DAI-Labor
Distributed Artificial Intelligence Laboratory
GT ARC
GERMAN TURKISH
ADVANCED RESEARCH CENTRE FOR ICT

A|O|T

System Design Specifications of Autonomous Warehousing: Multi-Robot Planning and Navigation

CONFIDENTIAL!

Applications of Robotics and Autonomous Systems

Team Members: David Mezey, Fynn Boyer, Jingsheng Lyu, Ricardo Gabriel Herdt, Sefika Efeoglu

Table of Contents

1 Introduction

As mobile robot technology becomes more and more affordable warehouses around the world utilize the possibility of automation using groups of robot-agents. The expansion of internet based trading in general imposes several challenges to guarantee an efficient distribution of products. In some cases such automation is crucial to keep the quality of goods high while keeping the costs of daily business low. As soon as orders arrive to a warehouse around the clock, automation becomes a necessity to make same-day delivery a widespread reality. Such example can be a warehouse of groceries and green goods [22] where the precise timing of the delivery from the order process has a vast influence on customer experience. The phenomenon that numerous providers (Walmart [23], DHL [24], Amazon [25]) have already started to use autonomous robot systems (ARS) in their warehouses is paving the way to a widespread and general [26] usage of applied robotics to optimize available resources such as used warehouse space and manpower in each segment of warehousing from order to delivery.

1.1 Motivation and Problem Definition

Currently developed fully automated autonomous warehouse solutions are mostly using a *grid-like floorplan* to operate a large number of mobile robots to find ordered articles and move them to a packaging area efficiently. Furthermore the exact plan of which task belongs to which robot-agent is *planned offline* meaning that the plan a robot-agent is complying to might have been planned several hours or even a day before. In these cases, unexpected changes in the environment, such as robot-failures, accidents, package-failures, etc. may lead to a full shutdown of the automated system until the source of the failure has been removed (usually by human manpower). Such warehouse outages may lead to many hours of delay in package-delivery and a complete recomputing of the offline task plans.

To further optimize efficiency and increase robustness against such scenarios it might be useful to operate robots online and remove the constraint of a grid-like space in which the robot-agents can move. As such, robot-agents shall have the possibility to either autonomously handle failure situations, or the central planning system shall handle the failure on the fly without the need of a shutdown and external human interference.

The current project follows an approach, where the planning of the tasks in the warehouse happens online (i.e. the tasks are dynamically assigned as they arrive to the facility) as well as robot-agents may travel on arbitrary paths in space. We suggest that this framework is possibly achieving an overall better performance (measured for instance in terms of the rate of delivery of packages) and higher robustness w.r.t. handling unforeseen scenarios.

In our model warehouse packages are being transported from designated pick-up areas to assigned package sockets by multiple robot-agents. Generating the orders/packages and planning which socket they must be transported to is out of the scope of our project and is already implemented. The floorplan and the exact position of environmental elements such as pick-up areas, package sockets, charging stations are given. Our goal is to implement a solution to assign continuously arriving orders/tasks to agents and to coordinate the motion of these robot-agents to fulfil the assigned tasks. Our goal is to implement a solution that solves the following problems:

1. **Task assignment:** Upon announcement of a new task/package a robot needs to be chosen according to its current position, current tasks and battery state. “Task assignment” includes the problem of defining and prioritizing the available information about robot-agents to choose the best (according to a well-defined metric) robot-agent for the task.

2. **Path planning:** Given the assigned tasks, the system shall compute for each robot an appropriate path to fulfill the task (i.e. to transport the package of interest from a pickup area to its goal location). The problem of "Path planning" is to plan such paths in a way that no robots collide with each other or with environmental obstacles. Furthermore, a part of the problem is to update paths in unforeseen situations so that they can be solved efficiently on the fly (such as in deadlock situations, robot failures, etc.). According to the architecture to be implemented these can be further detailed as follows:

   a. **Centralized case:**

      The computation of all paths is done by a central component that has access to global information. This approach enables the system to compute (close to) optimal solutions for the problem, but usually suffers from complexity issues.

      Centralized approaches can be further classified as:

      - **Coupled:**

        In this case, a compound state of all robot configurations is considered while computing paths. Centralized coupled algorithms can achieve optimality and completeness (i.e. an existing solution is guaranteed to be found), but due to the complexity of the underlying problem these solutions doesn't scale.

      - **Decoupled:**

        In a decoupled approach, paths are computed incrementally. Thus the problem is solved by decomposing the global solution in subproblems. For instance the path of each robot can be considered in turn, using a single-robot path planning algorithm for each instance. Decoupled algorithms usually provide efficient solutions, but aren't complete if an optimal solution is needed.

   b. **Decentralized case:**

      In a decentralized system the robots plan their paths without the help of a central unit. This decreases computational load and allows local handling of uncertainty but also gives rise to more conflicts and the chance for deadlocks. In order to resolve any occurring conflicts the robot need communication protocols to negotiate solutions or traffic laws that guide them. These measures must ensure that the system will run stable without breakdown that might be caused by undefined conflict cases or the inability to find a solution.

3. **Charging management:** Using multiple robot-agents to solve dynamically assigned tasks give rise to the problem of "Charging management". Namely, the battery status of robot-agents must be considered during task assignment as well as robots with low battery level should be guided to charging stations. The problem of "Charging management" includes defining low charge levels, handling the task assignment of such robots and when needed guiding these robots to find the closest available charging station. How do we choose which charging station shall be used and how long specific robots should charge are questions also belonging to the problem.
4. **Cross Track Error / Low level motion:** For the Low-Level-Task, we should focus on how the robot can smoothly move to the goal point. In addition, the robot should avoid all obstacles as fast as possible. However, in actual situations, the autonomous robots often cannot follow the expected trajectory. There is an error between the theoretical value and the real value. Our task is to minimize this error as possible and to make the automatic driving smoother. This error that still exists as a constant offset in the cross track is due to a steering bias in the vehicle. For this control, we can calculate the feedback gain of the PID controller. The cross track error influences the commanded curvature. Change the commanded curvature by changing the PID controller coefficient of this cross track error.

1.2 Objectives

The project takes place in the following scenario: A factory is to be fully automatised where upon a new online order from a customer the product of interest is being produced from raw material with some methods. The products are then being packaged and the packages are transported to specific end stations from which they are being delivered to the customer. Our objective is to realise one part of this scenario, namely transporting the packages (including goods) from the end of the packaging process to the end stations so that they can be delivered. In this framework our solution receives the following inputs and sends the following outputs in the overall scenario.

Inputs:

1. Ready-made packages are appearing on the end of the packaging station in package sockets in a state that they can be transported with any of the robot agents.
2. With a given package our solution receives information about where (in which end station) should it be transported from the central Task PLanner of the factory.
3. A physical environment is implemented from which the robot-agents are able to gather sensory information.
4. The physical environment includes charging stations that are able to charge a robot-agent when it is in the direct neighborhood of the charger.

Outputs:

1. The solution transports the appearing packages from the end of the packaging station to the assigned end stations.
2. The end result is that arbitrarily chosen packages can be delivered together and the output of our solution serves as an input to the delivery system.

Our detailed objectives to realize the solution are as follows:

1. Implement a “Task assignment agent” (TAA) to find the optimal robot-agent for a new task/package given by the “Task planner”. The TAA shall assign a new task to a robot-agent according to well-defined criteria such as:
    a. Battery life
    b. Package/task/robot priority
    c. Distance (with some metrics) from the start position
    d. Additional criteria if needed[1]
2. Implement new/Use existing method to plan the path of a single robot according to an assigned task. The algorithms chosen to plan the path of a single robot agent (or of multiple robot agents if applicable to the algorithm and architecture) include but are not limited to:
    a. Centralized architecture:
        i. Push-and-Swap or Push-and-Rotate multi-agent algorithms for fast offline path planning.
        ii. Heuristics to deal with imprecisions of the system (for instance mismatch between current positions of agents and their planned paths).
        iii. Efficient recomputation strategies in case of collisions and detected obstacles (possibly computing only paths of affected agents). Replanning of all paths shall be avoided.
    b. Decentralized architecture:
        i. A* / Theta* for local, independent path planning based on the roadmap & dynamic obstacle information.
        ii. Push and swap or Push and rotate for local central planning with completeness guarantee and scalability in case of a growing local network caused by deadlocks / collision propagation (one collision avoidance movement causes another collision).
    c. Hybrid architecture:
        i. A* / Theta* for local, independent path planning based on the roadmap, dynamic obstacle information, learned previous traffic flows and expected paths of other robots.

[1] Additional criteria: all information about a robot-agent which can be used to optimize the TAA. The chosen criteria and their definition might be a subject of change throughout the project.

   ii. Push and swap / Push and rotate for local central planning with completeness guarantee and scalability in case of a growing local network caused by deadlocks / collision propagation (one collision avoidance movement causes another collision).

3. Implement “Motion planning” (MP) of a single robot, i.e. the ability of a robot-agent to navigate on a given planned path “smoothly” and to handle the package of interest according to its start or end position. In other words, include low-level functionalities to the solution. MP might include the following subtasks:
   a. Use an efficient trajectory-following strategy to continue along the given trajectory closely both in the time and space dimension to ensure collision-avoidance and estimated arrival times.
   b. Calculate the cross-track error to steer the wheels via a PID controller.
   c. Be able to execute an emergency stop in case an obstacle gets too close.
4. Include “Obstacle detection” (OD) in a single robot-agent which is able to:
   a. Use the sensors of the robot-agents to detect and differentiate between obstacles of the environment and other robot-agents. In the latter case the ID of the other robot-agent shall be collected (e.g. for communication purposes).
   b. Make decisions about the “riskiness” of the situation, either by collecting path data from a central source, other robots in the vicinity or by interpolating the movement of the obstacle (if applicable).
   c. Communicate to MP to avoid collisions in unforeseen situations (caused by an obstacle)
5. Extend point 2.) to multiple robots if not yet extended (depends on the chosen algorithm and architecture).
6. Implement “Charging management” (CM) and connect to “Task assignment” (TAA). The following aspects shall be implemented:
   a. CM should be able to collect information about the battery level of robot-agents and act accordingly.
   b. CM shall be able to interact with TAA in a way that the risk of a robot-agent fully running out of battery is minimized.
   c. CM should dynamically handle the prioritization of low-charged robot-agents (if there are many and applicable).
7. (Optional) Include “Traffic planning” (TP) which can bias the planned paths according to the estimated traffic. TP is one of the following or a mixture/composite solution of the following:
   a. Low resolution estimated traffic maps in 3D (X-Y-t) or 4D (X-Y-t-p) to keep the level of traffic per unit space under a threshold value.
   b. Learned flow field of the environment can be incorporated during path planning to minimize the possibility of risky states. [7]

1.3 General Constraints

1. Communication between robots and other systems is loss-free, sufficiently fast and always available. If a robot runs out of energy or breaks down, communication will not be possible anymore though.
2. Computational resources are available in the magnitude that could be expected in the context of each system.
3. Robots can move freely through the environment without any limitation (e.g. by a network of lanes).
4. Tasks appear online at random times. There is no task schedule created at a previous point.
5. Robots carry a sufficient set of sensors to observe their environment, especially in front of them.
6. The expected throughput of the system is attainable if it runs efficiently. Factors that could be limiting include the number / attributes of the robots (e.g. speed, battery size, ...), the layout of the warehouse and the number of charging stations.

2 Use cases / scenarios

Hereby we define the main use cases of the given scenario. Please note, that according to the architecture we implement these use cases change and therefore we distinguish between Centralized (C) and Decentralized (D) use cases when it is applicable.

Use Case 1: A new package is ready to be transported

Primary Actor(s): Package Management, Task Planner

Details: Upon a new package has been created the task is to transport this new package from the end of the Packaging zone to the Delivery zone. According to the given package a task is generated for the robot-agents to transport the package from its start to its end position.

Main Success Scenario: Upon a new package appears a task is being generated to transport this package with a valid start and end position.

Use Case 2: A new task is ready to be assigned

Primary Actor(s): Task Assigner, Agents, Charging management

Details: Upon a new task has been generated (namely to transport a package from a starting position to an end/goal position in the environment) it needs to be assigned to a robot-agent to be completed. For this reason the Task Assignment Agent (TAA) collects useful information about the available robot-agents, such as current distance from the start position, battery level, etc. The TAA then assigns the task to the "best" available agent and stores this now assigned task in a que for the agent.

Main Success Scenario: Upon a new task is generated the TAA is able to find the optimal robot-agent for the task and to assign the created task to it.

Possible Misuse case: The TAA either doesn't find the best available agent for the task or it distributes the tasks in a way that merely the task assignment causes problems in other Use Cases.

Use Case 3: Path Generation

Use Case 3.a: Path generation according to an assigned task (Centralized, C)

Primary Actor(s): Path Planner

Details: Upon assignment of a task, a path has to be computed for the corresponding robot, such that it can move collision-free to it's target. The corresponding path is communicated to the agent that starts moving along it.

