Journal of Robotics and Control (JRC)
Volume 3, Issue 6, November 2022
ISSN: 2715-5072, DOI: 10.18196/jrc.v3i6.16298

Fleet Management System for an Industry Environment

Jakub Hažík [1], Martin Dekan [2], Peter Beňo [3], František Duchoň [4*]
[1,2,4] Institute of Robitics and Cybernetics, Faculty of Electrical Engineeringand Information Technology, Slovak University of Technology, Bratislava, Slovak Republic
[3] Photoneo s.r.o., Bratislava, Slovak Republic
Email: [1] hazik.jakub@gmail.com, [2] martin.dekan@stuba.sk, [3] beno@photoneo.com, [4*] frantisek.duchon@stuba.sk
*Corresponding Author

***Abstract* — The article deals with the management of a fleet of AMR robots that perform logistics in production. The entire system design is implemented in the ROS environment - state of the art for the development in robotics. Four already available solutions for fleet management in ROSe are analyzed in detail in the article. These solutions fail when there is a need to change the route plan in a dynamically changing environment. Likewise, some did not sufficiently synchronize the movement of the robots and collisions occurred or, with a larger number of robots, represented an enormous computational load. Our solution was designed to be as simple and reliable as possible for industrial use. It is based on a combination of semi-autonomous and centralized approach. A hybrid map is used for planning the movement of the robot fleet, which provides the advantages of both a metric and a topological map. This route map for a fleet of robots can be easily drawn in readily available CAD software. Synchronization of robots was designed on the principle of semaphore or mutex, which enabled the use of bidirectional paths. The results are verified in simulations and were aimed at verifying the proposed robot synchronization. It was confirmed that the proposed synchronization slows down the robots, but there were no collision situations. By separating route planning from synchronization, we simplified the entire fleet management process and thus created a very efficient system for network and hardware resources. In addition, the system is easily expandable.**

Keywords— Fleet management; ROS; Path; Mobile robot

I. INTRODUCTION

Automatic vehicles have been used in the industrial environment for several decades. However, their deployment is often problematic and unprofitable. Therefore, these vehicles must be replaced by smarter AGVs [14], able to move even in an environment that can be changed dynamically. It is assumed that such robots will move into the factory, according to the Industry 4.0 model. These robots must move along common paths and share critical information such as their location and environment map. Due to the industrial environment's safety requirements, this fleet must be controlled by a superior system - a supervisor.

In the context of Industry 4.0, new innovative approaches to the operation of AGV robots are beginning to take shape, which promises higher efficiency, higher adaptability, and much more. A new type of industrial robot called AMR is starting to emerge [15]. The first difference between AGV and AMR follows from the name (Fig. 1). The AGV is a guided vehicle, i. e., it needs a built-in support guidance system, mostly in magnetic strips in the floor or various signs in the environment. The advantage of the AMR robot is that it does not need anything like that. Thanks to this, its deployment or change of environment is much faster and cheaper. AMR is a robot with a certain degree of autonomy. The path planning between point A and point B is left exclusively to localization and navigation algorithms located directly in the robot or the master computer [16]. Almost all AGV systems are designed to operate without hindrance and out of reach of people. In the event of an obstacle, they will stop immediately for safety reasons. In contrast, most AMRs are designed to interact with the environment as much as possible. For example, if the robot detects an obstacle, and there is enough space to avoid it, it will bypass the obstacle to optimize its cycle time.

Fig. 1. AGV from company CEIT (left) and AMR from company Photoneo (right)

Most AGVs operate independently and must be manually programmed to avoid collisions. The AMR robot fleet should resolve mutual collisions directly in their essence, without the integrator having to program and handle all collision situations. Although AMR robots are far superior to AGV robots, their algorithms are still evolving and still need a lot of attention to compete with conventional AGVs in terms of reliability and safety.

In a distributed system such as the AMR robot fleet [17], the supervisor, or other words, the superior system, has also significant mission. It should play the role of monitoring other robots (subsystems) and should also provide higher-level control for all the robots. In practice, this can mean distributing necessary information for robots, such as an environment map, the location of workplace and charging stations, the required tasks, and the implementation of operational interventions by the operator. As a running process, the supervisor must also guarantee the safe operation of the entire system. For this reason, and especially from the

supervisor's nature, it is essential that it does the work on a dedicated computing resource (PC) and not on one of the robots.

The research contribution is a novel approach to the fleet management system. Compared to existing solutions, this system is simple, reliable and easily expandable. Compared to other solutions, which are analyzed in detail in Section III, our system separated the planning process from the synchronization. Thanks to this, it is not so complex and is much more flexible in the event of a need to trigger a change in the route plan. The system designed in this way can also be used for a larger number of robots, because it is not demanding on hardware and communication networks. The entire framework of our solution is described in detail in part IV. Subsequently, further improvements, which resulted from the experiments, are described in detail in Section V. Finally, there is a section describing the results that demonstrate the high efficiency and reliability of our approach.

II. FRAMEWORK

The proposed algorithms are implemented using ROS tools and libraries. ROS [3] provides communication interfaces and tools that greatly simplify the writing of software for robots. A control system is arranged in ROS nodes and communication channels, where the ROS node represents a process that can communicate with other equivalent processes via communication interfaces. It also contains useful tools for visualizing, monitoring, and debugging the programs.

A hybrid map [18] was used to represent the environment (a combination of metric and topological maps). This takes advantage of both representations. With a metric (grid occupancy) map, a very detailed model of the environment can be achieved. The robots can map well and then locate themselves in space using rangefinders. However, due to the required computational power to work with such a map, it turns out that a metric map is not so ideal for planning, especially in the case of large environments. Therefore, a topological representation will also be used. Such a model has the form of a graph and is more suitable for searching for possible routes.

Concerning mobile robots, it is necessary to consider the use of a wireless communication network primarily [19]. When monitoring and controlling AMR robots' fleet by some superior system, building a quality network in a given environment is required. The industrial network [4], in contrast to the conventional one, must meet high requirements in several parameters such as coverage, reliability, security, durability, scalability, and fast response. Several network architectures and industrial protocols are currently being developed, including Wi-Fi, ZigBee, Bluetooth, RFID, and other proprietary protocols. In industrial mobile robotics, Wi-Fi technology represents the vast majority, thanks to the sufficiently fast data flow, acceptable price, and long-range. Although Wi-Fi was not designed for industrial purposes, it can meet industrial requirements due to protocol and device-level modifications. One of the network's critical parameters for controlling mobile robots is the response speed, mainly due to strict safety certifications. Table 1 shows the maximum allowed network latencies for individual actions with the mobile robot. For example, to use a given network to control the robot's movement in real-time, it is necessary to ensure a latency lower than 1 ms.

TABLE I. INDUSTRIAL NETWORK REQUIREMENTS FOR SOME MOBILE ROBOT OPERATIONS [10]

Operation	Latency (end-to-end)	Message size
Monitoring system	50 ms – 1 s	40 – 250 bytes
Control	<10 ms	40 – 250 bytes
Positioning	< 1ms	40 – 250 bytes

III. ANALYSIS OF AVAILABLE SOLUTIONS

Even before creating our solution, the necessary analysis of available solutions for fleet management was performed. The following is a summary analysis of these solutions.

A. Tuw_multi_robot [1]

Tuw_multi_robot is a package of the ROS that was developed on the Vienna University of Technology premises. It is a set of algorithms that can control a fleet of robots from the initials to the target stations in any map without collisions. In the first step, a graph is generated based on a metric map using a Voronoi diagram [21], or the graph is directly defined by the user. Subsequently, the robots' target positions are selected, and synchronized routes through the graph's vertices are planned. Route synchronization consists in the fact that, for example, robot A is allowed to reach a given node if robot B has successfully passed this node, as it is shown in Fig. 2. More details can be found in [20].

