Collaborative Embedded Systems – A Case Study

Bernd-Holger Schlingloff
Humboldt Universität zu Berlin, and
Fraunhofer FOKUS, Berlin

Henry Stubert
InSystems Automation GmbH, Berlin

Wojciech Jamroga
Institute of Computer Science,
Polish Academy of Sciences

***Abstract*—A collaborative embedded systems (CES) is an intelligent agent in a cyber-physical system which cooperates with others by negotiation to fulfill a common task. In this paper, we consider autonomous transport robots as CES. These robots are used in production environments like factories and storage halls to realize the flow of materials within the production process. We describe the software hierarchy of the agents with self-localization, route planning and job scheduling. Then we discuss implementation strategies and possible benefits of a collaborative approach. Finally, we report on our modeling of the system for simulation and verification.**

I. Introduction

An *embedded system* is a computational system which is a fixed component of a technical system. Embedded systems have become ubiquitous in our technical world — for example, consider an electronic control unit in an automobile, a medical device implanted in the human body, or a CNC machine in an automated factory. Presently, these systems are increasingly being connected, to form complex *cyber-physical systems* (CPS). In a cyber-physical system, several embedded systems are cooperating to achieve a common goal. Examples are a network of driver assistants in a car achieving road safety, a body-area network monitoring vital functions of a patient, and an assembly line supervision system managing the production of some goods. However, for present-day CPS,

- the task which is to be performed,
- the actors which are to cooperate on the given task,
- the environment in which the task is set, and
- the way how to perform the task

are completely pre-determined. That is, at design time it is fixed which components constitute the CPS and which role each computational system has, in which way it shall contribute to the overall function, and what the intended environment for the deployment is. For example, in a car the behaviour and interaction of the anti-blocking system, electronic stability control, and traction control are completely fixed when the car leaves the factory. For this reason, current CPS are limited in their flexibility and area of application. They can hardly adapt to changing conditions, reconfigure themselves, and modify or optimize their behaviour for changing tasks. Including, e.g., a new electronic braking assistant in a given car design, or a new machine in a given production chain currently is a challenging task.

In order to achieve more flexibility and wider application areas, future CPS must be designed in a way such that their components collaborate with one another. A *collaborative embedded system* (CES) is an embedded system which can interact, negotiate and cooperate with others to fulfill various tasks. That is, several CES can be grouped together dynamically and for different tasks to form a *collaborative CPS*. The physical environment in which the composed system shall operate is only partially known at design time. Moreover, each actor in the group is more or less independent in the way it contributes to the common goal. Even the goal may vary to a certain extent; a collaborative CPS can adapt to changing tasks and side conditions. Examples of collaborative CPS would be

- a fleet of self-driving cars, where each car collaborates with others to optimize road traffic and avoid collisions,
- a group of humanoid robots playing (and winning) against a human soccer team, and
- a factory adapting its production capacities to customer demand, up to and including "lot size 1".

It is clear that truly collaborative CPS do not yet exist; however, large efforts are being taken towards this end. In this paper, we describe our enterprise to construct an industrial collaborative CPS, its challenges and preliminary results.

Before doing so, we compare our notion of "collaboration" with related concepts from the literature. Collaborative embedded systems can be seen as a special case of a broader metaphor, namely that of agents in a multi-agent systems [Wei99], [Woo02], [SLB09]. A *multi-agent system* (*MAS*) is a system that involves several autonomous entities that act in the same environment. No precise definition of an agent is commonly accepted, but most authors agree that an agent can be:

- *autonomous*, i.e., operating without direct intervention of others having some kind of control over its actions and internal state,
- *reactive*, i.e., reacting to changes in the environment,
- *pro-active*, i.e., taking the initiative whenever appropriate,
- *goal-directed*, i.e., acting to achieve a goal,
- *rational*, i.e., selecting their action optimally given the knowledge that it possesses,
- *social*, i.e., interacting with others (which can take the form of cooperation, communication, coordination, and/or competition),
- *embodied*, i.e., using sensors and effectors to read from and make changes to the environment, and
- *intelligent*... whatever that means.

