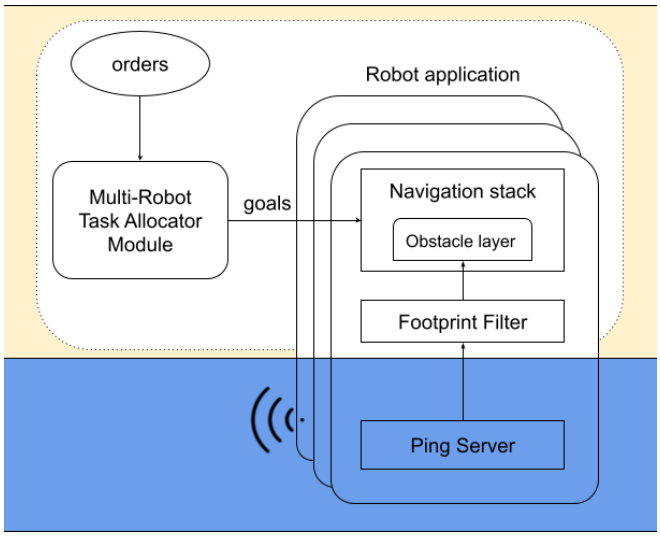


Fig. 5: Architecture and message flow diagram

The obstacle layer[2] of the robot's navigation stack subscribes to the filtered footprints to update the robot's costmap accordingly. The navigation stack sends the velocity control commands to the robots in Gazebo.

VI. Experimental Setup

We define the following two cases for each experiment that we perform:

1) Multi-robot task execution without using a network. This is referred to as the *non-network case* from hereon.
2) Multi-robot task execution using the network. This is referred to as the *network case* from hereon. For these experiments, we assume that an infrastructure-assisted WiFi network is available with coverage of the whole warehouse.

We use ROS2 as our communication middleware. NAV2 [42] has been used as the navigation stack. We use CORNET 2.0 for the end-to-end simulation. For the exact environment, we used a small warehouse world provided by AWS robotics (Fig. 6). While the robots are positioned inside the centre squares, the shelves on the right-hand side act as pickup stations where cargo is picked up from by the robot. The racks on the left-hand side act as delivery stations and cargo has to be delivered here by the robots.

We perform the following experiments,

1) Changing the number of robots in the environment while we kept the max velocity of the robots fixed, at $1.6m/s$
2) The maximum allowed velocity is varied for a fixed number of robots, 4 in this case.

For these experiments, we consider that robots possess only a $2m$ lidar to detect the obstacles. Also for both experiments, we allocate four tasks to each robot for a sufficient comparison.

[2]We have used a custom implementation of obstacle layer plugin provided with the Costmap-2D.[42]

Fig. 6: AWS Small Warehouse World in Gazebo

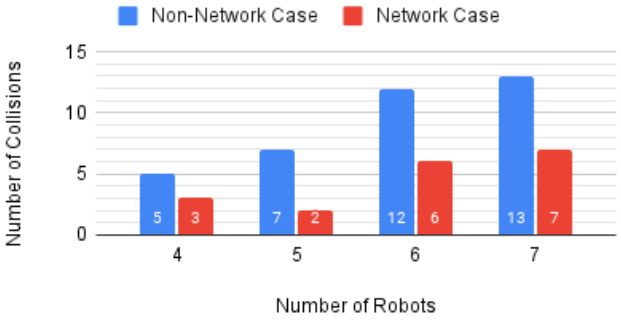


Fig. 7: Number of collisions when the Number of Robots is varied

VII. Results

Fig. 7 reports the number of collisions that occur between the robots for the non-network and network cases, as the number of operating robots increase from 4 to 7. From this we can conclude that the number of collisions in the network case is much lower than in the non-network case. In the network case, the robots can proactively avoid the obstacles using information shared over the network as the collision avoidance time horizon is higher. Hence, the change in velocity required to avoid the obstacle is much less than reactive avoidance. Consequently, this also increases the robots' average velocity, as shown in Fig. 8.