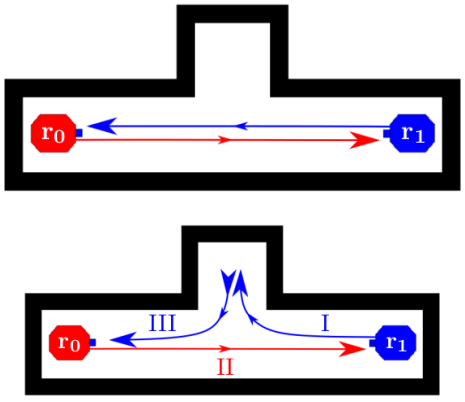

Fig. 2. Deadlock solution: UP - Initial and required robot positions representing the problem DOWN - Solving the problem by avoiding.

Addressing specific situations like this makes the package a potent tool in route planning for a robot fleet in environments where these situations occur. It is necessary to address them at the planning level. On the other hand, there is an issue in the result of the scheduler, which has to generate one synchronized route plan for all robots at the same time. If one of the robots reached the target and wanted to continue to the new target while the other robots are still moving, there are only two options. A new route plan could be created and immediately implemented, or the robot would have to wait for all the other robots to reach their goals. In both cases, however, we encounter side effects. After successfully running the simulation, we discovered issues where the route synchronization was not correct, and the robots collided. We

also analyzed the source code in more depth, and we judged that implementing any modifications will be extremely challenging.

B. Multi_robot_collision_avoidance [2]

Another important project is the work of scientists from the Dutch University Maastricht University, which has also grown into an available ROS package. This project seeks to solve the problem of mutual collisions between robots in real-time. The basis of the solution (Fig. 3) consists of known robot positions due to the AMCL localization in conjunction with modified methods ClearPath and Optimal Reciprocal Collision Avoidance. More details can be found in [22].

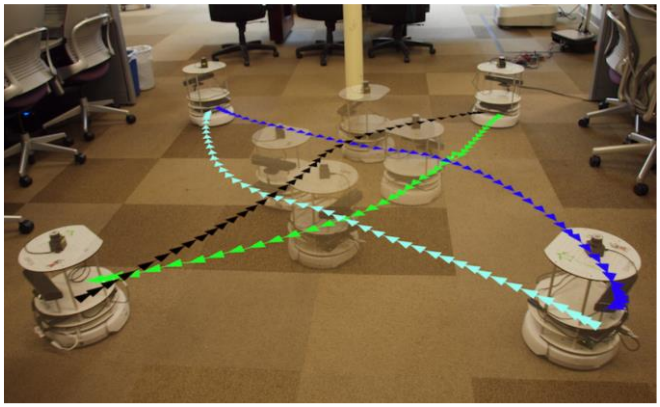

Fig. 3. A real example of the use of dynamic collision avoidance [2]

C. LibMultiRobotPlanning [5]

It is a C ++ library, and the primary purpose is to provide search algorithms for systems with the navigation of one or more robots. The individual algorithms are based on several different scientific methods, such as A *, A * epsilon, SIPP, CBS, ECBS, CBS-TA, ECBS-TA, etc. However, similar to tuw_multi_robot, one synchronized route plan is created for all robots when using multiple robots at once. Modifying the scheduler to generate a path for one robot without disrupting other robots' currently performed paths is not an easy task. More details can be found in [23].

D. Coordination_oru [7]

This research project, which was created at the Swedish University Örerbo University in the Multi-Robot Planning and Control Lab, represents an entirely different planning approach than in previous cases. To control the fleet of robots, it uses the so-called online planning, thanks to which the individual routes are synchronously independent of each other. So, if one of the robots requests a new path, the consequence of its creation is not so drastically invasive for the paths of the other robots. Part of the project is creating a usable ROS package with decent documentation and simulation tools. Despite efforts to achieve a reliable robot control system [6], it is an online control tool that poses particular risks in a planning or communication system's failure. It also places exceptionally high demands on the communication network, and the use of a large number of robots can lead to an unbearable situation. More details can be found in [24].

IV. CONCEPT AND METHODOLOGY

The standard V-model (Fig. 4) was applied for the development phase of the entire solution. The requirements for fleet management were defined as the creation of a simple and reliable system capable of running in real time so that the movement of robots is as efficient as possible and at the same time without any collisions. Subsequently, the available architectures in the ROS environment were analyzed (Section III) and it was found that none of these approaches meets the defined requirements. For this reason, a new approach was designed (Section IV). Based on tests in various scenarios, the shortcomings of the proposed approach were revealed. Based on these experiments, improvements were proposed (Section V) that created a system meeting the defined requirements (Section VI).

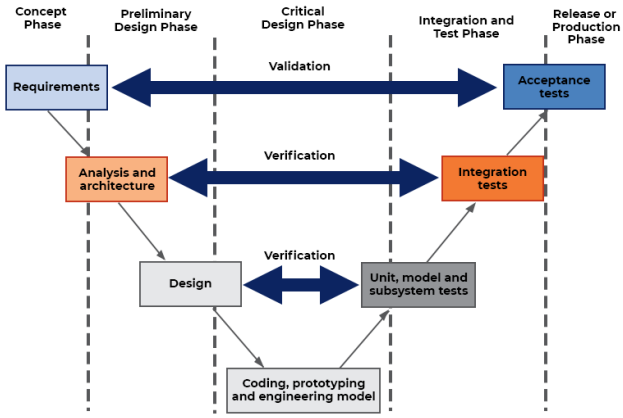

Fig. 4. V-model diagram [13]

The primary task for each of the robots is to get from point A to point B. When achieving this goal, it is necessary to plan routes so that there is no collision with the environment and other robots.

The **simplest approach** is that each robot will have its own planner to calculate the given robot's route without collisions with the environment [25]. Any conflicts with other robots would be resolved locally. In this case, the supervisor has a monitoring task and would not interfere in planning the movement or resolving the collisions. This can work without significant issues when a set of paths on which the robot can move and their ordered motion direction are introduced. It is also necessary to define mutual rules at intersections so that robots can operatively resolve collisions. This solution is relatively simple and easy to use in cases where there is a possibility to define only one-way paths. Contrarywise, this solution is inapplicable when the robots must move in a narrow corridor in both directions.

The second (**semi-autonomous**) approach differs from the previous one in that the planning task is taken over by a superior system (supervisor) [26]. The primary motivation for assigning this supervisor is to optimize the movement of robots between stations. As the supervisor has information on all robots and their paths, it may plan more suitable routes with this data. The planning algorithm can also consider the statistics of the time delay on individual routes and thus adjust the route planning so that it is at least suboptimal. In the semi-autonomous supervisor, the solving of local conflicts, e. g.

synchronization at intersections, is the task of the individual robots. The described method of control, like the previous one, requires a network of clearly defined one-way paths. However, with certain modifications in the supervisor and the warranty of sufficiently large space at intersections, it is possible to use bi-directional paths. Thanks to the possibility of optimizing the arrival of robots to the target station, this planning method has an advantage in certain situations. This is especially true in environments where collision objects often enter the robot's workspace, due to which, for safety reasons, the robots have to stop and wait for them to be removed.