Main Success Scenario: The robot arrives at its target location without collision and in a close to optimal time frame.

Possible Misuse case: Due to imprecise localisation information and varying velocities agents could collide (in which case they could stop based on a collision avoidance strategy). The frequency of these conflicts could force several path recomputations, leading to long delays where agents have to wait in order to move further.

Use Case 3.b: Path generation according to an assigned task (Decentralized, D)

Primary Actor(s): Agent, (Traffic Planning)

Details: After a task has been assigned to a robot, it has to plan a path to reach its target point. This path has to avoid any static obstacles. Avoiding areas with a high expected traffic volume should also be avoided if possible.

Main Success Scenario: The robot will use information about its current pose, the target pose and the static obstacle map to generate a path with a simple algorithm like A* or Theta*. If a traffic management system is in place, the robot will use available traffic data to optimize its path. Once a path was planned it will be provided to the traffic management to guide other robots in their planning process.

Possible Misuse case:

- If information about the current pose is incorrect, an invalid path will be generated.
- If the traffic predictions by the traffic management are not reliable the path optimization will return a meaningless result.

Use Case 4: Robot movement is smooth and reliable

Primary Actor(s): Agent

Details: Whenever a robot-agent is moving the movement is "smooth" in space and reliably predictable.

Main Success Scenario: There are no unforeseen jitters in the movement of the robot. The movement is precisely controllable and predictable.

Possible Misuse case: Jitters in the movement causes the robot-agent not to be able to follow its path smoothly.

Use Case 5: Robot-agent position data is reliable

Primary Actor(s): Agents, Motion Planning

Details: During all possible movement of a given robot-agent any distributed data about its position is reliable and precise (with some error interval) even if the collected sensory information of the robot-agent is noisy.

Main Success Scenario: The position data distributed by the robot-agent is reliable and as such it is usable for other nodes to e.g.: plan paths, monitor local and global traffic, or in decentralized case it is adequate for other agents to use it in local scenarios.

Possible Misuse case: Due to noise in sensory data and/or Motion Planning failure the position data calculated by the robot-agents is biased and therefore it is not sufficient to be used by other nodes. Unreliable sensory data pollutes path- and traffic planning and causes failures, crashes or suboptimality in general.

Use Case 6: Blocked Path

Use Case 6.a: An agent can not move further on its path (C)

Primary Actor(s): Agent, Path Planner, Obstacle Detection, (TAA)

Details: An agent is currently trying to fulfill its task which has been assigned before, moving on a path generated by path planner. The Obstacle Detection system signals that an unforeseen obstacle is in the path of the agent.

Main Success Scenario: The Obstacle detection system first identifies the obstacle. If the obstacle is a package which is identifiable and the robot-agent is available for a new task, a new task should be assigned to the robot using TAA. If it is another robot-agent, new paths are computed for both robots, possibly affecting other agents until a feasible solution is found. If the obstacle is any other environmental element which blocks the robot, the path of the robot-agent is recalculated with the Path Planner, possibly changing paths from other agents as before.

Possible Misuse case: The obstacle is not being detected which causes a collision with the obstacle, or the obstacle is misidentified.

Use Case 6.b: An agent can not move further on its path (D)

Primary Actor(s): Agent, Path Planner, Obstacle Detection, Negotiator, (TAA)

Details: An agent is currently trying to fulfill its task which has been assigned before, moving on a path generated by the agent. The Obstacle Detection system signals that an unforeseen obstacle is in the path of the agent.

Main Success Scenario: The local obstacle detection system of the robot determines the type of obstacle and acts accordingly.

- If it is another robot, communication with the other robot will be initiated in order to negotiate collision avoidance actions.
- If it is a package, the robot will publish the task to the task planner which will then assign the task to a robot.
- If it is an unknown object, the robot will adjust its planned path to avoid the object. If the robot is unable to generate a path because the obstacle obstructs the only path to the goal (e.g. package blocks a storage container) then the robot will wait in place.
- In case of the local path planner is unable to generate a path around the obstacle because it is the only way to get to the goal (e.g. package blocks storage container) the robot should wait at its current position.

Possible Misuse case:

- The negotiation between the robots concludes without a result creating a deadlock. The robots are stuck.
- Obstacles are not or falsely detected by the robot-agents.
- The robot-agent is not able to recall previous obstacles and drives in circles between different obstacles.

Use Case 7: Battery level of a robot-agent has dropped

Primary Actor(s): Agents, Charging Management

Details: During fulfilling a task the battery-level of a robot-agent has been dropped under a predefined threshold.

Main Success Scenario: The Charging Management unit is able to identify the agent with “low battery” and is further able to plan the charging process of the robot-agent accordingly. This includes changing the assigned tasks database and deciding on which charging unit and how long the robot-agent should use to charge its batteries.

Possible Misuse case: The Charging Management node is not able to identify the robot-agent on time and therefore a risky situation arises in which the robot-agent is at risk of “battery-death”.

Use Case 8: Traffic related risk is minimized

Use Case 8.a: Traffic related risk is minimized (C)

Primary Actor(s): Agents, Path Planning, Traffic Planning

Details: During fulfilling the tasks areas with high traffic in the environment arise due to the floorplan of the transportation area.

Main Success Scenario: The central traffic planner unit is able to identify areas with high robot-agent traffic and to act accordingly, i.e.: traffic should be kept under a predefined threshold. This either happens with directly modifying the path planned by the path planner unit, or rather using a recommendation system using traffic flow plans learned dynamically from previous task fulfilling cycles of the robot-agents.

Possible Misuse case: The traffic planner unit is not able to identify areas of high traffic or it fails to give an appropriate solution to reduce traffic in those areas. A possible misuse case is that traffic is reorganized as such that the situation gets worse or cyclically changing.

Use Case 8.b: Traffic related risk is minimized (D)

Primary Actor(s): Agents, Traffic Planning

Details: During fulfilling the tasks areas with high traffic in the environment arise due to the floorplan of the transportation area.

Main Success Scenario: The central traffic management system has dynamically gathered experience of previous traffic flow by being provided current robot positions at every time step. In addition, robots provide information about their planned paths so that the traffic management can make predictions for the future. When a robot plans a new path it can request traffic information from the traffic management (e.g. in the form of heatmaps) in order to plan a path avoiding congested areas.

Possible Misuse case: If inaccurate information is provided to the traffic management or the path optimization algorithm interprets the data falsely, the robots are unable to avoid traffic in a meaningful way, leading to unnecessarily long paths or an even worse traffic situation.

3 State-of-the-Art

3.1 Task planning

Modern, fully automated warehouses are operating with a vast volume of goods at any given time which leads to a problem, namely the *multi robot task assignment problem* (MRTA). Even though many robot-agents might be available for given tasks such as transporting the goods between designated areas of the warehouse, how should one choose which task belongs to which robot. In other words, how should one decide which robot is the most suitable for a given task. The answer depends on which priorities the overall Task Planner has and what kind of information it can access about the available robot-agents. As an example the Task Planner might assign tasks to the robot-agents according to the distance (with some well-defined metrics) between the robot-agent and the starting coordinates of the task. However, taking only the position of robots into account leads to a suboptimal solution. The task assignment process, therefore, needs to assign a given task according to many criteria regarding the robot agents as well as to minimize some environmental cost (distance, battery usage, etc) at the same time.

According to the taxonomy of task planning problems proposed by [16], our case consists of a “single-task robots, single-robot tasks” (ST-SR) problem i.e. robots can fulfill only a single task at once, and tasks need a single robot to be accomplished. Regarding assignments, the authors distinguish “instantaneous assignment (IA)” from “time-extended assignment (TA)” problems. In the former case, there is no information regarding future tasks (which is our case), while in the latter case either all tasks are known in advance, or at least there is a model of how tasks are created. ST-SR-IA problems belong to the simplest task allocation problems and can be seen as an instance of the Optimal Assignment Problem (OAP). In case there is a central task planner available, the Hungarian method is known to find an optimal solution in polynomial time [16]. A distributed bidding approach is proposed by [17], which can be easily parallelized and outperforms the Hungarian method on large sparse problems.

Our system has also the property that tasks arrive while robots are already assigned, and thus qualifies as an “online assignment” variant of the ST-SR-IA problem [16]. We also don’t allow reassignments, what would cause extra efforts for task and path planning. For this particular case, the MURDOCH assignment algorithm is shown to have the best performance possible [16] [18] . It simply assigns new tasks to the most fit robot currently available.

Since our instances are rather small (up to 8 robots with a sparse generation of tasks) the Hungarian method shall be implemented for the starting assignment, and further assignments will be done using the MURDOCH assignment algorithm.

3.2 Path planning

Ideally a solution to the *multi-robot path planning* problem (MRPP) should be *complete* (i.e. a solution is always found when it exists), *optimal* regarding time/resource related criteria and *efficient* [2]. Finding an optimal solution is known to be an intractable problem [1], therefore, any solution has to trade off some desired properties. Existing solutions to MRPP can be categorized as follows [2]:

1. **Centralized approach**

   In centralized path planning approaches the planning happens at a central component that has access to global information regarding the map, robot configurations and task assignment information. In other words the system of robot-agents is handled as a single composite robot entity. Centralized approaches are further classified as:

   a. **Coupled**

      A centralized solution to the MRPP is coupled if the path planner searches for an optimal solution considering all agents simultaneously. Such an approach is complete and optimal, but intractable. An example of a centralized coupled algorithm is described by [3], where a composite roadmap is constructed and optimal paths are derived from it. As pointed out by the authors, it’s usable to a problem with up to five robots.

   b. **Decoupled**

      Decoupled approaches to multi-robot path planning typically trade off solution quality (such as completeness) for efficiency by solving some aspects of the problem independently from others [2].

      The 2 main strategies to decouple aspects of the planning problem are “Prioritized Planning” and “Path Coordination”.

      In the former case, tasks are being assigned to robot-agents with different calculated priorities (which calculation might change over time) and paths are planned individually afterwards, starting with those with the highest priority. The resulting framework might lead to dead-lock situations when a path can not be planned by the planner-agent even if it exists, usually when a higher priority path blocks the only feasible path for a low-priority agent.

      The latter strategy is based on splitting the planning task in two separate subtasks [5]. First the spatial paths are planned, followed by a velocity planning that avoids collisions of the planned paths using the time-domain.

      Other approaches target for efficient, but suboptimal algorithms that achieve completeness,. The Bibox algorithm [15] for instance, is shown to have $O(n^3)$ complexity (where *n* stands for the number of vertices in the roadmap), but assumes a biconnected graph and at least two vertices empty. Similarly the “push-and-swap” algorithm [6] is shown to be scalable to relatively large scenarios with more than 100 robot agents. It follows a simple strategy of moving agents towards their goals (“push” operation) and switching positions of agents when two agents are in conflict, i.e. one blocks the other towards its goal (“swap” operation). As pointed out by [14] “push-and-swap” is not a complete algorithm though, as originally thought by the authors. Thus an alternative is suggested the “push-and-rotate” algorithm [14]. It basically improves the previous approach by decomposing the given graph into subgraphs, which allows different prioritizing of agents, and by introducing a new kind of operation called *rotate* to deal with problems detected. This algorithm is shown to be similarly efficient as the Bibox algorithm with the advantage of not being restricted to biconnected graphs.

Due to the simplicity and efficiency of the push-and-swap approach, we shall consider it for our problem. In case its limitations affect our use cases, we shall improve it with the strategies of the push-and-rotate algorithm, a feasible strategy considering the similarity of both approaches.

2. Decentralized approach

In decentralized approaches the paths of robots are not calculated in advance. Instead, the robots calculate their initial route independently of other robots. As the robot paths are based on incomplete information, conflicts between robots will arise. These are resolved online by negotiation between robots, traffic laws, or behavior-based avoidance rules.

An approach using negotiation between robots is described in [8]. The robots move in a network of roadways and need to resolve conflicts at intersections. This is done by appointing one robot involved in the conflict to be the leader. This robot will calculate a solution and distribute it to the other robots.

Traffic policies have been proposed e.g. by [9] and [10]. [9] categorizes traffic laws into different types, applied to the robot position, different conditions (e.g. overtaking, avoiding obstacles), and laws to ensure safety and handling of failures. In [10] basic rules are proposed to give robots guidelines in case of a collision.

Solutions using behavior-based rules have been proposed for example in [11] where robots stop or change direction to avoid collision. Using online techniques should also take into consideration the kinematic model of the robot as has been done in [12] to consider that robots can’t stop instantly.

Another approach [13] uses local dynamic networks between nearby robots. Within these networks the robots use a central planner to plan collision-free paths. Otherwise the robots use decoupled planners.

Decentralized approaches are fast to compute, scale well as the number of robots increases and are not dependent on the existence of a central unit. But since the initial path planning is based on incomplete information, the robots need a set of tools to resolve complex conflict situations.

