Distributed supervisory control for a system of path-network sharing mobile robots

Elżbieta Roszkowska* Bogdan Kreczmer* Adam Borkowski† Michał Gnatowski†
*The Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology, Wrocław, Poland
†Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland

***Abstract*— The paper presents a distributed control system for a multiple mobile robot system (MMRS). The robots share a common workspace, i.e., a network of paths, that is further partitioned into a number of sub-networks. Each sub-network has a separate controller, responsible for supervising the robot motion in its respective area, and communication with the other controllers. We discuss the architecture of the control system, the formal foundations underlying the control concept and ensuring its correctness, as well as their concrete implementations. The considerations are illustrated with a number of screens captured in the computer system initially developed to assist the synthesis of AGV network control [9], and now being tailored for the needs of MMRS. The reported work is still under the construction, yet the most crucial part has already been done.**

***Index Terms*— multi-robot system, deadlock avoidance, path network**

I. Introduction

In this paper we focus on supervisory control for a fleet of mobile robots moving in a known indoor environment such as, e.g., a hospital, a museum, a hotel. Coordination of the motions of mobile robots as they perform their task in a shared workspace has been, in recent years, a widely studied problem in robotics. The theoretical works in this area have mostly concentrated on motion planning with respect to collision avoidance and performance optimization [3]. The realization of such motion plans is, however, an open-loop, time-based control, that is highly sensitive to the system randomness. In a system of autonomous, asynchronously operating robots, that accomplish randomly arriving tasks, the eventual applicability of such plans is rather questionable. On the other hand, most research on the real-time control for multiple mobile robot systems (MMRS) has been directed towards software development, based on ad-hoc, rather than rigorous models, developed directly in programming languages and providing no formal guarantee of their correctness. A few works have proposed a more prospective approach to MMRS supervisory control, employing Petri nets as a modeling formalism, e.g., [5, 4]. However, also these papers focus on specific applications rather than deal with a general methodology or essential robot coordination problems such as deadlock avoidance.

In this paper we propose a control system for MMRS that is deprived of the above mentioned insufficiencies, i.e., a high-level control system based on a general mathematical model of formally proved correctness and, due to its closed-loop character, robust and immune to the system randomness.

Since the robots operate in a known environment, it is possible to assume that they move within a network of paths, whose geometry is established at the system-design stage. This approach substantially simplifies the problem of path-planning, basically reducing it to the selection of paths in the graph modeling the network. Moreover, such a solution allows us to employ a similar abstraction and methods to coordinate the concurrent robot movement as those that have been recently considered for AGV (*Automated Guided Vehicles*) systems in, e.g., [6, 11, 2] as well as in our earlier works [7, 10, 9, 8].

This new, evolving approach is based on a DES (*Discrete Event System*) representation, and *event-driven supervisory control* of the vehicle system. The contributions mentioned above differ with respect to such features as the type of the system (open vs. closed systems), the routing scheme (pre-determined vs. dynamically established routes), the modeling formalism (Petri nets, deterministic automata, processes/resources OS-like representation), and the research tendency (analysis-oriented vs. synthesis-oriented), yet, basically, all of them focus on the structural control and its central problem – collision-free and deadlock-free AGV model synthesis. In this paper, we build on the concepts established in our previous work (cited above), and using the proposed AGV model, we adapt it to the purposes of the considered system of mobile robots, and develop a complete distributed-control system for MMRS.

II. Control principles

As stated in the introduction, in this paper we consider a fleet of mobile robots that, similar to AGVs, travel within a network of paths. According to [10], the AGV system is characterized by the following features: it is *guidepath-based, zone-controlled, dynamically routed* and *closed*. More specifically, a guidepath-based traffic system consists of a number of vehicles (or, in our case, autonomous robots) that travel among a number of locations while following some predetermined paths that form a connected *guidepath network*. Links of this guidepath network can be traversed in both directions, but the motion of the vehicles on these links is unidirectional, i.e., a vehicle cannot reverse the direction of its motion while on any certain link. The tasks or *missions* of the vehicles consist of visiting a specified sequence of locations. The traffic system is characterized as *dynamically routed*, that is, the routes between the consecutive locations are