The third (**centralized**) approach [8] [9] combines the logic of route planning and synchronization, similar to the solution of tuw_multi_robot3. This means that the robots are not creating individual route plans, but (one) central planner makes them for all robots at once [27]. It would not be necessary to deal with robot conflicts and collisions while performing paths, as the plan itself already resolves them. The advantage of this solution is that the paths do not have to be fixed. This is because the deadlock can be solved by the planner with a suitable synchronized route plan. As a result, it is possible to cover more complicated cases, which makes the solution more robust. It is also possible to achieve a higher efficiency of robot movement because permissible paths are more extensive. The problem that makes implementing this control method more difficult is when one of the robots reaches the target station and asks for a new route. If the other robots are moving and still realizing their routes, a non-moving robot must wait, or the entire plan must be recalculated from the beginning. There are several solutions to how to create a collision-free path for a waiting robot. For example, the robot will wait for the other robots to reach the target station before a new route plan is generated for all robots. Collision-free operation is achieved, and at the same time, it is not necessary to adjust the route of other robots. It is a straightforward solution, but it is not very efficient. Another possibility is to request a new path and then generate and immediately implement a new route plan for all robots. But again, many other problems should be algorithmically solved so that the intervention in the actual movement of robots is as small as possible. Deployment of such a planning structure is also inappropriate when collision objects often enter the robot workspace. Every encounter requires stopping the robot and waiting for the obstacle to be removed for safety reasons. As a result, this has the effect of breaking the interconnected synchronization of all robots' paths.

We decided to proceed further with the development of algorithms for robot fleet management in a form that will be simple, to begin with, but with the potential to expand. The main simplification prerequisite will be introducing one-way paths in an environment defined in advance by the user. Simultaneously, it is the standard principle of operation of classic AGV robots in factories, from which comes the largest percentage of demand for such robots. Thanks to this simplistic assumption, we will focus on verifying and debugging individual algorithms to achieve a stable and reliable system. It will combine the control mentioned above structures' advantages, namely a semi-autonomous and centralized approach.

The role of the robot is to perform tasks coming from the superior system [11] [12]. This will be defined as a request to move to a point in space, while the supervisor system will also send the path itself. If a dynamic obstacle enters the path, the robot will respond by stopping and wait until the obstacle is removed. For safety reasons, such stopping should be solved using a control system directly on the robot. Therefore, the proposed fleet management control will not deal with dynamic obstacles in the environment. When dealing with the intersections, the robot will communicate with the superior system because the supervisor knows all the robots' current positions. Therefore, if the robot locates itself in front of the intersection zone, it will ask the superior system for information on whether another robot is in the given area. If so, the robot will wait in the current location and enter this zone only when it is free.

The Gazebo simulation environment was chosen to verify the solution. A pre-prepared environment with Kobuki robots was used [28][29]. A differentially driven chassis characterizes these robots. IMU sensors and buffer switches are also used. For the needs of robot control, the platform was enhanced by a simple lidar (RPL Lidar). An individual platform's control is a simple position controller, which can receive commands from positioning and implement the corresponding action with feedback control. This controller is described in the following pseudo-code.

Algorithm 1 Position controller

Input: CurrentPose *pc*, RequiredPose *pr*
Output: *angularSpeed*, *linearSpeed*
errorAngle, *errorDistance* := **GetError**(*pc*, *pr*)
angularSpeed := *Kp*,*rotation* ∗ *errorAngle*
angularSpeed := **AcceptMaxAcceleration**(*angularSpeed*)
angularSpeed := **Saturation**(*angularSpeed*). Výpočet akčného zásahu - translácia
linearSpeed := *Kp*,*translation* ∗ *errorDistance*
linearSpeed := *linearSpeed*− **abs**(*angularSpeed*) ∗*speedsBalanceCoefficient*
linearSpeed := **AcceptMaxAcceleration**(*linearSpeed*)
linearSpeed := **Saturation**(*linearSpeed*)
function GetError(*pc*, *pr*)
errorDistance := **sqrt**((*pc.x* − *pr.x*)2 + (*pc.y* − *pr.y*)2)
goalAngle := **atan2**(*pr.y* − *pc.y*, *pr.x* − *pc.x*)
robotAngle := *pc.theta*
if HasTheSameSigns(*robotAngle*, *goalAngle*) **then**
return *{goalAngle* − *robotAngle*, *distanceError}*
end if
if *robotAngle* > 0 **then**
errorAngle := −1 ∗ **abs**(*goalAngle*) + **abs**(*robotAngle*)
if *errorAngle* < −π **then**
errorAngle := *errorAngle* + 2π
end if
else
errorAngle := **abs**(*goalAngle*) + **abs**(*robotAngle*)
if *errorAngle* > π **then**
errorAngle := *errorAngle* − 2π
end if
end if
return *{errorAngle*, *errorDistance}*
end function

Existing CAD software tools were used to create paths in the environment (Fig. 5). There are a few rules to follow when drawing the paths. Only lines and curves (Arc) should be used, and a color-coded arrow defines the direction of individual paths. The arrow, together with the appropriate line or curve, is inserted into the block named path_X. The network of roads created in this way is saved in the standardized .dxf format. The output is an oriented graph (Fig. 6). Each node of the graph contains these attributes: ID - unique identifier of the node and [X, Y] - Cartesian coordinates of the position in the 2D map. The edges of the graph also contain an attribute - the distance between the vertices. The A * algorithm is used for path planning [30].

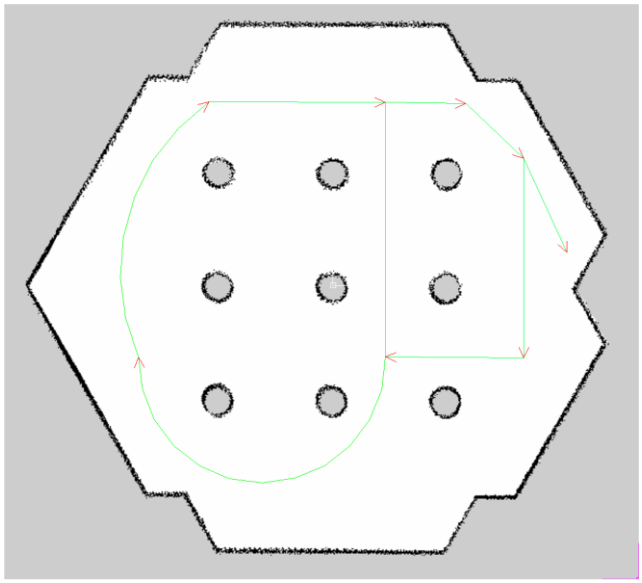

Fig. 5. Defining paths using AutoCAD software

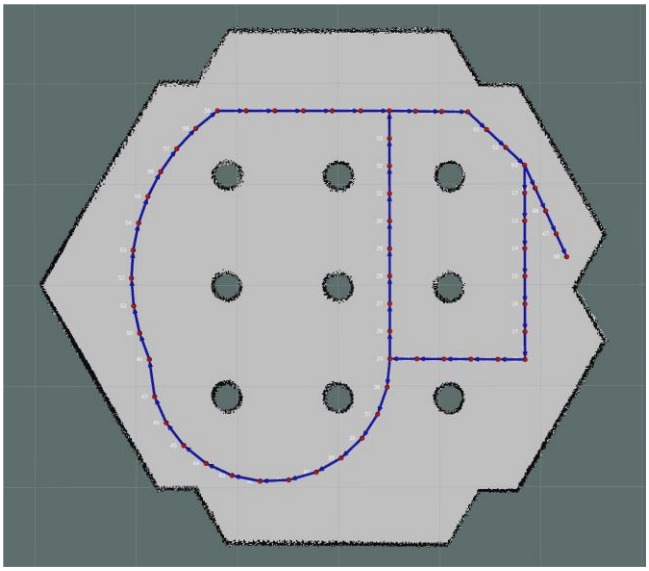

Fig. 6. Oriented graph generated based on pre-drawn paths (Rviz)