We will use a similar approach as presented in [13] using dynamic networks to enable locally complete solutions. The robots will locally communicate their planned trajectories but with a limited time scope to account for the uncertainty growth as time progresses to prevent unnecessary planning. Only if they detect a collision in near-future trajectories will they initiate local central planning. To take care of collision propagation (a collision was created because another collision was averted) and allow the planner to potentially reach completeness (to resolve deadlocks), local networks can grow up to the full system size if necessary. This approach will give the same capabilities as a central planning system while demanding less computational resources and handling uncertainty by limiting the replanning scope to near-future events that are well predictable.

3.3 Charging Management

Energy management is one of the most significant issues that needs to be solved to provide the continuity of any autonomous system. This issue is managed different ways depending on the decentralized and centralized architecture in the literature. For instance, Kiva robots (used by Amazon) drive to the charging station when their battery level drops below a threshold. This energy management approach is defined as self-learning system in decentralized decision. In [19] scheduling approaches for the charging of the autonomous systems are based on three key components :

1. The timing of charging processes
2. The selection of a charging station
3. The duration of the charging processes

In this paper, flexible scheduling approach is used different methodologies on these key components: threshold for timing of charging, heuristic algorithm for the selection of the charging station and threshold/swap time for the duration of the charging. Another approach [20] in the literature to find the charging station is Online Connected Dominating Sets (OCDS), which are used in undirected graphs, when the battery level of the robot agent drops below a critical level. In this approach, any two charging stations shall not be farther from each other than a specified distance. In [21], the authors manage the energy of autonomous systems via the docking and charging system. This approach is needed to let the robot itself handle autonomous docking and charging problem.

Charging management is defined as centralized in our system, so our system uses the threshold for the timing of the charging processes and standard heuristic algorithm (Euclidean distance, Diagonal distance, or Manhattan distance) to find the nearest available charging station. Charging management tracks the battery level of the robot-agent in our system, and also to find the nearest available charging station is managed by it. Our system considers the same key components with [19] for charging management.

4 System Requirements, Architecture and Specifications

4.1 Requirements.

- The system starts with idle robots.
- Tasks are generated by the existing system. A task consists of the coordinates of a package (which appears in a storage) and a designated end station.
- Upon arrival of tasks, Task Assignment Agent decides which agent shall take it. Agent is selected according to expected time, ability to carry package (e.g. weight) and battery status.
- (CA) TAA communicates to Central Path Planner the assignment, providing the agent's ID, package position and end station.
- (CA) Central Path Planner plans a close to optimal path for accomplishing the given task. The planning shall consider other paths planned, and existence of obstacles.
- Collisions between robots are not allowed and will be avoided both by carefully planning of paths and local sensing.
- (Optional) An agent may stop working. System has to consider it as an obstacle, and it's package (if any) shall be picked up by another agent. Selecting an appropriate agent and path are done as for regular tasks. Thus: TAA and Path Planning shall be designed to accommodate this special case.
- (CA) If an agent detects a possible collision (either with an obstacle or other agent) it shall immediately stop and communicate this (and the obstacles coordinates) to the Central Path Planner.
- (CA) The system has to deal with mismatches between planned paths and current agent configurations. It shall send readjustment signals in case a danger of collision is reported.
- (DA) The system has to be able to resolve any deadlocks or conflicts arising because of incomplete planning via local negotiation between the robots.
- Scenarios are provided which shall be used to run our solution in a MORSE based simulation.

- Our solution has to scale up to at least 5 agents, ideally dealing for a longer period in the 8 agent case.
- The solution has to be efficient. This means that paths chosen are reasonable, and the system as a whole doesn't suffer from long delays (for instance while selecting agents and planning paths). The system shall not stutter due to delays in such computations.
- Agents shall navigate their paths smoothly (i.e. following path with high precision). Velocity shall be predictable, so that planned paths stay indeed collision free.
- At least two different package types should be available.
- At least two different robot types with different capacity should be available.

4.2 System Architecture

Centralized Architecture

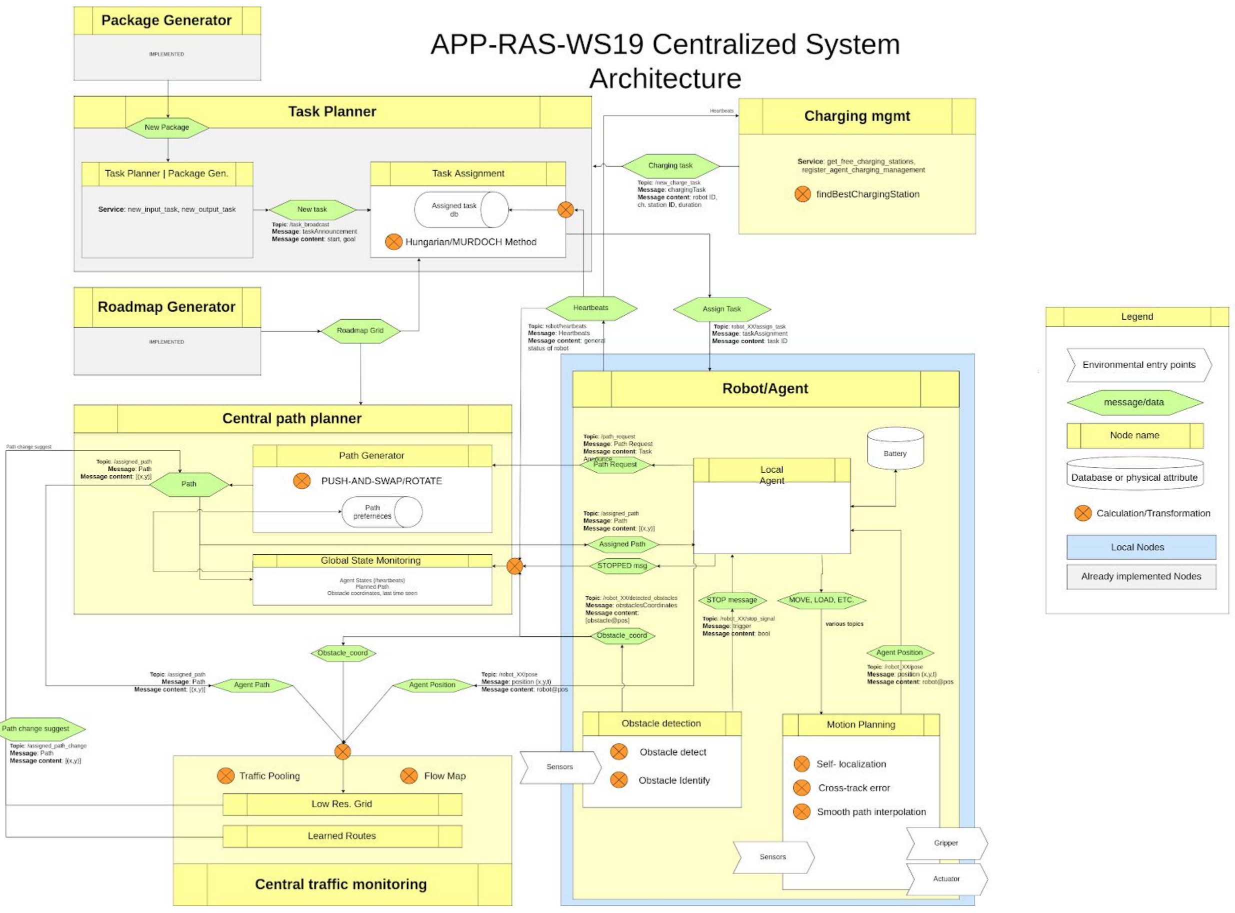


Figure Centralized System Architecture / Flow-Chart Diagram

Decentralized Architecture

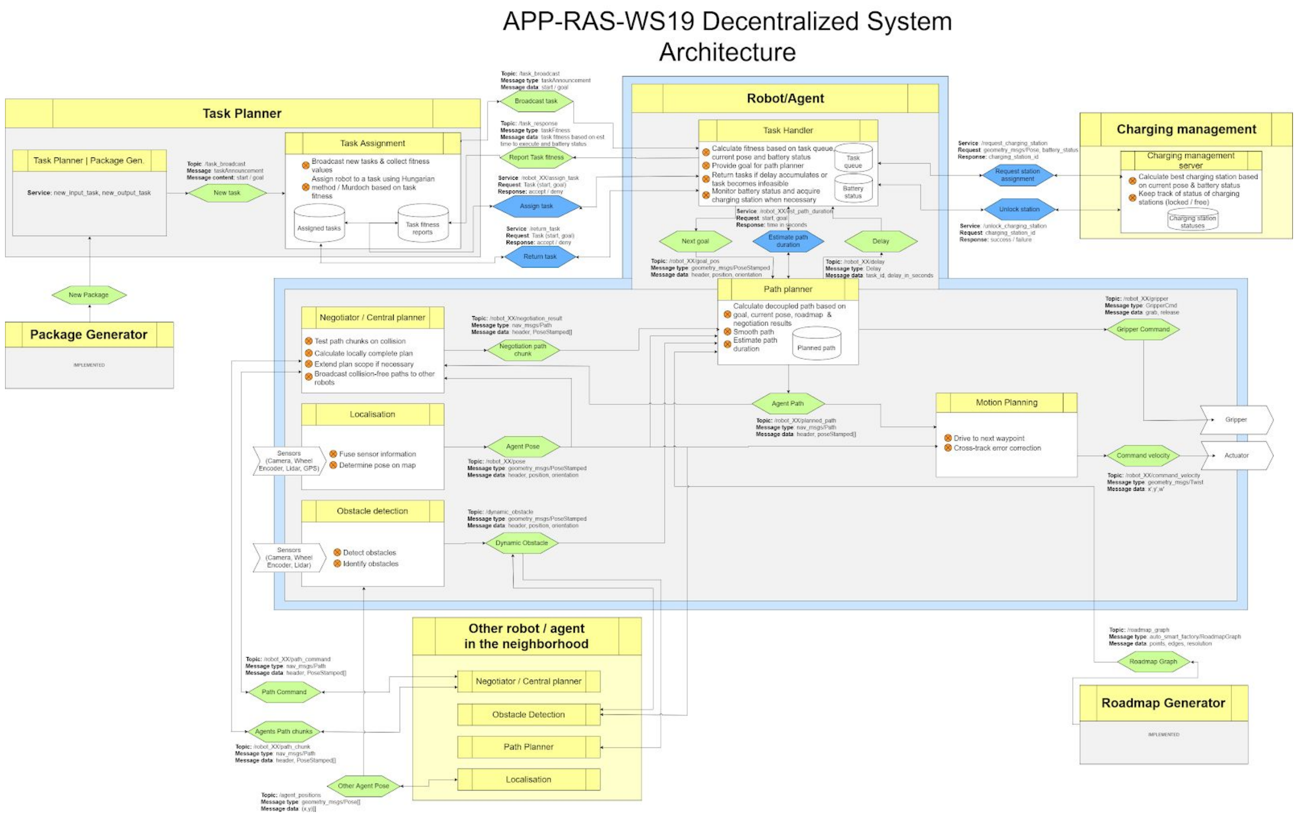

APP-RAS-WS19 Decentralized System Architecture
Task Planner
Task Planner | Package Gen.
Task Assignment
Broadcast new tasks & collect fitness values
Assign robot to a task using Hungarian method / Murdoch based on task fitness
Assigned tasks
Task fitness reports
New task
Broadcast task
Report Task fitness
Assign task
Return task
Robot/Agent
Task Handler
Calculate fitness based on task queue, current pose and battery status
Provide goal for path planner
Return tasks if delay accumulates or task becomes infeasible
Monitor battery status and acquire charging station when necessary
Task queue
Battery status
Next goal
Estimate path duration
Delay
Charging management
Charging management server
Calculate best charging station based on current pose & battery status
Keep track of status of charging stations (locked / free)
Charging station statuses
Request station assignment
Unlock station
New Package
Package Generator
Path planner
Calculate decoupled path based on goal, current pose, roadmap & negotiation results
Smooth path
Estimate path duration
Planned path
Negotiator / Central planner
Test path chunks on collision
Calculate locally complete plan
Extend plan scope if necessary
Broadcast collision-free paths to other robots
Negotiation path chunk
Localisation
Fuse sensor information
Determine pose on map
Sensors (Camera, Wheel Encoder, Lidar, GPS)
Agent Pose
Obstacle detection
Detect obstacles
Identify obstacles
Sensors (Camera, Wheel Encoder, Lidar)
Dynamic Obstacle
Agent Path
Motion Planning
Drive to next waypoint
Cross-track error correction
Gripper Command
Gripper
Command velocity
Actuator
Other robot / agent in the neighborhood
Negotiator / Central planner
Obstacle Detection
Path Planner
Localisation
Path Command
Agents Path chunks
Other Agent Pose
Roadmap Graph
Roadmap Generator

