Workshop on Towards Fully Decentralized Multi-Robot Systems: Hardware, Software and Integration
2013 IEEE International Conference on Robotics and Automation, ICRA 2013
Karlsruhe, Germany, May 6 - 10, 2013, **http://icra2013mrs.tuebingen.mpg.de/**

Communication Integrated Control Architecture in Multirobot Systems

Hakan Karaoğuz, Halûk Bayram and H. Işıl Bozma

***Abstract*— Communication plays important role in exploiting the advantages of a multi-robot system. This paper extends the sense-act loop of control architectures to sense-communicate-act paradigm. In this new paradigm, the control architecture has to incorporate two new modules: communication and network topology update. The communication module enables the exchange of information with neighboring robots as deemed by the current communication network. As the robots and environment are both dynamic, the communication network topology has to be evolving dynamically. In network topology update module, the topology is updated periodically considering the communication or task related objectives. There are two alternative approaches – centralized or decentralized network update. The two modules are integrated into ROS. With this extended architecture, multi-robot scenarios with network based communication can easily be developed.**

I. Introduction

Many multirobot tasks require the robots to augment their individual sensing via communicating and exchanging information with other robots [1]. Interestingly, most control architectures consider some variation of sense-act paradigm[1] [2]. When the robots are endowed with communication abilities, the control architecture has to be modified to incorporate sense-communicate-act framework. The focus of this paper is on this topic – namely the design of multirobot control architectures with communication capabilities.

A. Related Literature

The development of robust real-time software for multi-robot systems is a challenging task that involves integrating a large number of components [3]. One approach has been to develop a software architecture so that expended effort can be recouped through their reuse with different robots in future projects [4], [5]. The various architectures that have been proposed can be grouped into three categories: deliberative or hierarchical, reactive or behavioral architectures and hybrid [6]. Deliberative architectures are based on sense-think-act paradigm [2], [7]. As planning may be quite problematic in dynamic settings, reactive approaches follow only the sense-act paradigm with no or very limited planning [8], [9]. As this may lead to problematic behaviour in unknown environments, hybrid architectures that follow again sense-think-act sequence with some reactivity embedded [10]. Despite their differences, all of these approaches rely on the main idea that an intermediate layer is required to fill the gap between the functional and symbolic worlds which may lead to inconsistencies [11]. Various architectures address this issue by enabling the coordination of planning and execution with robot behaviors and control [12]. The partitioning of the decisional layer into separate resources enhances the flexibility in handling different tasks [11].

In all these architectures, communication module is viewed as handling the interactions among modules at different layers [13], [6]. If the robots are to be endowed with communication capabilities, then these architectures have to be extended to incorporate inter-robot communication [14]. Thus, it will be possible for the robots to periodically exchange information. Due to limitations associated with broadcasting, point-to-point communication is preferred [15]. However, the communication complexity in all-to-all schemes increases with the square of the number of robots [16]. Thus, with the growth of robot team size, all-to-all communication faces serious scalability challenges [17], [1]. One remedy is to consider communication related objectives and evolve the network topology either via decentralized [18], [19] and centralized [20] approaches.

B. General Approach

In this paper, we present a novel control architecture that extends the common sense-act paradigm to sense-communicate-act framework. This new architecture incorporates two new modules: communication and network update. The communication module enables exchange of information with neighboring robots as deemed by the current communication network. As the robots and environment are both dynamic, the communication network topology has to be evolving dynamically. In the network update module, the topology is updated periodically based on task related objectives. Two alternative approaches to network update are possible: Centralized via a network coordinator or decentralized. The proposed architecture is developed in the framework of Robot Operating System (ROS) - an open-source robot meta-operating system. With this extended architecture, multi-robot application scenarios with network based communication can easily be developed. The outline of the paper is as follows: The robot architecture for a multi-robot system is presented in Section II. The communication module is explained in Section III. In Section IV, the network update module is described. An implementation of this architecture on a three robot team is presented in Section V. The paper concludes with a brief summary including ongoing work.