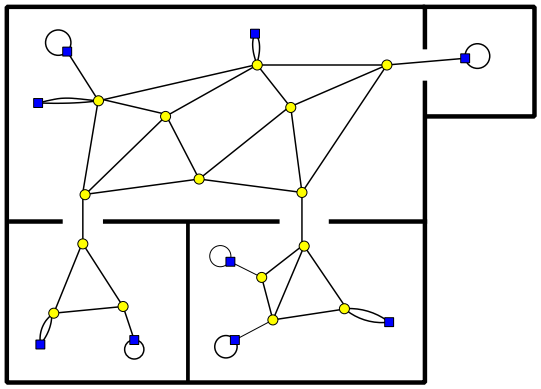

Fig. 1. Example layout of the control areas and the path network. Squares mark destination nodes and circles makr remaining nodes.

developed in real-time. In order to avoid the physical collision of the various vehicles, the traffic is *zone-controlled*. That is, the entire guidepath network is partitioned into a number of segments, or *zones*, and only one vehicle is allowed in any segment at any point in time. Finally, a traffic system is viewed as *closed*, which means that vehicles with no current mission remain in the guidepath network, either idling on some guidepath link or moving among various links in order to clear the way for some other vehicles.

While maintaining this basic characterization of the AGV system, we also make the following specific assumptions about the here considered MMRS.

1) The robot workspace is partitioned into a number of disjoint areas, covered by the respective, mutually disjoint path sub-networks. Particular pairs of areas are connected through special transit paths.
2) Each workspace area is supervised by its own controller. The controllers can communicate one with another, as well as with the robots currently located in their respective areas (see Fig. 1).
3) The guidepath network includes three types of nodes:
   a) the nodes that are destination points, able to accommodate one robot at a time
   b) the root-node, able to accommodate all the robots at a time (representing, e.g., the docking station)
   c) the remaining nodes, where the presence of robots can only be temporary.
4) In the initial state all the robots are located in the root node, where each of them is assigned a mission specifying the destination node for its travel.
5) Having received a mission, each robot reports to its current controller, which then negotiates with the other controllers a global route for the robot, that is, a sequence of workspace areas to be passed on its way to the destination node.
6) When passing a particular area, the motion of a robot is supervised by the respective area controller. Each robot can freely move within a zone, while zone changing requires permission of the local controller, which also plans dynamically the robot's route within the area it controls.
7) After attaining its current destination point, a robot becomes idle until it is assigned another mission. Then its new global route is established, and the robot sets out for the next trip.

III. DES MODEL OF MMRS

In this section, first we recall the concept of the DFSA (Deterministic Finite State Automaton) [1], and then employ it to establish an automaton model of MMRS.

A. Deterministic Finite State Automata

Definition 1: A deterministic finite state automaton (DFSA)is a 5-tuple $G = (S, E, \Gamma, \delta, s_0)$, where:

1) S is the set of *states*.
2) E is the set of *events*. The occurrence of an event causes a state transition in G.
3) $\Gamma : S \to 2^E$ is the *feasible event function*. Event $e \in E$ is feasible (i.e., can occur) in state $s \in S$ iff $e \in \Gamma(s)$.
4) $\delta : S \times E \to S$ is the *transition function*. δ is a partial function defined for pairs (s, e) such that $e \in \Gamma(s)$. $s' = \delta(s, e)$ is the new state resulting from the occurrence of event e in state s.
5) $s_0 \in S$ is the *initial state*.

In each state s of a DFSA, only such an event e can occur for which the transition function $\delta(s, e)$ is defined. The occurrence of e induces a new state $s' = \delta(s, e)$. The following definitions describe two more FSA concepts that will be useful further in this work.

Definition 2: The *reachability set* of a state $s \in S$ is a subset $R(A, s) \subseteq S$ defined inductively as follows: (i) $s \in R(A, s)$; (ii) for every pair $(s', e) \in S \times E$ such that $s' \in R(A, s)$ and e is feasible in s', the state $s'' = \delta(s', e)$ is also in $R(A, s)$. The *transition graph* of s, $RG(A, s) = (R(A, s), D)$, is a directed multi-graph with vertex set $R(A, s)$, and edge set D that contains edge d_e from vertex s' to s'' iff event e is feasible in state s' and $s'' = \delta(s', e)$.

Definition 3: An FSA $A_{res} = (S, E, \Gamma_{res}, \delta, s_0)$ is a restriction of $A = (S, E, \Gamma, \delta, S_0)$ iff $\forall s \in S$, $\Gamma_{res} \subseteq \Gamma$. The reachability set and the reachability graph of s in system A_{res} will be denoted by $R_{res}(A, s)$ and $RG_{res}(A, s)$, respectively.