Figure Decentralized System Architecture / Flow-Chart Diagram

Hybrid Architecture

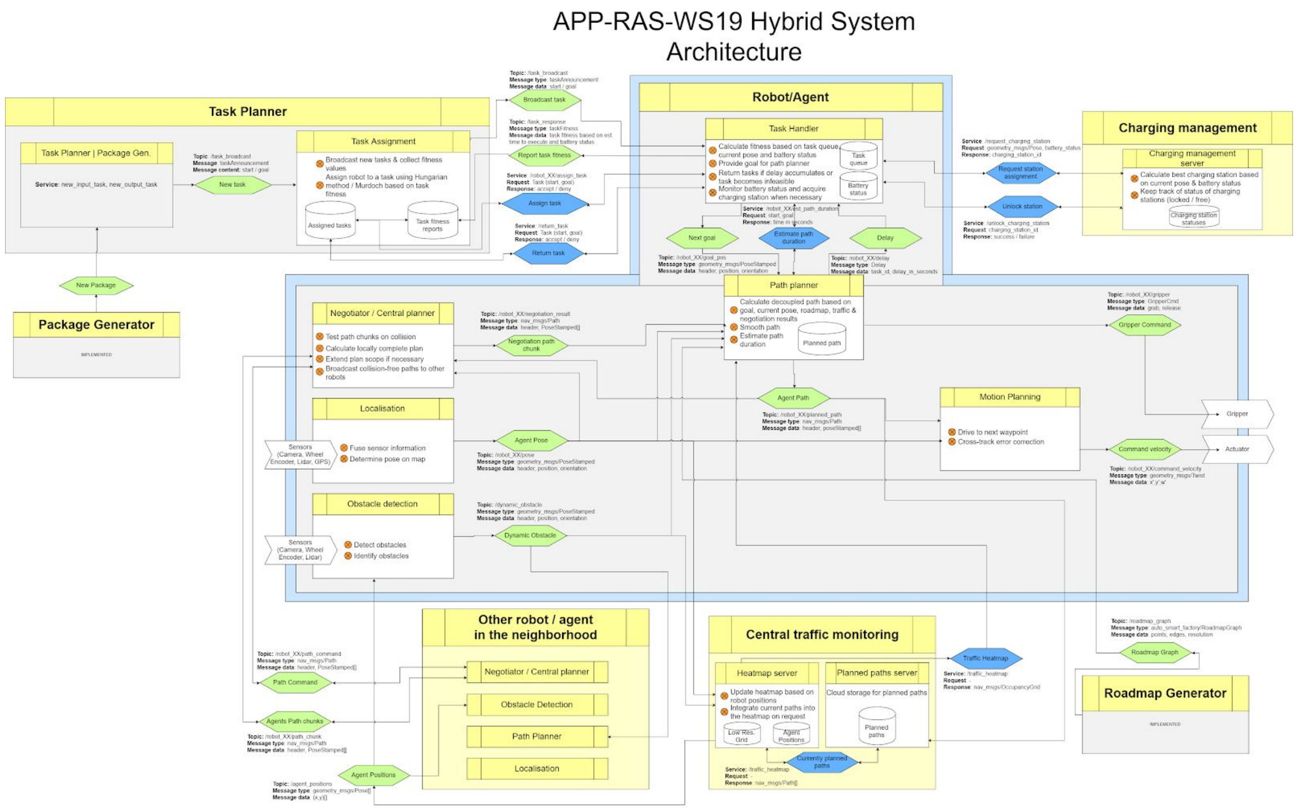
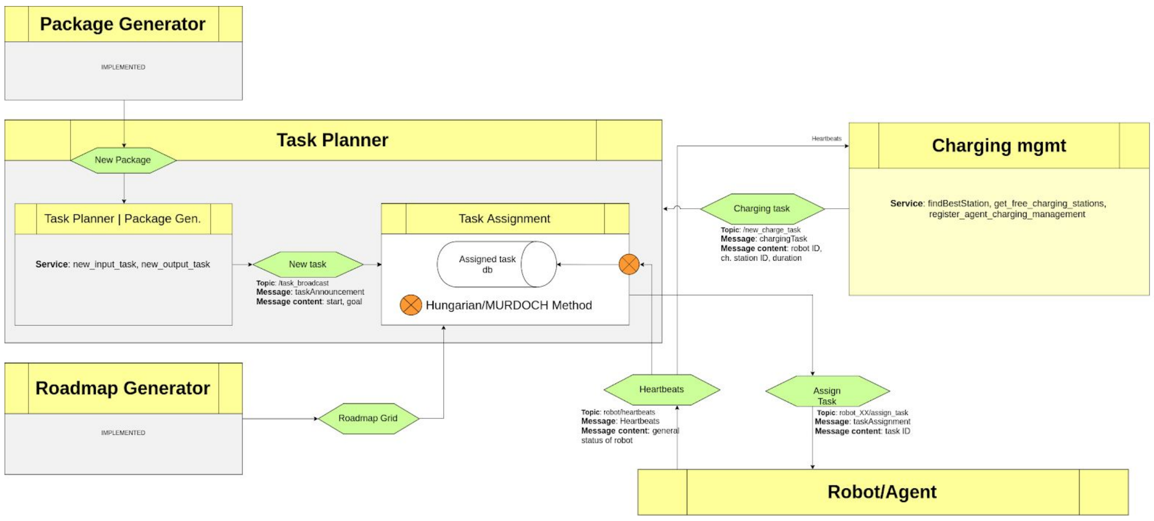

APP-RAS-WS19 Hybrid System Architecture
Task Planner
Task Planner | Package Gen.
Service: new_input_task, new_output_task
Topic: /task_broadcast
Message: taskAnnouncement
Message content: start / goal
New task
Task Assignment
Broadcast new tasks & collect fitness values
Assign robot to a task using Hungarian method / Murdoch based on task fitness
Assigned tasks
Task fitness reports
Topic: /task_broadcast
Message type: taskAnnouncement
Message data: start / goal
Broadcast task
Topic: /task_response
Message type: taskFitness
Message data: task fitness based on est. time to execute and battery status
Report task fitness
Service: /robot_XX/assign_task
Request: Task (start, goal)
Response: accept / deny
Assign task
Service: /return_task
Request: Task (start, goal)
Response: accept / deny
Return task
New Package
Package Generator
IMPLEMENTED
Robot/Agent
Task Handler
Calculate fitness based on task queue, current pose and battery status
Provide goal for path planner
Return tasks if delay accumulates or task becomes infeasible
Monitor battery status and acquire charging station when necessary
Task queue
Battery status
Service: /robot_XX/est_path_duration
Request: start, goal
Response: time in seconds
Next goal
Estimate path duration
Delay
Topic: /robot_XX/goal_pos
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Topic: /robot_XX/delay
Message type: Delay
Message data: task_id, delay_in_seconds
Charging management
Charging management server
Calculate best charging station based on current pose & battery status
Keep track of status of charging stations (locked / free)
Charging station statuses
Service: /request_charging_station
Request: geometry_msgs/Pose, battery_status
Response: charging_station_id
Request station assignment
Unlock station
Service: /unlock_charging_station
Request: charging_station_id
Response: success / failure
Path planner
Calculate decoupled path based on goal, current pose, roadmap, traffic & negotiation results
Smooth path
Estimate path duration
Planned path
Negotiator / Central planner
Test path chunks on collision
Calculate locally complete plan
Extend plan scope if necessary
Broadcast collision-free paths to other robots
Topic: /robot_XX/negotiation_result
Message type: nav_msgs/Path
Message data: header, PoseStamped[]
Negotiation path chunk
Localisation
Sensors (Camera, Wheel Encoder, Lidar, GPS)
Fuse sensor information
Determine pose on map
Agent Pose
Topic: /robot_XX/pose
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Obstacle detection
Sensors (Camera, Wheel Encoder, Lidar)
Detect obstacles
Identify obstacles
Topic: /dynamic_obstacle
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Dynamic Obstacle
Agent Path
Topic: /robot_XX/planned_path
Message type: nav_msgs/Path
Message data: header, poseStamped[]
Motion Planning
Drive to next waypoint
Cross-track error correction
Topic: /robot_XX/gripper
Message type: GripperCmd
Message data: grab, release
Gripper Command
Gripper
Command velocity
Actuator
Topic: /robot_XX/command_velocity
Message type: geometry_msgs/Twist
Message data: x',y',w'
Other robot / agent in the neighborhood
Negotiator / Central planner
Obstacle Detection
Path Planner
Localisation
Topic: /robot_XX/path_command
Message type: nav_msgs/Path
Message data: header, PoseStamped[]
Path Command
Agents Path chunks
Topic: /robot_XX/path_chunk
Message type: nav_msgs/Path
Message data: header, PoseStamped[]
Agent Positions
Topic: /agent_positions
Message type: geometry_msgs/Pose[]
Message data: (x,y)[]
Central traffic monitoring
Heatmap server
Update heatmap based on robot positions
Integrate current paths into the heatmap on request
Low Res. Grid
Agent Positions
Planned paths server
Cloud storage for planned paths
Planned paths
Currently planned paths
Service: /traffic_heatmap
Request: -
Response: nav_msgs/Path[]
Traffic Heatmap
Service: /traffic_heatmap
Request: -
Response: nav_msgs/OccupancyGrid
Topic: /roadmap_graph
Message type: auto_smart_factory/RoadmapGraph
Message data: points, edges, resolution
Roadmap Graph
Roadmap Generator
IMPLEMENTED

Figure Hybrid System Architecture / Flow-Chart Diagram

4.3 System Specifications

Centralized / Decoupled

1. *Task Planner*



The Task Planner in our solution is reliable to generate new tasks according to the output of the Package Manager and to assign these online generated new tasks to robot agents according to some calculations. The task generation process is already implemented and therefore it is out of scope for our project. For the Task Assignment Agent (TAA) which shall be a part of or very strongly connected to the Task Planner we will implement a dedicated ROS Node. The Node will subscribe to the "/task_broadcast" topic of the Task Planner, so that upon each new package-related task the TAA can start the assignment process to a robot-agent. For the assignment process TAA also collects the "/heartbeats" information of the robot-agents, including the battery level, eta, idle status, position and orientation. The collected data will be used as an input to the assignment process using the Hungarian method or in later iterations the MURDOCH assignment algorithm. After the decision (i.e.: which task should be assigned to which robot-agent), TAA uses the robot-agent's "/assign-task" service to finalize the process. The assigned task as well as the ID of the robot agent is stored locally in the TAA node as well. To estimate "distance" of the starting position of the task and the possible robot-agents TAA uses the roadmap grid generated by the Roadmap Generator.

Furthermore, the task planner is reliable to dynamically handle "not package related tasks" as well, such as charging tasks. In this case according to the input of the Charging Management Unit (CM) the Task Planner shall create a new task according to which charging station is chosen by CM for the given robot-agent. This new charging task is then assigned to the robot-agent with low battery level. If a robot-agent is currently fulfilling a task and the battery level drops TAA dynamically reassigns the task in process to another robot-agent and assigns an adequate charging task to the robot-agent with dropped battery level. As a Summary:

a. Task Assignment Agent (TAA)

i. Functionality & Methodology

New tasks are assigned to robot-agents using the Hungarian method or the MURDOCH assignment algorithm

ii. Inputs & Outputs

New Task (Input, Topic Subscription): Receives information containing the start and end station of a package to be transported.

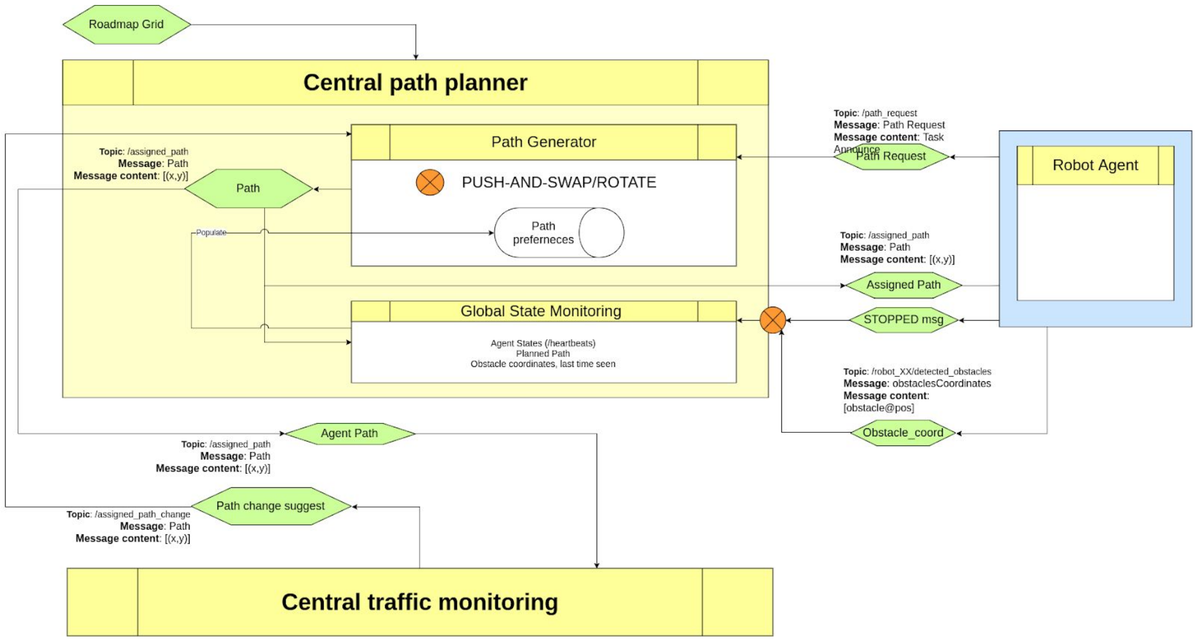
Roadmap Graph (Input, Topic Subscription): Receives the static connectivity map of the warehouse

Heartbeats (Input, Topic Publication): Receives regular status update messages from all robots.

