robotics

MDPI

Article

Multi AGV Coordination Tolerant to Communication Failures

Diogo Matos [1,*], Pedro Costa [1,2], José Lima [1,3] and Paulo Costa [1,2]

[1] Centre for Robotics in Industry and Intelligent Systems (CRIIS)—INESC TEC, 4200-465 Porto, Portugal; pedrogc@fe.up.pt (P.C.); jllima@ipb.pt (J.L.); paco@fe.up.pt (P.C.)
[2] Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal
[3] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, 5300-252 Bragança, Portugal
* Correspondence: diogo.m.matos@inesctec.pt

Abstract: Most path planning algorithms used presently in multi-robot systems are based on offline planning. The Timed Enhanced A* (TEA*) algorithm gives the possibility of planning in real time, rather than planning in advance, by using a temporal estimation of the robot's positions at any given time. In this article, the implementation of a control system for multi-robot applications that operate in environments where communication faults can occur and where entire sections of the environment may not have any connection to the communication network will be presented. This system uses the TEA* to plan multiple robot paths and a supervision system to control communications. The supervision system supervises the communication with the robots and checks whether the robot's movements are synchronized. The implemented system allowed the creation and execution of paths for the robots that were both safe and kept the temporal efficiency of the TEA* algorithm. Using the Simtwo2020 simulation software, capable of simulating movement dynamics and the Lazarus development environment, it was possible to simulate the execution of several different missions by the implemented system and analyze their results.

Keywords: multi-AGV control; path planning; Timed Enhanced A*; tolerance to communication faults

Citation: Matos, D.; Costa, P.; Lima, J.; Costa, P. Multi AGV Coordination Tolerant to Communication Failures. *Robotics* **2021**, *10*, 55. https://doi.org/10.3390/robotics10020055

Academic Editor: Xinjun Liu

Received: 7 February 2021
Accepted: 23 March 2021
Published: 27 March 2021

1. Introduction

The constant technological development felt at present creates a need for a constant adaptation on the part of the industrial sector in order to fulfil the demands of the corporate market. To remain competitive, industries must generate a demand for new innovative solutions in an attempt to create value. These solutions do not always reflect a direct valorization of the final product. In most industries, the production costs have a significant impact in their market competitiveness, which leads to a significant evolution of the automated systems. These systems grant the possibility of reducing the labour cost and simultaneously optimizing production time. As a product of this evolution, Automated Guided Vehicles Systems (AGV) were created and saw their first use in an industrial environment in 1954. Since then, the use of this type of system has seen a steady increase, and is a common sight in industry at present [1].

Currently, the AGV are predominantly used in the moving of products, being mainly used as a mean of transporting material between the production lines and the storage sectors. However, their use is not restricted to the industrial sector; AGV, for example, can also be used in hospitals and distribution centres. The mass use of AGV creates questions about their efficiency and productivity in scenarios where multiple robots are operating in restricted environments and exposed to communications faults.

The coordination of a fleet of autonomous vehicles is a very complex task, most multi-robot systems rely currently on static and pre-configured interactions between the robots [2]. When the unpredictability associated with communication flaws is added, this task becomes even more difficult. Due to this fact, the study of trajectory planning

algorithms allied to methods of detection and mitigation of communication faults, has increased in the later years. These algorithms use the robots' localization information, either predicted or measured, to control the traffic of the robot fleet, and attempt to plan safe and optimized routes in an industrial environment.

This work provides an implementation of a traffic control system, in a multi AGV environment, based on the TEA* algorithm [3] and robust communication faults. The main focus consists of obtaining a cooperative movement between all robots that are part of the system, avoiding, simultaneously, any situation that could lead to a mutual block and subsequently lead to the not finishing all the assigned tasks. This system should also be able to operate efficiently when subjected to communication faults of one or more robots. This system should also keep the efficiency and time optimizations offered by the TEA* in [4], expanding the use of this algorithm to environment where communication flaws are common, by using binary semaphores [5] to control the access to areas affected by communication faults.

Until this point, the TEA* algorithm has never been implemented in an industrial environment where both delays in robots movements as well as communication faults are common occurrence. As such, the paper's main contribution is the implementation of a traffic control system based on this algorithm capable of operating under these circumstances. This system will be composed of two parts: the TEA* planning algorithm and the Supervision module.

The rest of this paper is organized as follows: Section 2 presents a brief overview of other works related to the topic. Section 3 presents the overall system architecture as well as the lower level control implemented in each robot. Section 4 details the modifications implemented to the TEA* algorithm. The supervision module will be presented in Section 5. Section 6 will validate the performance of the implemented system. Finally, a few concluding remarks are made in Section 7.

2. State of the Art

The coordination of a fleet of robots falls under the *Multi-Agent Path Finding* (MAPF) category of problems. This problems comprise of finding a set of paths for the agents that are encompassed by the system. These paths have the objective of moving the agents from their current vertices to their targeted vertices while avoiding conflicts and reducing as much as possible the cost for the movement of the agents [6]. MAPF has practical applications in video games, traffic control, and robotics [7]. Solving MAPF problem in an optimal way is NP-Hard (non-deterministic polynomial-time hardness) in terms of complexity [8,9].

Presently, several different methods are commonly used to plan the movement and actions of fleet of mobile robots. This methods can range from the use of potential fields [10] to the use of Reinforcement Learning [11,12].

From all these methods, the ones based on Graph Search algorithms are of special importance not just because they can discover the possible paths but also because they allow the discovery of the most efficient possible paths for the robots [13].

As the name itself suggests, these methods consist of two stages: the initial decomposition of the workspace into a graph, and the application of a graph search algorithm in order to discover efficient paths for the robots. The initial decomposition of the workspace, in most cases, can be easily achieved by either using an approach based on Roadmaps [14,15] or Cell Decomposition [16].

After obtaining the graph, necessary the implementation of a method of searching it is necessary. Of particular importance in this field is the A* algorithm and its derivatives [17]. The standard A* algorithm, introduced by [17], uses both the cost already incurred from the initial node and the cost until the finish node. This characteristic allows this algorithm to obtain optimal and complete solutions.

Later, new variants of the A* algorithm were developed with the intention of being used in Multi-AGV systems. Such as the Dynamic A* algorithm proposed by [18], this

algorithm stands out as it is capable of operating in situations where the environment is unknown or just partially known.

Another notable variant is the Lifelong Planning A* algorithm, also called LPA*, proposed by [19]. This algorithm allows the use of previous information in order to reduce the processing time. More recently, in 2011, the 3D A* algorithm was introduced by [20], which adds the temporal dimension, allowing for a coordinate representation of (x,y,t). This detail allows for the representation of stationary movement, where the x and y coordinates maintain the same through iterations and the t coordinate is increasing.

Another variant of the A* algorithm is the M* [21], this variant was created since planning trajectories for large numbers of robots is very computationally expensive. Therefore, the M* algorithm allows for the creation of a relatively cheap path for the robots by sacrificing a small portion of the overall efficiency of the system. Although this method does not generate the most efficient solution possible for the robot routing problem, it generates good enough solutions that avoid conflicts between the robots.

Recently, a new graph search algorithm was introduced for the implementation in Multi-AGV systems. The Time Enhanced A* algorithm, also known as TEA*, introduced by [3] and later modified by [4], is a graph search algorithm created due to the need for an algorithm suitable for Multi-AGV systems, which avoids collisions, deadlocks and guarantees the efficient execution of a set of tasks [3]. The TEA* algorithm was created by introducing the concept of temporal layers to the A* algorithm, which gives the possibility of the algorithm of executing the path planning in an online mode [3] using the estimation of the position of the robots for any given time instance. This makes the TEA* algorithm capable of taking into account possible changes that can occur in the environment, unlike its predecessor. The TEA* algorithm is also capable of dynamically shifting the priorities of each robot in order to resolve possible conflict situations [4], unlike the static priorities used by [22].

All the methods based on graph search algorithms referred to above rely on a centralized architecture [2,23], which expands on the idea of a decentralized multi-robot system by using a service-oriented architecture. However, this idea is still in its infancy.

In the field of tolerance to communication faults, most advances are focused either on regeneration of communication faults [24] or in applying estimation methods to deal with sudden sporadic communication faults [25]. However, there are few studies on multi robot systems that are capable of dealing both with sporadic faults, as well as, dealing with entire zones of the environment where there is no communication with the robots.

3. Overall System Architecture and Lower Level Control

To better comprehend the function and implementation of both the TEA* algorithm and the supervision module, the system's overall architecture is explained, as well as its overall composition.

The implemented system is based on a centralized control architecture [23]. This configuration was chosen, since the TEA* Graph Search Algorithm needs prior knowledge of all the current robots positions in order to correctly generate a set of paths for all the robots controlled by the systems [4].

The implemented system is responsible for controlling the movement of a team of robots. This team of robots would consist of four robots similar to the ones used in the Factory lite 2019 edition [26]. These robots locomotion system is made up of two differential traction wheels and are simulated using the Simtwo2020 simulation software (CRIIS, Porto, Portugal), created by Paulo Costa. This software is capable of simulating differential traction robots [27], as well as already having implemented the simulation model for the robots from the Factory Lite competition [26], these robots were developed and built by Centre for Robotics in Industry and Intelligent Systems (CRIIS).

In Figure 1, a top level representation of the implemented system is shown. As seen in this figure, the implemented control system is comprised by three core modules. The TEA*

adapted library and supervisor modules located in an application that represents the central control unit and the trajectory control module located in each robot control unit. Information is transmitted between the four robots and the central controlling application via the UDP/IP communication protocol.

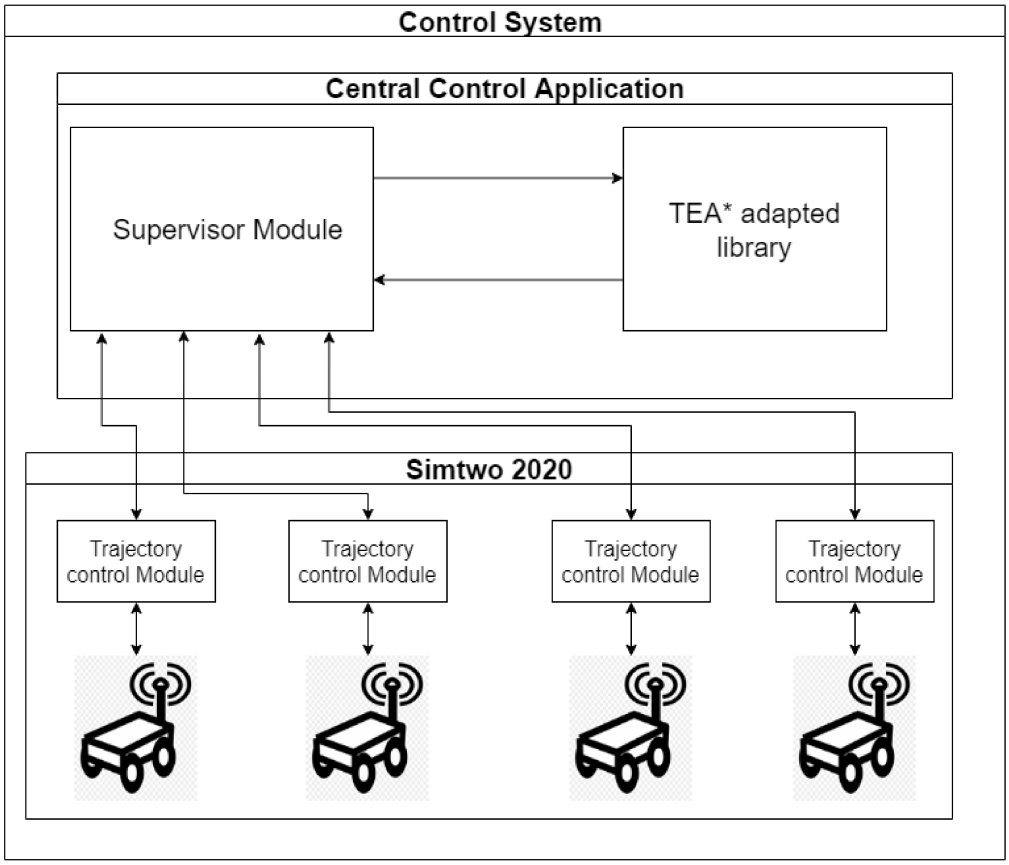

Figure 1. Top level view of the Control system (TEA*—Time Enhanced A*).

The central control unit uses the interaction between the supervision module and the modified TEA* algorithm, to control each robot planned path, this path is then transmitted to the robot. The robot control algorithm is only responsible for controlling the velocities of each of the robots wheels, so that it can accurately execute its assigned path.

The simulation of communication faults in this simulated environment was executed by a special module that was implemented between the simulation platform and the centralised control platform. In the case of static communication faults, this module will analyze the position of the robot and checks if it is within the area defined by the user. If this condition is verified, the module will not transmit any data regarding that robot to the control system. For sporadic faults, this module uses a fixed probability to decide whether to transmit or not the data to the central control unit. It is also capable of simulating sporadic faults within a defined area or in the entire environment. All the data used by this module is isolated from the rest of the system to maintain the validity of the experiment results.

4. Modified TEA* Algorithm

As described before, several modifications to the algorithm presented by [4] were implemented during the design and creation of the central control unit. Three major modifications were implemented into this algorithm, these modifications are the following:

- Implementation of a Task Scheduling function;

- Modification of the method that associates the current robot position with a node from the graph;
- Implementation of an algorithm that creates a binary semaphore when a communication fault is detected.

These modifications had the objective of allowing the application of the algorithm into industrial scenarios, and also allowing the system to safely plan paths for the robots during communication faults.

4.1. Task Scheduling

The main difference between the method used by [4] and the implemented Task Scheduling function, was that the later would move the robot to a charging post when all of the tasks assigned to the robot were concluded. This would allow for the robot to recharge while no new tasks are assigned to it.

To make the robots move to their charging stations when no tasks are assigned to them, both the function that obtains the next task, as well as the function that adds new tasks to the robot need to be modified.