Agents in a MAS are often software entities, as opposed to CPS where the actors are physical artifacts supervised

by an intelligent controller. Thus, CPS have to cope with the imprecision and imperfection of physical sensors and actuators, and the results from MAS literature should be taken with a grain of salt when applied to the design of CES. On the other hand, multi-agent systems have been studied in computer science and AI for almost thirty years now, and the topic of collaboration was featured very prominently in the research. In particular, some cooperation and negotiation mechanisms have been developed for collaborative MAS that can be used in CES as well, cf. [Wei99] and especially [San99]. Such mechanisms typically borrow from game theory, and use mechanisms based on auctions and/or bargaining [OR94], [LBS08] to create a "market" where agents split the workload and the profits in a distributed autonomous way. A survey of market mechanisms for multi-robot coordination has been presented in [DZKS06].

Autonomous systems are characterized by their ability to accomplish complex tasks independent from human operators. This is of particular interest in contexts where human interaction is too expensive or too dangerous. Therefore, autonomous systems are being developed for various tasks (autonomous driving, autonomous flying, autonomous manipulator robots, etc.). CES may or may not be autonomous; the focus in CES is on interaction rather than autonomy.

Open systems are systems which are designed for an unknown or only partially known application area and environment. This term has been coined as a counterpart to "closed systems" which can be validated and tested in a "closed control loop". Various calculi have been proposed for the verification of open systems. CES can be seen as a special type of open systems, acting in a physical environment.

Self-adaptive systems can modify their behaviour autonomously, to react to changing environments. According to our above definition, a collaborative CPS is a self-adaptive system. Other "self-*" properties used to describe introspection of technical systems into themselves are: *self-configuring, self-aware, self-healing, self-learning, self-believing,* and others. Clearly, a CES can or must possess all of these properties to a certain degree. However, since in technical contexts these terms are used very inhomogeneously, no concrete design principles for CES can be derived from them.

In this paper, we exhibit challenges and solution paths for a concrete case study of industrial CES: a system of transport robots to be deployed in storage facilities and production plants. The system as a (non-collaborative) CPS has been in use for several years and at several plants. We describe the domain, localisation, path planning and job scheduling of the system currently in use. Then, we discuss the challenges for the current design and the benefits to be gained by a collaborative control. We describe different modeling approaches and implementation strategies. Then, we give some preliminary simulation results on simulation and verification of the new system. Finally, we summarize our discussion with an outlook on future work.

Figure 1. The proANT line of autonomous transport robots.

II. A CASE STUDY: AUTONOMOUS TRANSPORT ROBOTS

Our case study is the proANT line of autonomous transport robots (see [InS16]). Autonomous transport robots are vehicles designed for carrying loads in factories and storage sites. A typical fleet consists of 4-20 robots which can carry 50-200 kg load each. Their main use is in production logistics to ensure a timely delivery and disposal of material to different sites in a production process. In contrast to stationary transport systems such as conveyor belts or powered roller conveyors, driverless vehicles offer some significant advantages:

- Scalability: Increased production demand can be met simply by deploying more vehicles, instead of costly upgrading and remounting the conveyors;
- Changeableness: Changing the layout of a production process can be done easily, no stationary equipment has to be rebuilt;
- Redundancy: If a single vehicle fails, it can be replaced by others. With stationary equipment, even a small failure of a hardware module (drive, sensor) often means that the whole process is halted;
- Reduced space requirements: In general, vehicles use less space than conveyors; moreover, they can be stowed away if not in use.

In contrast to traditional automated guided vehicles (AGV), transport robots not only drive autonomously, but can navigate freely in their environment, without being bound to a fixed track. Given the description of a target point, the robot autonomously determines its current position, the optimal route to the destination and the maximal speed along this route. Therefore, it multiplies the benefits of driverless vehicles in production: A fleet of transport robots can be extended even during operation of the factory; no intervention in the running operation is necessary. Changing the location of a machine requires simply to adapt the corresponding target point in the internal map of the robots; it is not even necessary to designate new routes. With AGVs, the flow of material is distributed among several vehicles; however, due to the fixed track guiding, a broken vehicle can cause road blocks. Autonomous transport robots can try to circumvent or overtake such obstacles. The dynamic path planning also allows autonomous transport robots to share walkways and corridors with human workers; temporary road blocks such as people and deposited loads can be passed.