H. Karaoğuz, H. Bayram and H.I. Bozma are with the Intelligent Systems Laboratory, Department of Electrical and Electronics Engineering, Bogazici University, Bebek 34342 Istanbul Turkey, e-mail: {hakan.karaoguz, hbayram, bozma}@boun.edu.tr

[1]Note that with some architectures, this is replaced by sense-plan-act paradigm.

II. Robot Architecture

A multirobot system consists of a set of $\mathcal{R} = \{1, \ldots, r\}$ robots. Each robot operates based on an internal model. In the sense-communicate-act paradigm, this model is extended to incorporate a communication state. The control architecture realizes all the components of this model.

A. Robot Model

We assume that each robot has a unique identifier i. Each robot $i \in \mathcal{R}$ has a multitude of different type of internal states including physical size, position, sensor readings and sensory features. The physical size is defined variables such as radius $\rho_i \in \mathbb{R}$. The position $b_i(t) \in \mathbb{R}^2$ refers to robot's position in its workspace. For the sake of simplicity, we assume that configuration state is subsumed by position. These internal states are common to all robot architectures.

The new addition is the communication state. The communication state specifies the desired communication links. It is defined as

$$a_i(t) = \left[a_{i1}(t) \ldots a_{i(i-1)}(t)\, a_{i(i+1)}(t) \ldots a_{ir}(t)\right]^T$$

where $a_{ij}(t) \in \mathcal{B}$ with $\mathcal{B} = \{0, 1\}$. The value $a_{ij} = 1$ if and only if robot i wants a direct link with robot $j \neq i$. Otherwise $a_{ij} = 0$. Note that the states a_{ij} are not necessarily symmetric. Mutual consent is needed to establish a direct link – that is a link ij is created if and only if $a_{ij} = a_{ji} = 1$. We will omit the time argument whenever time dependency is clear from the context.

The information set $\chi_i(g) \subset \mathcal{R}$ of each robot is a set-valued function that specifies the set of robots with which the robot is getting information from [21]. In this work, we assume that each robot i share position information via inter-robot communication. By definition,$\{i\} \subset \chi_i(g)$. If $\chi_i(g) = \{i\}$, it only knows about itself and is completely oblivious to other robots. If the information set consists of itself and immediate neighbors as $\chi_i(g) = \{i\} \cup \mathcal{N}_i(g)$, then the robot gets configuration state information from neighboring robots. In this case, as network changes, so does each information set $\chi_i(g)$. If the cardinality of this set is $|\chi_i(g)| = r$, then this implies that the robot is getting information from all the robots. If $|\chi_i(g)| << r$, then the robot is getting information only from a subset of robots.

B. Control Architecture

The control architecture is comprised of three layers [6]. The physical layer consists of actuator and sensors modules. The actuator module consists of motors and their drivers. The sensors read the internal state of the robot as well as providing sensory feedback. This layer is hardware specific and needs to be modified as robotic platform or components are changed.

The next layer is the hardware abstraction layer. This layer serves as an intermediate layer between the physical layer and robotic modules. They are independent of the robotic platform or the hardware components. We use ROS which provides hardware abstraction, device drivers, message-passing and package management. In particular, the following properties are extremely convenient for multirobot applications [22]:

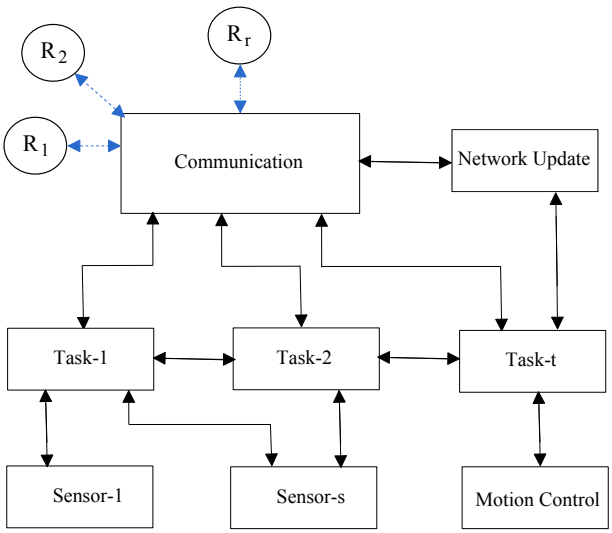

Fig. 1. Functional layer. Rectangles represent the nodes in the ROS. Circles show the other robots.