B. Structure of MMRS

We will develop the model of MMRS as a composition of the sub-systems associated with the separately controlled areas. Thus, we will view the system's path network as a set $\mathcal{U}$ of disjoint sub-graphs U_i, $i = 0, \ldots, n+1$, where U_0 abstracts the docking station, and U_{n+1} is constituted by the transit edges that connect the remaining, mutually disjoint sub-graphs, corresponding to the n separately controlled area of MMRS. Each sub-graph, except U_{n+1}, is biconnected[1], and represented by the triple $U_i = (V_i, Z_i, \zeta_i)$, $i = 0, \ldots, n+1$, such that: (i) Z_i is the set of the graph *edges*, corresponding to the set of zones defined in the sub-network U_i; (ii) $V_i = W_i \cup D_i$ is the set of the *vertices* of U_i, among which D_i is the

[1] An undirected graph is biconnected if each of its edges lies on a cycle.

set of destination points, and W_i is the set of the remaining vertices, and (iii) $\zeta_i : Z_i \to 2^{V_i}$ is the *edge incidence function*, that associates with each edge $z \in Z_i$ a set of two vertices, if z is a proper edge of U_i, and a singleton, if z forms a self-loop. It is assumed that when completing their tasks at the destination points, the robots do not block the path network, which is modeled as a loop with two edges (see Fig. 4) connecting the respective vertex $v \in D_i$ with the remaining part of the network. Moreover, we assume that sub-graph U_0 has a specific structure, namely it is constituted by $|H|$ edges that form a cycle, where $|H|$ is the cardinality of the robot set H.

Consequently, the total path network is given by the graph $U = (V, Z, \zeta)$, where $V = \bigcup_i^n V_i$, $Z = \bigcup_i^n Z_i$, and $\zeta : Z \to 2^{V_i}$ is the edge incidence function s.t. for each $i \in 1, .., n$, the restriction of the domain of ζ to Z_i results in $\zeta|_{Z_i} = \zeta_i$, and for each transit edge $z \in Z_{n+1}$, $\zeta(z) = (v, v') \in V_i \times V_j$ s.t. $i, j \in 0, \ldots, n$ and $i \neq j$, i.e., the vertices of each transit edge belong to two distinct sub-graphs. For each sub-network U_i, $i \in 0, \ldots, n$, we will distinguish the subset of transit edges $T_i = Z_{n+1} \cap Z_i$ that have a vertex in U_i. Finally, the whole MMRS will be given by the pair $HU = (H, U)$, specifying its two components – the robot set H and the path network U.

C. Feasible dynamics of MMRS

While MMRS as a whole is a closed system, each of the local sub-networks is open, as robots can travel from U_i to U_j through transit zones $z \in T_i \cap T_j$. Thus, when describing the operation of the sub-systems associated with particular sub-networks U_i, we also take into consideration the adjacent transit zones T_i. The dynamics of these sub-systems will be represented by the following automata.

Definition 4: Consider a MMRS specified by the pair $HU = (H, U)$. The DFSA $A_i(HU) = (S_i, E_i, \Gamma_i, \delta_i, s_{0,i})$, $i \in 1, \ldots, n$, abstracting the i-th, $i \in 0, \ldots, n$, sub-system of the MMRS is defined as follows.

1) The state set S_i is the set of vectors $s = [s(z)|z \in Z_i \cup T_i]$, where $s(z) \in (H \times \zeta_i(z)) \cup \{\texttt{null}\}$ describes the state of zone z; zone z with $s(z) = \texttt{null}$ is an *empty* zone in state s, while, for non-empty zones, the first component of $s(z)$, $s(z; 1)$, indicates the vehicle $h \in H$ occupying this zone, and the second component of $s(z)$, $s(z; 2)$, indicates the vertex $v \in \zeta(z)$ towards which vehicle h is moving on z.
2) The event set E consists of all the triplets $e = (z', z'', h) \in Z_i \cup T \times Z_i \cup T \times H$ such that $z' \neq z''$ and $\zeta(z') \cap \zeta(z'') \neq \emptyset$; each of these events corresponds to the transition of robot h from zone z' to its neighboring zone z''. If $z' \in T_i$ ($z'' \in T_i$) then event e represents an entry (exit, resp.) of robot h to (from, resp.) network U_i from (to, resp.) transit edge z' (z'', resp.). Otherwise e is a local event.
3) For each $s \in S_i$, $e \in \Gamma(s)$, i.e., event $e = (z', z'', h)$ is feasible in state s iff $s(z'; 2) = v \in \zeta_i(z') \cap \zeta(z'')$ and $s(z'') = \texttt{null}$, i.e., there is a robot in zone z', moving towards vertex v shared with z'', and zone z'' is empty.
4) If $\delta(s, e)$ is defined, the resulting state $s' = \delta(s, e)$ is given by: $s'(z') = \texttt{null}$; $s'(z'') = (h, v')$, where $v' \in \zeta(z'')$ and $v' \neq v$ if zone z'' corresponds to a proper edge, while $v' = v$ if zone z'' is a self-loop; finally, $s'(z) = s(z)$ for all remaining zones z.
5) The initial state $s_{0,i}(z) = \texttt{null}$ if $i \neq 0$ or $z \in Z_{i+1}$, i.e., each zone outside the docking station is empty. The initial state of the zones that lie on the cycle $c = v_1, z_1, v_2, \ldots, v_{|H|}, z_{|H|}$ that constitutes the docking station U_{n+1} is given by $s_{0,i}(z) = (h_j, v_j)$ s.t. $h_i \neq h_j$ if $i \neq j$.