When a robot reaches its target point, this new function checks if there are anymore tasks assigned to the robot and in the case that there are no more missions, it orders the robot to move to the closest recharging station; this attribution is represented via a flowchart in Figure 2.

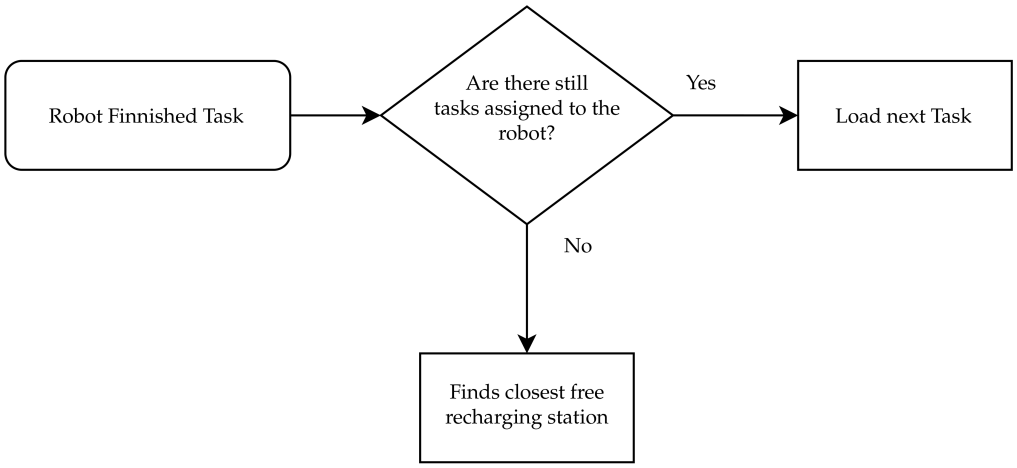

Figure 2. Flowchart representation of the attribution of charging stations to the robots

When a new task is added, the system checks whether the robot is either moving to or stopped at a charging station or if the robot is executing another task. If it ends up being the latter, it will add the new task to the subtask array as the last element and will also increment the number of tasks. Otherwise, it will change the target point to the new task node and add that task to the subtask array. It is also necessary to increment both the current task point as well as the number of tasks.

At the beginning of each planning cycle the nodes corresponding to the positions of all robots that are currently charging, will be considered to be "Obstacle Inaccessible Robot". This new node status represents a node that is currently occupied by a robot which is currently recharging or that has suffered a communication fault, therefore meaning that this robot cannot be moved to give way to another robot. This allows the planning of path around the position of these robots and avoiding the movement of these robots.

For this reason, the recharging positions should be placed in locations where they will not block traffic, such as at the extremities of dead end pathways. After a robot has completed all of its tasks, the robot will move to the nearest unoccupied recharging station and will remain there until another task is assigned to it.

4.2. Planning during Communication Faults

It is also necessary to modify the TEA* library so that it is able to plan safe and efficient paths even when one or more robots are experiencing communication faults. The library created by [4] does not take into account this possibility, therefore, when one or more robots are subjected to communication faults, it is not able to guarantee a safe and efficient planning, having a high probability of occurring collisions and deadlocks.

The method implemented in this project, in order to create a system robust to communication faults and to areas where there is no communication, consists in the interaction between both modules of the central control unit.

Each communication fault will have a set of nodes associated with it, these nodes are generated by the supervision module and their generation is explained in Section 5. Then the modified TEA* library places the nodes associated with the current active faults as immovable obstacles, for all robots and during all steps of the planning, until the affected robot exits the communication fault area and communication can be re-establish. This prevents any other robot from entering the communication fault zone. Therefore, creating a binary semaphore [28] that regulates the traffic of robots entering and leaving the zone. The new library will also not plan any new paths for the robots experiencing communication faults, this serves to reduce the risk of the TEA* algorithm inadequately assuming that these robots are executing a different path than the one they are actually taking, which could lead to a loss of efficiency when planning the paths of the remaining robots.

This additional control is important since when a robot is experiencing a communication fault situation there are no guarantees that any alterations to the planned path will reach and be executed by the robot, therefore it is necessary to limit the presence of several robots inside the same communication fault zone. However, in some cases this is impossible as it is shown in Section 6.

5. Supervision Module

This module is responsible for controlling when the robots paths need to be replanned and for detecting, measuring and handling communications faults. This module is composed by two sub-modules hierarchically related with each other. Each of these sub-modules is responsible for handling one of the tasks mentioned earlier. These sub-modules will be referred as the Planning Supervision Sub-Module (PSSM) and the Communication Supervision Sub-Module (CSSM), respectively.

The Figure 3 represents the hierarchical relation between all the parts that constitute the implemented control system. Represented in blue in this figure are both the sub-modules that constitute the System Supervision Module. In this figure, it is also possible to observe that the CSSM is located above the PSSM, showing that the CSSM commands have a higher authority than the PSSM ones.

The relation between the two sub-modules is based on a loop and interrupt-based control. On environments where no communication faults occur the CSSM does not intervene and the replanning of paths is handled by the PSSM. In this case the CSSM's only job is to detect that a communication fault is occurring. However if a communication fault is detected, the PSSM is overruled and the replanning of the robots paths is then controlled by the CSSM. When the fault has ceased, the control is handed back to the PSSM.

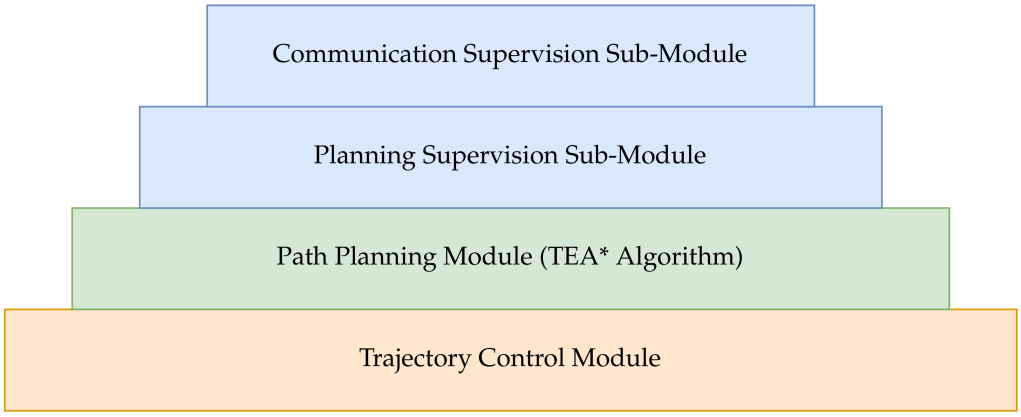

Figure 3. Representation of the hierarchical relation between all the parts of the system.

5.1. Planning Supervision Sub-Module (PSSM)

This sub-module is responsible for controlling when the robots need to have their path replanned in situations where no communication fault is detected. In order to accomplish this task, it needs not only to detect when one of the robots is delayed or ahead of time but also detect when a robot has completed the current step of his path.

Firstly, it is necessary to define when a robot is considered delayed or ahead of time. When using the TEA* algorithm, it is necessary that all robots are synchronized, meaning that all robots must be in the same step of their path, in order to avoid collisions and deadlocks. Therefore, delayed and ahead of time robots are robots that are one or more steps behind and one or more steps forward than the remaining robots.

However, in practice, this is not so linear since it would be impossible to synchronise all the robots in the system, thereby originating a very ineffective system where the paths are constantly being replanned. For this reason, a detection system must be created that allows small delays when those delays do not constitute a problem for the overall safety of the robot. To accomplish this goal, the PSSM executes three different verifications every time the robots position is updated. These verifications are useful to decide if it is necessary or not to replan the paths for all robots .

- Check whether any robots are too distant from their supposed position;
- Check whether the maximum difference between steps is 1;
- Check whether any robot is moving to a position currently occupied by another robot.

However, before any of these verifications can be executed it is necessary to know the type of movement the robot is executing. This movement could be classified into one of three categories:

- Robot is stopped;
- Robot is rotating;
- Robot is currently moving along a link.

After obtaining the type of movement of the robot, the sub-module checks all of the robots current steps in order to obtain the lowest step value possible. It then compares the current robot coordinates to the coordinates that the robot would have if it was currently executing the same step as the robot with the lowest step value. If the distance between this coordinates is greater than the stipulated threshold, the robots paths are replanned.

However, this function only checks for robots that are currently moving between nodes, not taking into consideration any delays during the rotation or even during communication with the robot. Another problem of this function is that it assumes that all the links have a dimension superior to the stipulated threshold.

Therefore, another function is also required to guarantee that the robots that are rotating or stopped are also synchronized with the other robots, as well as, guarantee that the synchronization and safe planning of paths for the robots can also be executed, even if

the link dimension is shorter than threshold distance. This function checks whether the maximum step different between all the robots in the system is superior to one. If this condition is verified it means that the robots are not synchronized therefore their paths must be replanned.

Even in this example, the desynchronization was artificially induced; in an industrial environment it can be caused by a lot of factors, from small differences in the wheels diameter to differences in the grip coefficient.

In some cases, even a small desynchronization can cause a collision, as illustrated in Figure 4. In these cases, the most viable solution in order to maintain the safety of the system is to constantly replan the robots' paths as soon as the position of one of the robots is updated. Therefore the sub-module must detect these cases and act accordingly.

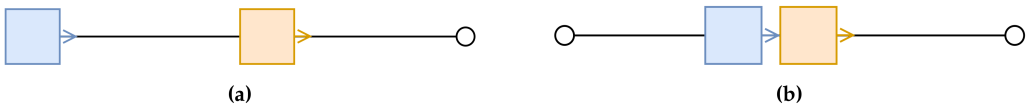

Figure 4. Example of a small desynchronization leading to a collision. The subfigures a,b show the movement of the robot in this example, these subfigures use square to represent the location and orientation of the different robots. In this figure the circles represent the different nodes associated with that part of the map. (**a**) Initial position of the robots; (**b**) Collision originated by a delay of the movement of the orange robot.

To detect these specific cases, a new function was implemented; this function checks whether the node that corresponds to the current step of the robot path is equal to the current position of any of the remaining robots. In the case of a robot being currently located in a link between two nodes, the function checks the coordinates of both nodes that are part of the link, in order to guarantee that no other robot is moving to those nodes. If any of the described conditions happen, the robots paths are immediately replanned so that collisions are avoided.

When no replanning of the robots path is being executed, this sub-module needs to detect when a robot has completed its current path step and increment that step. However, these points can be a problem in the case where some robots are waiting for other robots to finish their movement. Therefore, it is necessary to stipulate when the steps of these robots can be increment in a safe way.

If the robot movement is a rotation or if the robot is travelling between two nodes, determining when the robot has finished its current step of the planned path can be done by comparing the current robot position and orientation with the desired robot position and orientation at the end of that step.

In the case where the robot has stopped and is waiting for other robots to finish their movement, the robot step counter only increments when all moving robots have finished that step, preventing the movement of the robot before the path is cleared. This method assumes the worst case scenario where the movement of the stopped robot depends on the movement of all the moving robots of the system. It would be relatively costly in computational terms, to check all of the systems robots path and determine which robots needs to move in order to allow the movement of the stopped robot.

5.2. Communication Supervision Sub-Module (CSSM)

In order to create a system robust to communication faults the Communication Supervision Sub-Module (CSSM) was implemented. It is responsible for detecting communication faults, calculating their size and forcing the replanning of the robots paths to take into consideration said faults. This sub-module will predominantly be faced with two different fault situations:

- Situations where one area in the factory floor map consistently has no communication with the central control unit;
- Situations where temporary loss of connection happens with the central control unit.

In either case, the system will not have any prior knowledge of the existence and location of these faults. For this reason it will have to constantly map the factory floor map, since faults can be dynamically created and destroyed. This allows the modified TEA* algorithm to then plan a path accordingly, as explained in Section 4.2. Figure 5 shows a flowchart of the normal functioning of this sub-module.

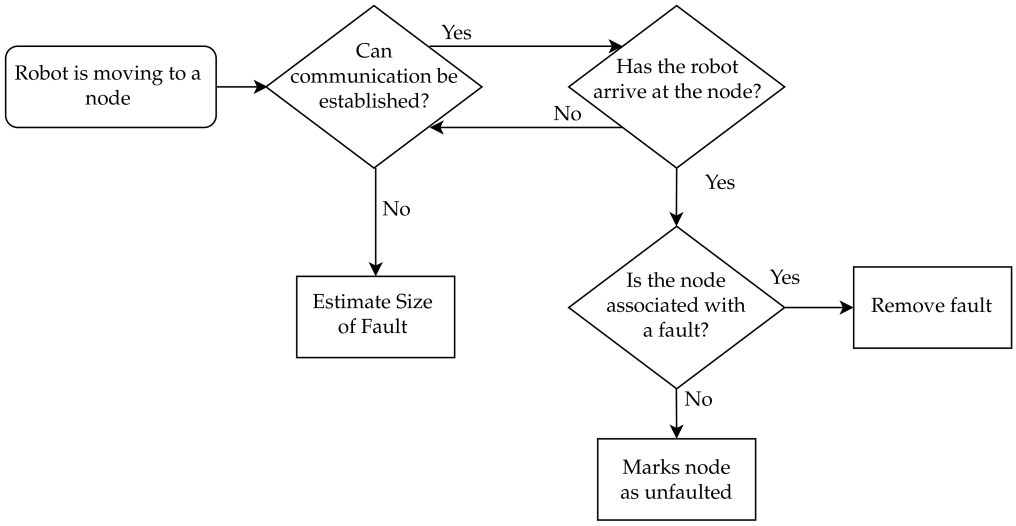

Figure 5. Flowchart representation of the Communication Supervision Sub-Module.

On the detection of a communication fault in one or more of the robots, the CSSM generates an interrupt, where the normal functioning of the central control unit is halted and the size of the fault is calculated. Afterwards, it forces a new replanning event, in order to take into account the estimated fault size; and return the system back to its normal functioning. During this replanning event, the nodes associated with the active faults are placed as occupied, therefore preventing other robots' entry into the zone affected by an active fault.