- Modularity: Separate processes (navigation, network update, mapping) can all be separated out. They interact through an interface in which software processes (a.k.a. "nodes" in ROS) communicate about shared "topics" in ROS. Publish/Subscribe allows each node to send and receive only the desired data (messages).
- Separation of physical and messaging interface helps avoid hardware dependencies.
- Inter-module communication is enabled via asynchronous callback functions that are called whenever data is available for processing.

The final layer is the functional layer. It is comprised of a collection of robotic modules.

- Motion control: This module updates the velocity of the wheels and providing the odometry information.
- Sensing: These modules are responsible for providing sensory information to the other modules from the sensors. A separate module is associated with each different sensor.
- Communication: These modules are associated with inter-robot communication.
  - Communication: This module is responsible for communication with other robots via sending or receiving messages.
  - Network update: This module updates the communication state a_i.
- Task modules: These modules associated with the different capabilities and functionalities of robots. Some of these modules are essential – meaning that all the robots will have these modules. Other modules are optional and the robot may not have them at all.
  - Navigation: This module generates the velocity profile for moving the robot in its workspace and

Control Byte	Message Type ID	Message Data

Fig. 2. Message structure.

thus affects b_i. It sends the current b_i to the communication module every T_c seconds.

– Localization: This module localizes the robot and generates b_i for each robot individually.

Each module is active concurrently with the other modules and may interact possibly with many of them. These modules are implemented as "nodes" in the ROS as shown in Fig. 1. Each node behaves like a separate process, but can interact with the other nodes via the publish/subscribe mechanism provided by the ROS. As the robots are engaged in a particular task, some of the task modules will be activated depending on what's required of the robot.

III. Communication Module

The communication module of robot i is responsible for the incoming and outgoing messages. We use TCP based communication protocol. This module performs the following:

- Open a socket with each robot $j \neq i$ in the team.
- Receive the incoming messages from the sender robots.
- After decoding the message, publish the incoming message to the related node(s).
- After receiving the outward bounded messages from the nodes, encode the outgoing message and send it to the receiver robot.

Note that all the sockets are opened at the beginning of operation in order not to make an extra effort on opening and closing sockets as the network topology changes. However, only some of these sockets will be in use depending on the network topology. The remaining sockets will be idle. Let it be noted that having opened, but idle sockets requires only negligible computation and communication overhead. This is because both end-points generate and respond to periodic TCP-generated keep-alive messages that occur at least once every 2 hours [23].

The message format is based on the modified version of CVS (comma-separated values). Each message contains three parts as shown in Fig. 2.

- Control Byte: This byte is for error checking purposes and is set to "AA".
- Message Type ID: This byte specifies the type of information in the message.
- Data: This part is of variable length as depending on the message type.

Note that communication is done in a robot-to-robot manner instead of broadcast. Hence the message data does not have any robot identity information regarding the sender robot.

IV. Network Update Module

The network topology is defined by the collective communication state of the robots $a \in \mathcal{A}$. Here $a = \sum_{i \in \mathcal{R}} a_i \otimes e_i$ and $\mathcal{A} = \underbrace{\mathcal{B}^{r-1} \times \ldots \times \mathcal{B}^{r-1}}_{r \text{ times}}$ denotes the communication space. Each communication state $a \in \mathcal{A}$ induces an undirected network $g(a) \in \mathcal{G}$. In the sequel, we will omit the a argument whenever it is clear from the context. The set $\mathcal{G} = \{g' | g' \subseteq g^r\}$ is the set of all possible graphs on $\mathcal{R}$ and g^r is the complete graph. Each graph $g = (\mathcal{R}, \mathcal{E})$ is such that the set of edges $\mathcal{E}(g) \subseteq Q$ represents the robot pairs ij between which there is a direct communication link. From each robot i's perspective, the corresponding set of links is defined as $\mathcal{E}_i(g) = \{ij \mid j \in N_i(g)\}$. The case when $g = g^r$ corresponds to all-to-all communication - namely every robot communicates directly with every other robot. For each robot i, its neighboring robots $\mathcal{N}_i(g)$ are robots with which direct links exist – namely $\mathcal{N}_i(g) \triangleq \{j \in \mathcal{R} \,|ij \in \mathcal{E}_i(g)\}$. Hence, network update is based on modifying the collective communication state a. This can be achieved via two alternative approaches: centralized and decentralized.