New Charging Task (Input, Topic Subscription): Receives information containing the details of a new charging task decided by the CM unit. Includes the robot ID with low battery, the battery level, the charging station to which the robot should be guided and the duration of the charging process which serves as a refractory period for the TAA w.r.t. the robot-agent, in other words during the charging process no new task can be assigned to the robot.

Task Assignment (Output, Service call): Uses the task assignment service of a chosen robot-agent to assign a task to it.

2. *Central Path Planner*


Roadmap Grid
Central path planner
Path Generator
PUSH-AND-SWAP/ROTATE
Path prefernece s
Topic: /assigned_path Message: Path Message content: [(x,y)]
Path
Populate
Global State Monitoring
Agent States (/heartbeats)
Planned Path
Obstacle coordinates, last time seen
Topic: /path_request Message: Path Request Message content: Task Announce
Path Request
Robot Agent
Topic: /assigned_path Message: Path Message content: [(x,y)]
Assigned Path
STOPPED msg
Topic: /robot_XX/detected_obstacles Message: obstaclesCoordinates Message content: [obstacle@pos]
Obstacle_coord
Agent Path
Topic: /assigned_path Message: Path Message content: [(x,y)]
Path change suggest
Topic: /assigned_path_change Message: Path Message content: [(x,y)]
Central traffic monitoring

The Central Path Planner has to compute paths for all robots which have tasks assigned, so that they can safely (i.e. collision-free) travel over the scenario to accomplish their goals. Thus paths shall be planned to move robots between their current position and the

i. Package station
ii. End-Station
iii. Charging Station

depending on their current tasks (a task in this sense may also mean recharging, or going back to the package station.) The path for a given assigned task is generated upon request from the agent. In other words, the task is first assigned to the robot-agent, which, according to this task, will request a path from the path planner.

The Central Path Planner consists of a Path Generator and a Global State Monitoring component, which may be implemented in a single ROS node for efficiency reasons.

The Path Planner (in later iterations) is also "biased" to generate paths according to current and estimated traffic scenarios. This happens with an interface with the Central Traffic Monitoring Unit (TM). The path generation process is using the "/assigned_path_change" topic of TM which includes either Flow Map data estimated to have low traffic or pooled traffic data of the environment.

b. Path Generator

i. Functionality & Methodology

Paths are planned using the Push-and-Swap algorithm, which allows incremental computation of paths, which fits the online property of our system.

ii. **Inputs & Outputs**

Path Request (Topic Subscription): Receives information from an agent containing the Agent's ID, the type of task and the extra information needed to accomplish it (package location, end-station etc.)

Roadmap Graph (Topic Subscription): Receives the static connectivity map of the warehouse

Path Change Suggest (Topic Subscription): Receives Flow Map or pooled traffic data for path generation

Path Preferences db (Input, Database): Database populated by Global State Monitoring including information about possible failures and obstacles in the environment that can influence the path generation process.

Assigned Path (Topic Publication): A description of the path assigned to a given robot.

b. Global State Monitoring

i. **Functionality & Methodology**

Stores global data received from other components. For instance agent states (location, assigned tasks etc.), detected obstacles etc.

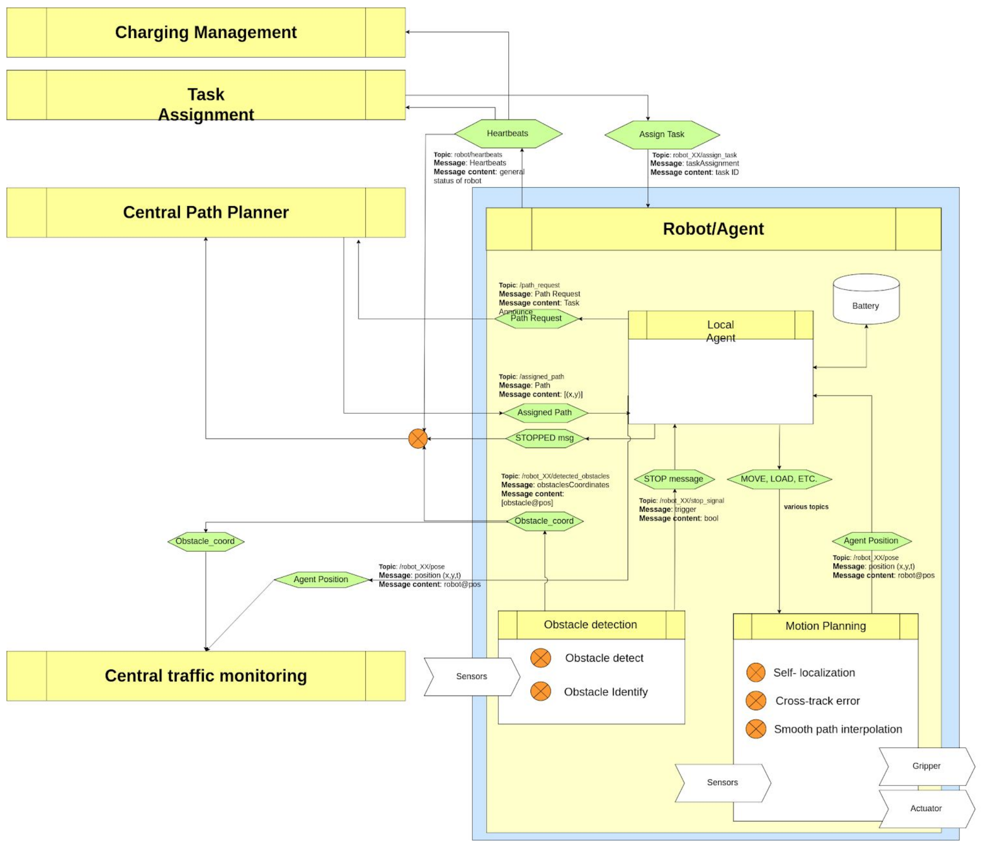
ii. **Inputs & Outputs**

Obstacle Coordinates (Topic Subscription): Coordinates of a detected obstacle.

Assigned Path (Topic Subscription): A description of the path assigned to a given robot.

Path Preferences db (Output, Database): Populating the Path Preferences database with possible failures or obstacles in the environment.

3. Robot/Agent



The Robot Agent (RA) node of the architecture includes all nodes, services and topics that belongs to a robot-agent. These are all coordinated via the Local Agent node of RA. When a new task is generated and is assigned to an agent TAA assigns it using the "/assign_task" service of RA. Then, RA requests a new path to accomplish the task from PP using the "/path_request" topic. When the path is generated it is assigned to RA via the "/assigned_path" topic. The RA is then uses the Motion Planner unit (which is a part of RA) to smoothly navigate through the assigned path and to fulfill the task goal, i.e. unloading a package if necessary. The used methods to load, unload or carry a package are already implemented, hence they are not included in this specification. In case the assigned path of RA is not followable due to an obstacle blocking it, the Obstacle Detection Unit shall stop RA movement with a dedicated STOP message ("/robot_xx/stop_signal)" as well as send the obstacle type and coordinates to the PP. In this case the STOP signal triggers a new path request from the agent, but at this time the obstacle coordinates and type is being considered during the path planning process. The RA is furthermore publishes general status information on a regular basis to the "/heartbeats" topic which serves as an input for other central nodes, such as the TA, the CM and the PP.

a. Local Agent

i. Functionality & Methodology

Responsible for the communication between the Agent and both the Central Nodes such as the Task Planner, the Central Path Planner, Traffic Management and Charging Management.

ii. Inputs & Outputs

Assigned Path (Topic Subscription): A description of the path assigned to a given robot.

STOP Message (Input, Topic Subscription): Emergency high frequency trigger signal to stop all current movements of the agent.

Heartbeat (Topic Publication): Contains information regarding current status of the robot (if idle or not task, estimated time of arrival, current position, battery level and orientation).

Path Request (Topic Publication): Publishes a request to get a new path of PP which is adequate to fulfill the task. Includes the Agent's ID, the type of task and the extra information needed to accomplish it (package location, end-station etc.)

b. Obstacle Detection

i. Functionality & Methodology

Responsible for dynamic obstacle detection in the environment using the sensors of the robot-agent as well as for acting with an emergency stop signal with unforeseen scenarios and for communicating the obstacles to central nodes.

ii. Inputs & Outputs

Sensory Data (Environmental entrypoint): Using the "/laser_scanner" topic information to estimate environmental obstacles.

STOP Message (Output, Topic Publication): Sending emergency high frequency trigger signal to stop all current movements of the agent when an obstacle is detected.

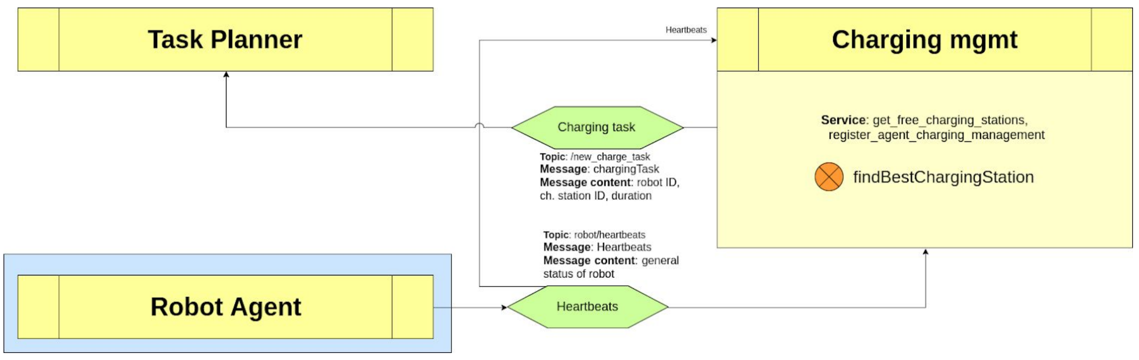
Obstacle Coordinates (Topic Publication): Type and coordinates of a detected obstacle using the ground truth position information of the robot.

c. Motion Planning

i. Functionality & Methodology

Using the available environmental information (using sensors) the Motion Planning unit is responsible to reliably estimate the position of RA which will serve as a ground truth for the Local Agent. Furthermore, the Motion Planning Unit will calculate the cross track error and use it in conjunction with a PID controller to provide a smooth and reliable movement along a given path without any jitters.

4. Charging Management



The Charging Management Unit in the architecture is responsible to act on unforeseen battery related issues regarding the robot-agents. CM continuously monitors the battery level of RA using the "/heartbeats" topic. When the battery level of a given agent is low CM identifies the agent. According to the position of the agent, and the currently available charging stations ("/get_free_charging_stations") CM chooses an optimal charging station ("/findBestStation") and decides on how long the robot-agent should be at the charging station. After these decisions are made the CM uses the "/new_charging_task" topic to convey the information to TAA.

i. Functionality & Methodology

Using the available information about RA and the environment the CM unit creates optimized charging tasks and introduces them to the TAA so that robots with low battery can be scheduled to be recharged.