As shown in Figure 5, while the robot is unaffected by communication faults the current position of the robot is recorded has locations where it is possible to establish communication with the robot. These nodes will be referred to as unfaulted nodes and will be used in the estimation stage of the method to delimit the area subjected to faults. However, this designation does not mean that it is always possible to establish communication with a robot located in them. These nodes can still suffer from sporadic faults, as well as the appearance of new communication faults in areas where previously communication could be established. Therefore, their status as unfaulted nodes may not be permanent. Figure 6 shows an example of the mapping of unfaulted.

The estimation stage of this method happens when a robot enters a fault state. When entering this stage there are three possible situations that can happen: either the robot is entering a fault that is still not mapped, it is entering a fault that has already been mapped previously, or it is entering a unmapped extension of a previously mapped fault. Each of these situations is treated differently, as shown in Figure 7.

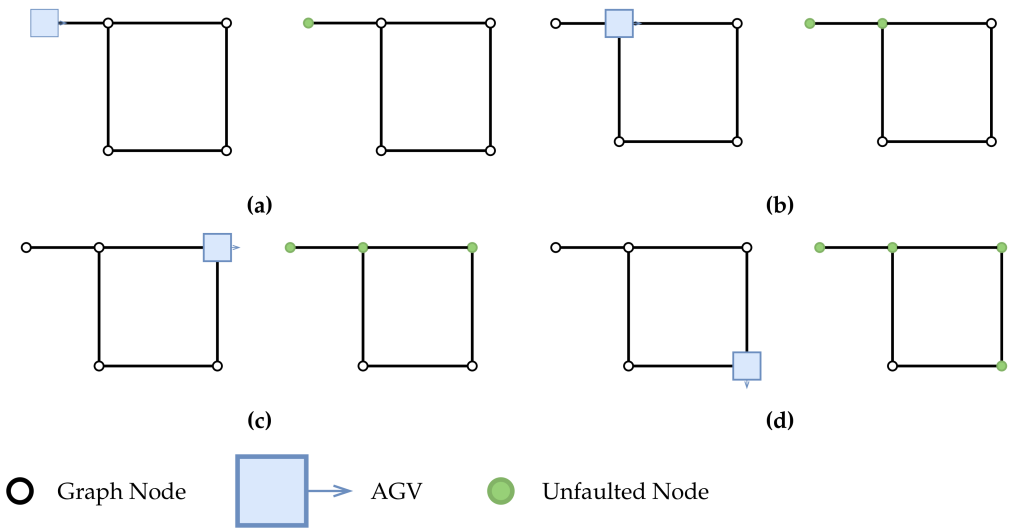

Figure 6. Example the mapping of unfaulted nodes. The progression of this mapping is shown via the subfigures a,b,c,d showing the Autonomous ground vehicle (AGV) movement on the left image and the status of the mapped nodes on the right one. (**a**) Initial state of the system; (**b**)System state at the end of the first step; (**c**)System state after the second step; (**d**) System state after the third step.

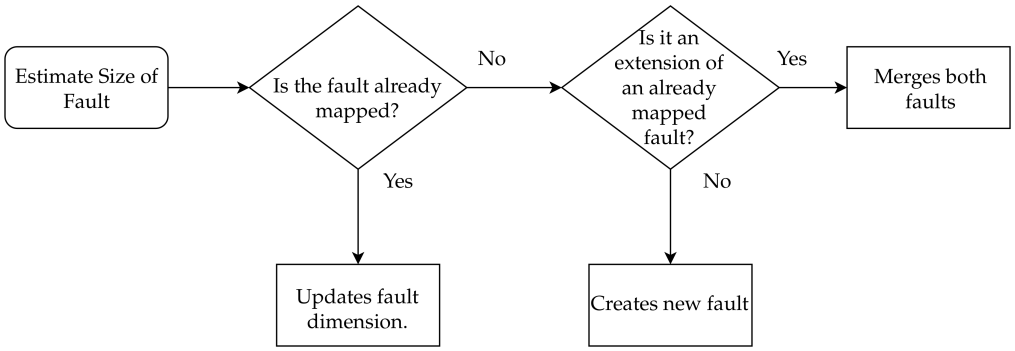

Figure 7. Flowchart representation of the different types of faults that the estimation method can encounter.

To estimate the dimension of a fault, the planned path for the robot is analyzed, so that it can determine the most likely exit node, node where communication can re-establish. To achieve this, each of the nodes corresponding to the next step of the planned path is sequentially compared with the records of the unfaulted nodes. This comparison goes on until either one of the path nodes is found to be an unfaulted node or the planned path ends. Afterwards, this method records all of the nodes that comprised the robot path between the current robot position and the exit node as a set of faulted nodes. Each set of faulted nodes therefore represents an area where no communication can be established with the robots, and have associated with it an array of possible entry/exit nodes. After the structure of nodes that represent the fault is created, the id value of the robot that detected it is associated with the fault.

In this method, when a fault is detected in the middle of a link, the entry/exit points of a fault zone are chosen in a worse case scenario; therefore these points are the last known nodes with good communication with the robot and the first point where communication could be re-established, respectively.

This method also accounts for the fact that a robot can exit sooner than the estimated exit node, especially in a situation where only a small percentage of the graph node has already been mapped. In these cases, an interrupt is generated when communication is re-establish with the robot. During this interrupt, both the faulted node set, as well as the

entry/exit node set associated with that fault are corrected. This correction is executed by removing from the faulted node set all the robots path nodes that are between the actual exit point and the estimated exit node. The status of the removed nodes are changed to *not mapped*. Finally, the node where communication was re-established is placed into the entry/exit set. On exiting the fault, the robot will no longer be associated with that fault.

Figure 8 shows an example of the detection and mapping of a new fault; in this example the nodes represented in green are nodes considered to be unfaulted nodes, the nodes represented in orange are part of the entry/exit set associated with this fault and the nodes represented in red are faulted nodes associated with this fault. In this figure, the button graph representation serves to illustrate the current status attributed to each node.

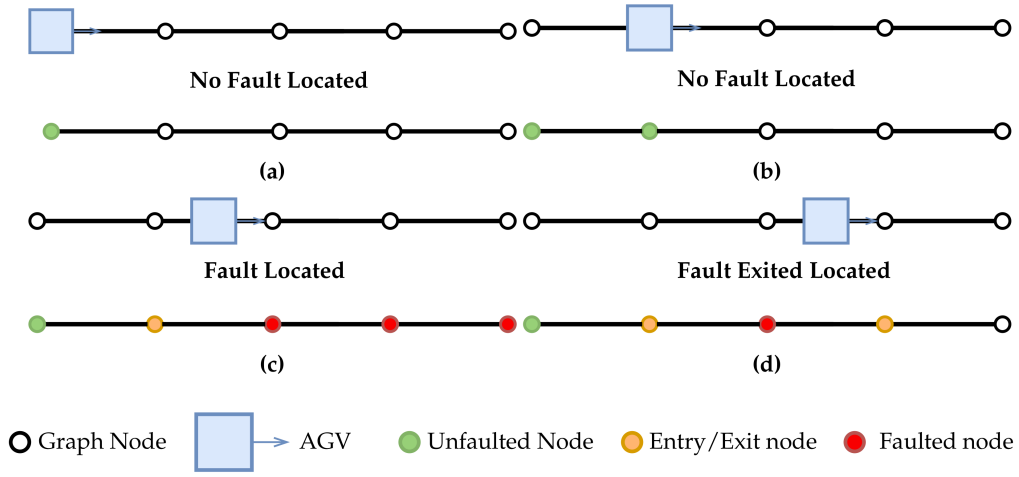

Figure 8. Example the mapping of a new fault. The progression of this mapping is shown via the subfigures showing the robot movement on the top image and the status of the mapped nodes on the bottom one. (**a**) Initial state of the system; (**b**) Mapping status and robot location at the end of the first step; (**c**) Mapping status and robot location when the communication fault is detected, initial estimation of the fault size; (**d**) Mapping status and robot location when communication is reestablished, readjustment of the fault size.

After the estimation of the fault dimension, the group of nodes associated with that fault is analyzed to see if they belong to an already existing fault or if they can be associated with an already existing fault as an extension of it, as shown in Figure 7.

First, the method starts by checking whether at least two of the nodes marked as entry/exit nodes are also entry/exit nodes of an already mapped fault. If this is not the case, the method will analyze each of the nodes that comprise the affected node set, individually and see if they belong to other already mapped faults. If any of these conditions are true both node sets are merged. In the rare cases where the estimated fault nodes sets are associated with more than one already mapped fault, all the faults associated with the estimated fault are merged into only one fault.

The mapping methodology presented is a methodology where the efficiency is directly proportional to the time spent, by moving the robots. An increased movement of the robots to different nodes would lead to more nodes being mapped, therefore leading to a more efficient estimation of the affected area upon the detection of a fault. To deal with sporadic faults, this idea was of an increase in efficiency with the passing of time was improved upon.

For starters, the CSSM would only assume that a robot is in a communication fault status, if there is no new message received after a time equivalent to the execution to four of the robot control cycles has passed since the last message from that robot was received. In the implemented system, this means that a gap of approximately 400 ms elapsed, taking into account the central unit control cycle period, before any action is taken. In terms of the

robots' movement, this gap is too small, when used in conjunction with the PSSM, to cause any collisions or any significantly loss of efficiency, as shown in Section 6. Assuming a linear nominal velocity of 10 cm per second the gap would correlate with a movement of 4 cm if the robot was going full speed.

During this gap, the robot currently experiencing the fault will attempt to resend its data at least three times, therefore reducing any affect that random packet loss could induce a communication fault situation.

Applying the idea of increasing the efficiency of the algorithm with the passing of time, a method for dealing with sporadic faults was implemented. This method is comprised of two parts, the first one is executed every time a communication is established with the robot in a node that was previously flagged as suffering communication faults and the second one is executed when a communication fault is detected in an unfaulted node.

The first part of this function verifies whether the node where the robot is communicating from is considered a faulted node. If so, the fault associated with that node is removed and all the faulted nodes from that are placed as not mapped, except for the node from where the communication originated which is placed as unfaulted. All the entry/exit nodes associated with that fault are still kept as unfaulted nodes. Figure 9 shows an example of the removal of a fault; in this figure both the faulted nodes, represented in red, originated from to a sporadic fault that is currently inactive. Therefore, when a robot enters this fault it does not experience any communication loss, this leads to all the nodes being considered to be unfaulted nodes, represented in green.

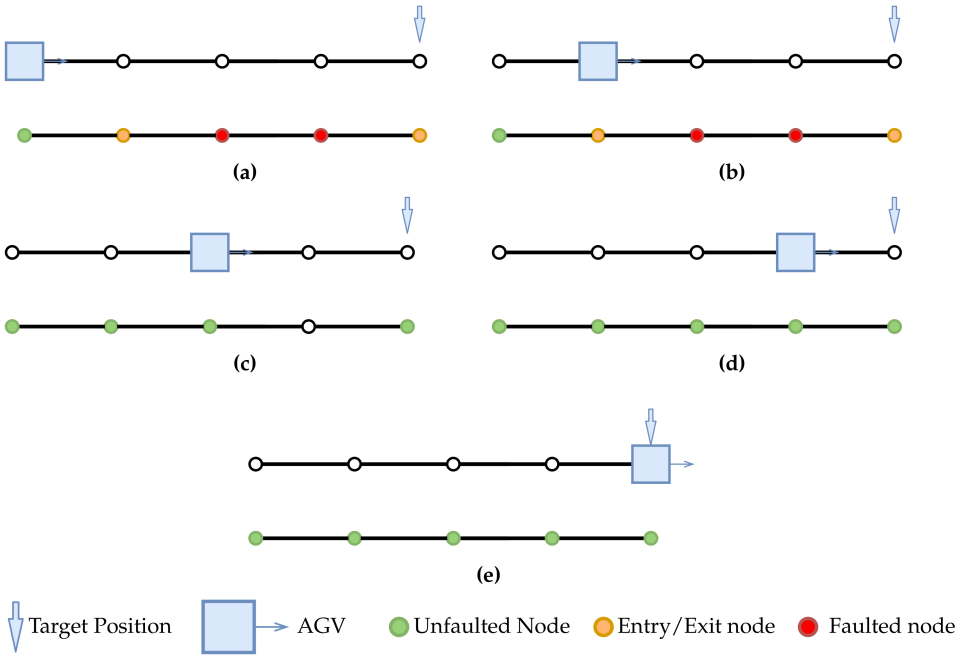

Figure 9. Example of the removal of fault.(**a**) Initial state of the system; (**b**) Mapping status and robot location at the end of the first step; (**c**) Mapping status and robot location after the robot enters the area previously affected by a communication fault, the nodes associated with that fault are placed as not mapped; (**d**) The robot proceeds to remap the nodes; (**e**) System state when the robot reaches its target point with all of the nodes fully mapped.

This first part serves to remove any sporadic faults that might occur. This removal is especially important for the efficient planning of the paths since it avoids the unnecessary creation of binary semaphores.

In situations when a sporadic fault occurs on an entry/exit node of another already mapped fault, the sporadic fault is associated with that fault. When the sporadic fault is removed, the nodes of the already mapped fault are placed as not mapped. This allows for

the storing of the dimension associated with that fault, permitting an efficient remapping of the fault if a robot enters it.

The Figure 10e shows an example of the removal of a sporadic fault that has been merged with a fault that corresponds to an area where there is no communication with the robot. As with Figure 9, the robot starts by placing both the faulted nodes as unmapped nodes. However, when the robot enters the second unmapped node it experiences a communication fault for being in an area inaccessible by the communication network Figure 10d, therefore the fault dimension estimation algorithm is called leading to the remapping of that fault.