A. Centralized Network Update Strategy

In the centralized approach, a network coordinator is responsible for determining the network topology. Hence, the functionality of network update module depends on whether the particular robot is a regular robot or a network coordinator. In case of a regular robot, the network module simply sends position information b_i to the coordinator and receives the updated communication state a_i as:

- Send position state b_i to the network coordinator periodically every T_g seconds.
- Update the collective communication state a_i based on received information from the network coordinator.

In turn, the coordinator does the following:

- Receive position b_i from all the robots.
- Find a network topology acceptable to all the robots based on received information and update a. It ensures that sent information is consistent - namely $a_{ij} = a_{ji}$.
- Send a_i to each robot.

The topology update is based on robots' communication objectives $v_i : \mathcal{G} \times \mathcal{F} \to \mathbb{R}$ that encode their communication strategies while they are engaged in their tasks. The set $\mathcal{F}$ refers to the set of all robots positions. A sample case is as shown in Fig. 3(b). All the robots send their location states to the coordinator. In turn, the network coordinator determines a new topology and transmits this to all the robots as also seen in Fig. 3(c). For details, the interested reader is referred to [20].

B. Decentralized Network Update Strategy

In this approach, each robot decides for its neighbors $\mathcal{N}_i(g)$ independently of other robots by updating its communication state a_i every T_g seconds. The network update proceeds as follows:

- This module is activated every T_g seconds.
- It attempts a series of temporary direct communications with other robots and updates a_i directly.

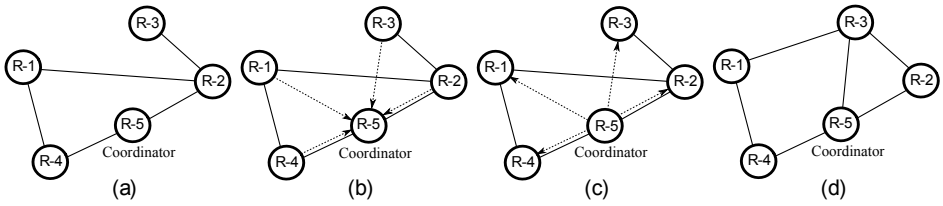

Fig. 3. Robot-5 is acting as a network coordinator. Solid links represent the network topology. One-way dashed links show the temporary communications. (a) Initial network topology; (b) At the onset of the network update process, all the robots send their information to the coordinator; (c) At the end of the update process, the coordinator sends the network information to the robots; (d) Final network topology.

- The update process is continued until a predefined time ΔT_g is reached.

As robots are resource constrained, each robot is viewed as maximizing its own benefit from participation in the communication network. Thus, the updating of the network topology is based on an payoff function $v_i : \mathcal{G} \times \mathcal{F} \rightarrow \mathbb{R}$. For details, the interested reader is referred to [18], [19].

C. Centralized vs Decentralized Network Update

The two alternative network update approaches have both pros and cons. In the centralized approach, it will be easier to achieve optimal network topologies since the network coordinator has a global view of the overall system. Of course, this comes with an extra overload on the coordinator. Furthermore, it is dependent on the successful operation of the network coordinator robot. However, if the network update process does not require any specific ability, one of the remaining robots can take on the role of network coordinator very easily. In contrast, decentralized approaches are easy to implement as they do not require a coordinator. However, the decision process on a final topology may take a long time to converge depending on the objectives. Furthermore, it is not easy to check whether the network topology is acceptable to all the robots or not. In practice, since many multirobot systems do not conform to a strict centralized/decentralized dichotomy, many largely decentralized architectures utilize leader/coordinator robots for certain parts of their tasks.