Notice that local events that belong to two distinct sub-systems are independent. That is, for any A_i and A_j, $i \neq j$, the feasibility of a local event e in A_i neither depends on the state of A_j, nor its occurrence induces any changes in the state of A_j. In contrast to that, since each transit zone is shared by the two sub-systems that contain its two respective edges, an entrance or exit events that involves this edge is observed in both subsystems. Therefore the total system can be defined as the following composition of its sub-systems.

Definition 5: Given the sub-systems of MMRS $A_i(HU) = (S_i, E_i, \Gamma_i, \delta_i, s_{0,i})$, $i \in 1, \ldots, n$, the DFSA abstracting MMRS is a tuple $A(HU) = (S, E, \Gamma, \delta, s_0)$ such that:

1) $S = [s_1, s_2, \ldots, s_n]$, $E = \bigcup_{i=1}^n E_i$,
2) for each $e \in E_i \subseteq E$, $e \in \Gamma(s)$ iff $e \in E_i$ and $e \in \Gamma_i(s_i)$,
3) for each $e = (z', z'', h) \in E_i \subseteq E$, $\delta(s, e) = s'$ s.t.:
   a) if $z', z'' \in Z_i$, i.e., e is a local event in A_i, then $\delta(s, e) = [s_1, s_2, \ldots, s'_i, \ldots, s_n]$, where $s'_i = \delta_i(s_i, e)$,
   b) if $z' \in Z_i$ and $z'' \in T_i \cap T_j$, i.e., e represents the event of leaving, by robot h, sub-system A_i for transit edge z'' shared with A_j, then $\delta(s, e) = [s_1, s_2, \ldots, s'_i, \ldots, s'_j, \ldots, s_n]$, where $s'_i = \delta_i(s_i, e)$, $s'_j(z'') = s'_i(z'')$, and $\forall z \in Z_j \cup T_j$ s.t. $z \neq z''$, $s'_j(z) = s_j(z)$,
   c) if $z'' \in Z_i$ and $z' \in T_i \cap T_j$, i.e., e represents the event of entering, by robot h, sub-system A_i from transit edge z'' shared with A_j, then $\delta(s, e) = [s_1, s_2, \ldots, s'_i, \ldots, s'_j, \ldots, s_n]$, where $s'_i = \delta_i(s_i, e)$, $s'_j(z'') = s'_i(z'')$, and $\forall z \in Z_j \cup T_j$ s.t. $z \neq z''$, $s'_j(z) = s_j(z)$,
4) $s_0 = [s_{0,1}, s_{0,2}, \ldots, s_{0,n}]$.

It is convenient to view a state of MMRS in a graphical form, as a partially directed graph (PDG) $G = G(U, s)$, i.e., a graph that has the structure of U, and both undirected and directed edges. In $G(U, s)$, empty zones are represented by undirected edges, while a zone occupied by vehicle h moving towards vertex v is represented by a directed edge pointing to v and labelled by h.