The cognitive abilities of (a system of) autonomous transport robots can be structured into three layers: self-localization, route planning, and job scheduling.

A. Self-localization

A basic prerequisite for autonomous mobile robots is self-localization. The robot needs to know where it is in order to behave intelligently. Indoor localization is usually done by sensing different parameters of the environment. In proANT robots, an integrated 2D laser scanner continually measures the distance to neighboring objects (walls, machines, obstackles, humans). This laser scanner is mounted in max. 200mm height above ground, parallel to the floor. It can scan a 2D profile of the environment of up to 270°, with an angular resolution of approx. 0.5°. The detected distance ranges from few cm up to 30m with a measurement fault (compared to the actual distance) below 3%. The sampling takes around 80 ms.

Laser scanners were originally included and certified in mobile robots for functional safety. The European Norm EN1525 "Safety of industrial trucks – driverless trucks and their systems" requires that standing objects of diameter above 70mm must be detected and the vehicle must stop before them. If the laser scanner detects an object in the immediate vicinity of the robot, it shuts down the power to the drive and thus stops any movement. Besides this basic safety feature, the laser scanner can be used for self-localization as follows.

For relative positioning, each robot has an odometric system. It consists of an incremental encoder of the movement of the drive wheels to measure the traveled distance, and a gyroscopic sensor to measure the relative angle of the robot in space. Since these measurements are not very precise, they must be adjusted by the information from the laser scanner. As a setting-op operation, one robot from the fleet is moved around manually (e.g., by joystick) in the environment in which it shall operate autonomously later on. During this initiation, all distance profiles from the laser scanner are recorded and combined with the data from odometry. From this, a map is produced which reflects the contours of objects as sensed by the scanner in 200 mm height. This map is downloaded from the robot and manually edited. A graphical editor is used to add one-way passages and no-go-areas to the map. Moreover, also target positions (unique name, coordinates and approaching angle) are added to the map. The edited map is uploaded to every robot in the fleet.

Upon start-up, each robot starts an initial localization to determine its current position. This is done by comparing the currently received distance profile from the laser scanner with a coherent fraction of the map. The matching yields a position and the probability of correct positioning. Using thresholds, this probability can be used to trigger various actions (e.g., switching on a green light for successful initial localisation, or sounding an alarm if the position is not found).