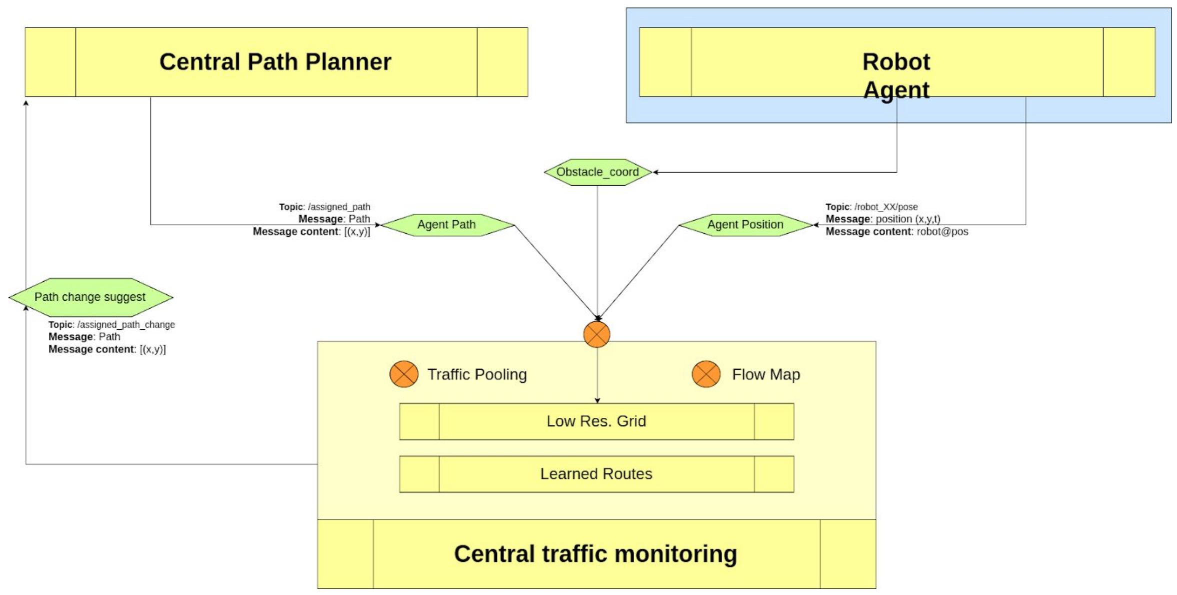
ii. Inputs & Outputs

Heartbeat (Topic Subscription): Contains information regarding the current status of the robot agents (if idle or not, task, estimated time of arrival, current position, battery level and orientation).

Available Charging Stations (Topic Subscription): Receives environmental information about which charging stations are currently available. ("/get_free_charging_stations")

New Charging Task (Topic Publication): Publishes information about a new charging task, i.e.: which robot, for how long and at which charging station should be recharged. ("/new_charge_task")

5. Traffic Management



Due to traffic related issues monitoring and acting according to current and estimated traffic is useful. In our architecture the Traffic Management Unit is providing a Flow Map according to learned paths of previous task fulfillment as well as it continuously monitors the actual and planned position of the robot-agents to provide a traffic map. The information is then used to bias the path generation process of the PP.

i. Functionality & Methodology

To estimate traffic level across the provided simulation environment the TM uses the planned and actual position of RA, as well as found obstacles in the environment. At the end of calculation (which uses [7] and pooling based methods) the output is a suggestion map. A newly generated path is then biased by TM and if necessary a recommendation of path change is communicated to the PP.

ii. Inputs & Outputs

Agent Path (Topic Subscription): A description of the path assigned to a given robot.

Agent Position (Topic Subscription): Receives the ground trough position information of all given robot-agents. (“/robot_XX/pose”)

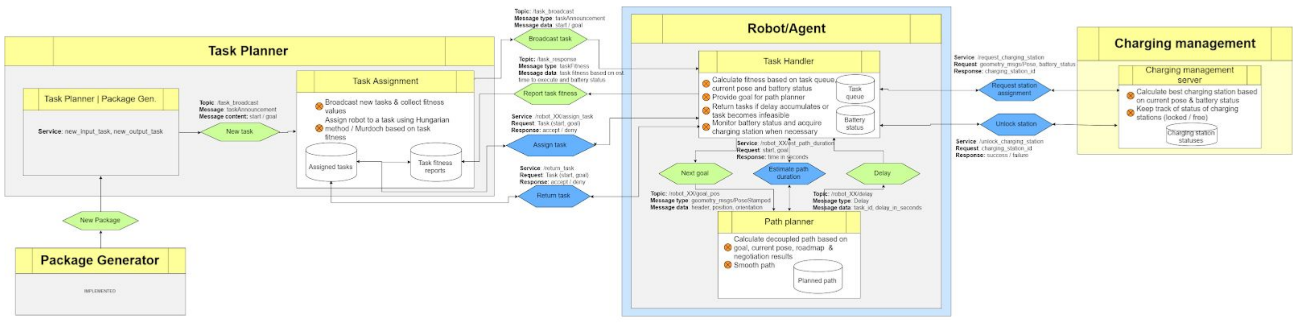
Obstacle Coordinates (Topic Subscription): Receives information about the type and coordinates of found obstacles in the environment.

Path Change Suggest (Topic Publication): Publishes information about recommended changes in assigned tasks considering traffic scenarios.

Decentralized / Hybrid

1. *The task planning process*

The task planning process is used to determine which robot will fulfill a newly arrived task.
To do so the task will be broadcasted to all robots which will then provide the task assignment node with a fitness value indicating how well the robot is suited to perform the task. The task will then be assigned to the robot with the best fitness value. Robots can also return a task if it becomes infeasible or the robot accumulates delay.


Task Planner
Task Planner | Package Gen.
Service: new_input_task, new_output_task
Topic: /task_broadcast
Message: taskAnnouncement
Message content: start / goal
New task
Task Assignment
Broadcast new tasks & collect fitness values
Assign robot to a task using Hungarian method / Murdoch based on task fitness
Assigned tasks
Task fitness reports
New Package
Package Generator
IMPLEMENTED
Topic: /task_broadcast
Message type: taskAnnouncement
Message data: start / goal
Broadcast task
Topic: /task_response
Message type: taskFitness
Message data: task fitness based on est. time to execute and battery status
Report task fitness
Service: /robot_XX/assign_task
Request: Task (start, goal)
Response: accept / deny
Assign task
Service: /return_task
Request: Task (start, goal)
Response: accept / deny
Return task
Robot/Agent
Task Handler
Calculate fitness based on task queue, current pose and battery status
Provide goal for path planner
Return tasks if delay accumulates or task becomes infeasible
Monitor battery status and acquire charging station when necessary
Task queue
Battery status
Service: /robot_XX/est_path_duration
Request: start, goal
Response: time in seconds
Next goal
Estimate path duration
Delay
Topic: /robot_XX/goal_pos
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Topic: /robot_XX/delay
Message type: Delay
Message data: task_id, delay_in_seconds
Path planner
Calculate decoupled path based on goal, current pose, roadmap & negotiation results
Smooth path
Planned path
Service: /request_charging_station
Request: geometry_msgs/Pose, battery_status
Response: charging_station_id
Request station assignment
Unlock station
Service: /unlock_charging_station
Request: charging_station_id
Response: success / failure
Charging management
Charging management server
Calculate best charging station based on current pose & battery status
Keep track of status of charging stations (locked / free)
Charging station statuses

a. *Task assignment*

i. **Functionality & Methodology**

The task assignment broadcasts new tasks to the robots and awaits their corresponding fitness. Once all fitness values have arrived (or after a maximum time limit) the robots will be assigned using the Hungarian method and the Murdoch algorithm (choose robot with the highest fitness value). If a robot returns a task, the task will be broadcasted as if it had just arrived.

ii. **Inputs & Outputs**

New Task (Topic Subscription): Receives new tasks created by the package generator

Broadcast Task (Topic Publication): Publishes new tasks to all robots

Task fitness(Topic Subscription): Receives the fitness corresponding to a certain task by the robots

Assign Task (Service Request): Assigns a task to the robot with the best fitness value

Return Task (Service Response): Accept / decline request of a robot to return a task

b. *Task handler*

i. **Functionality & Methodology**

The task handler decides which action the robot will take next. It creates a fitness value when a new task is broadcasted, keeps track of the currently assigned tasks (with their estimated time) and makes sure the robot will not run out of battery.

To calculate the fitness for a new task, the robot takes into account its current pose (if it doesn't have a task assigned), its task queue and its battery status. It will make an estimate how much time the currently assigned tasks plus the new task will take. It will also estimate how much battery power will be left at the end of all tasks. Based on these values it will create a fitness value. An algorithm for the fitness value could be:

$$fitness = \frac{time\ to\ execute\ all\ tasks\ [s]}{battery\ power\ after\ execution\ [\%\ left]}$$

Using this formula a low fitness value is better.

If the robot falls below a threshold value for battery power after execution of all currently assigned tasks, it will request a charging station from the charging management station and drive there to replenish its energy. If the battery level falls below a critical value at any time of execution, it will return all tasks to the task assignment node and charge up immediately. This critical value has to be chosen so that the robot will have enough power to reach the charging station.

The task handler will assign new goals to the path planner depending on the task at hand (get package, store package, charge up). It will also track how much delay has accumulated to return tasks if a threshold delay is reached.

ii. Inputs & Outputs

Broadcast Task (Topic Subscription): Receives new tasks

Task Fitness (Topic Publication): Publishes calculated fitness value for a certain task

Assign Task (Service Response): Accepts / decline assigned task

Return Task (Service Request): Request to return a task

Next goal (Topic Publication): Next goal pose to drive towards

Estimate path duration (Service Request): Plan path for a start & goal pose and estimate how long it will take (used in the fitness calculation process)

Delay (Topic Subscription): Add / reduce delay based on recalculations of the path planner

Request station assignment (Service request): Asks the charging management to respond with a charging station to be used for recharging

Unlock station (Service Request): Notifies the charging management that recharging has finished

c. Charging management server

i. Functionality & Methodology

The charging management assigns charging stations if a robot requests one. Once a charging station was assigned to a robot, the charging management will keep a lock on it to prevent other robots from using that station. To decide which charging station will be assigned to a robot a simple heuristic like Euclidean or Manhattan distance will be used. The charging management searches for the charging station that minimizes the distance between robot and station. If a robot finished charging it will send a notification so the station can be unlocked again.

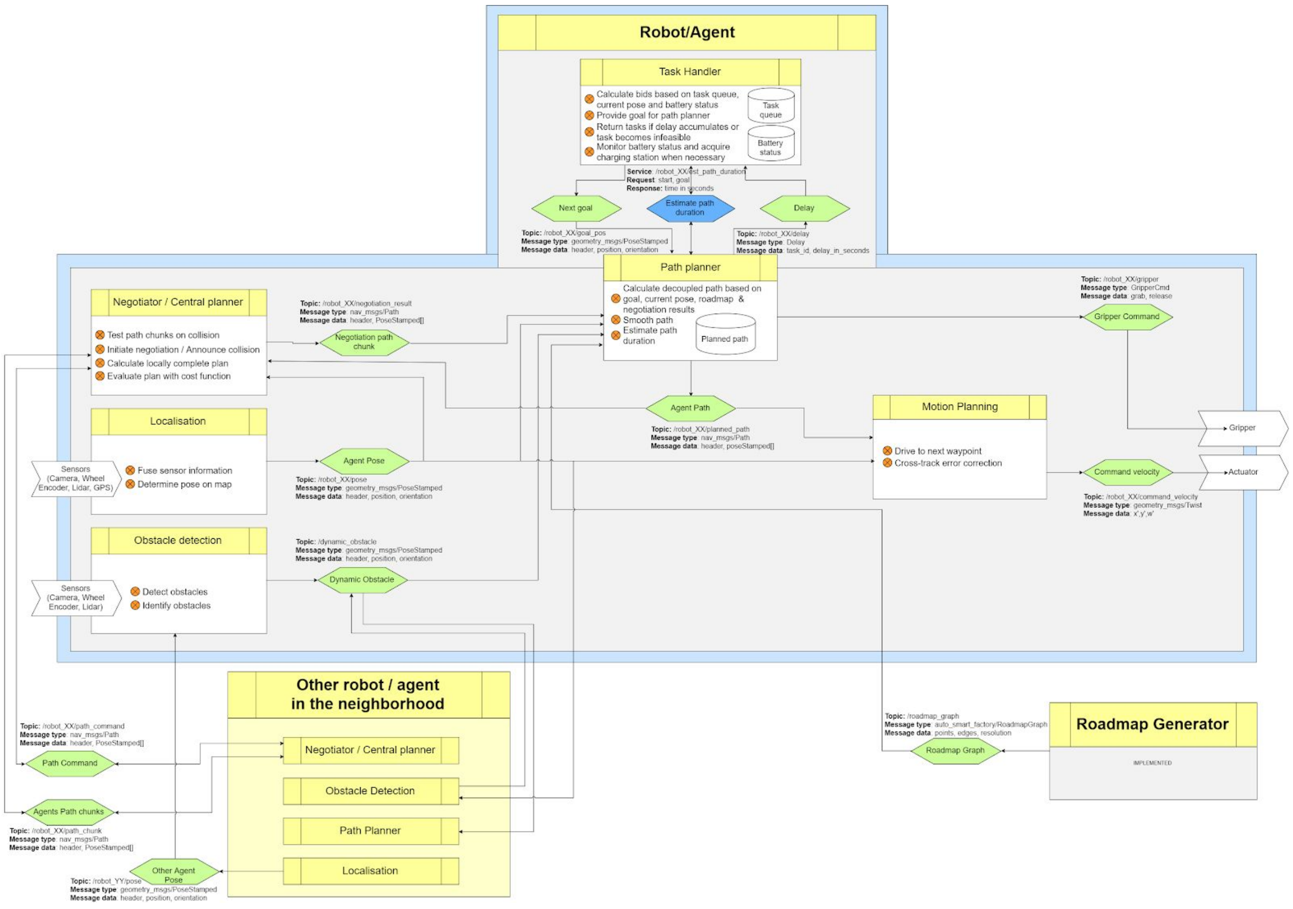
ii. Inputs & Outputs

Request station assignment (Service response): Responds with the charging station that is assigned to a requesting robot

Unlock station (Service Response): Notifies the robot if unlocking the station was successful

2. *The path planning & execution process*

Given a goal pose by the task planner, the job of the path planning & execution process is to reach that goal pose in an efficient manner while avoiding collisions. In order to do that the robot has to localize itself, plan a path, detect and circumnavigate obstacles, negotiate with other robots, and control its actuators so as to follow a trajectory leading to the goal.