After the first tests, the movement of the robots was recorded. The problems were identified at road intersections (crossroads) and in the case of different robot speeds. The proposed implementation does not take such conditions into account, and collisions between robots occur. Thus, it is necessary to create a motion synchronization system that will ensure the smooth and safe movement of robots on dedicated roads. The paths are represented in a graph, and each of them is marked with a unique identifier, which allows the allocation of individual vertices for a given robot. So, the robot cannot enter vertices already allocated for another robot. Such functionality is very similar to a general principle of semaphore or mutex [31]. This semaphore is represented as a separate ROS node running on a supervisor. The individual robots ask for a single ROS service to allocate the node that the robot wants to reach. If the server returns a successful allocation response, the robot can directly move to that node. Otherwise, if another robot already allocates this node, the server returns a failed allocation message, the robot stops and resubmits after some time. The number of nodes allocated for each of the robots is defined by a configurable parameter set by the user concerning the physical length of a particular robot. It is also possible to consider the connected carts in this parameter. Since the configuration defines the maximum number of vertices, it is not necessary to deallocate redundant vertices. This is done automatically by the algorithm after exceeding the limit. Fig. 7 shows a specific situation in which a collision is avoided thanks to the used semaphore. The color-coded dots (purple and yellow) represent the robots' allocated dots, with the robot standing to the left waiting for the freeing of the intersection.

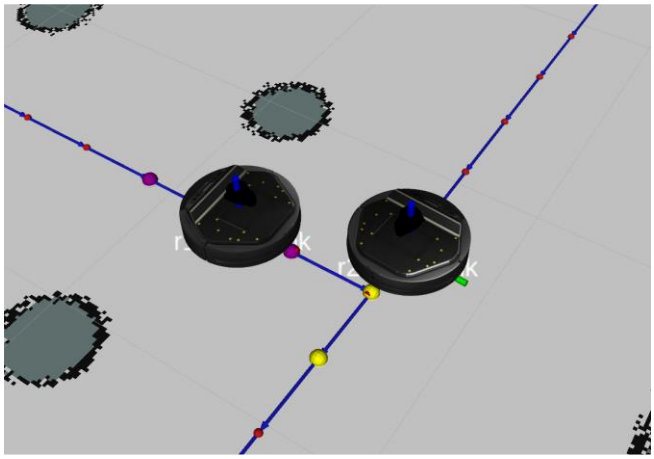

Fig. 7. Solution of a collision of the robots

Successful usage of AMR robots also needs to define the starting and target points and some other conditions. However, moving from point A to point B may not be the only task that the robot must perform from a practical perspective. After reaching the target station, the robot does not leave it immediately with a new target. The robot must remain in the station to load or unload the material, and only then it can move to the new target station. This defines a new task for the robot: "wait for the operator to confirm the loading/ unloading." But also, other tasks such as "park in the charging dock," "let yourself be controlled remotely," or "do nothing, wait for new tasks" must be defined in the fleet management control. Two ROS nodes were created to achieve the required behavior reasonably, one of which is part of the server - it stores and distributes tasks for individual robots. The second ROS node runs on individual robots - it receives the assigned tasks from the server and then assigns them to the robot's control elements.

The fleet management control must also provide some necessary control actions over the entire system or even individual robots. Such activities can be useful in debugging system configuration and security risks analysis or unexpected situations. ROS itself offers a CLI interface [32][33] for calling ROS services and topics, with which the required actions in the system are performed. Nevertheless, even a seemingly simple action can be a set of large calls to various processes in a multi-robot system. Therefore, some necessary control actions were implemented in the GUI (Fig. 8), thanks to which the operator can relatively easily and comfortably perform basic operations in the system. From the available ROS packages, Gmapping [35][36] and AMCL [37][38] were used for the self-localization of individual robots in space.

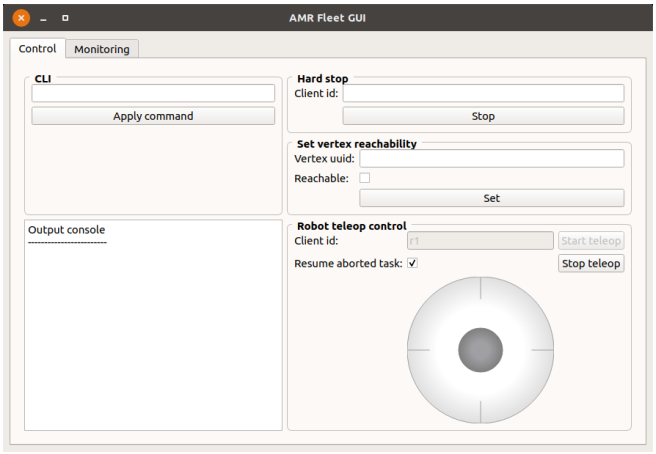

Fig. 8. Simple control interface (GUI)

V. IMPROVEMENTS IN THE CONCEPT

After several experiments, a weakness in our solution was identified in using one-way paths, due to which robots are unable to reach some parts of the environment. For example, if a narrow place in the environment does not allow to create of two one-way paths next to each other, the area behind such a location will not be reachable. Fig. 9 shows this situation. It is evident that it is possible to get to the target point, but if this path is one-way, the robot will not get back. This may be a real case when the robot must transport material closer to a line or machine where the possibility of maneuvering is reduced.

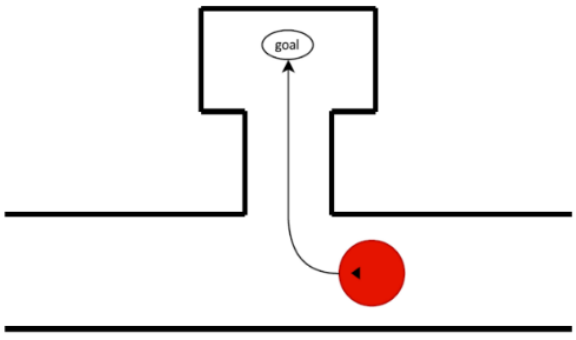

Fig. 9. A situation in which the target station is unreachable due to the narrow space

This issue was solved by introducing bidirectional paths. Still, it will need a completely different planner type because a simple search for the shortest path in the graph is not enough to find the right combinations of paths. So that the robots do not get into collisions and deadlocks. This has been done by not using bidirectional paths on the whole map but only where necessary (Fig. 10). The correct individual paths without collisions were achieved by the synchronization using the semaphore. The messages used for the communication and graphical interface were also modified.

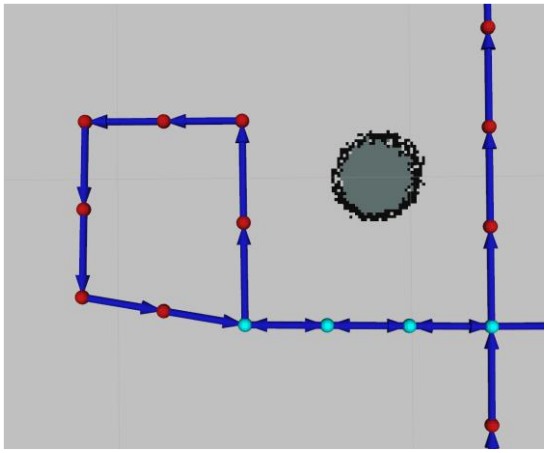

Fig. 10. Graphical representation of bidirectional paths in a graph

The sync semaphore has a crucial role in our approach. It coordinates the execution of individual paths so that they are without any collisions, even in bidirectional paths. It also ensures that the robots do not block each other in areas with bidirectional paths. To prevent two robots from meeting each other on one road, we must ensure that there is always only one robot on a bidirectional path at a given moment. Therefore, if the robot requests to lock a node from the bidirectional path, all other vertices lying on that path will be locked automatically. So, another robot will not be able to enter these nodes. However, in order not to unnecessarily prevent the movement of robots that are aimed in the same direction, we gradually unlock the nodes that have already been reached by the given robot. This principle is shown in Fig. 11 (where the robot, due to its physical dimensions, permanently maintains two locked nodes). Suppose the robot's path does not point through a bidirectional path but only passes through a node that is part of a bidirectional path. In that case, all nodes belonging to the bidirectional path do not unnecessarily lock (only the node through which the robot passes is locked).