D. Admissible dynamics of MMRS

In order that MMRS maintains its operational integrity and flexibility, it is essential that all of the robots preserve their ability to access every zone in the network. A system state that supports this feature will be called *live*, and defined as follows:

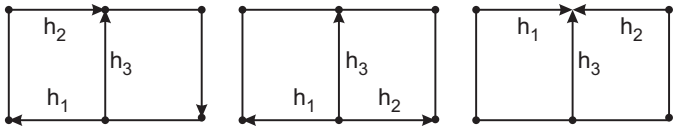

Fig. 2. Examples of MMRS states: a live state (left), a state from which a deadlock is unavoidable (middle), and a deadlock state.

Definition 6: In a MMRS $A(HU)$, state $s \in S$ is *live* iff for each robot h and each zone $z \in Z$, the strongly connected component of the transition graph $RG_{res}(A, s)$ contains a state s' s.t. $s'(z; 1) = h$, that is, s.t. robot h is located in zone z.

A phenomenon that can deprive robots from their ability to visit each zone in the system is the deadlock. Fig.2 depicts example live and not live states (using the aforementioned convention of representing MMRS states as PDGs).

If all states of a particular $A(HU)$ are live then the system is live. Otherwise one needs to consider a liveness enforcing supervisor $\Delta : S \to 2^E$ that indicates the *admissible* events $\Delta(s)$, and restricts the dynamics of $A(HU)$ to a live system $A_{res}(HU) = (S, E, \Gamma \cap \Delta, \delta, s_0)$. To ensure this, we build on the results of [10] that, due to the same zone-controlled traffic model, can also be applied to MMRS.

Theorem 1: In MMRS $A(HU)$, state $s \in S$ is live iff there exists a state $s' \in R(A, s)$ such that each directed edge in PDG $G(U, s')$ lies on a cycle.

A supervisor that employs this property as a sufficient condition for testing liveness of state s (and checks the reachability of a required state s' through subsequent condensations of $G(s)$) was proposed in [7].

With such a supervisor, the liveness of the whole MMRS is ensured in the following way:

1) The state of each sub-system A_i is kept locally live, i.e., such that it is always possible to reach in A_i, a state s' satisfying Theorem 1 without the necessity of any robot to leave A_i.
2) As discussed in Section II, the global route for a robot is established in the negotiation process between the controllers of the sub-networks $U_{i_1}, U_{i_2}, \ldots, U_{i_k}$ that the route intersects. Once a route is accepted, in each such a subnetwork U_{i_j}, $j = 1, \ldots, k$, we reserve a unit of its capacity c_{i_j}, where c_{i_j} is equal to the number of zones in U_{i_j}. A route can only be accepted in state s if it is admissible in s, that is, if for each $j = 1, \ldots, k$, $sl_{i_j}(s) \geq 2$, where $sl_{i_j}(s)$ is the slack of U_{i_j} in state s, defined as the difference between the capacity of U_{i_j} and the sum of the current numbers of robots and the reserved capacity units in U_{i_j}.

The formal description of the problem presented in this section allows us to present the architecture of the proposed implementation of the system.

IV. Distributed architecture of the MMRS controller

A fundamental part of MMRS is the system of subnetwork controllers, whose communication structure reflects the connections between the subnetworks $U_{i_1}, U_{i_2}, \ldots, U_{i_k}$. The proposed design assumes that each controller is constituted by three modules: *Local Network Supervisor* (LNS), *route planner* (RP), and *communication module* (CM).

The role of LNS is to steer the robots towards their goals in particular subnetworks U_i. LNS is also responsible for keeping each subsystem A_i locally live. It means that decisions considering the robots' movement must be determined by LNS in such a way that ensures the behavior of A_i (when viewed as a separate system) consistent with the requirement of Def.6.

The role of the second module, RP, is to determine a sequence of subnetworks $U_i, U_{k_1}, U_{k_2}, \ldots, U_{k_r}, U_j$ which a robot must pass in order to reach its final goal in the case when it is located in another subnetwork U_j. To perform this task the planner must know the structure of connections between subnetworks $U_1, U_2, \ldots, U_n$. Technically, we represent these connections by introducing another subnetwork, denoted by U_{n+1}, whose nodes correspond to the subnetworks $U_1, \ldots, U_n$, and whose edges correspond to the transit edges of U. The information on this connection structure is obtained by each RP with help of the respective CM.

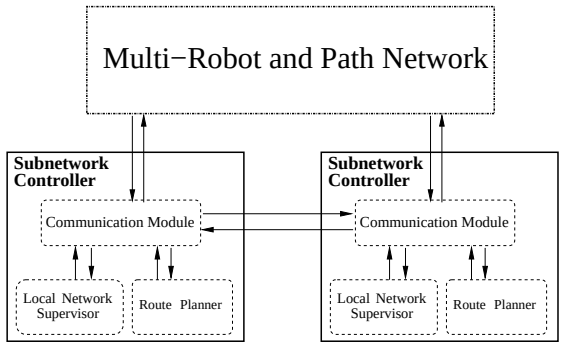

Fig. 3. MMRS Structure.