Robot/Agent
Task Handler
Calculate bids based on task queue, current pose and battery status
Provide goal for path planner
Return tasks if delay accumulates or task becomes infeasible
Monitor battery status and acquire charging station when necessary
Task queue
Battery status
Service: /robot_XX/est_path_duration
Request: start, goal
Response: time in seconds
Next goal
Estimate path duration
Delay
Topic: /robot_XX/goal_pos
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Topic: /robot_XX/delay
Message type: Delay
Message data: task_id, delay_in_seconds
Path planner
Calculate decoupled path based on goal, current pose, roadmap & negotiation results
Smooth path
Estimate path duration
Planned path
Negotiator / Central planner
Test path chunks on collision
Initiate negotiation / Announce collision
Calculate locally complete plan
Evaluate plan with cost function
Topic: /robot_XX/negotiation_result
Message type: nav_msgs/Path
Message data: header, PoseStamped[]
Negotiation path chunk
Localisation
Sensors (Camera, Wheel Encoder, Lidar, GPS)
Fuse sensor information
Determine pose on map
Agent Pose
Topic: /robot_XX/pose
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Obstacle detection
Sensors (Camera, Wheel Encoder, Lidar)
Detect obstacles
Identify obstacles
Topic: /dynamic_obstacle
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Dynamic Obstacle
Agent Path
Topic: /robot_XX/planned_path
Message type: nav_msgs/Path
Message data: header, poseStamped[]
Motion Planning
Drive to next waypoint
Cross-track error correction
Topic: /robot_XX/gripper
Message type: GripperCmd
Message data: grab, release
Gripper Command
Gripper
Command velocity
Actuator
Topic: /robot_XX/command_velocity
Message type: geometry_msgs/Twist
Message data: x',y',w'
Other robot / agent in the neighborhood
Negotiator / Central planner
Obstacle Detection
Path Planner
Localisation
Topic: /robot_XX/path_command
Message type: nav_msgs/Path
Message data: header, PoseStamped[]
Path Command
Agents Path chunks
Topic: /robot_XX/path_chunk
Message type: nav_msgs/Path
Message data: header, PoseStamped[]
Other Agent Pose
Topic: /robot_YY/pose
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Topic: /roadmap_graph
Message type: auto_smart_factory/RoadmapGraph
Message data: points, edges, resolution
Roadmap Graph
Roadmap Generator
IMPLEMENTED

a. *Path planner*

i. **Functionality & Methodology**

The path planner will provide a path to reach the goal point provided by the task planner. It will calculate a smooth path independently of other robots based on the goal and the current pose. If any conflicts occur the path chunks resulting of negotiations will also be taken into consideration. To plan a path the robot will search the roadmap of the warehouse via a simple algorithm like A* or Theta*. Dynamic Obstacle information will trigger replanning if it obstructs the robot path. The obstacle will be integrated into the roadmap to guide the search. Any path chunks that are provided by negotiation will have priority and can't be changed by the local planner. It will resume planning from the point where the negotiated path chunk ends.

Everytime a path is planned the duration to execute it will be estimated based on a simple kinematic model of the robot. This duration will be used as an expected value when creating the fitness value for a task and can be used to update the delay of ongoing tasks.

The path planner will also keep track of the current job of the robot in a state. Once a path has been driven, the state will indicate if any further action is necessary (e.g. grab / store package, attach to charging station).

At each time step the path planner will also keep track if the estimated pose of the robot is close to the expected pose in the path at that time step. If both values differ too much from each other, a new plan will be created.

ii. **Inputs & Outputs**

Next goal (Topic Subscription): Next goal pose to drive towards

Estimate path duration (Service Response): Plan path for a start & goal pose and estimate how long it will take (used in the fitness calculation process)

Delay (Topic Publication): Add / reduce delay based on recalculations of the path planner

Negotiation Path Chunk (Topic Subscription): Receives near-future, collision-free path chunks that have been negotiated between robots

Agent Pose (Topic Subscription): Receives the current pose of the robot

Dynamic Obstacle (Topic Subscription): Receives unknown obstacles detected by this or other robots

Roadmap Graph (Topic Subscription): Receives the static connectivity map of the warehouse

Agent Path (Topic Publication): Publishes the path planned by the robot (it could also be imagined to use a ROS action for this purpose but it's not clear if other robots)

Gripper Command (Topic Publication): Publishes commands to the gripper based on robot state at the end of a trajectory

b. *Localisation*

i. **Functionality & Methodology**

The localisation node will use sensor and map information to determine the pose of the robot in the world. The GPS signal will provide noisy absolute position information. When the robot moves, sensor readings like wheel encoders will be used to make a prediction of the movement. These sensor readings are noisy as well so it will increase position uncertainty. By using a Kalman filter we can reduce noise and fuse sensor information to get a good estimate of the robot pose.

ii. **Inputs & Outputs**

Sensor Measurements (Sensor Input): Receives information from sensors scanning the environment

Agent Pose (Topic Publication): Publishes estimate of the robot pose

c. *Obstacle Detection*

i. **Functionality & Methodology**

The obstacle detection node has to detect obstacles and differentiate between other robots and unknown obstacles (such as dropped packages) in the environment. The robot will use sensor information such as LIDAR readings to scan for close objects. If it detects an object it will compare the position of the object with the positions of other robots in the vicinity to identify the obstacle. If another robot was seen, the obstacle will be ignored as the negotiating node will prevent collisions. If an unknown object was detected, that information will be provided to the path planner, which will plan a path around the obstacle. The obstacle information will also be provided to other robots so they can adjust their paths as well if necessary.

ii. Inputs & Outputs

Sensor Measurements (Sensor Input): Receives sensor information such as LIDAR or camera readings

Other Agent Pose (Topic Subscription): Receives position of other robots in the area

Dynamic Obstacle (Topic Publication): Publishes position of any unknown obstacle that was found

d. Negotiator / Central Planner

i. Functionality & Methodology

The task of the negotiator is to avoid collisions and resolve deadlocks between local robots. To do that it will employ a complete planner that will be executed by one robot in the collision system.

Each negotiator publishes a near-future chunk of the robot path. It doesn’t publish the whole path as it becomes more unlikely as time progresses. Publishing only near-future chunks prevents unnecessary replanning.

The negotiator of each robot constantly tracks the trajectories of other robots in the vicinity on possible collisions. If it detects a collision it will initiate a negotiation process. One robot will be chosen to calculate the plan (e.g. the one with the lowest id). It collects the local goals via the trajectory information of the robots involved in the collision and creates a new plan that avoids the collision.

After that it will test if the newly created trajectories collide with other trajectories in the network. If it does, it will replan taking the new robot in the collision-system into account.

It then broadcasts the generated paths to the other robots in the collision. If any robot receives a path command from another robot it has to obey that order and will continue its path planning starting from the end of the commanded path.

ii. Inputs & Outputs

Commanded Path Chunk (Topic Publication): Publishes the path chunk that resulted for the robot after the negotiation

Agent Path (Topic Subscription): Receives the currently planned path of the robot

Agent Pose (Topic Subscription): Receives the current pose of the robot

Agents Path Chunks (Topic Subscription / Publication): Publishes the currently planned path of the robot within a limited time scope

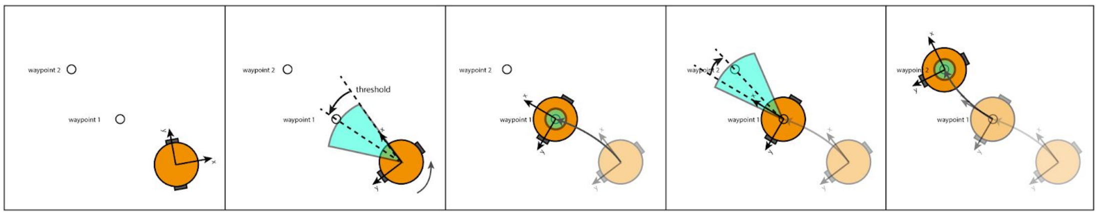
Path Command (Topic Subscription / Publication): The robot that was chosen to calculate the local central plan will publish the resulting paths to the corresponding other robots

e. Motion Planning

i. Functionality & Methodology

The task of the motion planning is to match the pose of the agent with the expected pose in the planned path at each time point. To do so it will calculate the difference between the current robot orientation and the orientation necessary for a direct heading towards the next waypoint. If the difference is above a threshold, the robot will turn on the spot until the difference is below the threshold. Otherwise the robot will start moving forward and correct any deviations from the trajectory using the cross-track-error in conjunction with a

PID controller. Once the robot is within a threshold area of the current waypoint, the goal waypoint will switch to the next waypoint and the process starts anew.



ii. Inputs & Outputs

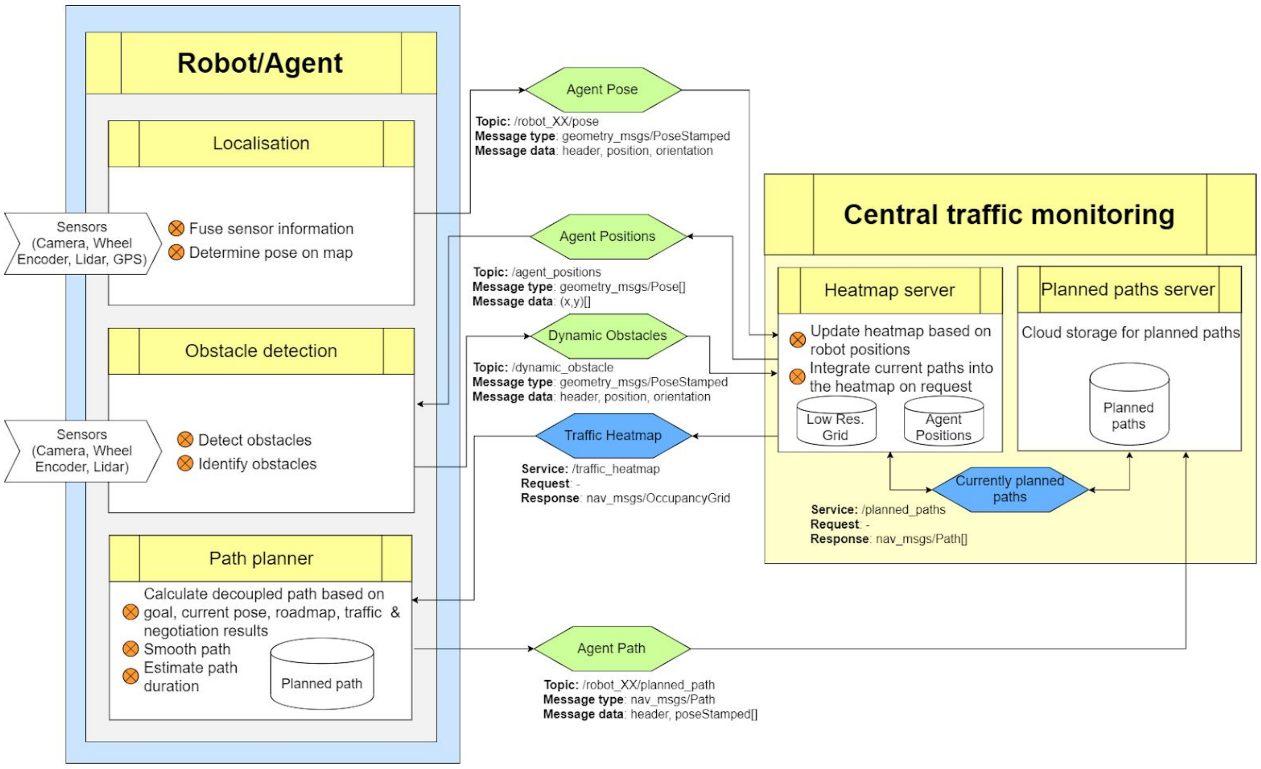
Agent Path (Topic Subscription): The path to follow

Agent Pose (Topic Subscription): The current estimation of the agent pose

Command Velocity (Topic Publication): Control commands to the wheels (linear & angular speed)

3. The traffic management system (hybrid architecture only)

The traffic management system is an extension for the decentralized architecture to improve the planning of the initial path the robot will follow. In the decentralized architecture robots will calculate the shortest paths only. This will lead to many robots taking very similar paths which will lead to conflicts. Instead the robots should take into account experience from the past and expectations of the future in order to avoid areas that are prone to traffic congestions. The traffic management will extend the area in which the robots will drive to avoid conflicts before they occur.