V. TURTLEBOT TEAM

We use a three-robot team. All the robots are Turtlebot having a differential driving mechanism and on-board processing is done with an ASUS netbook. The robots all have cylindrical bodies with radius 17.5 cm and the maximum speed is limited to 0.1 meters/second. Each robot has onboard encoders and Hokuyo laser range scanner with 4m field of depth - both of which are used in localization and generating individual position information. Communication is required so that the robots are informed of each others' position. The robots communicate over an ad hoc network. In this implementation, network update is done in the centralized manner. Hence, one of the robots acts as the network coordinator and updates the network topology as explained in Section IV-A.

A. Functional Modules

All the robots are endowed with six modules as shown in Fig 4(left) for the standard robot and Fig 4(right) for the coordinator robot.

- Motion control: This module, Turtlebot node, is provided by ROS. It sends the desired velocity to the Turtlebot's control unit and provides the odometry of the robot.
- Laser sensor: This module manages Hokuyo laser range scanner.
- Localization: Adaptive Monte Carlo localization (AMCL) module that is available in ROS is used.
- Navigation: Each robot navigates to its a priori specified goal position without any collisions along the way. A modified version of multirobot navigation functions [24] is implemented in a decentralized manner. Control law depends on the information set $\chi_i(g)$.
- Communication: This module manages the incoming and outgoing messages.
- Network Update: In all the robots except the network coordinator robot, this module requests the updated network. In the coordinator robot, this module considers all the robots' communication related objectives and updates the network accordingly. The details of this are presented in [20]. It then sends the relevant information to all the robots.

B. Inter-Robot Messages

The list of possible messages depends on the requirement of the application. In our case, we define three types of message in the implementation.

1) Message Type ID = 1. In this case, the data contains robot's position state b_i in the global coordinate system. A sample message for this type is `''AA'', ''1'', ''0;0''` which says that my current position is at (0,0).
2) Message Type ID = 2. In this case, the data contains detailed robot information including the robot's current position, radius, and goal position. A sample message for this type is `''AA'', ''2'', ''0;0;17.5;500;500''`, which indicates that I am a robot with 17.5 cm radius at (0,0) location and my desired position is at (500,500).
3) Message Type ID = 3. This message is sent from the coordinator to a robot. The message data contains a