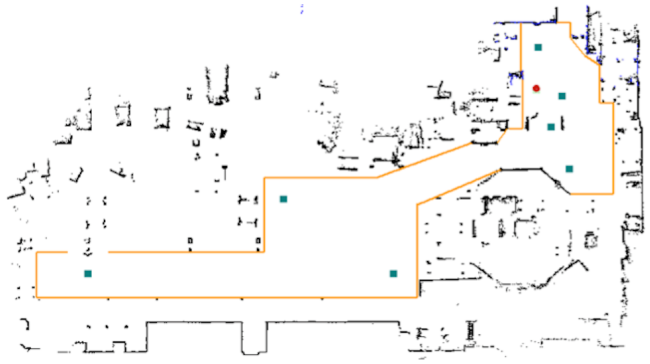

Figure 2. Map with outlines, no-go-areas and target locations

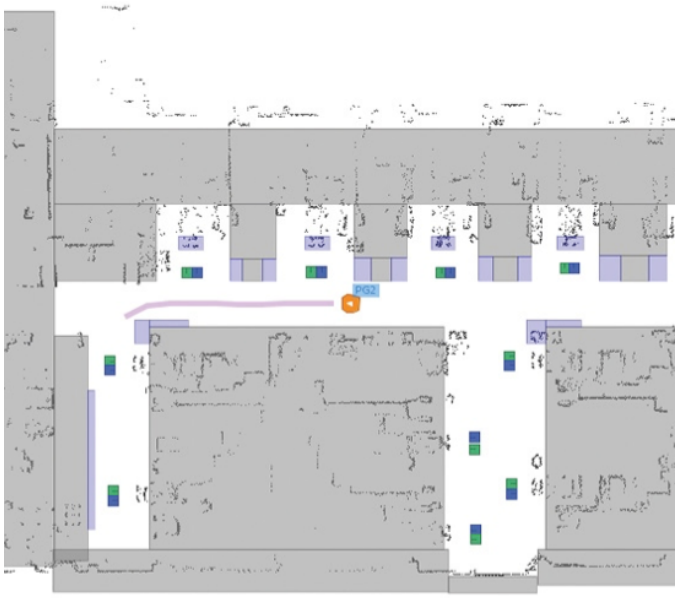

Figure 3. Robot with planned path

During autonomous operation, the localization algorithm is triggered at regular intervals, depending on the information about the traveled distance from odometry. If the probability of correct localization is above threshold, the calculated position is assumed to be correct and the odometric position is updated with this information. This way, the robot is constantly aware of its own position within the mapped area. The above algorithm turns out to be quite stable and so far never caused any problems.

B. Route planning

During autonomous operation, the robot has to travel to specific target locations. It thus has to find the optimal route from its current position to this target location, given the map which determines the accessible floor space. An important factor for route planning is the size of the robot, i.e., its width in the direction of motion. The map is divided into squares which are a fraction of the robot size. From the map it is determined whether a square is drivable or in a no-go-area. All no-go-squares and adjacent squares up to half of the robot size are marked occupied and exempt from path planning. For all remaining squares, a cost is calculated which decreases with the distance to occupied squares. Then, a path is calculated which constitutes a smooth spline function through the lowest cost squares.

Since the robots are operating in a dynamic, real-life environment, they have to be able to react to unexpected obstacles and road blocks. In particular, this also includes other robots from the fleet which travel in the opposite direction. Here, currently the "dynamic window approach" is being used [FBT97]. If the robot during its trip detects an obstacle which is not in the map, the corresponding squares are marked temporarily occupied, and costs are recalculated. Since the robot keeps moving towards the obstacle, the cost function is dynamic, respecting current speed and acceleration. This dynamic guarantees a fast travel towards the destination, while at the same time avoiding collisions. Since for the robot human workers and other robots are moving obstacles, the dynamic window calculation is triggered cyclically. Thus, the path is frequently recalculated. To avoid that the new path deviates too much from the previous, the cost of the currently planned path is reduced and it thus is preferred. Since the path calculation uses considerable computational efforts, the path is calculated only within the range of the laser scanner.

C. Job Scheduling

Currently, job scheduling is done by a centralized server which can communicate with all robots of the fleet via WLAN. If a machine is in need of supply or disposal of material, it sends a request to a master control station. This station inquires the current availability with the storage management system. It then generates a transport job description, including origin and destination as well as the priority of the job. The transport job is assigned to a particular robot from the fleet, according to a given scheduling strategy. Moreover, corresponding messages are sent to the storage and the production machine. Monitoring handshake messages between robot and machine or storage, the master control system supervises the correct execution of transport jobs. The times of status changes of all transports are logged in a data base.

The master control station also monitors the battery status of all robots in the fleet. If the available energy drops below a certain threshold, it generates a recharging job for the corresponding robot. The robot then moves to the designated charging station and waits until it gets the next job.

The scheduling strategy determines the assignment of tasks to robots. A good strategy both guarantees a timely accomplishment of jobs and a high degree of capacity utilisation. A high use of capacities allows to minimize the size of the fleet, reducing investment and maintenance costs. Thus, the scheduling should avoid empty trips as well as "lazy" robots. Important factors to be considered during scheduling are

- the distances of unassigned robots to the origin of a job,
- the current battery status of unassigned robots,
- the current traffic situation, and
- the priority of a job.

Whenever a request for a transport job arrives, the job scheduler checks which robot is available. If there is more than one robot available, it is determined which of these has the best conditions to fulfill the job, according to the above criteria. This robot is then assigned the task. If there is no robot available, the job is postponed until some robot finishes its current job. If there are several waiting jobs, the one with highest priority is scheduled first. Thus, jobs are basically scheduled in a first-come-first-served way.

A potential improvement would be to consider also robots which are currently executing a job for scheduling. The distance of the target positions of a loaded robot to the origin of a new task might be smaller than the distance of all idling robots. However, due to dynamic changes in the traffic situation, the time for traversing a given distance varies almost unpredictably. Thus, this improvement is not (yet) implemented in the actual proANT system.