The way of locking the nodes, where two robots could meet each other, does not occur. But there are different critical situations. For example, such a problem arises when all the bidirectional nodes from one path are locked, even those on which the robot is not currently located. Then it may happen the second robot, whose path does not lead through a bidirectional path, needs to enter one of the blocked nodes (Fig. 12, blue robot). Because his next node is blocked and not allowed to enter, it must wait for the red robot to pass through. Therefore, unless the red robot is immediately in front of the node, the blue robot will lock this node. At this point, the one who requested the lock last will have the right to enter it. Thanks to this, the blue robot can enter this critical section. If necessary, the red robot waits until the blue leaves.

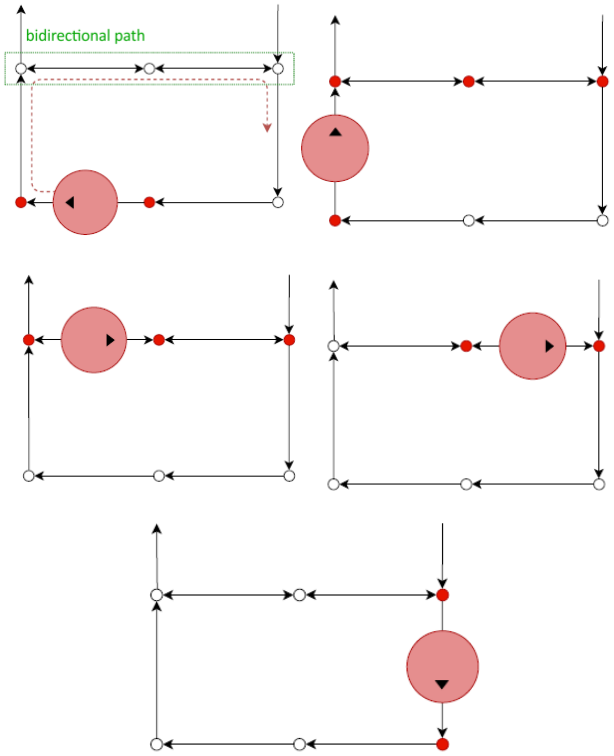

Fig. 11. The principle of locking the nodes on bidirectional paths (small red dotes are blocked nodes)

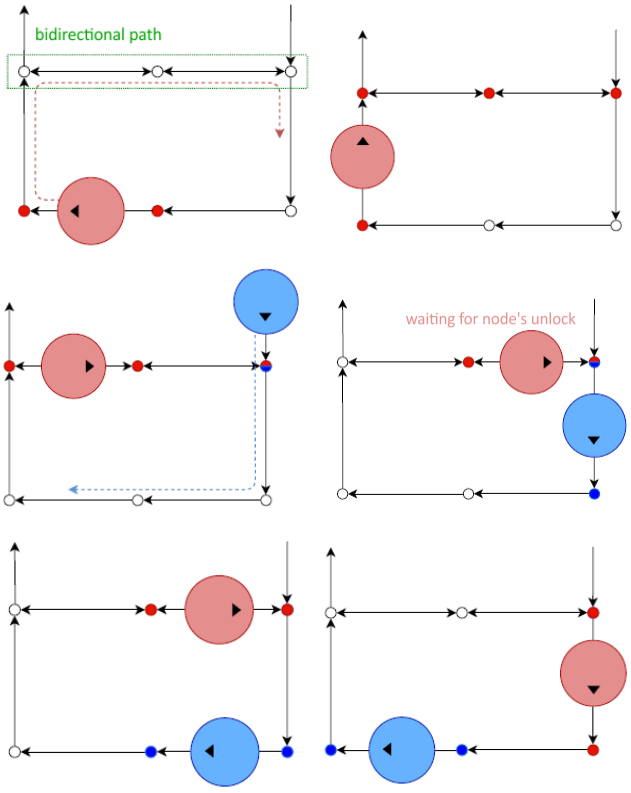

Fig. 12. The principle of locking the nodes on bidirectional paths in a situation with two robots

However, there is another critical situation. If a bidirectional path leads to a closed circle path, after the meeting of several robots, a deadlock can occur in the given area (Fig. 13)

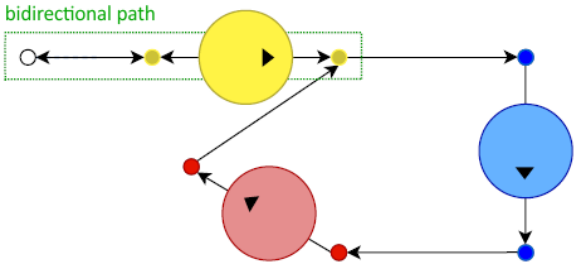

Fig. 13. Dead-lock of robots in the area, which was created by the introduction of bidirectional paths

A new framework in the semaphore was created to avoid this situation. It evaluates the locking of nodes within defined areas. The user specifies the area through the configuration file by naming the nodes belonging to the area and the maximum number of robots allowed in the area. It is also possible to generate such a configuration by a suitable algorithm based on the generated graph. If the maximum number of robots occupies a given area and another of the robots requests the node's locking belonging to this area, this locking will be rejected. So, the robot will have to wait until the number of robots in the area decreases.

Generally, by introducing bidirectional paths, the fleet's operation is more effective, and above all, it allows to move robots through narrow corridors and into dead ends. However, it should be noted that a significant part of the graph should be formed from one-way paths, and the bidirectional paths should be used only in necessary cases.

Suppose the individual robots in the system have different speeds, for example, because they are fully loaded with material or empty. In that case, situations arise where the slower robot gets ahead of, the faster one. The side effect of this situation is that the faster robot must slow down to the slower one's speed. As a result, the overall possible usability of the system also decreases. The faster robot will allocate a longer part of the path in front of it to eliminate this effect.

Consequently, the probability that a slower robot will get in front of a faster robot is reduced. The data variable storing the path length that the robot can allocate in front of it is easily parameterizable. It can be changed during operation according to the current speed of the robot.

Finally, the principles of creating the routes for the fleet management are summarized as follows:

- Bidirectional paths should only be used when necessary. And if they are used, they should be as short as possible. This is because there are extensive downtimes in front of bidirectional paths, where the robot waits for the other robot to leave this path.
- It is vital to avoid logical errors, such as the wrong combination of road directions at an intersection. Suppose the robot must remain for some time in the station, necessary for loading or unloading material. In that case, it is required to place this station off the main path. By creating a detached edge for the station, the standing robot will not block other robots that do not need to stop at this station.

VI. RESULTS

The proposed solution should be compared with other available solutions for robot fleet management. However, because the solution is unique, it is impossible to compare it directly quantitatively with another. Therefore, we will try to evaluate the system based on a series of simulation tests. For this reason, the experimental environment was created. In the map, bidirectional paths, one-way paths, areas with condensed intersections, and fewer crossings were created. We tried to place the stations on the map so that the paths' usability was evenly distributed. In Fig. 14, the stations are shown by white circles. The robots do not stay in the stations in the simulation setting, but after reaching them, the robot immediately points to the next station.