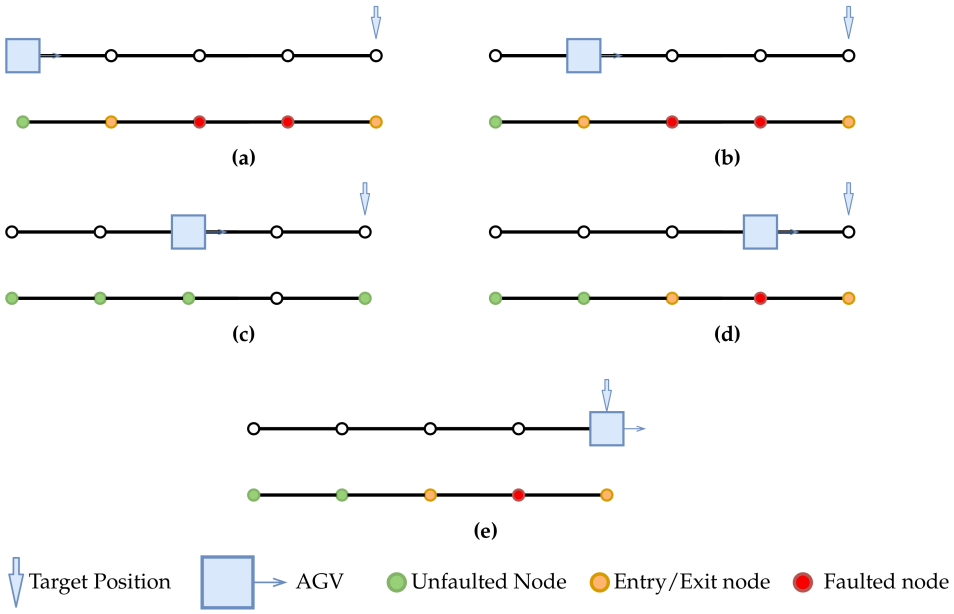

Figure 10. Example of the removal of a sporadic fault in an entry/exit node. (**a**) Initial state of the system; (**b**) Mapping status and robot location at the end of the first step; (**c**)The robot enters an area previously affected by a communication fault, since no fault is detected all nodes associated with the previously mapped fault are placed has not mapped; (**d**) The robot proceeds to map the remaining nodes that are still affected by communication faults; (**e**) Final system state, the new size of the fault is now fully mapped.

The second part of this function runs when a fault is detected in an unfaulted node. When this happens, this fault is assumed as a sporadic fault, so its dimension is limited to the node where it happened. This node is removed from the unfaulted set and the mapping methodology is applied. Similar to the static faults when communication is reestablished with the robot, the size of the fault is updated.

6. Tests and Validation

In this section, all the implemented modules as well as the interaction between them will be tested and validated. It will also present the situations where this system is incapable of guaranteeing a safe execution of the mission.

These tests will be divided into two categories, those with static communication faults and those with sporadic and static communication faults. The results of these tests, as well as other tests executed to the system, are represented in the appendixes of this document. For the representation of the results of these tests, a methodology where the robots are represented by a full square, their destination point is represented by an empty square of the same colour as the robot and their executed path between images is represented in the same colour as the robot. For ease of comprehension, numbers with the same colour as the robot have been added to the figures, in order to demonstrate the task order associated with each robot.

For these tests, four robots will be used. Each robot will be assigned a set of tasks that it needs to complete. When all tasks assigned to all of the robots that comprise the system are completed and all robots have reached their charging post the test is deemed as finished.

All tests were executed using the same factory floor map. This layout was inspired by a real-life factory.

6.1. Planning Supervision and Overall System Performance when Subjected to Static Communication Faults

The first category will analyze the behaviour of the system when only exposed to areas where there is no communication. The second set of tests will expose the system not only to areas where there is no communication but also to packet loss and sporadic communication faults.

For the first set of tests, the initial conditions represented in Figure 11 will be used.

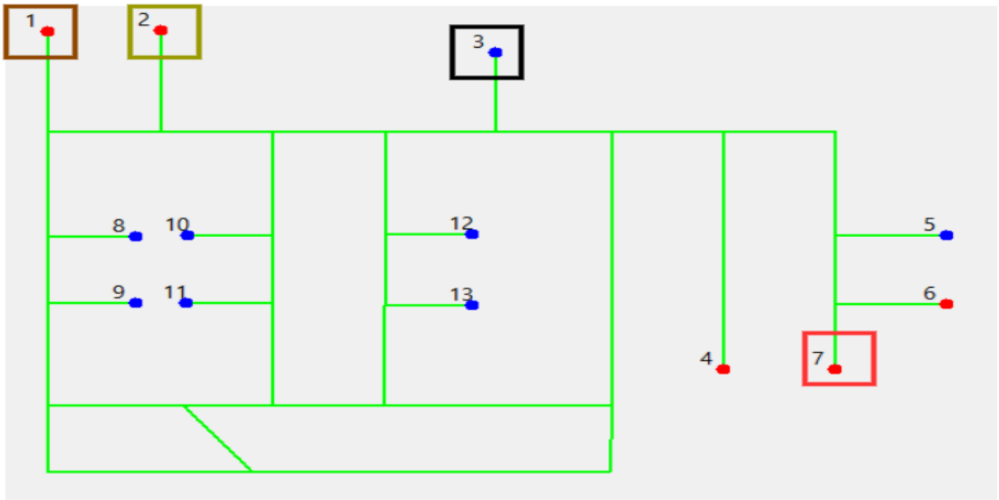

Figure 11. Representation of the workstation distribution used in this set of tests. The coloured squares indicate the initial position of the robots.

In this set of tests the no communication section chosen encompasses both workstations 10 and 11, this area is represented in Figure 12 by the colour red.

To test the system response, the creation of a new mission is also necessary. As the intent of these tests is to analyze the implemented system response to areas where there is no communication, a mission comprised of only one task to each robot is enough. Table 1 shows each robot assigned task as well as its initial priority.

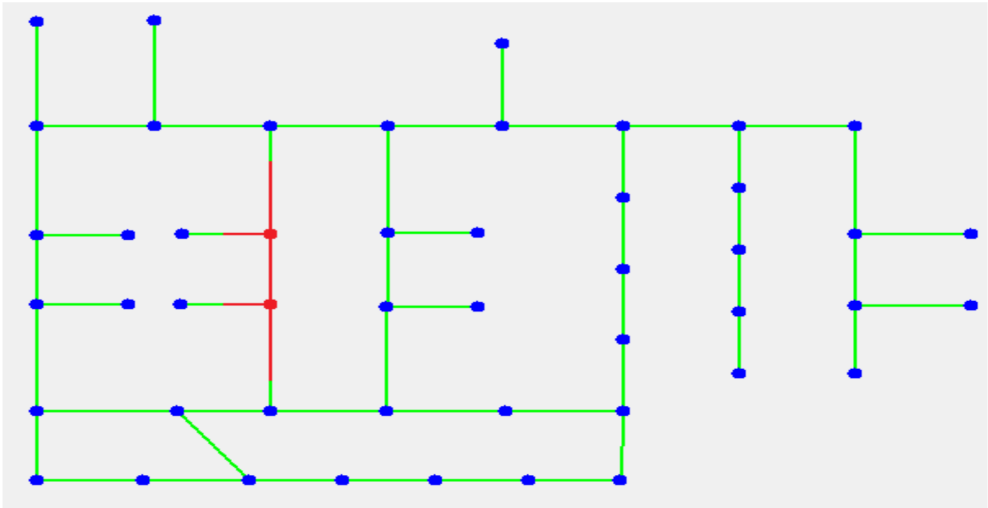

Figure 12. Representation of the chosen area where no communication can be established with the robots.

Table 1. Tasks and initial priority of each robot. The numbers under the task column represent the workstation, as identified in Figure 11, to which the robot is ordered to move to.

Id Robot	Colour	Initial Priority	Task
1	Brown	1	9
2	Dark Green	2	10
3	Black	3	11
4	Red	4	5

Figure A4 shows the paths travelled by the robots during the execution of this mission, as well as showing the mapping of the factory floor in order to determine which nodes are experiencing communication faults. In this mapping, the nodes represented in green are considered unfaulted nodes, the nodes represented in yellow are considered entry/exit nodes of the faults that have been mapped, the nodes represented in purple are considered possible entry/exit nodes of the faults that have yet to be mapped, and finally the red nodes are the faulted nodes.

By analyzing Figure A4, it is possible to observe that the system managed to successfully complete the assigned mission avoiding any collisions or deadlocks, even when two of the tasks required the robots to travel through the area that wasn't covered by the communication network. The system was also capable of correctly mapping the fault by using the data from the two robots that travelled through it.

The system first detected the fault in Figure A4e; initially the critical zone was considered to be part of the fault as seen in Figure A4f. On detecting this fault a replanning event is triggered and this leads the black robot to stop and wait for the dark green robot to exit the fault.

When the dark green robot exits the fault, Figure A4i, the fault dimensions are adjusted and the status of the node representing workstation 10 is changed from faulted node to an entry/exit node, as seen in Figure A4j. The black robot can now advance in its course, since the fault is no longer occupied, therefore a new replanning event is executed given the current position of the dark green robot, the black robot is now forced to plan a path that accesses the area via the bottom side instead of using the top side as initially planned.

When the dark green robot re-enterw the fault, in Figure A4k, the fault is once again placed as occupied. However, this time since the black robot is accessing the fault via the

bottom side there is no need for it to stop immediately. An important thing to note is that in this moment only the top node of the fault has been mapped as faulted node since it was the only one used by the robots until this point, as seen in Figure A4l. This leads to the black robot only stopping when it reaches the bottom entrance of the zone as seen in Figure A4m.

After the dark green robot once again exits the fault, in Figure A4o, the black robot enters the fault from the other side and therefore maps the other faulted node, as seen in Figure A4p. Since one of the new mapped faulted nodes is the same as one of the possible entry/exit nodes of the previously mapped fault, the system merges both faults.

A link to a video showing the execution of this test can be found in Appendix E.

This test was repeated 10 times in order to show that the presented sample was not a sporadic success. Analyzing the data from these tests, it is possible to conclude that the system is capable of successfully dealing with areas that are not covered by the communication network, since in all samples the system executed the mission successfully. It also allows for the stipulation of a baseline execution time for this mission of 1 min and 47 s. When this mission is executed without any communication faults, its mean execution time is of 1 min and 34 s. As expected, the overall efficiency of the system decreases when the system is exposed to communication faults.

6.2. Planning Supervision and Overall System Performance when Subjected to Static and Sporadic Communication Faults

The second set of tests has the objective of verifying that the implemented system is capable of completing an assigned mission, in a safe and efficient way, when exposed to both sporadic faults, as well as, zones where there is no communication with the robot.

To accomplish this, the CSSM must first not only detect the area where there is no communication but also detected the occasional loss of communication in a specific node. After the detection, this sub-module must also check if any of the detected faults have ceased to exist, as explained in Section 4.2.

In order to compare results of both sets of tests, the same mission and initial conditions used in the first set will be used, as shown in Figure 13 and Table 2.

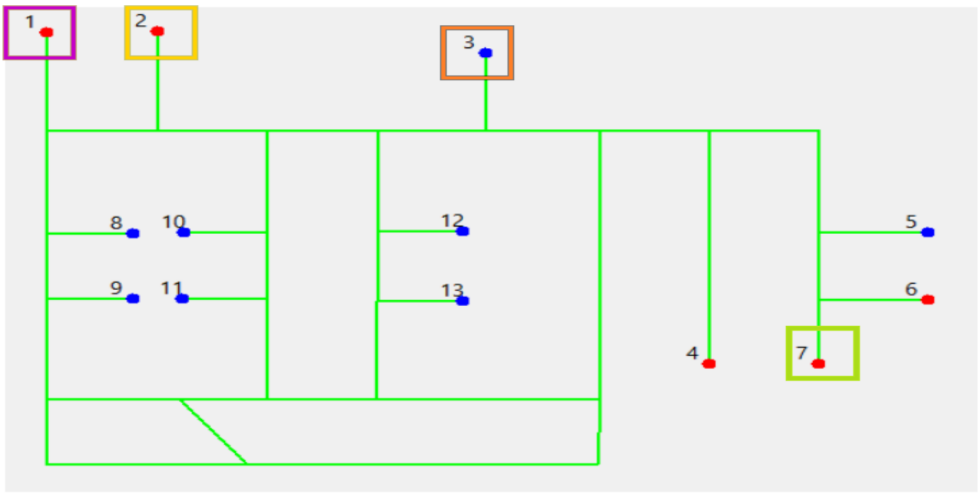

Figure 13. Representation of the workstation distribution as well as of the initial position of the robots used in this set of tests. The coloured squares indicate the initial position of the robots.

Table 2. Tasks and initial priority of each robot. The numbers under the task column represent the workstation, as identified in Figure 13, to which the robot is ordered to move to.

Id Robot	Colour	Initial Priority	Task
1	Purple	1	9
2	Yellow	2	10
3	Orange	3	11
4	Green	4	5

The sector of the map that is not covered by the communication network is also the same as the one used in the previous set of tasks, as shown in Figure 14.

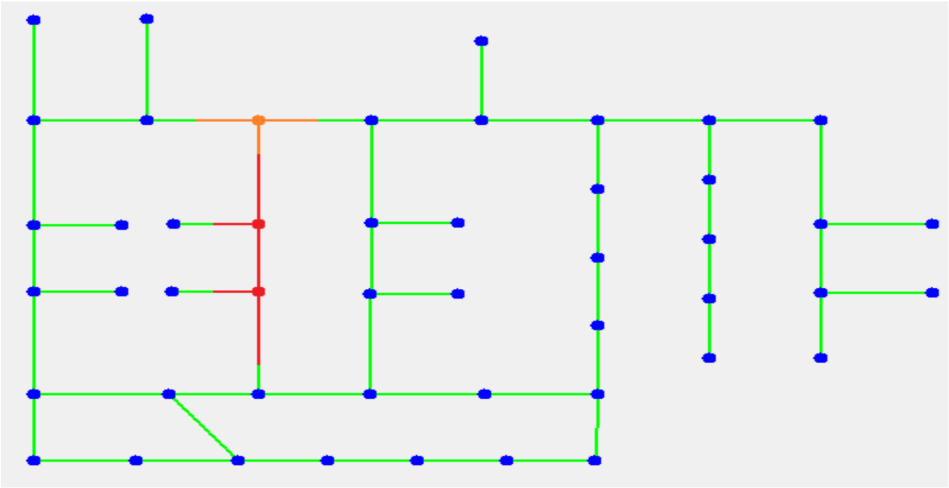

Figure 14. Representation of the chosen area where no communication can be established with the robots.