Simulations done by Yan et al. ([ZYAC12]) indicate that the average time for job completion could be reduced by up to 20% if jobs could be scheduled in advance. If the frequency of requests by certain machines or for certain materials is known in advance, idling robots could be sent on their way even before a request is issued. However, in the applications we are targeting this is not a realistic assumption.

Challenges: The proANT system has been in use at seven customer sites for several years and has proved to be stable and reliable. However, there is always room for improvement.

Firstly, the central master control station is a single point of failure. If this software should break down, then production would stop. Of course, there could be a redundant server; however, the principal problem remains.

Secondly, wireless communication may not always be reliable. In harsh factory environments, the wireless connection may be distorted, and certain areas may be shadowed by walls and machines. Therefore, robots may not be able to contact the server for an unpredictable amount of time.

Thirdly, a number of problems may disrupt the smooth execution of the production chain: Robots or machines may fail, roadways may be blocked, there can be delays and traffic jams on passways, the battery capacity of a robot may be exhausted due to long waiting times, etc. If such unforeseen circumstances occur, usually the whole order status must be re-initialized.

To cope with these issues, we are currently implementing a collaborative approach to job scheduling which is described in the next section.

III. Collaborative transport robots

In order to improve the overall performance, we are investigating a collaborative approach for fleets of proANT robots. That is, we are implementing a distributed decision making algorithm, where each robot decides upon which jobs to accept. The central server is just the interface to the process, accepting requests for delivery and deposit of materials. The requests are sent via broadcast to all receiving robots. The robots then negotiate among themselves who takes the job.

Decentralized methods for robots to allocate and re-allocate resources and tasks among themselves have been extensively studied in multi-agent systems, cf. [San99] for a theoretical introduction and [DZKS06] for an application-oriented survey. The challenge is to obtain effective coordination and

reasonably optimal decision making for a team acting in a dynamic, partially known and time-constrained domain where different tasks and resources can interfere, either positively or negatively. Such mechanisms have been used in AI for at least 35 years [Smi80]. The typically used mechanism is that of an auction, see [KKT10] for an excellent overview.

Auction-based coordination is easy to understand, simple to implement, and broadly applicable. A natural idea would be to endow each robot with "virtual money". On one hand, the robot would earn new money by completing transport jobs. On the other hand, it would spend money to "buy" a job when having successfully bidden in an auction. Appropriate money would be also deduced from the robot's account according to the time and energy costs of completing the job. Each robot would freely decide how much it is willing to bid for which job, so that it maximizes its payoff. Hence, coordination would emerge from local decision making of self-interested rational parties. A similar approach can be applied to route planning, and even to resolution of local conflicts and collision avoidance through peer-to-peer bargaining.

To most ends and purposes, however, the above scheme involves a level of sophistication that is not really needed for coordination of simple robots. In most existing applications, robots bid their *costs*, and the robot with the *lowest* bid gets the job. Thus, for each offered job, every robot calculates the cost of its execution, depending on the current position, load status, and already accepted jobs. Then, it places a bid for the job containing the calculated cost, in competition with the other robots. The robot with the lowest overall cost wins the bid and gets the job. It is also possible that, whenever a new job has been generated, all the jobs that are not yet in execution are offered up for sale again and being reallocated.

The envisaged advantages of the decentralized auction-based approach are:

- More flexibility and robustness: Robots need no longer be centrally registered and administered. There is no single bottleneck in the system anymore (the central server) whose failure would inevitably lead to a complete breakdown. Moreover, robots which are – permanently or temporarily – unavailable do not participate in the negotiations and thus are not assigned jobs.
- Higher throughput: Since each robot is responsible for its own tasks, it can try to optimize the jobs it accepts.
- Better scalability: If the transport capacity of a fleet is to be increased, new robots can simply be added. Even heterogeneous fleets, where each robot has different capabilities, can be administered this way. Moreover, the job issuing instances need not know which robots are available.
- More security: Taking effective control of a decentralized system by an intruder is much more difficult than for a centralized one.