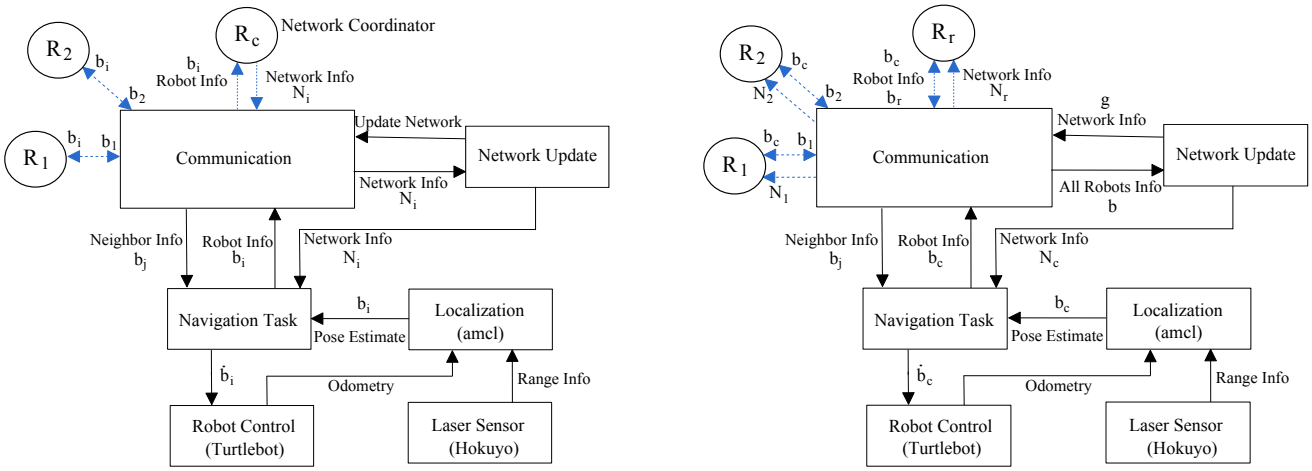

Fig. 4. Robotic modules: Left: Standard robot R_i; Right: Network coordinator robot R_c.

Fig. 5. The snapshots of the experiment with Robot 3 (the robot with red cone) acting as a network coordinator.

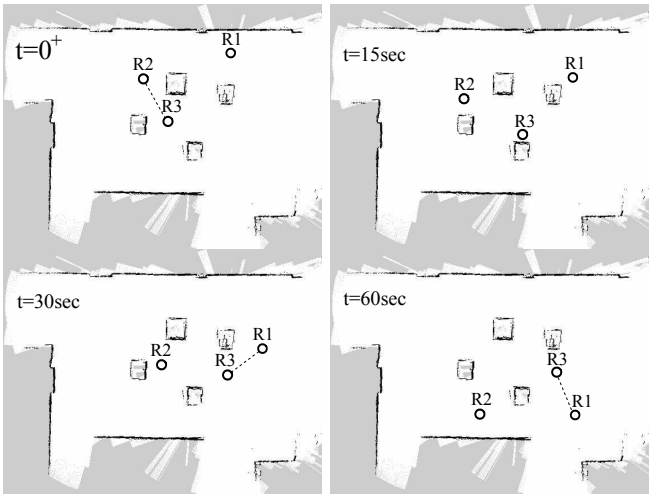

Fig. 6. Time evolution of the experiment. Robot 3 is acting as a network coordinator. Dotted lines represent the pairwise communication links.

list of $N_i(g)$ of each robot. A sample message for this type is `''AA'', ''3'', ''1; 2''`, which indicates that Robot1 and Robot2 are your neighbors. If there is no neighbors, the message would become `''AA'', ''3'', ''0''`.

C. Experiments

The experiments are done using robot teams consisting of 3 robots navigating in a workspace of about 6m $\times$ 6m as shown in Fig. 5. Each robot should navigate to its a priori specified goal position simultaneously with the other robots without any collisions along the way. The map of workspace is built using gmapping package in ROS with one of the robots prior to the experiment. This map is then shared by all the robots. In the experiments, the values of robot-to-robot communication period $T_c = 3$ seconds which implies that each robot sends its position information b_i to its neighbors every 3 seconds. Robot 3 is assigned to be the network coordinator. The network update period $T_g = 15$ seconds which implies that each robot sends its position information b_i to the network coordinator 15 seconds. The link cost in the robot's communication payoff function v_i is set to 2.

Sampled snapshots from one exemplary task is as shown in Fig. 5. The corresponding evolution of the network topology is shown in Fig. 6. At the beginning of the task, the network g is empty - meaning that none of the robots are communicating. As robot 3 (network coordinator) receives the position information from the other robots in the team, it updates the network topology so that robot 2 and robot 3 start communicating. In the next network update, this link is broken so that robots 2 and 3 stop communicating. In the following network update, the coordinator updates the network topology where robot 1 and robot 3 start communicating. This process continues as long as the robots are operational. Let it be noted that the network topology depends on the objectives selected and programmed in the network update module. If alternative objective functions are used, the network topology will change accordingly.