Robot/Agent
Localisation
Sensors (Camera, Wheel Encoder, Lidar, GPS)
Fuse sensor information
Determine pose on map
Obstacle detection
Sensors (Camera, Wheel Encoder, Lidar)
Detect obstacles
Identify obstacles
Path planner
Calculate decoupled path based on goal, current pose, roadmap, traffic & negotiation results
Smooth path
Estimate path duration
Planned path
Agent Pose
Topic: /robot_XX/pose
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Agent Positions
Topic: /agent_positions
Message type: geometry_msgs/Pose[]
Message data: (x,y)[]
Dynamic Obstacles
Topic: /dynamic_obstacle
Message type: geometry_msgs/PoseStamped
Message data: header, position, orientation
Traffic Heatmap
Service: /traffic_heatmap
Request: -
Response: nav_msgs/OccupancyGrid
Agent Path
Topic: /robot_XX/planned_path
Message type: nav_msgs/Path
Message data: header, poseStamped[]
Central traffic monitoring
Heatmap server
Update heatmap based on robot positions
Integrate current paths into the heatmap on request
Low Res. Grid
Agent Positions
Planned paths server
Cloud storage for planned paths
Planned paths
Currently planned paths
Service: /planned_paths
Request: -
Response: nav_msgs/Path[]

a. Heatmap Server

i. Functionality & Methodology

The heatmap server has a low resolution grid of the warehouse. At each time step it will receive the positions of the robots on the map. The map cells that relate to the positions of the robots will be set to a certain value (indicates high traffic). As the robots move around the heatmap will be filled with high traffic value cells. To give recent data more relevance, the initial value in the map will decay over time until it reaches zero again.

In addition to the robot positions the heatmap server will be provided with obstacle information. Obstacles will be treated just like high traffic areas. Robots will avoid them in their planning process. As the obstacle traffic value will decay over time, robots will move towards that path again, automatically checking whether the obstacle has been cleared.

On request the heatmap will be provided to the robots to guide them in their path planning process. Before the heatmap is provided to the robot the heatmap server will request the currently planned paths from the planned paths server. Each path will be integrated as traffic into the heatmap but as each path progresses farther into the future its traffic value will be discounted to take uncertainty into consideration.

An alternative to the map approach would be to just save current and recent positions of robots until they have fully decayed. That way you would only save cells that have non-zero data in it which can save transfer size and lookup time.

ii. **Inputs & Outputs**

Agent Pose (Topic Subscription): Receives the pose of each robot

Agent Positions (Topic Publication): Publishes the current positions of all agents

Currently planned paths (Service Request): Receives the paths that all robots plan to drive at the point of request

Traffic Heatmap (Service Response): Responds with the current traffic heatmap

b. Planned paths server

i. **Functionality & Methodology**

The planned paths server is used as a cloud storage for the planned paths of every robot. When a robot finished calculating its plan it will publish the planned path to the planned paths server. The storage will store one path per robot, so robots can update their paths by sending a new one.

When a robot plans a new path it will request the traffic heatmap. The heatmap server will then request the currently planned paths of all robots to include them in the traffic heatmap.

ii. **Additional Inputs & Outputs**

Agent Path (Topic Subscription): Receives a new plan a robot calculated

Currently Planned Paths (Service Response): Responds with all paths currently planned

c. Localisation (hybrid extension)

i. **Functionality & Methodology**

The localisation node is extended to provide the current pose estimation to the heatmap server at each time step.

ii. **Additional Inputs & Outputs**

Agent Pose (Topic Publication): Publishes the current estimation of the robot pose

d. Obstacle detection (hybrid extension)

i. **Functionality & Methodology**

The obstacle detection receives the position of all agents via the heatmap server to identify obstacles by comparing the obstacle position with known robot positions. This will prevent classification of a detected robot as an unknown obstacle. It will also provide the heatmap server with dynamic obstacle information to share it with other robots.

ii. **Additional Inputs & Outputs**

Agent Positions (Topic Subscription): Receives the positions of all agents

Dynamic Obstacles (Topic Publication): Publishes the position of dynamic obstacles

e. *Path planner (hybrid extension)*

i. **Functionality & Methodology**

The path planner will be extended to take past and future traffic information into account. The path will be generated with A* / Theta* based on the roadmap, dynamic obstacle information, recent traffic flows and planned paths of other robots. Obstacles, recent and future traffic flow will all be conveyed to the robot via the heatmap as high traffic zones. Obstacle information and high traffic areas will be encoded as high weights in the graph to discourage the algorithm to plan a path through that area without prohibiting it completely.

ii. **Additional Inputs & Outputs**

Traffic Heatmap (Service Request): Requests the current traffic heatmap

Agent Path (Topic Publication): Publishes a newly calculated path

5 Timeline

Hereby we planned the following Timeline for the project. The underlying logic is that we first would like to implement a minimal viable product, that is a centralized architecture with only necessary nodes, i.e. task assignment, path planning and smoot path following (w.r.t. the robot agent). Due to the modular architecture this can be later extended with Charging and Traffic Management. We also dedicated 3 weeks when we would like to decentralize our solution if needed or to optimize already implemented nodes.

Weekly Plan:

W5.: Finalizing and Submitting the SDS document

W6.: Task Assignment Node (TAA)
- a. Implement the node in the file architecture
- b. Implement the Hungarian/MURDOCH method of task assignment
- c. Implement all topic publishing and topic subscription between TAA and available nodes

W7.: Path Planning (Centralized)
- a. Implement the node in the file architecture
- b. Implement the PUSH_AND_SWAP algorithm of central, incremental path generation
- c. Implement all topic publishing and topic subscription between PP and available nodes
- d. TEST: environment with 1-2 robots getting tasks assigned and fulfilled

W8.: Low-level tasks (Motion Planning, optimization)

W9.: Low-level tasks (Obstacle detection)
- a. Implement the node in the file architecture
- b. Implement solution to read in data from the laser scanner and use it to:
  - i. estimate distance from an object
  - ii. distinguish between robot, package and environmental obstacles.
- c. Implement all topic publishing and topic subscription between Obstacle Detection and available nodes as in the architecture
- d. TEST: environment with 1 robot and “teleported obstacles”

e. Create demonstrative example to MVP and presentation

W10.: **Milestone 2:** Working Centralized Minimal Viable Product

a. Finalize the Optimization sprints, decide on used decentralization techniques according to what is feasible and needed

W11.: Charging Management

a. Implement the node in the file architecture
b. Implement message type and channel for new charge tasks
c. Finalize workflow decisions (what is the critical battery level, etc)
d. Implement all topic publishing and topic subscription between CM and available nodes as in the architecture
e. TEST: environment with 1 circulating robot (a charge task should disrupt the circulation)
f. TEST: using MVP extended with CM

W12.: Traffic Management

a. Implement the node in the file architecture
b. Implement “Learned Flow Map” algorithm if applicable and/or traffic pooling
c. Finalize workflow decisions (how traffic should bias the path planning)
d. Implement all topic publishing and topic subscription between CM and available nodes as in the architecture
e. TEST: extend solution with TM

W13.: Optimization 1 - Problems, Planning
W14.: Optimization 2 - Progress
W15.: Optimization 3 - Progress, Done
W16.: **Milestone 3:** Optimized full solution, Final Presentation
W17.: Writing Final Report
W18.: Final Report Submission

6 Frameworks and Tools

- Robot communication is implemented using ROS Kinetic.
- Simulation environment uses MORSE.
- ROS nodes are implemented in Python and C++ in case more performance is needed.
- Path Planning uses in the Centralized approach the “push-and-swap” algorithm (possibly being changed to “push-and-rotate” if needed). An implementation of “push-and-swap” is available at
- A* / Theta* will be used for initial local path planning in the decentralized approach, planning within a network will be done using the algorithms from our centralized approach
- Hungarian method / Murdoch algorithm for task assignment
- Traffic Management will either create learned Flow Maps [7] according to robot movements over time or use traffic pooling.
- Euclidean distance is used for finding the nearest available charging station in charging management.

7 References

[1] Hopcroft, John E., Jacob Theodore Schwartz, and Micha Sharir. "On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the" Warehouseman's Problem"." *The International Journal of Robotics Research* 3.4 (1984): 76-88.

[2] Parker, Lynne E. "Path planning and motion coordination in multiple mobile robot teams." *Encyclopedia of complexity and system science* (2009): 5783-5800.

[3] Švestka, Petr, and Mark H. Overmars. "Coordinated path planning for multiple robots." *Robotics and autonomous systems* 23.3 (1998): 125-152.

[4] Erdmann, Michael, and Tomas Lozano-Perez. "On multiple moving objects." *Algorithmica* 2.1-4 (1987): 477.

[5] Kant, Kamal, and Steven W. Zucker. "Toward efficient trajectory planning: The path-velocity decomposition." *The international journal of robotics research* 5.3 (1986): 72-89.

[6] Luna, Ryan J., and Kostas E. Bekris. "Push and swap: Fast cooperative path-finding with completeness guarantees." *Twenty-Second International Joint Conference on Artificial Intelligence*. 2011.

[7] Jansen, M. Renee, and Nathan R. Sturtevant. "Direction Maps for Cooperative Pathfinding." *AIIDE* 2008 (2008): 185.

[8] Yuta, Shin'ichi, and Suparerk Premvuti. "Coordinating autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots." *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*. Vol. 3. IEEE, 1992.

[9] Kato, Shin, Sakae Nishiyama, and Jun'ichi Takeno. "Coordinating mobile robots by applying traffic rules." *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. IEEE, 1992.

[10] Asama, Hajime, et al. "Collision avoidance among multiple mobile robots based on rules and communication." *Proceedings IROS'91: IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91*. IEEE, 1991.

[11] Matarić, Maja J. "From local interactions to collective intelligence." *The biology and technology of intelligent autonomous agents*. Springer, Berlin, Heidelberg, 1995. 275-295.

[12] Pallottino, Lucia, et al. "Decentralized cooperative policy for conflict resolution in multivehicle systems." *IEEE Transactions on Robotics* 23.6 (2007): 1170-1183.

[13] Clark, Christopher M., Stephen M. Rock, and J-C. Latombe. "Motion planning for multiple mobile robots using dynamic networks." *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. Vol. 3. IEEE, 2003.

[14] de Wilde, Boris, Adriaan W. ter Mors, and Cees Witteveen. "Push and rotate: cooperative multi-agent path planning." *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[15] Surynek, Pavel. "A novel approach to path planning for multiple robots in bi-connected graphs." *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009.

[16] Gerkey, Brian P., and Maja J. Matarić. "A formal analysis and taxonomy of task allocation in multi-robot systems." *The International Journal of Robotics Research* 23.9 (2004): 939-954.

[17] Bertsekas, Dimitri P., and David A. Castanon. "The auction algorithm for the transportation problem." *Annals of Operations Research* 20.1 (1989): 67-96.

[18] Kalyanasundaram, Bala, and Kirk Pruhs. "Online weighted matching." *Journal of Algorithms* 14.3 (1993): 478-488.

[19] Selmair, Maximilian, et al. "Scheduling Charging Operations of Autonomous AGVs in Automotive In-House Logistics." 2019

[20] Hamann, Heiko, et al. "Pick, Pack, & Survive: Charging Robots in a Modern Warehouse based on Online Connected Dominating Sets." *9th International Conference on Fun with Algorithms (FUN 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[21] Cassinis, Riccardo, et al. "Docking and charging system for autonomous mobile robots." *Department of Electronics for Automation, University of Brescia, Italy* (2005).

[22] Vincent, James. "Welcome to the automated warehouse of the future: How British supermarket Ocado is using robots to make online grocery shopping faster". The Verge, 8 May 2018, https://www.theverge.com/ 2018/5/8/17331250/automated-warehouses-jobs-ocado-andover-amazon.

[23] Crets, Stephanie. "Are the robots taking over?". Digital Commerce 360, 31 January 2019, https://www .digitalcommerce360.com/2019/01/31/are-the-robots-taking-over/.

[24] O'Brien, Mike. "DHL Investing $300M in Warehouse Robotics, Automation". Multi-Channel Merchant, 3 December 2018, https://multichannelmerchant.com/operations/dhl-investing-300m-warehouse-robotics-automation.

[25] Simon, Matt. "Inside the Amazon Warehouse Where Humans and Machines Become One". Wired, 6 May 2019, https://www.wired.com/story/amazon-warehouse-robots/.

[26] Oyster, Bay. "50,000 Warehouses to Use Robots by 2025 as Barriers to Entry Fall and AI Innovation Accelerates". ABI Research, 26 March 2019, https://www.abiresearch.com/press/50000-warehouses-use-robots-2025-barriers
-entry-fall-and-ai-innovation-accelerates.

[27] John Connors and Gabriel Hugh Elkaim "Trajectory Generation and Control Methodology for an Autonomous Ground Vehicle". National Science Foundation under NSF grant 0521675, Autonomous Systems Laboratory, Computer Engineering Department, University of California, Santa Cruz. 2008.