Note that, in our case, jobs are possibly interrelated. For example, it may be beneficial for a robot to combine delivery of an item with picking up another item from a neighbouring machine. There are several approaches to auctioning such jobs. So called *parallel auctions* offer each item independently. This approach is computation-efficient, but often results with non-optimal allocation where positive synergies between jobs are neglected. At the other end, we have *combinatorial auctions* where agents bid for *bundles of goods* – in our case, subsets of transport jobs. Those are efficient allocation-wise, but require the robots to generate and exchange exponentially many bids, which is seldom feasible in real-time applications. A reasonable compromise between the two extremes is offered by *sequential single-item (SSI) auctions*, where the "goods" are allocated in one multi-round auction. During each round, every robot bids on each unallocated transport job. The outcome of the round is to allocate one additional job to one bidding robot. It has been proved that the output of SSI auctions is guaranteed to be reasonably close to the optimal allocation, and experimental results showed very promising performance [LMK$^+$05]. Our plan is to first use parallel auctions, and refine it to SSI auctions in the second step.

The more sophisticated approach with "virtual money" has some further advantages. First, it allows for more reliability, as jobs with a higher priority can be given a higher reward, and thus they will be treated earlier. Moreover, it can further improve throughput: since "idling" costs money, each robot has an incentive to accept as many jobs as it can handle. Last but not least, it ensures fairness of auction outcomes in a long run and supports fair resolution of conflicts, which leads to a balanced use of robots in the team. However, the virtual money approach will require much more fine-tuning than the simple auctioning with costs, and therefore it is planned for later future.

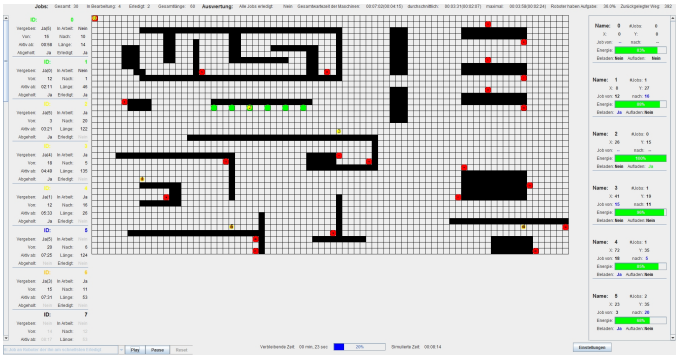

Figure 4. Simulation environment with robots, machines, storage, and power outlets

IV. Preliminary Results

In order to identify the significant parameters for the actual system, a simulation environment was set up in [Sit16]. A screenshot of this simulation environment is shown in Figure 4. In this simulation, it is possible to vary the number of robots, stations, and jobs. The simulation allows to choose between different job assignment strategies:

- First-come-first-served: The first free robots gets the next job (as in the current proANT system).
- Best-fit: The robot with highest energy is scheduled.

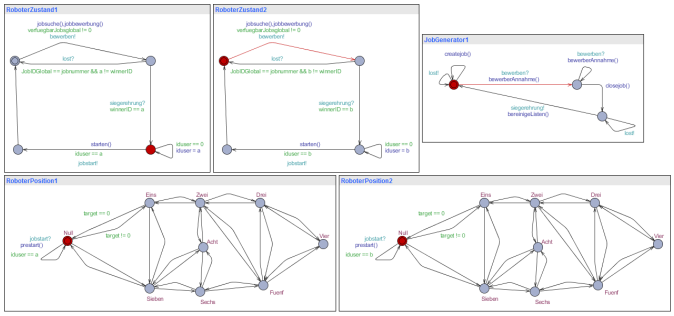

Figure 5. Automata model for verification

- Collaborative: The jobs are assigned via bargaining.
- Random: A new job is assigned to an arbitrary robot (for comparison only).
- Optimal: Jobs are scheduled in advance to the robot most suited (for comparison only).

A preliminary observation is that the simulation results are only "interesting" if the load is within a "balanced" range. If there are many available robots and only few tasks, all tasks are immediately scheduled, and the strategy is unimportant. If there are only few robots and much more tasks than these robots can handle, overall performance deteriorates with all strategies.