These sets of tests will test the occurrence of a sporadic fault in an entry/exit node of the fault that represents the area where no communication can be established with the robot, this area is represented in orange in Figure 14.

The representation of the results of the first part of this set of tests is represented in Figure A6. As with the previous tests, the nodes represented in green are considered unfaulted nodes, the nodes represented in yellow are considered entry/exit nodes of the faults that have been mapped, the nodes represented in purple are considered possible entry/exit nodes of the faults that have yet to be mapped and finally the red nodes are the faulted nodes.

By analyzing Figure A6, it is possible to observe that when the yellow robot moves to the top entry node of the fault corresponding to the section of the factory that is not covered by the communication network, referred as the static fault for brevity, a sporadic fault occurs in this node. Since this fault occurs in an entry/exit node this node is associated with the static fault, as seen in Figure A6a–f.

Similar to the previous tests, the orange robot advances until the edge of the fault and then stops and waits for the yellow robot to exit the fault, and therefore places the binary semaphore then corresponds to that fault in the free state before proceeding, seen in Figure A6e.

On exiting the fault a second time, as shown in Figure A6g, the yellow robot sees that the sporadic fault has ceased since the system can now communicate with the robot in a zone where it previously couldn't. This leads to the system reclassifying the node where communication was re-established with the robot as an unfaulted node. The system

also reclassifies the node belonging to the static fault as an unmapped node, due to the impossibility of the system knowing if it belongs to the static fault or to the sporadic fault.

Similar to the previous tests after the yellow robot exits the fault, the orange robot is cleared to proceed with the execution of its task. This leads to the orange robot entering the static fault, as shown in Figure A6i. The executing of this robot tasks leads to the remapping of the static fault and this time the dimension is correct since no sporadic fault occurs when the orange robot exits the fault for a second time. This process is shown in Figure A6i–p.

A link to a video showing the execution of this test can be found in Appendix E.

The execution of this test was repeated 10 times. In all samples, the system was able to successfully operate in environments where sporadic communication faults as well as static communication faults can occur, and manage to safely execute its assigned tasks.

6.3. Cases Where the Implemented System Can't Guarantee a Safe Execution of the Mission

Even due the overall results of the implemented system tests were very successful there are some situations where the implemented system is incapable of guarantee a safe execution of its assigned mission.

These situations normally occur due to either external interference or due to a communication fault occurring in an unfortunate time or having a large dimension. These situations are the following:

- Two or more robots entering the same fault at the same time;
- Impossibility of a robot to move away from the exit node of a fault.

In the first one, if two or more robots enter the same unmapped static fault at the same time, a collision can occur as the system has no way to control the synchronization between robots as it is incapable of communicating with the robots. This normally occurs when a communication fault has a large dimension.

The second situation where this may occur is if the entry/exit of a fault is located in a link between two nodes. There is a chance of a situation where one robot is currently traversing that link and another robot enters the fault through another entry point. The first robot will be in a deadlock situation since both of its possible movements choices are now locked until the second robot exits the fault.

7. Conclusions

The article presents the implementation of a control system capable of controlling the traffic of a fleet of robots, i.e., plan and control the movement of a fleet of robots that have multiple tasks assigned. The implemented system must plan safe and efficient paths, avoid deadlocks and be immune to network failures. It uses the TEA* algorithm, proposed in [4], as a base for the path planning and has a high tolerance to communication faults. The implemented system is comprised of several modules. This system not only planned the robots paths but also supervised their execution and was on guard against any communication failures.

Several modifications of the TEA* algorithm, proposed by [4], were also implemented in order to make it compatible with the new environment conditions that the algorithm would have to face. This modifications also intended to move the TEA* algorithm closer to a real life industrial application.

As shown in Section 6.2, the implemented system is capable of safely executing a set of tasks in environments where both static and sporadic communication faults occur. On previous implementations of the TEA* algorithm [4] , when the robots were subjected to communication faults, there would be a high probability of the occurrence of deadlocks, since the TEA* algorithm requires the synchronization of the movements and positions of all the robots of the system in order to be effective.

Although the system has a good tolerance to communication faults, it still lacks total effectiveness when dealing with communication faults, as demonstrated by in Section 6.3.

Author Contributions: The contributions of the authors of this work are pointed as follows: Conceptualization: D.M., P.C. (Pedro Costa) and J.L.; Methodology: D.M., P.C. (Pedro Costa) and J.L.; Software D.M., P.C. (Pedro Costa), J.L. and P.C. (Paulo Costa); Validation: D.M., P.C. (Pedro Costa) and J.L.; Writing—Review and Editing: D.M., P.C. (Pedro Costa) and J.L.; Supervision: P.C. and J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is financed by National Funds through the Portuguese funding agency, FCT—*Fundação para a Ciência e a Tecnologia* within project UIDB/50014/2020. This work has been supported by the European Regional Development Fund (FEDER) through a grant of the Operational Programme for Competitivity and Internationalization of Portugal 2020 Partnership Agreement (PRODUTECH4S&C, POCI-01-0247-FEDER-046102).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: This study didn't report any new data.

Acknowledgments: The authors of this work would like to thank the members of INESC TEC iiLab for all the support rendered to this project.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TEA*	Time Enhanced A*
AGV	Autonomous ground vehicle
MAPF	Multi-Agent Pathfinding
LPA*	Lifelong Planning A*
Mutex	Multiple exclusion object
NP	Non-deterministic Polynomial-Time
PSSM	Planning Supervision Sub-Module
CSSM	Communication Supervision Sub-Module

Appendix A. Execution of the First Mission with no Communication Faults

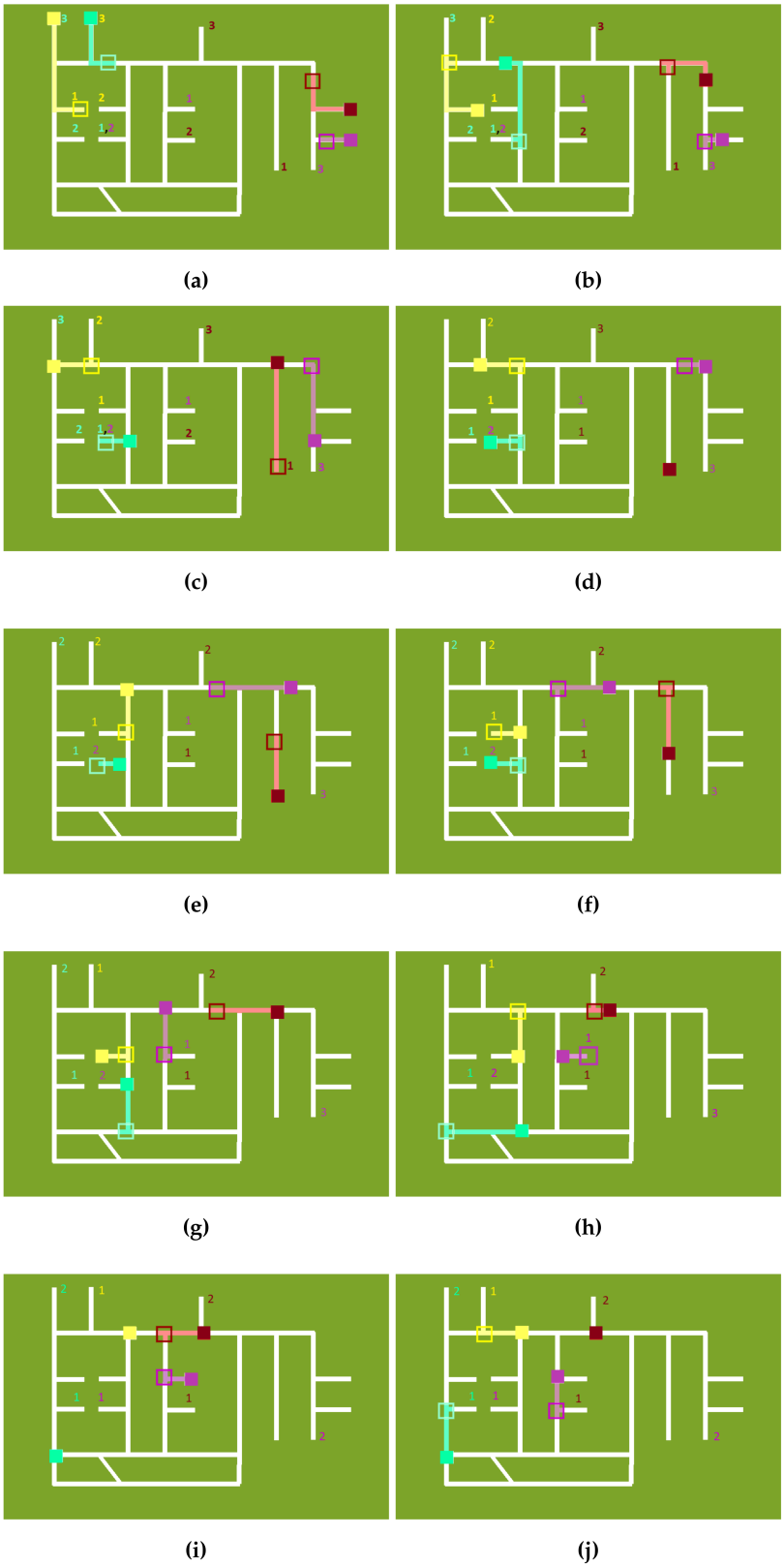

Figure A1. *Cont.*

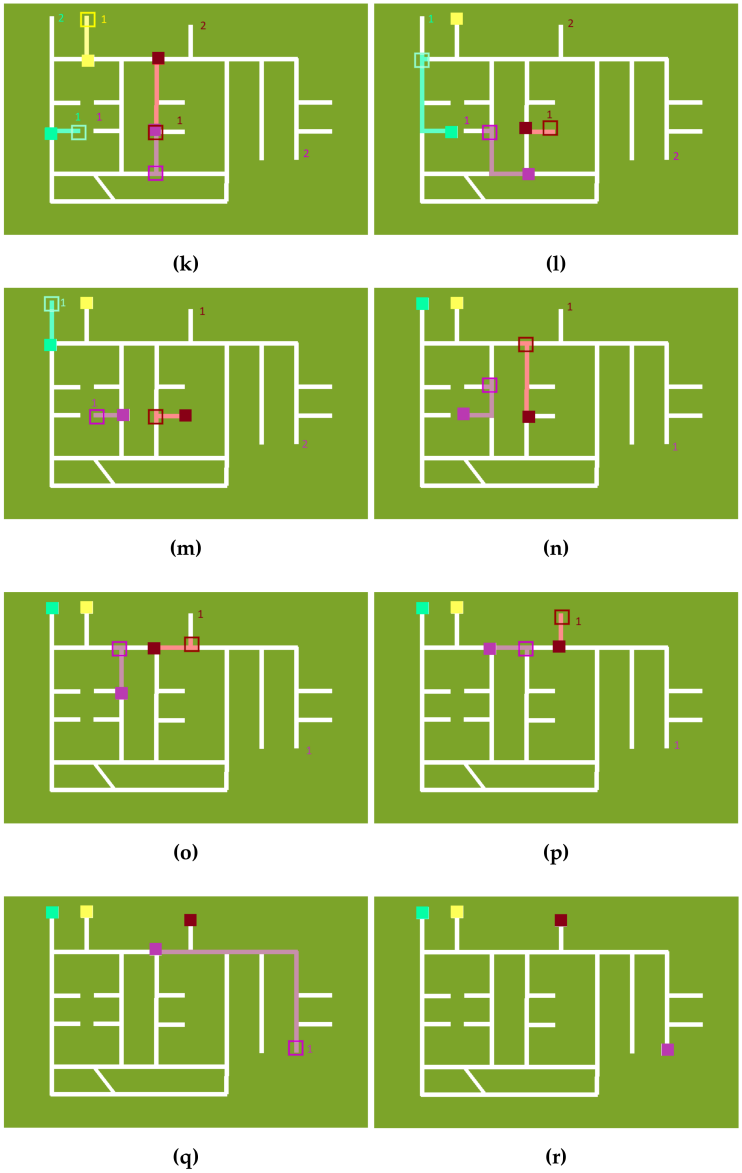

Figure A1. Movement of the robots during the execution of the first mission, each robot as two tasks assigned to it. In subfigures (a-r), the robot is represent by the full coloured square, while its movement between each of the subfigures is represented by the outlined square of the same colour. The numbers presented in the subfigures represent the task order assigned to each robot. (**a**) Initial positions of the robots; (**b**) Positions of the robots after one task is completed by the yellow robot; (**c**) Position and movement of the robots during the execution of the first task assigned to both the red and blue robots; (**d**) The purple robot must wait until the red robot starts its movement before it can advance; (**e**) A delay in the movement of the light blue robot originates a replanning event; (**f**) The robots keep executing their planned paths without any need of replanning; (**g**) Position and movement of the robots during the execution of the second tasks assigned to the yellow robot; (**h**) The robots keep executing their planned paths without any need of replanning; (**i**) Position and movement of the robots during the execution of the first tasks assigned to the purple; (**j**) The red robot waits until the purple robot moves from its current stop to initiate its movement; (**k**) The red robot shadows the movement of the purple robot until it reaches the entry node of its target point; (**l**) Movement of the yellow robot into its charging post and execution of the second task assigned by to the blue robot; (**m**) Movement of the blue robot into its charging post and execution of the second task assigned by to the red robot; (**n**) Positions of the robots when the purple robot finishes its second task; (**o**) All tasks are completed and the robots move to their charging posts; (**p**) Both robots move

towards their charging stations; (**q**) Movement of the red and purple robots towards their charging posts; (**r**) Final positions of the robots after all tasks are completed.