When two robots collide, there is a high probability of the robots becoming unstable and detecting the floor as an obstacle, which leads them to replan or go into recovery mechanism. This result in a longer path, which increases the overall distance traveled to complete the tasks. In network cases, as the number of collisions is less, the number of times the recovery mechanisms is triggered in lowered. This leads to a decrease in the total distance traveled by the system. Conversely, as the number of robots increases, there is also a chance that the robot would take a longer path to avoid the other robots proactively and hence even if collisions are less, there might be an increase in the total distance traveled by the system. This was the case for the setup involving seven robots, shown in Fig. 9. The total distance traveled is slightly higher in the network case as a trade-off for avoiding collisions.

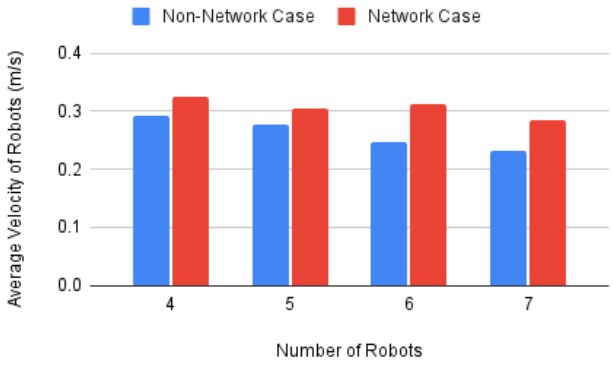


Fig. 8: Average Velocity of Robots when the Number of Robots is varied

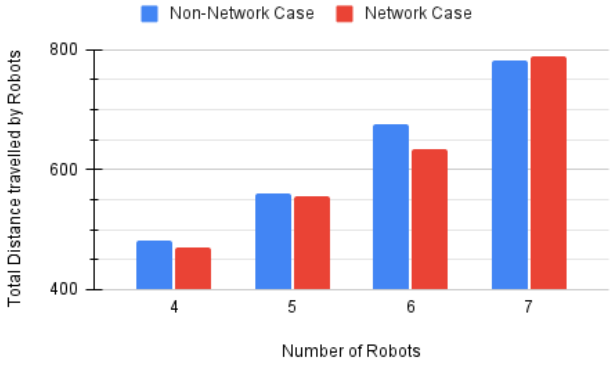


Fig. 9: Total Distance travelled by Robots when the Number of Robots is varied

Fig. 10 compares the non-network and network cases when the maximum allowed velocity of the robots is increased. It illustrates two trends:

- As the maximum velocity increases, the minimum non-collision distance (MNCD) decreases both in the network and the non-network cases because of lesser reaction time for collision avoidance.
- For every value of maximum velocity, the MNCD is higher in the network case compared to the non-network case, illustrating that there is a higher collision avoidance time horizon for the network case.

There are some interactions between robots that lead to collisions even in the network case. These are when the robots tend to go in the same direction for proactive avoidance, and hence they still collide. Future works can ensure a better or complete collision avoidance where the global trajectories resolve all conflicts. A conflict is defined as when two robots go in the same direction to avoid each other. While this can ensure better collision avoidance, it would also require higher compute effort [25].

VIII. Discussion and Conclusions

This paper describes the benefits of having a simulation infrastructure to research and develop solutions in the field

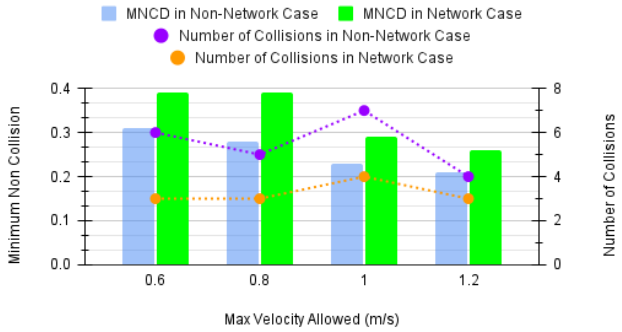


Fig. 10: Analysis by varying Max. Allowed Velocity

of networked robotics. In particular, we leverage a network-robot co-simulation framework, CORNET 2.0, which is agnostic to middleware. With the rapid progress in both networking technologies such as WiFi6 and 5G, the availability of such frameworks will hold the key for rapid prototyping of network-aware robot automation.