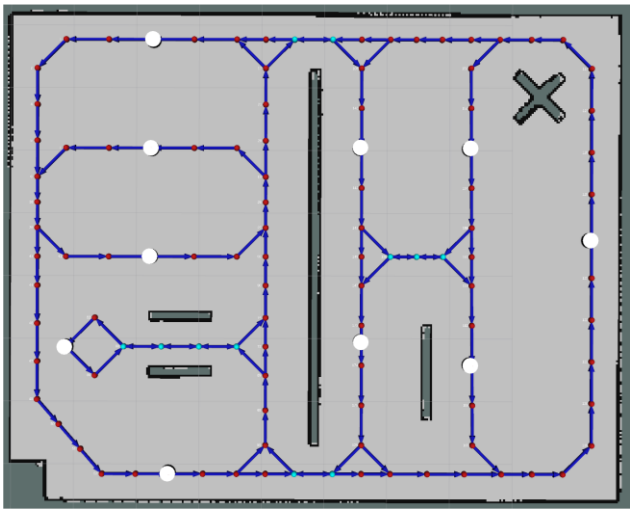

Fig. 14. Map of simulation environment with paths and stations (width 7.8 m, length 9.8 m, the total length of roads 71.81 m, the total number of nodes in graph 110, max. length of graph edge 0.7 m, number of stations 10)

In the first test, path synchronization was turned off, and the supervisor system was limited only to path planning and task assignment. Collisions between robots begin to occur, so their effects in the physical model were turned off. Although such a setting is not very informative about the system's final performance, it will nevertheless serve as a useful reference for comparison with other results. This test will also show a kind of "ideal state," where there are no unnecessary downtimes, and the potential usability of individual robots is up to 100 percent. Since we also want to evaluate the impact of improvements that consider the speed of robots, the speed limit to divide the robots into two groups was used: faster and slower. The Table 2 shows a list of system parameters for each simulation.

TABLE II. SIMULATION PARAMETERS

Simulation	No. 1	No. 2	No. 3
Number of robots	5/5	5/5	5/5
Maximal speed (m/s)	0.2/0.5	0.2/0.5	0.2/0.5
Sync semaphore	Off	On	On
Length of allocated path (m)	-	0.7/0.7	0.7/0.25

The results for the individual simulations can be found in Figs. 15 and 16. The graphs show the monitored variables in the time range of 30 minutes. As logically expected, the introduction of path synchronization led to a decrease in robot mobility, and thus a reduction of overall utilization is observed. This decrease is approximately 25 to 29 percent in the specific map used.

On the other hand, 266 mutual collisions of robots were averted. Let's compare experiments 2 and 3 (Table 2). Another impressive result is that the improvement in allocating a longer section of path ahead, with respect to the speed of the robot, did not have any significant effect on the system's usability. Improved functionality does not bring the desired result because the faster robot gets priority at the intersections, but the slower one has to stop and wait for the faster robot. In the worst case, if there is another robot behind the slower robot, suddenly we have two standing robots, which results in an even worse result than in the case without this improvement.

Series of experiments were performed, increasing the speed difference between a fast and a slow group of robots. For the quicker group, the speed limit to 0.7 m/s was set, and all other parameters were at the previous values. Although a slight improvement over the last series of experiments can be seen, it has again been shown that this improvement has no dramatic effect on system usability. In the case of environments where long roads predominate over areas with intersections, a more considerable difference in the system's usability can be observed. This is because situations where a faster robot is blocked, prevail over situations where a slower robot is waiting at an intersection.

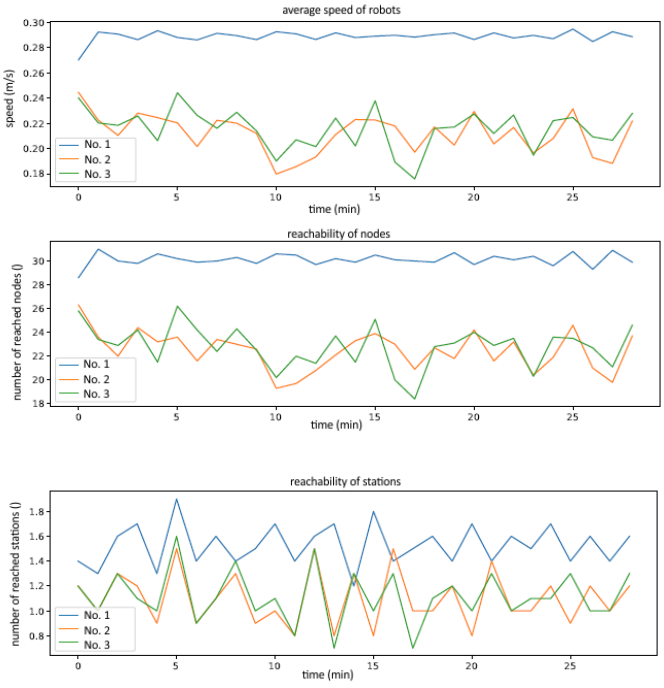

Fig. 15. Monitored variables in the experiments

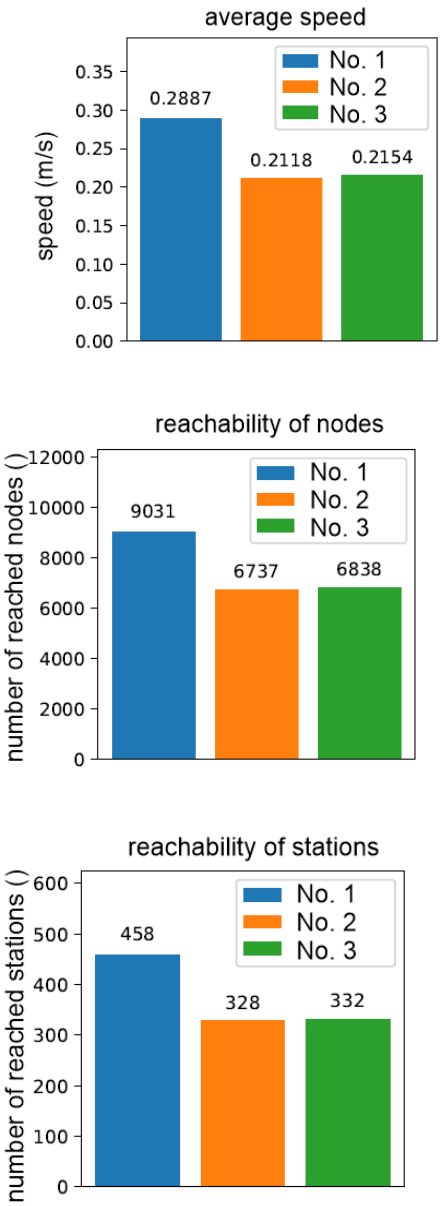

Fig. 16. Average values of robot speeds, reachability of nodes, and reachability of stations

During the development of the entire system, the burden on the communication network has been minimized. The unnecessary transfer of transformations (ticopic/tf) between the robots and the server was reduced. To clarify: ROS topic is peer-to-peer communication [34]. If individual robots send their status information on the topic and to servers, two processes (monitoring and task manager) need to take care of that message. Thus, this creates identical two data flows through the network. This redundancy was prevented by using the topic_tools/relay tool, which runs on the server, reads the requested topic sent from the robots, and forwards it to another topic with an identical message type. Forwarding already takes place within the server, so it does not load the network in any way. We performed four measurements, using 1, 3, 5, and 10 robots in the same map as in the previous case, and we used the same maximum speed of 0.5 m/s for all robots. The data flow was measured using Wireshark software, the simulated robots were run on a different computer than the server (Fig. 17 and 18).