Appendix B. Execution of the Second Mission with no Communication Faults

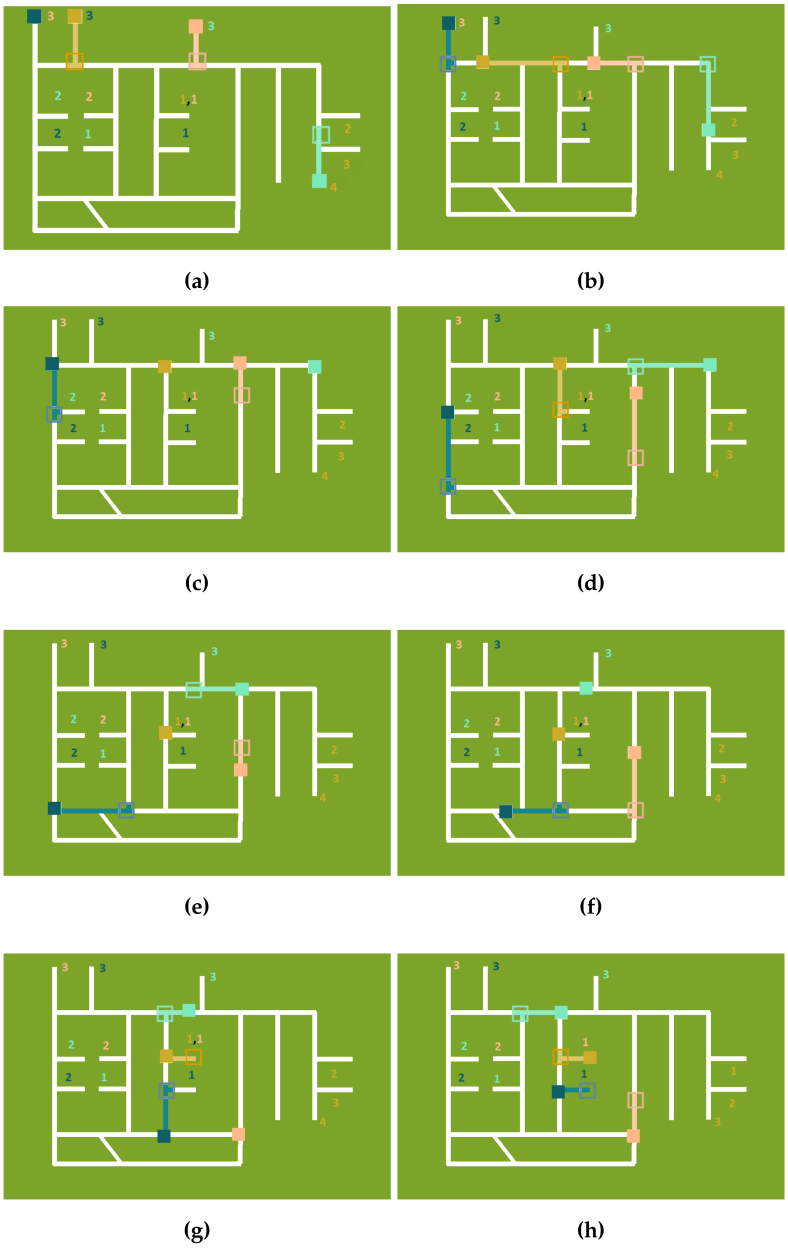

Figure A2. Movement of the robots during the execution of the second mission, first figure. In subfigures a–p, the robot is represent by the full coloured square, while its movement between each of the subfigures is represented by the outlined square of the same colour. The numbers presented in the subfigures represent the task order assigned to each robot. (**a**) Initial positions of the robots; (**b**) All robots keep following their planned paths; (**c**) Since the brown robot has a higher priority than the pink one, this last one is force to take the longer path in order to avoid any conflicts; (**d**) The light blue robot had to stop and wait for the pink robot to move from the intersection before resuming its movement; (**e**) Delays in the movement of the dark blue robot force the system to replan the path of the pink robot to avoid pathing conflicts; (**f**) Delays in the rotation of the brown robot force further

replanning of the robots paths; (**g**) No replanning is necessary since the robots are still synchronised; (**h**) Both the brown and dark blue robot finish their fist task.

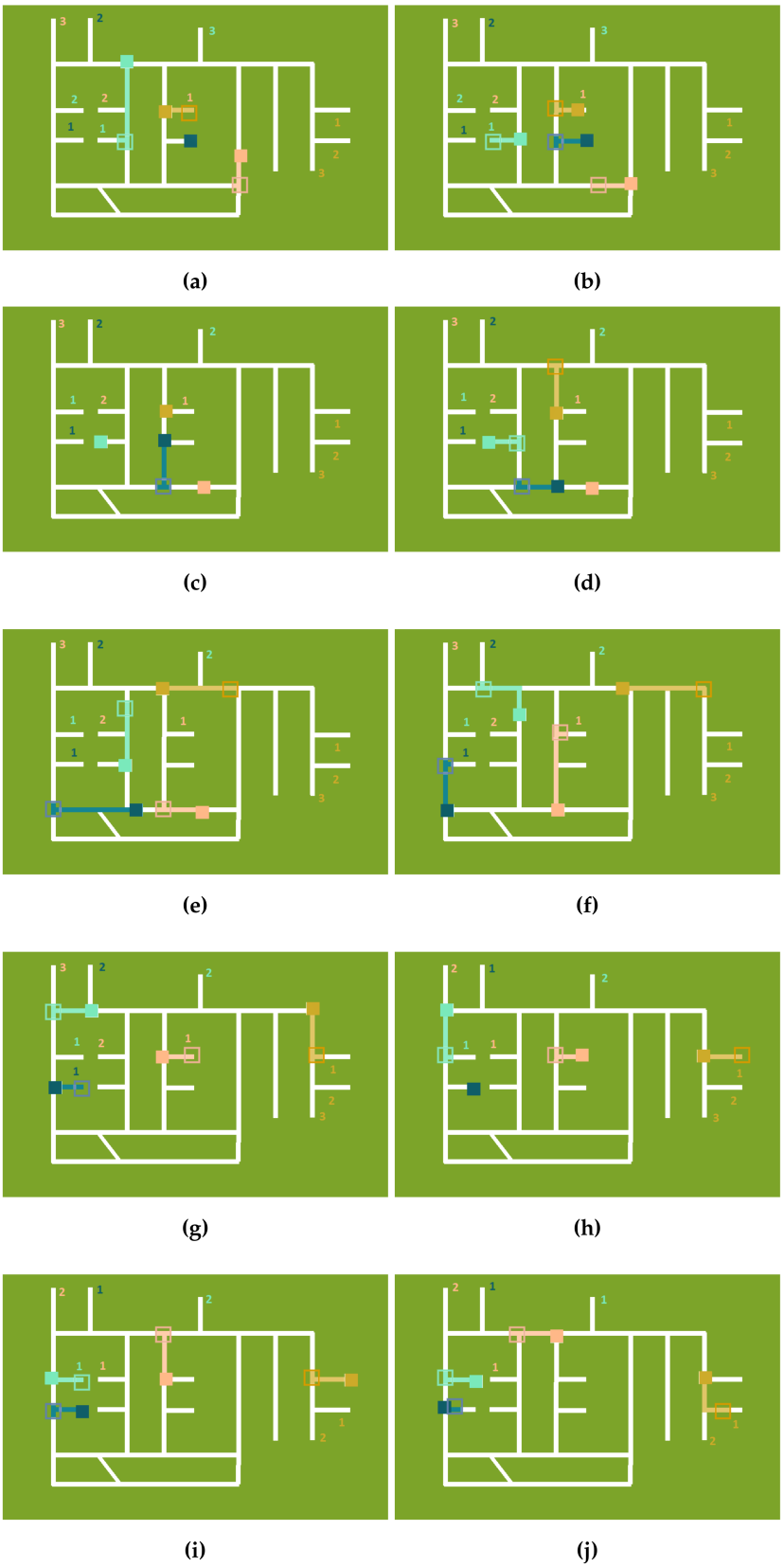

Figure A3. *Cont.*

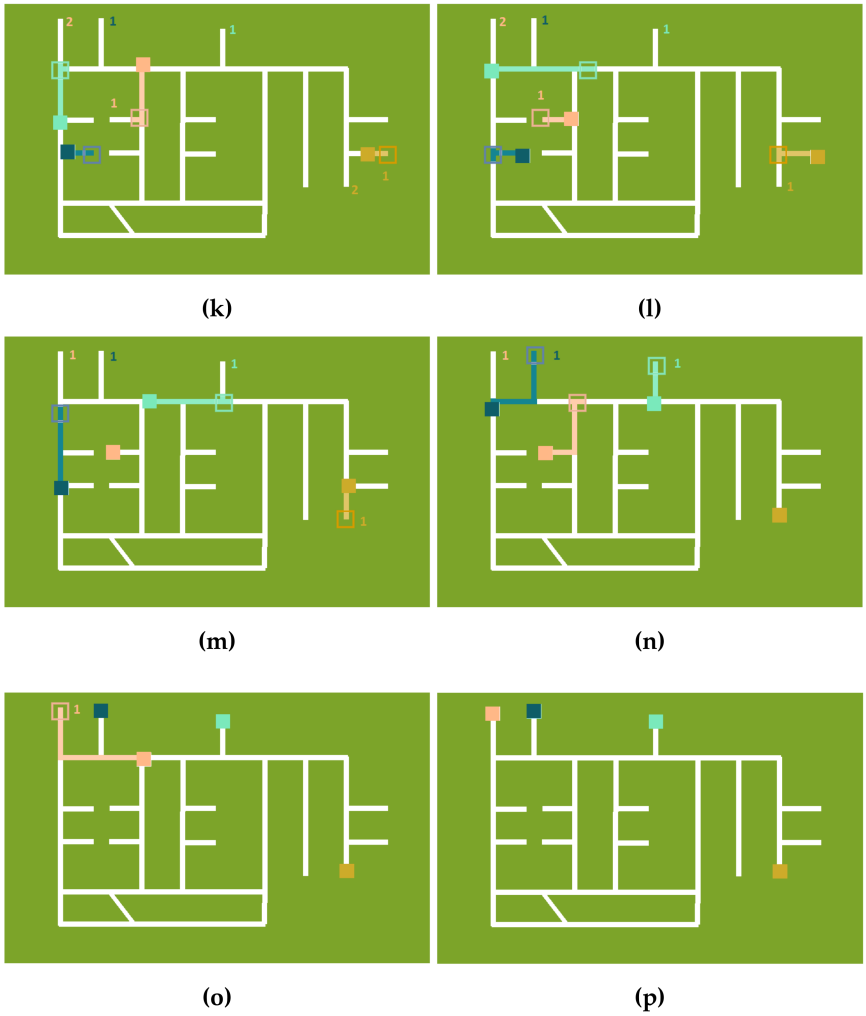

Figure A3. Movement of the robots during the execution of the second mission, second figure. In subfigures a–p, the robot is represent by the full coloured square, while its movement between each of the subfigures is represented by the outlined square of the same colour. The numbers presented in the subfigures represent the task order assigned to each robot. (**a**) Futher delays in rotation of the dark blue robot originate another replanning event; (**b**) The dark blue robot finishes its second task; (**c**) The pink robot has to wait until the dark blue robot exits from that branch of the map; (**d**) The light blue robot finishes its first task; (**e**) The robots keep executing their planned paths without any need of replanning; (**f**) Given the position of the robots the risk of a delay causing a pathing conflict is very low; (**g**) The robots keep moving to their target positions; (**h**) The dark blue robot and the pink robot finish their second and first task, respectively; (**i**) The pink and dark blue robots start moving to their charging positions; (**j**) The light blue robot and the brown robot finishes their second task; (**k**) New replanning event occurs due to a delay in the light blue robot movement; (**l**) The dark blue robot moves back in order to avoid a deadlock situation with the light blue robot; (**m**) The brown and pink robots finish their third and second tasks, respectively; (**n**) All three robots move towards their charging stations; (**o**) All robots move to their charging stations; (**p**) Final positions of all robots after all tasks are completed.