In order to acquire the information about the network structure, it is sufficient that initially each controller only knows its immediate neighbors. The defined communication protocol allows the module CM to broadcast the question to these neighbors, that is subsequently retransmitted by the controllers to their own neighbors. The answers sent by all controllers make it possible to create the graph representing the sub-areas' connection. In this way, each controller can build its own copy of the subarea network based on the current information about the network U. If a connection between two sub-areas is added or removed both controllers of these sub-areas must broadcast the message to their neighbors which than is retransmitted to other modules. It allows all the controllers to update the model of the subarea network in the case of any modification of its structure. This additional feature of the defined protocol is not necessary in the discussed project (because of a static structure of the path network), but has been included in the design for the sake of its further development.

When a new goal is assigned to a robot, it sends a message to the controller of the subarea in which it is currently located. The conveyed information includes the name of the node being

a new goal for the robot. The controller determines the location of the node and checks whether or not it is in this subarea. In the former case the task is handled by the LNS. It finds dynamically the path to the goal and coordinates the movement of all the robots in this subarea so that they avoid deadlocks. The control strategy applied by LNS is described in more detail in the next section. In the latter case, when the final goal is not in the considered subarea, the controller broadcasts the question to all other controllers in the MMRS network. The question contains the label of the goal node, which is unique for all the network. After identifying the sub-network U_j where the goal is located, the controller calculates a route to U_j in the sub-network graph according to the method is described at the end of subsection III-D). The route is represented by a sequence of sub-networks $U_i, U_{k_1}, U_{k_2}, \ldots, U_{k_r}, U_j$. In this case, LNS finds the path for the robot to an entry to U_{k_1}. After crossing the entry the management of the robot movement is taken over by the controller of the sub-network U_{k_1}. The discussed process is continued until the robot enters the sub-network U_j containing its goal. In this sub-network, the LNS module of the U_j controller supervises the robot movement, ensuring that eventually the robot attains its final goal.

By now, we have completed the implementation of the most crucial part of the control system, that is the LNS module. The other components of the network control system are conceptually ready, but undergoing the coding phase. In the next section, we present in more detail the construction of LNS, and give an example of its operation.

V. Local Network Supervisor

As mentioned in the previous section, the role of LNS is to guard the movement of the robots so that the state of the network is kept live, as well as to determine and supervise the execution of robots' routes that let them attain their local goals. The mechanism of liveness enforcement is based on Theorem 1, and implemented in the form of a cycle-condensation algorithm. If a considered state s is safe then the condensation procedure folds the partially directed graph representing s to a single vertex. For more details, we refer the interesting reader to [10] and [7].

The routes of the robots are planned dynamically. That is, each time a robot arrives at the end of its current zone, LNS plans its further route step, represented by an edge in the path network graph, and sends a respective message to the robot. Clearly, these decisions can reflect various routing strategies and robot prioritization schemes. In the current version of the project, we implement a heuristic algorithm, based on *the shortest path policy* concept. Generally speaking, the algorithm tries to enforce that each vehicle h takes the shortest route to its current goal by calculating for each vehicle the shortest path and, in the case when the first edge of the path is currently unavailable, delaying the vehicle until the edge becomes available or some arbitrarily determined waiting time n_{del} has passed. More formally, the strategy can be presented as given below.

The shortest-path policy. Associate with each vehicle $h \in H$ a time counter, set to $tc(h) = 0$ when h ends its travel in the current zone.

1) Create the set $\tilde{H}$ containing all the robots h_i that have attained the end of their respective zones $z_i = z(h)$, and set the counters $tc(h_i)$.
2) For each robot $h_i \in \tilde{H}$ find the shortest path π_{h_i} from vertex $v = z(h_i; 2)$, which is the end of zone z_i.
3) Create $\tilde{H'}$ containing all the robots h_i that meet the conditions: $h_i \in \tilde{H}$, and the event $e_j = (z(h_i), z'_j, h_i) \in \Gamma_{res}(s)$, where z'_j is the next zone determined by the path π_{h_i} and s is the current state of the system.
4) Find the robot h_k for which
   $tc(h_k) = max_{h_l \in \tilde{H'} \wedge tc(h_l) > n_{del}} tc(h_l)$.
   If such a robot exists then select the event $e_k = (z(h_k), z'_k, h_k)$ and finish the procedure.
5) Find the robot h_k for which
   $tc(h_k) = max_{h_l \in \tilde{H'}} tc(h_l)$.
   If a such robot exists then select the event $e_k = (z(h_k), z'_k, h_k)$. If a robot doesn't exist then no event is selected.