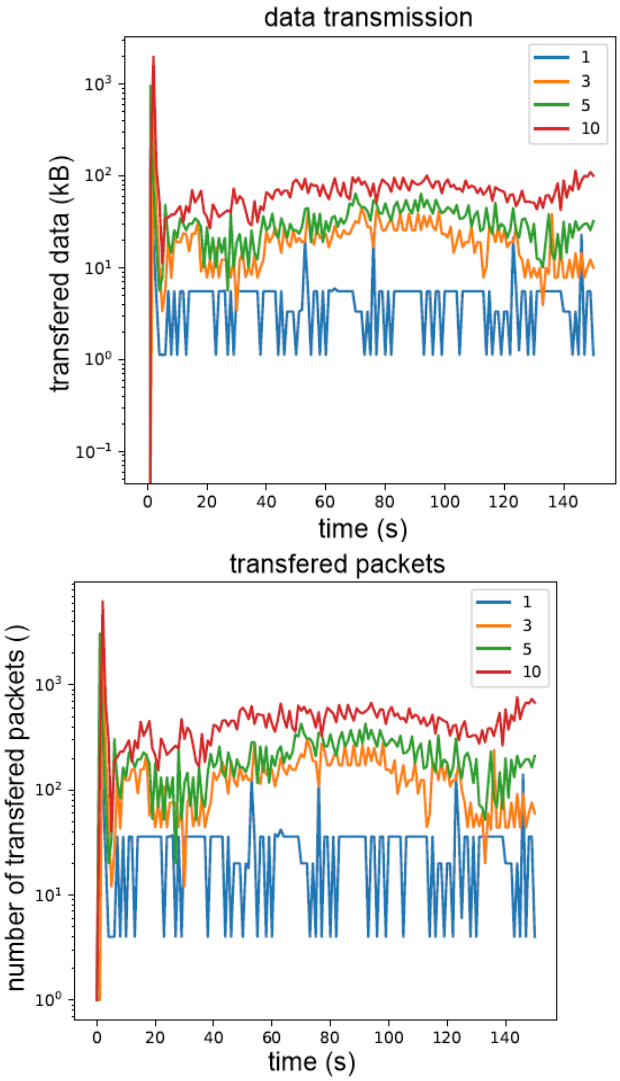

Fig. 17. Data flow between the server and the robots over time for different numbers of used robots

Based on the results, it can be stated that no large data is sent between the robots and the server. Likewise, the number of packets is tolerable for lower categories of network elements. Another useful fact is that the gradual addition of robots to the system does not increase the data flow exponentially. A higher increase in flow can be observed at the start of the system because a map intended for localization is distributed between the robots. Moreover, initialization calls are also made. We also tried to simulate a connection failure between the robot and the server by hard disconnecting and reconnecting the network connection. The system managed it without any problems. After disconnecting the network connection, the robot reached the allocated part of the path, stopped, and waited for the server's connection and permission to allocate another part of the path in front of it. After reconnection, communication was restored (thanks to the ROS communication's correct design), and the system could continue to run. The only re-establishment criterion is to preserve the original use of IP addresses for the server and robots

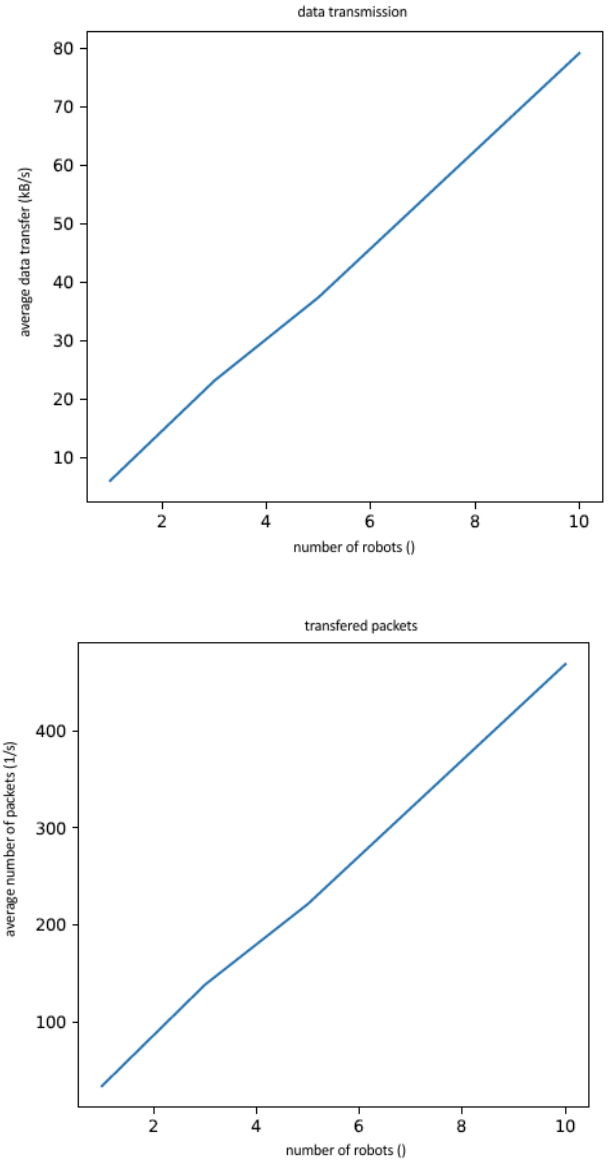

Fig. 18. Average data flow depending on the number of robots

VII. CONCLUSION AND FUTURE WORK

Our research's main goal was to analyze and implement fleet management for a group of robots. Although some methods are available in the ROS environment, none of them met our criteria for managing a fleet of robots. Each available technique has its own limitations and different advantages. Due to the identified issues, we proposed own approach for the fleet management. We decided to use a simple planner in combination with the correct synchronization of path execution. This is a significant advantage of our approach, as it ensures a fast and reliable synchronization and at the same time a collision-free solution for a group of robots. This was achieved also thanks to the use of bidirectional paths in critical areas, giving our solution a new dimension. Thanks to this, we can cover almost all cases of using a group of robots in real industrial environments. Unlike other existing solutions, ours differs mainly in that it has a separate planning process from synchronization – combination of semi-autonomous and centralized approach. Thanks to this, we avoided using too complex, inflexible algorithms for creating synchronous plans. So our solution is not only easy to use, but compared to other solutions, it also has low network and hardware requirements. Moreover, another advantage of our approach is its modularity and it can be modified and enhanced in different ways (e.g., methods other than standard A* can be used for local planning [39][41][42] or the solution can be extended to 3D for the usage of drones in industrial environment [40]).

Our next goal for future work is to verify this system in a real industrial environment. However, it is still necessary to complete many implementation details, which were not included in this work. These are mainly the details of a more friendly GUI, enabling convenient setting and control of individual operator's interventions and the entire system's configuration options to adapt to a wide range of real control requirements quickly. It would also be possible to improve the planning process by adding additional data because finding the shortest path at the same time does not mean that this path is also the fastest.

ACKNOWLEDGMENT

This work was supported by company Photoneo and research projects VEGA 1/0754/19, VEGA 1/0775/20, and VEGA 1/0599/20.

REFERENCES

[1] B. Binder, "Spatio-temporal prioritized planning," in Ph.D. Thesis, Wien, 2017.

[2] D. Claes et al., "Collision avoidance under bounded localization uncertainty," in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1192-1198, 2012.

[3] W. Guan, S. Chen, S. Wen, Z. Tan, H. Song and W. Hou, "High-Accuracy Robot Indoor Localization Scheme Based on Robot Operating System Using Visible Light Positioning," in IEEE Photonics Journal, vol. 12, no. 2, pp. 1-16, April 2020, Art no. 7901716, doi: 10.1109/JPHOT.2020.2981485.

[4] X. Li et al., "A review of industrial wireless networks in the context of industry 4.0," in Wireless networks, pp. 23-41, 2017.