Appendix C. Execution of a Mission with Static Communication Faults

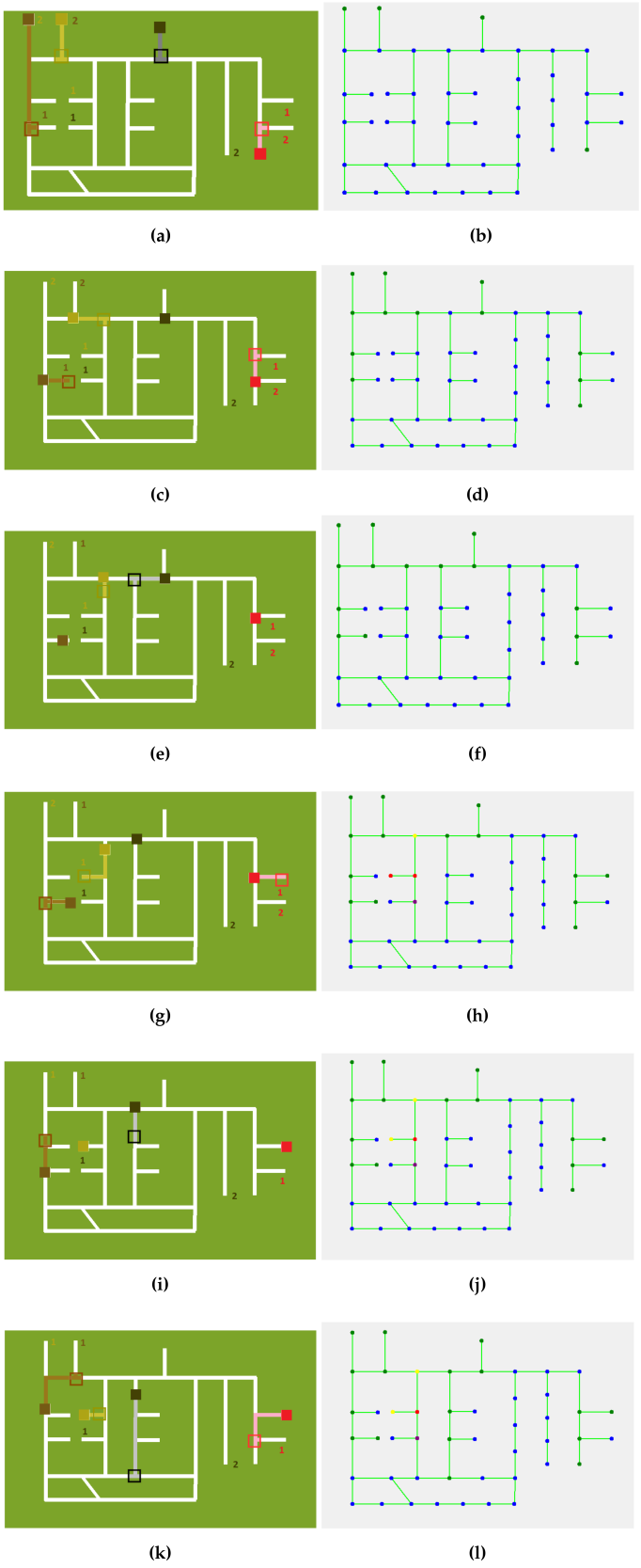

Figure A4. *Cont.*

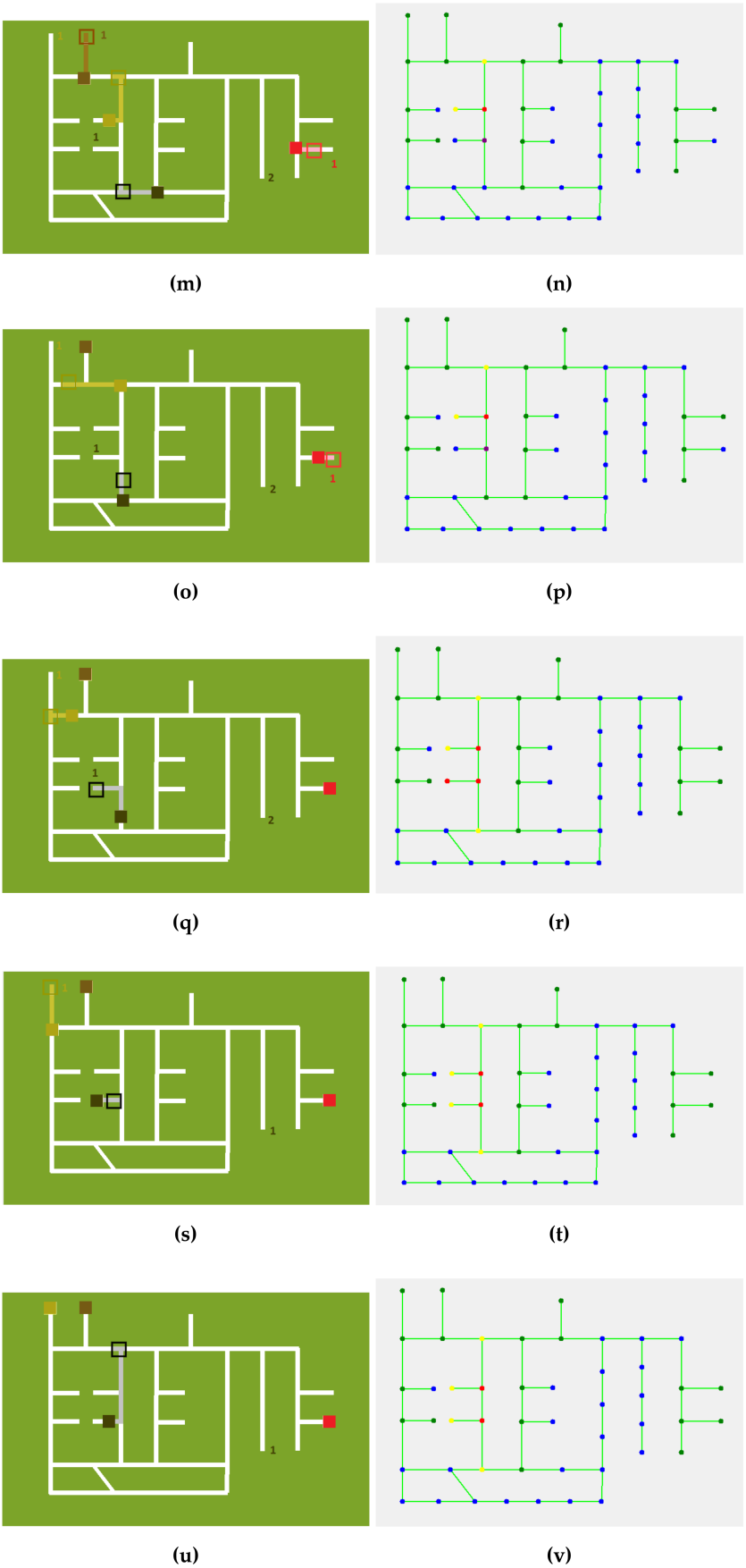

Figure A4. *Cont.*

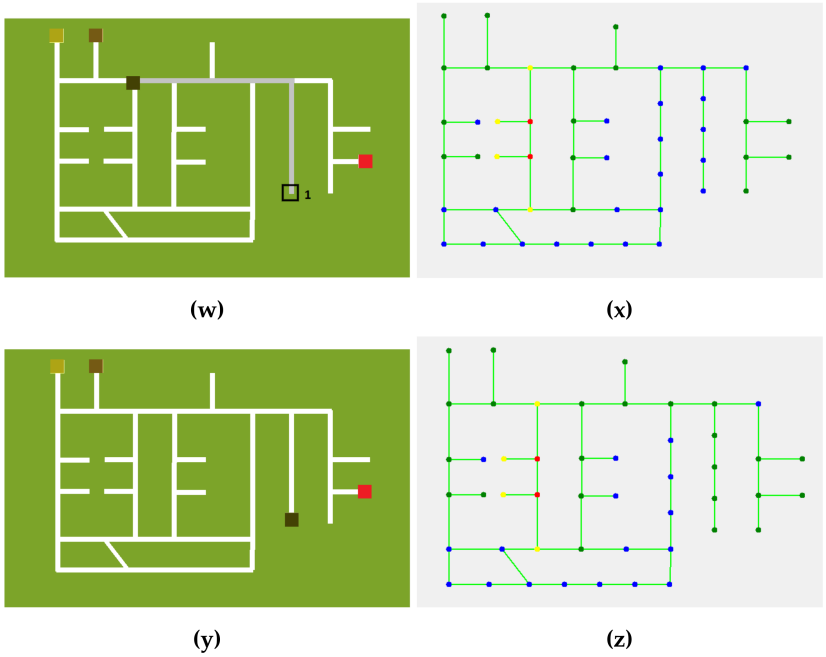

Figure A4. Behaviour of the system when subjected to an area without communication. In right subfigures, the robot is represent by the full coloured square, while its movement between each of the subfigures is represented by the outlined square of the same colour. The numbers presented in the subfigures represent the task order assigned to each robot. In the left subfigures the status of each node is displayed, blue equals unmapped, red equals faulted, yellow represents a entry/exit node and green represents an unfaulted node. (**a**) Initial positions of the robots; (**b**) Initial status of all the nodes of the system; (**c**) Position and movement of the robots during execution of the first task by the brown robot; (**d**) Status of the nodes during the movement of the robots; (**e**) The brown robot finishes its first task; (**f**) Status of the nodes during the after the brown robot finishes its first task, several nodes are now mapped as unfaulted nodes; (**g**) Entry of the light brown robot into the area affected by the communication fault; (**h**) Initial estimation of the size of the area affected by the communication fault; (**i**) The light brown robot exits the fault, the red robot also finishes its first task; (**j**) The size of the fault is now adjusted using the most recent data from the light brown robot; (**k**) The light brown robot reenters the area affected by the fault; (**l**) Status of the nodes when the light brown robot reenters the area affected by the fault; (**m**) The light brown robot moves through the area affected by the fault; (**n**) The new data is analysed however no modifications to the fault size is necessary; (**o**) The black robot is force to wait until the light brown robot exits the area affected by the communication fault; (**p**) Status of the nodes during this wait; (**q**) The black robot now enters the area affected by the fault via a previously unmapped entry; (**r**) The fault size is adjusted based on the last know location of the black robot and its planned path; (**s**) Communication is reestablished with the black robot; (**t**) The fault size is readjusted to take into account the new data; (**u**) The black robot reenters the fault; (**v**) no adjustments are necessary since now the fault is fully mapped, all of its entry/exit nodes have been mapped; (**w**) he black robot now moves to its charging station; (**x**) No more faults are detected during this movement; (**y**) Final position of the robots in the system; (**z**) Final status of all the nodes in the system.

Appendix D. Execution of a Mission with Static and Sporadic Communication Faults

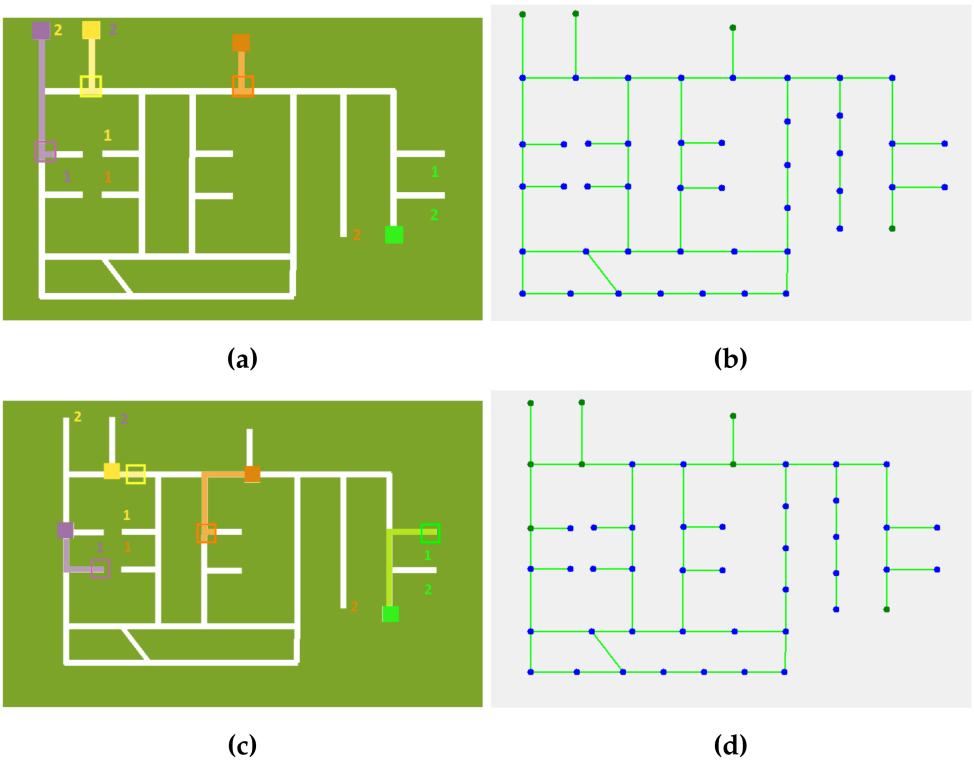

Figure A5. Behaviour of the system when subjected to a sporadic fault in an entry/exit node, first image. In right subfigures, the robot is represent by the full coloured square, while its movement between each of the subfigures is represented by the outlined square of the same colour. The numbers presented in the subfigures represent the task order assigned to each robot. In the left subfigures the status of each node is displayed, blue equals unmapped, red equals faulted, yellow represents a entry/exit node and green represents an unfaulted node. (**a**) Initial positions of the robots; (**b**) Initial status of all the nodes of the system; (**c**) The yellow robot enters the area affected by the communication faults; (**d**) Status of the nodes before the yellow robot enters the fault.