The selected event represents an advancement of some robot in its route, that in the current state of the system is expedient from the viewpoint of the assumed routing criterion. This selection is further translated by LNS into a decision of zone transfer, which is next passed to the considered robot. The procedure is repeated as long as there is a robot at the end of its current zone; otherwise it is suspended and triggered again by an event corresponding to the completion of the currently executed route step of some of the robots. If no event is selected, then the robots awaiting the allocation of a new zone must wait until some other robots attain the end of their zones and get a permission to enter new ones. This causes a change of the system state, which eventually enables the waiting robots to resume their travel. The mechanisms underlying the construction of the liveness enforcement policy and the routing policy formally ensure the control correctness, that is the occurrence of no deadlock or starvation phenomena, and hence, the ability of each robot to eventually reach its goal. More specifically, the system is kept live, as its of its subnetwork is kept live, and the routing policy guarantees that each robot can complete its task in a finite time.

Below, we illustrate the discussed concept with four screens obtained in the computer application supporting the development of the MMRS control system [9]. Fig. 4 gives an example of a number of robots travelling in a local path network, i.e., a sub-network supervised by a single controller. Each robot has to reach a node being its local goal.

A more complex task for a robot is a mission, i.e., a sequence of nodes to be subsequently visited before attaining the goal node. Fig. 5 shows the mission of the robot with the label `Robot_9`, which consists of the nodes `Node_2`, `Node_5`, `Node_4`, and finally `Node_21`. These nodes are marked by numbers *1*, *2*, *3*, *4*. The other robots have similar missions, containing at least three nodes. The example of mission lists is presented in the graphical window of the application in Fig. 5. The controller conducts the movement of the robots in the manner that they avoid deadlocks, complete their missions, and reach their goals. Fig. 6 shows the final state of the system after reaching their goals by all the considered robots. As an

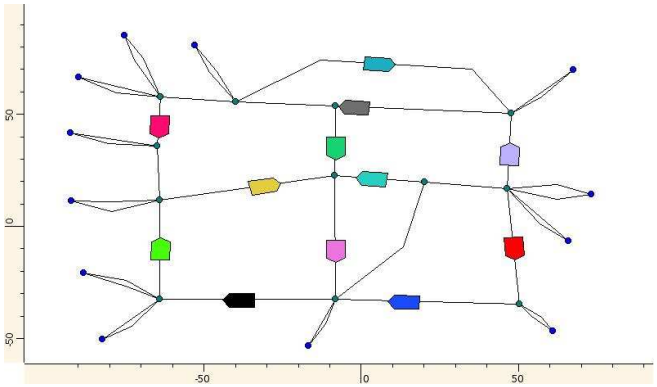

Fig. 4. The example of the graph representing path network and robots which are managed by a single controller.

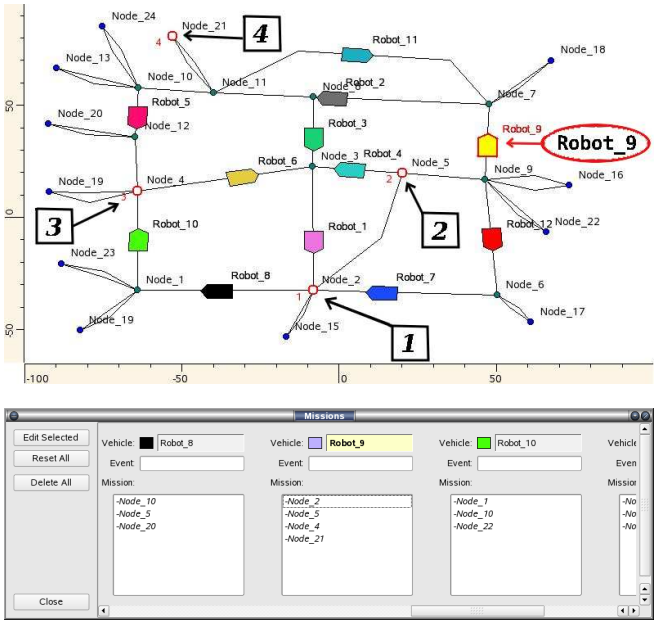

Fig. 5. The example of a mission for a robot `Robot_9`. The nodes which have to be visited before reaching the local goal are marked by numbers *1*, *2*, *3*. The number *4* marks the local goal of the robot.