We use this framework in the context of a basic Industry 4.0 use-case of a multi-robot task execution in a warehouse environment. We show proactive collision detection and avoidance in areas of less on-board sensor visibility, such as alleyway crossings and corners. These results underline how such a framework allows one to evaluate metrics impacting throughput in a networked warehouse automation context where a traditional non-networked automation relying solely on on-board sensors would not suffice. Such evaluations in turn enable making key assessments from a resource logistics perspective too.

The middleware-agnostic nature of CORNET 2.0 enables easy integration of network infrastructure aspects to an already existing robotic automation simulation. The framework also provides avenues to evaluate network specific characteristics such as channel and packet loss, planning for network coverage in a given area with minimal or no blind spots and so forth. We strongly believe that these characteristics are essential to accelerate further research and development in network-aware behavioral design solutions in robotic automation.

Acknowledgment

We would like to thank Nokia for their constant financial support to conduct this work. Furthermore, we are very grateful to Dr. Naveen Arulselvan, Shubham Sharma, Varghese Kuruvilla, Poornima JD and Sadgun Srinivas from ARTPARK for their support in the process of developing the software infrastructure used in this work.

References

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] W. K. SERIES, "The trend towards warehouse automation," Westernacher, Tech. Rep., 2017.

[3] Y. Fan, F. Deng, and X. Shi, "Multi-robot task allocation and path planning system design," *2020 39th Chinese Control Conference (CCC)*, pp. 4759–4764, 2020.

[4] 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.

[5] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2011, pp. 3260–3267.

[6] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[7] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous robots*, vol. 8, no. 3, pp. 325–344, 2000.

[8] J. Pugh and A. Martinoli, "Inspiring and modeling multi-robot search with particle swarm optimization," in *2007 IEEE swarm intelligence symposium*, IEEE, 2007, pp. 332–339.

[9] D. Portugal and R. Rocha, "A survey on multi-robot patrolling algorithms," in *Doctoral conference on computing, electrical and industrial systems*, Springer, 2011, pp. 139–146.

[10] S. Carpin, "Fast and accurate map merging for multi-robot systems," *Autonomous robots*, vol. 25, no. 3, pp. 305–316, 2008.

[11] G. Antonelli, F. Arrichiello, and S. Chiaverini, "Flocking for multi-robot systems via the null-space-based behavioral control," *Swarm Intelligence*, vol. 4, no. 1, pp. 37–56, 2010.

[12] M. Chowdhury and M. Maier, "Local and nonlocal human-to-robot task allocation in fiber-wireless multi-robot networks," *IEEE Systems Journal*, vol. 12, pp. 2250–2260, 2018.

[13] G.-Y. Shi, I. E. Rabban, L. Zhou, and P. Tokekar, "Communication-aware multi-robot coordination with submodular maximization," *ArXiv*, vol. abs/2011.01476, 2020.

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

[15] P. Mazdin and B. Rinner, "Distributed and communication-aware coalition formation and task assignment in multi-robot systems," *IEEE Access*, vol. 9, pp. 35 088–35 100, 2021.

[16] A. Tamjidi, R. Oftadeh, S. Chakravorty, and D. A. Shell, "Efficient recursive distributed state estimation of hidden markov models over unreliable networks," *Autonomous Robots*, pp. 1–18, 2020.

[17] A. Wasik, P. Lima, and A. Martinoli, "A robust localization system for multi-robot formations based on an extension of a gaussian mixture probability hypothesis density filter," *Autonomous Robots*, vol. 44, pp. 395–414, 2020.

[18] V. Varadharajan, D. St-Onge, B. Adams, and G. Beltrame, "Soul: Data sharing for robot swarms," *Autonomous Robots*, vol. 44, pp. 377–394, 2020.

[19] V. Ramaswamy, S. Moon, E. W. Frew, and N. Ahmed, "Mutual information based communication aware path planning: A game theoretic perspective," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1823–1828. DOI: 10.1109/IROS.2016.7759290.