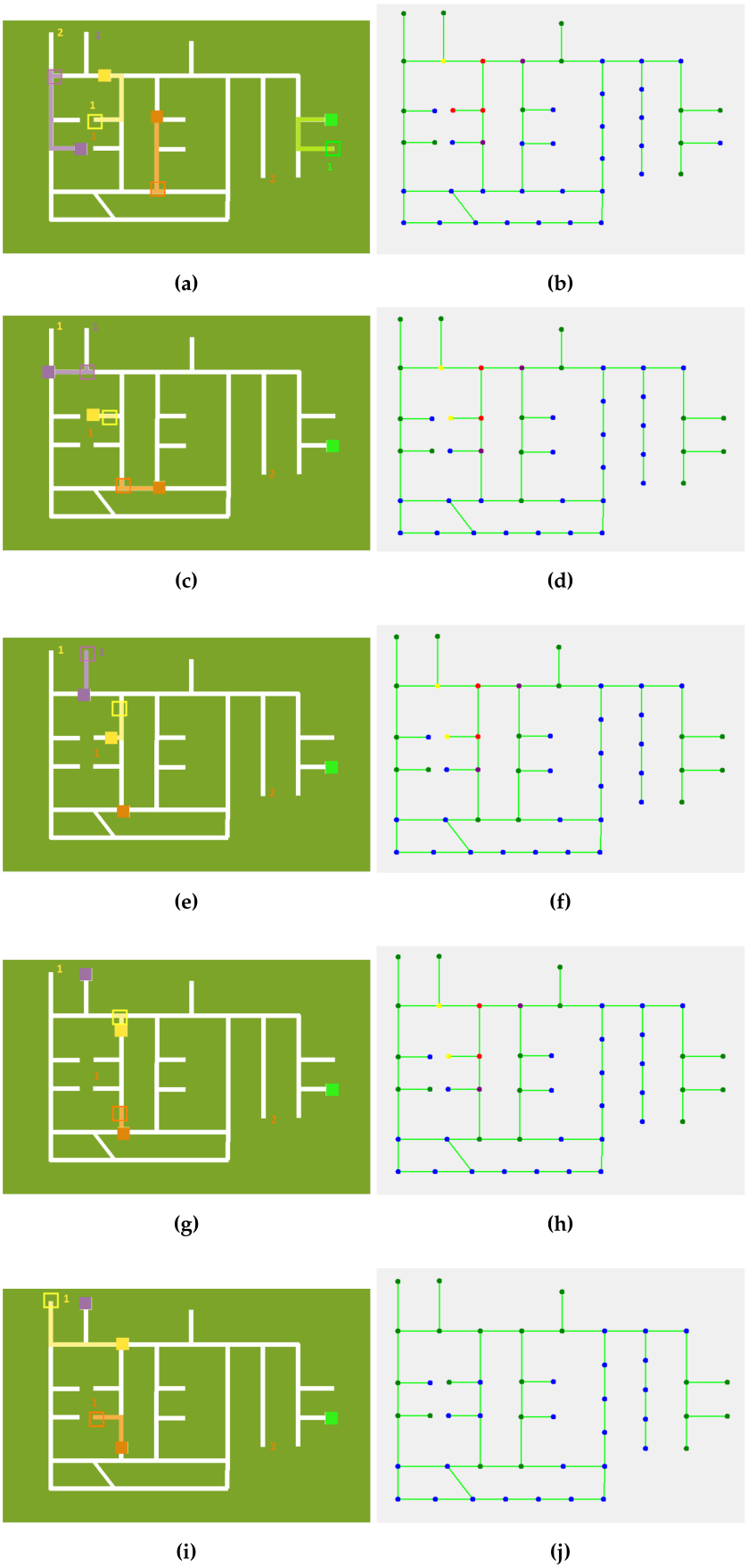

Figure A6. *Cont.*

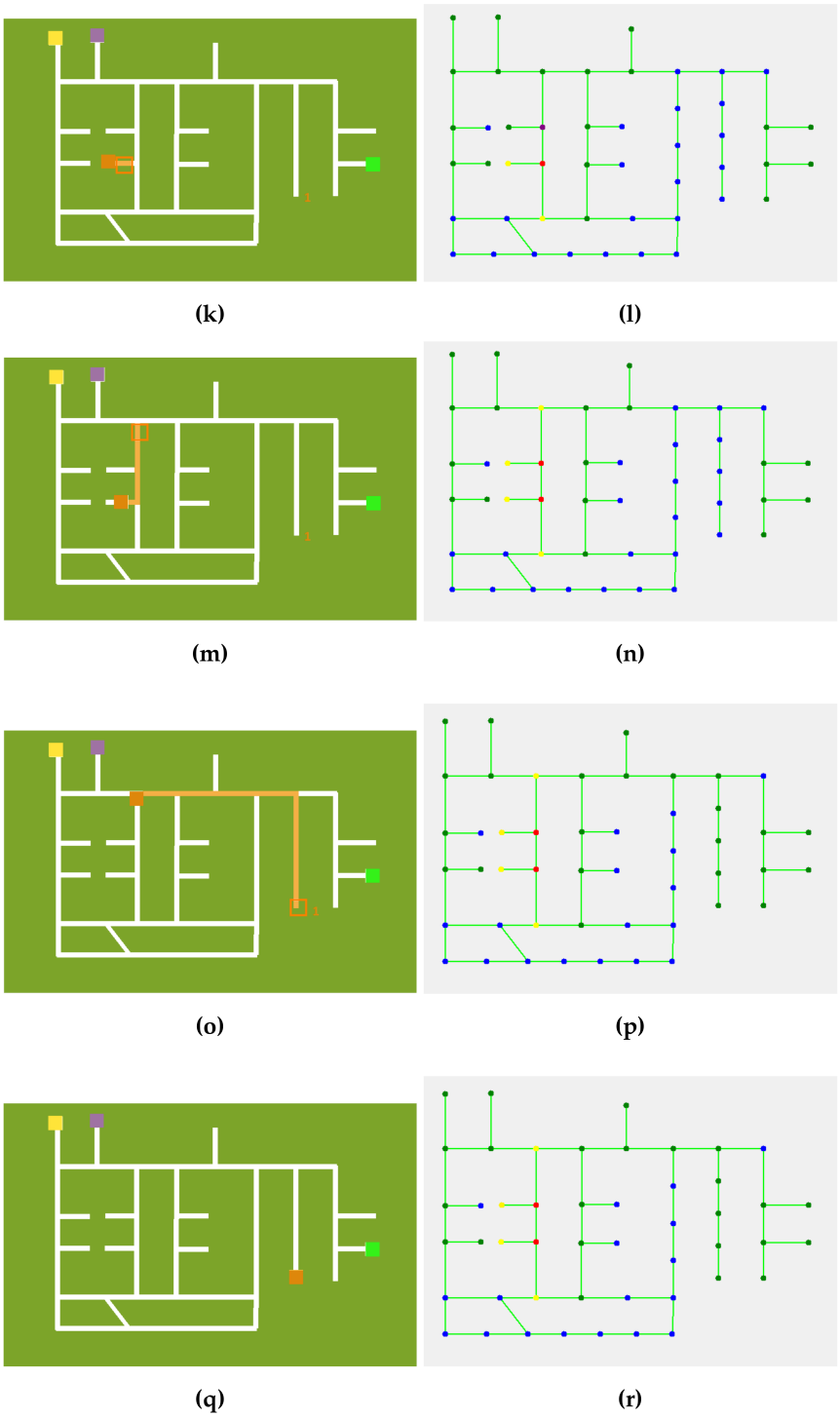

Figure A6. Behaviour of the system when subjected to a sporadic fault in an entry/exit node, second image. In right subfigures, the robot is represent by the full coloured square, while its movement between each of the subfigures is represented by the outlined square of the same colour. The numbers presented in the subfigures represent the task order assigned to each robot. In the left subfigures the status of each node is displayed, blue equals unmapped, red equals faulted, yellow represents a entry/exit node and green represents an unfaulted node. (**a**) The yellow robot moves through the faulted area; (**b**) Initial estimation of the fault size based on the path planned for the robot and the already mapped nodes; (**c**) Communication is reestablished with the yellow robot; (**d**) The size of the fault is adjusted; (**e**) The yellow robot reenters the fault, which forces the orange robot to wait until communication can be reestablish; (**f**) Status of the nodes before communication is reestablished; (**g**) The yellow robot moves through the faulted area; (**h**) No changes are executed to the size of the

mapped fault; (**i**) Communication is reestablish with the yellow robot in a node that was previously affected by a communication fault; (**j**) Since communication was establish in a previously faulted node the nodes associated with that communication fault are placed as unmapped; (**k**) The orange robot reenters the fault and starts remapping it; (**l**) Status of the nodes before the orange robot enters the fault; (**m**) The orange robot moves through the fault remapping it; (**n**) Half of the fault size is remapped; (**o**) The orange robot exits the fault and moves towards its charging station; (**p**) The fault size is fully mapped; (**q**) Final position of the robots in the system; (**r**) Final status of all the nodes in the system.

Appendix E. Video Links

Link to the video of the execution of the first mission by the implemented system, in an environment where no communication faults occur: https://youtu.be/7MnuJ7nOKug (accessed at 23 March 2021).

Link to the video of the execution of the second mission by the implemented system, in an environment where no communication faults occur: https://youtu.be/oxt4x80ydPw (accessed at 23 March 2021).

Link to the video of the execution of the a mission by the implemented system, in an environment where only static communication faults occur: https://youtu.be/rlSbgYZXpAg (accessed at 23 March 2021).

Link to the video of the execution of the a mission by the implemented system, in an environment where both static and sporadic communication faults occur: https://youtu.be/dl5ZbOh_vUg (accessed at 23 March 2021).

References

1. Ullrich, G. The History of Automated Guided Vehicle Systems. *Autom. Guid. Veh. Syst. Prim. Pract. Appl.* **2015**. [CrossRef]
2. Siefke, L.; Sommer, V.; Wudka, B.; Thomas, C. Robotic systems of systems based on a decentralized service-oriented architecture. *Robotics* **2020**, *9*, 78. [CrossRef]
3. Santos, J.; Costa, P.; Rocha, L.F.; Moreira, A.P.; Veiga, G. Time enhanced A*: Towards the development of a new approach for Multi-Robot Coordination. In Proceedings of the IEEE International Conference on Industrial Technology (ICIT), Seville, Spain, 17–19 March 2015; pp. 3314–3319. [CrossRef]
4. Moura, P.; Costa, P.; Lima, J.; Costa, P. A temporal optimization applied to time enhanced A*. *AIP Conf. Proc.* **2019**, *2116*, 1–4. [CrossRef]
5. Downey, A.B. *The Little Book of Semaphores*, 2nd ed.; v2.2.1; Green Tea Press: Needham, MA, USA, 2016.
6. Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; Zhou, N.F. Robust multi-agent path finding and executing. *J. Artif. Intell. Res.* **2020**, *67*, 549–579. [CrossRef]
7. Felner, A.; Stern, R.; Shimony, S.E.; Boyarski, E.; Goldenberg, M.; Sharon, G.; Sturtevant, N.; Wagner, G.; Surynek, P. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In Proceedings of the 10th Annual Symposium on Combinatorial Search, SoCS 2017, Pittsburgh, PA, USA, 16–17 June 2017; pp. 29–37.
8. Surynek, P. An optimization variant of multi-robot path planning is intractable. In Proceedings of the National Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; Volume 2, pp. 1261–1263.
9. Yu, J. Intractability of optimal multirobot path planning on planar graphs. *IEEE Robot. Automat. Lett.* **2016**, *1*, 33–40. [CrossRef]
10. Falcó, A.; Hilario, L.; Montés, N.; Mora, M.C.; Nadal, E. A Path Planning Algorithm for a Dynamic Environment Based on Proper Generalized Decomposition. *Mathematics* **2020**, *8*, 2245. [CrossRef]
11. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-robot path planning method using reinforcement learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
12. Yu, J.; Su, Y.; Liao, Y. The Path Planning of Mobile Robot by Neural Networks and Hierarchical Reinforcement Learning. *Front. Neurorobot.* **2020**, *14*, 1–12. [CrossRef] [PubMed]
13. Da Costa, P.L.C.G. Planeamento Cooperativo de Tarefas e Trajectórias em Múltiplos Robôs. Ph.D. Thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2011; p. 231.
14. Latombe, J.C. Introduction and Overview. Robot Motion Planning. In *Robot Motion Plan*; Springer: Boston, MA, USA, 1991; pp. 1–57. [CrossRef]

15. Lozano-Pérez, T.; Wesley, M.A. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *Commun. ACM* **1979**, *22*, 560–570. [CrossRef]
16. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006; 826p. [CrossRef]
17. Hart, P.E.; Nilsson, N.J.; Raphael, B. The Heuristic Determination. *IEEE Trans. Syst. Sci. Cybern.* **1968**, 100–107. [CrossRef]
18. Stentz, A. Optimal and Efficient Path Planning/or Partially Known Environments. In *Intelligent Unmanned Ground Vehicles*; Springer: Boston, MA, USA, 1997.
19. Koenig, S.; Likhachev, M. Incremental A*. In *Advances in Neural Information Processing Systems 14 (NIPS 2001);* Dietterich T., Becker S., Ghahramani Z., Eds.; MIT Press: Cambridge, MA, USA, 2002.
20. Wang, W.; Goh, W.B. Multi-robot path planning with the spatio-temporal A* algorithm and its variants. In *Advanced Agent Technology, Proceedings of the AAMAS 2011, Taipei, Taiwan, 2–6 May 2011*; Dechesne F., Hattori H., Mors A., Such J.M., Weyns D., Eds.; Springer Publishing: New York, NY, USA, 2011; pp. 313–329. [CrossRef]
21. Wagner, G.; Choset, H. Subdimensional expansion for multirobot path planning. *Artif. Intell.* **2015**, *219*, 1–24. [CrossRef]
22. Jansen, R.; Tg, C.; Sturtevant, N. A new approach to cooperative pathfinding. In Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, Taipei, Taiwan, 2–6 May 2008; Volume 3, pp. 1401–1404.
23. Cao, Y.U.; Fukunaga, A.S.; Kahng, A.B. Cooperative Mobile Robotics: Antecedents and Directions. *Auton. Robot.* **1997**, *4*, 7–27. [CrossRef]
24. Raynal, M. Fault-tolerant Agreement in Synchronous Message-passing Systems. In *Synthesis Lectures on Distributed Computing Theory*; Morgan & Claypool Publishers: Williston, VT, USA, 2010; Volume 1, pp. 1–189. [CrossRef]
25. Kandath, H.; Senthilnath, J.; Sundaram, S. Mutli-agent consensus under communication failure using Actor-Critic Reinforcement Learning. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence, SSCI 2018, Bangalore, India, 18–21 November 2019; pp. 1461–1465. [CrossRef]
26. Braun, J.; Fernandes, L.A.; Moya, T.; Oliveira, V.; Brito, T.; Lima, J.; Costa, P. Robot@Factory Lite: An Educational Approach for the Competition with Simulated and Real Environment. In Proceedings of the Robot 2019: Fourth Iberian Robotics Conference, Porto, Portugal, 20–22 November 2019; pp. 478–489. [CrossRef]
27. Costa, P.; Gonçalves, J.; Lima, J.; Malheiros, P. Simtwo realistic simulator: A tool for the development and validation of robot software. *Theory Appl. Math. Comput. Sci.* **2011**, *1*, 17–33.
28. Silver, D. Cooperative pathfinding. In Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE) 2005, Marina del Rey, CA, USA, 1–3 June 2005; pp. 117–122.