[5] Library with search algorithms for task and path planning for multi robot/agent systems. [online]. Available on: https://github.com/whoenig/libMultiRobotPlanning.

[6] A. Mannucci et al., "Provably safe multi-robot coordination with unreliable communication," in IEEE Robotics and Automation Letters, pp. 3232-3239, 2019.

[7] F. Pecora et al., "A Loosely-Coupled Approach for Multi-Robot Coordination, Motion Planning and Control," in ICAPS, pp. 485-493, 2018.

[8] V. Vavrík et al., "The design of manufacturing line configurations with multi-agent logistics system," Transportation Research Procedia, pp. 1224-1230, 2019.

[9] Hajduk et al., "Developing new behavior strategies of robot soccer team SjF TUKE Robotics: Category MiroSot," in International Journal of Advanced Robotic Systems, 2016, doi: 10.1177/1729881416663670.

[10] M. C., Lucas-Estan, "Emerging trends in hybrid wireless communication and data management for the industry 4.0," in Electronics, 2018, doi: 10.3390/electronics7120400.

[11] R. Jánoš and M. Sukop, "Modular robots on multiagent principe," in Глобальне управління та економіка, pp. 57-60, 2015.

[12] A. Janota et al., "Functional Behavior of Traffic Control Systems Learned Through Multiagent Systems," in International Workshop on the Educational Uses of Multi-Agent Systems (EduMAS), 2009.

[13] J. Adam, "What is the V-model approach to software development and testing?" [online]. Available on: https://kruschecompany.com/v-model-software-development-methodology/

[14] E. A. Oyekanlu et al., "A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5G-based smart manufacturing applications," in IEEE access, pp. 202312-202353, 2020, doi: 10.1109/ACCESS.2020.3035729.

[15] G. Fragapane et al., "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," in European Journal of Operational Research, pp. 405-426, 2021, doi: 10.1016/j.ejor.2021.01.019.

[16] A. Liaqat et al., "Autonomous mobile robots in manufacturing: Highway Code development, simulation, and testing," in The International Journal of Advanced Manufacturing Technology, pp. 4617-4628, 2019, doi: 10.1007/s00170-019-04257-1

[17] A. Singhal et al., "Managing a fleet of autonomous mobile robots (AMR) using cloud robotics platform," in European Conference on Mobile Robots (ECMR), pp. 1-6, 2017, doi: 10.1109/ECMR.2017.8098721.

[18] B. Kuipers et al., "Local metrical and global topological maps in the hybrid spatial semantic hierarchy," in IEEE International Conference on Robotics and Automation, pp. 4845-4851, 2004, doi: 10.1109/ROBOT.2004.1302485.

[19] P. Neher et al., "Identification and Classification of the Communication Data of Automated Guided Vehicles and Autonomous Mobile Robots," in 8th International Conference on Automation, Robotics and Applications, pp. 68-75, 2022, doi: 10.1109/ICARA55094.2022.9738572.

[20] B. Binder et al., "Multi robot route planning (MRRP): Extended spatial-temporal prioritized planning," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4133-4139, 2019, doi: 10.1109/IROS40897.2019.8968465.

[21] H. Dong et al., "The path planning for mobile robot based on Voronoi diagram," in Third International Conference on Intelligent Networks and Intelligent Systems, pp. 446-449, 2010, doi: 10.1109/ICINIS.2010.105.

[22] D. Hennes et al., "Multi-robot collision avoidance with localization uncertainty," in AAMAS, pp. 147-154, 2012.

[23] H. Ma et al., "Overview: A hierarchical framework for plan generation and execution in multirobot systems," in IEEE Intelligent Systems, pp. 6-12, 2017, doi: 10.1109/MIS.2017.4531217.

[24] F. Pecora et al., "A loosely-coupled approach for multi-robot coordination, motion planning and control," in Twenty-eighth international conference on automated planning and scheduling, 2018.

[25] J. Conesa-Muñoz et al., "Distributed multi-level supervision to effectively monitor the operations of a fleet of autonomous vehicles in agricultural tasks," in Sensors, pp. 5402-5428, 2015, doi: 10.3390/s150305402.

[26] P. Forte et al., "Online task assignment and coordination in multi-robot fleets," in IEEE Robotics and Automation Letters, pp. 4584-4591, 2021, doi: 10.1109/LRA.2021.3068918.

[27] M. Berndt et al., "Centralized Robotic Fleet Coordination and Control," in Mobile Communication-Technologies and Applications; 25th ITG-Symposium, pp. 1-8, 2021.

[28] A. Renawi et al., "ROS validation for non-holonomic differential robot modeling and control: Case study: Kobuki robot trajectory tracking controller," in 7th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO), pp. 1-5, 2017, doi: 10.1109/ICMSAO.2017.7934880.

[29] N. Gallardo et al., "Formation control implementation using kobuki turtlebots and parrot bebop drone," in World Automation Congress (WAC), pp. 1-6, 2016, doi: 10.1109/WAC.2016.7582996.

[30] F. Duchoň et al., "Path planning with modified a star algorithm for a mobile robot," in Procedia Engineering, pp. 59-69, 2014, doi: 10.1016/j.proeng.2014.12.098.

[31] C. Iordache et al., "Smart Pointers and Shared Memory Synchronisation for Efficient Inter-process Communication in ROS on an Autonomous Vehicle," inIEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6441-6448, 2021, doi: 10.1109/IROS51168.2021.9636018.

[32] B. Dieber et al., "Penetration testing ROS," in Robot operating system (ROS), pp. 183-225, 2020, doi: 10.1007/978-3-030-20190-6_8.

[33] S. Adam and U.P. Schultz, "Towards interactive, incremental programming of ros nodes," in arXiv preprint arXiv:1412.4714, 2014.

[34] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open source software, pp. 1-6, 2009.

[35] Y. Abdelrasoul et al., "A quantitative study of tuning ROS gmapping parameters and their effect on performing indoor 2D SLAM," in 2nd IEEE international symposium on robotics and manufacturing automation (ROMA), pp. 1-6, 2016, doi: 10.1109/ROMA.2016.7847825.

[36] R. K. Megalingam et al., "ROS based autonomous indoor navigation simulation using SLAM algorithm," in International Journal of Pure and Applied Mathematics, pp. 199-205, 2018.

[37] W. P. N. D. Reis et al., "An extended analysis on tuning the parameters of Adaptive Monte Carlo Localization ROS package in an automated guided vehicle," in The International Journal of Advanced Manufacturing Technology, pp. 1975-1995, 2021, doi: 0.1007/s00170-021-07437-0.

[38] W. P. N. D. Reis et al., "A quantitative study of tuning ros adaptive monte carlo localization parameters and their effect on an agv localization," in 19th International Conference on Advanced Robotics (ICAR), pp. 302-307, 2019, doi: 10.1109/ICAR46387.2019.8981601.

[39] I. Hassani et al., "Turning Point and Free Segments Strategies for Navigation of Wheeled Mobile Robot," in International Journal of Robotics and Control Systems, pp. 172-186, 2022, doi: 10.31763/ijrcs.v2i1.586.

[40] X. Wei-hong et al., "Review of Aerial Manipulator and its Control," in International Journal of Robotics and Control Systems, pp. 308-325, 2021, doi: 10.31763/ijrcs.v1i3.363.

[41] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," in international journal of physical sciences, pp. 1314-1320, 2012, doi: 10.5897/IJPS11.1745.

[42] M. N. Zafar and J. C. Mohanta, "Methodology for path planning and optimization of mobile robots: A review," in Procedia computer science, pp. 141-152, 2018, doi: 10.1016/j.procs.2018.07.01.