In order to assure that the overall system satisfies required properties, we are also verifying our approach via model checking. We are building various models (timed and untimed) for the UPPAAL model checker [UPP16], and are formulating properties in temporal logic. Then, we are employing UPPAAL to simulate and analyze our verification model. Figure 5 shows a verification model with two robots and seven stations [Wal16]. A list of properties is given in Table I.

Table I. Example properties for verification

Safety and Liveness	The system never deadlocks.
	Every job is eventually accomplished.
	Every job is accomplished within its given time.
	No robot ever runs out of energy.
Fault tolerance	The system continues if roads are blocked.
	The system continues if up to n robots fail.
	If a station fails, other jobs are not affected.
Flexibility	Additional robots do not affect the system.
	Additional stations do not affect the system.
	The system works for different floor layouts.
Security	External intruder cannot bring down the system.
	Traitor robots cannot bring down the system.
	Traitor stations cannot bring down the system.

Whereas safety, liveness, and fault tolerance can be verified by "classical" means, flexibility is verified by increasing and decreasing the number of stations and robots and varying the floor plan, respectively. A formal verification of flexibility would require parametric model checking. Security currently is neither implemented in the system nor in the model. To verify security, we need authentication of agents and assumptions about the encryption of messages.

V. Conclusion

We considered autonomous transport robots as collaborative embedded systems. In contrast to the traditional, centralized management structure we devised a system with distributed decision making, where each robot is responsible for its own actions. Each robot is autonomous for its localization and route planning, whereas several robots collaborate on the scheduling of jobs. Expected benefits are better scalability, easier maintenance, more robustness and better use of resources. We believe that the described architecture is typical for a number of similar applications, e.g., in industry 4.0. Preliminary results in simulation and verification are promising. However, for a practical deployment in real production sites, much more work needs to be done.

References

[DZKS06] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[FBT97] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[InS16] InSystems. Automation GmbH: proANT Automatic Navigating Transport Vehicle, 2016. `https://www.proant.de/`, last accessed Feb 2016.

[KKT10] Sven Koenig, Pinar Keskinocak, and Craig A. Tovey. Progress on agent coordination with cooperative auctions. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

[LBS08] K. Leyton-Brown and Y. Shoham. *Essentials of Game Theory: A Concise, Multidisciplinary Introduction*. Morgan & Claypool, 2008.

[LMK+05] M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, S. Koenig, A. Kleywegt, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Proceedings of the International Conference on Robotics: Science and Systems*, pages 343–350, 2005.

[OR94] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[San99] Tuomas W. Sandholm. Distributed rational decision making. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed AI*, pages 201–258. The MIT Press, Cambridge, MA, USA, 1999.

[Sit16] Franz Sitzmann. Simulation und Vergleich der Effektivität verschiedener Job-Scheduling-Verfahren für autonome Transportroboter. Bachelor's Thesis, Humboldt Universität zu Berlin, Institut für Informatik, 2016.

[SLB09] Y. Shoham and K. Leyton-Brown. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[Smi80] R. Smith. The contract net protocol: high level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29:1104–1113, 1980.

[UPP16] Dep. of Inf. Technology at Uppsala University and Dep. of Computer science at Aalborg University: UPPAAL Integrated tool environment for modeling, simulation and verification of real-time systems. last accessed Feb. 2016. `http://www.uppaal.org/`.

[Wal16] Florian Walter. Modellierung und Verifikation von Sicherheit und Lebendigkeit eines Systems mobiler Transportroboter mit Modellprüfung zeitbehafteter Automaten. Bachelor's Thesis, Humboldt Universität zu Berlin, Institut für Informatik, 2016.

[Wei99] G. Weiss, editor. *Multiagent Systems. A Modern Approach to Distributed AI*. MIT Press: Cambridge, Mass, 1999.

[Woo02] M. Wooldridge. *An Introduction to Multi Agent Systems*. John Wiley & Sons, 2002.

[ZYAC12] Nicolas Jouandeau Zhi Yan and Arab Ali-Ch´erif. Multi-robot heuristic goods transportation. *6^{th} International IEEE Conference 'Intelligent Systems'*, 2012.