[20] A. Mardani, M. Chiaberge, and P. Giaccone, "Communication-aware uav path planning," *IEEE Access*, vol. 7, pp. 52 609–52 621, 2019.

[21] S. Kemna, J. G. Rogers, C. Nieto-Granda, S. Young, and G. S. Sukhatme, "Multi-robot coordination through dynamic voronoi partitioning for informative adaptive sampling in communication-constrained environments," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2124–2130.

[22] F. Amigoni, J. Banfi, and N. Basilico, "Multirobot exploration of communication-restricted environments: A survey," *IEEE Intelligent Systems*, vol. 32, no. 6, pp. 48–57, 2017.

[23] A. Tamjidi, R. Oftadeh, S. Chakravorty, and D. A. Shell, "Efficient recursive distributed state estimation of hidden markov models over unreliable networks.," *Auton. Robots*, vol. 44, no. 3-4, pp. 321–338, 2020. [Online]. Available: http://dblp.uni-trier.de/db/journals/arobots/arobots44.html#TamjidiOCS20.

[24] A. Mannucci, L. Pallottino, and F. Pecora, "Provably safe multi-robot coordination with unreliable communication," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3232–3239, 2019.

[25] R. Luna and K. E. Bekris, "Network-guided multi-robot path planning in discrete representations," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 4596–4602.

[26] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, pp. 2149–2154.

[27] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, Springer, 2018, pp. 621–635.

[28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, PMLR, 2017, pp. 1–16.

[29] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility an overview," 2011.

[30] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[31] R. R. Fontes, S. Afzal, S. H. Brito, M. A. Santos, and C. E. Rothenberg, "Mininet-wifi: Emulating software-defined wireless networks," in *2015 11th International Conference on Network and Service Management (CNSM)*, IEEE, 2015, pp. 384–389.

[32] A. Varga, "Omnet++," in *Modeling and Tools for Network Simulation*, Springer Berlin Heidelberg, 2010, pp. 35–59. DOI: 10.1007/978-3-642-12331-3_3. [Online]. Available: https://doi.org/10.1007/978-3-642-12331-3_3.

[33] E. A. Marconato, M. Rodrigues, R. d. M. Pires, D. F. Pigatto, A. R. Pinto, K. R. Branco, *et al.*, "Avens-a novel flying ad hoc network simulator with automatic code generation for unmanned aircraft system," 2017.

[34] N. R. Zema, A. Trotta, E. Natalizio, M. Di Felice, and L. Bononi, "The cuscus simulator for distributed networked control systems: Architecture and use-cases," *Ad Hoc Networks*, vol. 68, pp. 33–47, 2018.

[35] S. Baidya, Z. Shaikh, and M. Levorato, "Flynetsim: An open source synchronized uav network simulator based on ns-3 and ardupilot," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM, 2018, pp. 37–45.

[36] S. Acharya, A. Bharadwaj, Y. Simmhan, A. Gopalan, P. Parag, and H. Tyagi, "Cornet: A co-simulation middleware for robot networks," in *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, IEEE, 2020, pp. 245–251.

[37] M. Kudelski, L. M. Gambardella, and G. A. Di Caro, "Robonetsim: An integrated framework for multi-robot and network simulation," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 483–496, 2013.

[38] M. Calvo-Fullana, D. Mox, A. Pyattaev, J. Fink, V. Kumar, and A. Ribeiro, "Ros-netsim: A framework for the integration of robotic and network simulators," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1120–1127, 2021.

[39] S. Acharya, B. Amrutur, M. Bharatheesha, and Y. Simmhan, *Cornet 2.0: A co-simulation middleware for robot networks*, 2021. arXiv: 2109.06979 [cs.RO].

[40] B. P. Gerkey and M. Mataric, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, pp. 939–954, 2004.

[41] A. Khamis, A. Hussein, and A. M. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," in *Advances in Social Media Analysis*, 2015.

[42] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. [Online]. Available: https://github.com/ros-planning/navigation2.