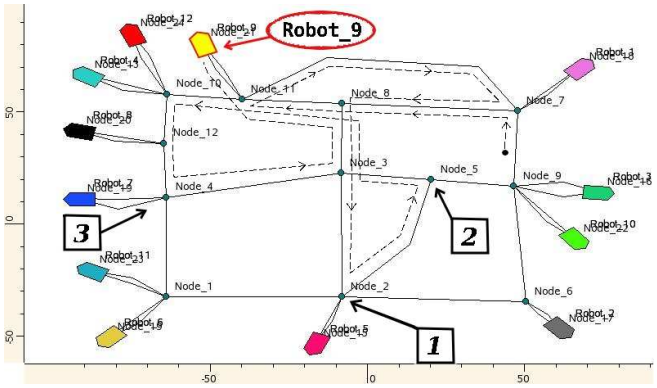

Fig. 6. The final state of the system after reaching by robots their goals. The path of `Robot_9` is depicted.

example this figure shows the path of `Robot_9`. Comparing the initial state presented in Fig. 5 and the final state of the system (see Fig. 6) it is worth to notice that the task which has been solved is not trivial.

VI. Conclusions

The paper presents a concept of distributed supervisory control for a system of mobile robots moving in a common path network. The control concept is based on a formal mathematical model, which guarantees the correct realization of its mission by each particular robot, as well as their correct co-existence in terms of collision and deadlock avoidance. The implementation of this model is still under development, however the crucial part of the control construction, i.e., the LNS module has already been done. The work on the controller is assisted by the development of a computer tool that allows to test the efficiency of routing strategies and/or robot and task prioritization schemes.

The idea of the distributed robot supervision was taken from the airplane and railway control. Such systems are more flexible, independent of the workspace size, and able to operate in complex environments. In the future, we plan to study the influence of the distribution level, in terms of the number of the subnetworks that constitute the system, on the MMRS performance. Moreover, we intend to experiment with giving more autonomy to the mobile robots, so that they could accomplish their tasks based on local calculations of the control decisions. The logics of the developed coordination model allow their further distribution, i.e., a direct implementation of the developed supervisory control in the robots' controllers, which then will also take over the responsibility of the communication with other robots in the system.

References

[1] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[2] M. P. Fanti. Event-based controller to avoid deadlock and collisions in zone-control AGVS. *Int. J. of Production Res.*, 40:1453–1478, 2002.

[3] Steven LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[4] Y.-H. Liu, S. Kuroda, T. Naniwa, H. Noborio, and S. Arimoto. A practical algorithm for planning collision-free coordinated motion of multiple mobile robots. In *IEEE Int. Conf. Robot. Automat.*, volume 3, pages 1427–1432, 1989.

[5] F.R. Noreils. Integrating multirobot coordination in a mobile-robot control system. In *IEEE Int. Workshop on Intelligent Robots and Systems*, volume 1, pages 43–49, 1990.

[6] S. A. Reveliotis. Conflict resolution in AGV systems. *IIE Transactions*, 32(7):647–659, 2000.

[7] E. Roszkowska. Liveness enforcing in closed AGV systems with dynamic routing. In *Proc. of ICRA'04*, pages 5165–5170. IEEE, 2004.

[8] E. Roszkowska. Formally correct asynchronous control for guidepath-based traffic systems. In *8th IFAC Symposium SYROCO'06*, 2006.

[9] E. Roszkowska and B. Kreczmer. Control and simulation system of transport vehicle motion in a guidepath network. In *Advancements in Robotics*, pages 107–116. WKŁ, Warsaw, 2006. (in Polish).

[10] E. Roszkowska and S.A. Reveliotis. On the liveness of guidepath-based, zone-controlled dynamically routed, closed traffic systems. Technical Report PRE/I-6/13, Wroclaw University of Technology, 2006. Also, under review in IEEE Trans. Automat. Control.

[11] N. Wu and M. Zhou. AGV routing for conflict resolution in AGV systems. In *Proc. of ICRA'2003*, pages 1428–1433. IEEE, 2003.