VI. Conclusion

This paper extends the sense-act loop of control architectures to sense-communicate-act paradigm. In this new frame-

work, the control architecture incorporates two new modules: communication and communication network update. The communication module enables the periodic exchange of information with neighboring robots as deemed by the current network topology. The network update module is responsible for updating the network topology periodically via considering the communication or task related objectives. In the centralized approach, they leave updating of the network topology to one of the robots - who additionally acts as the network coordinator. This robot updates the network – taking all the robots communication related objectives into consideration. Alternatively, in decentralized network update, the robots decide for themselves with whom to communicate. These modules are integrated into ROS and are implemented on a multirobot team. Our ongoing work is focusing on using this framework in multirobot tasks where achieving optimal network topologies is critical to performance.

ACKNOWLEDGMENTS

This work has been supported in part by TUBITAK Project 111E285 and by the Turkish State Planning Organization (DPT) under the TAM Project, number 2007K120610.

REFERENCES

[1] G. Dudek, M. Jenkin, and E. Milios, "A taxonomy of multirobot systems ," in *Robot teams: From diversity to polymorphism*, T. Balch and L. Parker, Eds. A.K. Peters, 2002, pp. 2 – 26.

[2] M. Siegel, "The sense-think-act paradigm revisited," in *1st Int. Workshop on Robotic Sensing,*, 2003.

[3] U. Saranli, A. Avci, and M. Ozturk, "A modular real-time fieldbus architecture for mobile robotic platforms," *IEEE Trans. on Instrumentation and Measurement,*, vol. 60, no. 3, pp. 916–927, 2011.

[4] J. Roberts, P. Corke, R. Kirkham, F. Pennerath, and G. Winstanley, "A real-time software architecture for robotics and automation," in *IEEE Int. Conf. on Robotics and Automation,*, vol. 2, 1999, pp. 1158–1163.

[5] E. Coste-Maniere and R. Simmons, "Architecture, the backbone of robotic systems," in *IEEE Int. Conf. on Robotics and Automation,*, vol. 1, 2000, pp. 67–72.

[6] S. Ali and B. Mertsching, "Towards a generic control architecture of rescue robot systems," in *IEEE Int. Workshop on Safety, Security and Rescue Robotics*, 2008, pp. 89–94.

[7] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen, "A deliberative architecture for auv control," in *IEEE Int. Conf. on Robotics and Automation,*, 2008, pp. 1049–1054.

[8] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[9] R. Stenzel, "A behavior-based control architecture," in *IEEE Int. Conf. on Systems, Man, and Cybernetics,*, vol. 5, 2000, pp. 3235–3240.

[10] R. C. Arkin and T. Balch, "Aura: Principles and practice in review," *J. of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 175–189, 1997.

[11] A. Degroote and S. Lacroix, "Roar: Resource oriented agent architecture for the autonomy of robots," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 6090–6095.

[12] T. Estlin, R. Volpe, I. Nesnas, D. Mutz, F. Fisher, B. Engelhardt, and S. Chien, "Decision-making in a robotic architecture for autonomy," in *Int. Symp. on Artificial Intelligence, Robotics, and Automation in Space,*, 2001.

[13] E. Park, L. Kobayashi, and S. Lee, "Extensible hardware architecture for mobile robots," in *IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 3084–3089.

[14] L. Xiaopen, M. Hongwei, and Z. Zhihui, "Research on open control architecture of autonomous mobile robot with multi-layer and modularization," in *2nd Int. Asia Conf. on Informatics in Control, Automation and Robotics*, vol. 1, 2010, pp. 511–515.

[15] Z. Wang, M. Zhou, and N. Ansari, "Ad-hoc robot wireless communication," in *IEEE Int. Conf. on Systems, Man and Cybernetics*, vol. 4, 2003, pp. 4045–4050.

[16] E. Klavins, "Communication complexity of multi-robot systems," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 275–292.

[17] R. Kravets, K. Calvert, and K. Schwan, "Payoff-based communication adaptation based on network service availability," in *Proc. IEEE Int. Conf. on Multimedia Computing and Systems*, 1998, pp. 33–42.

[18] H. Bayram and H. I. Bozma, "Multi-robot navigation with limited communication - deterministic vs game-theoretic networks," in *IEEE/JRS Int. Conf. on Robots and Systems*, 2010, pp. 1825–1830.

[19] ——, "Pairwise vs coalition game networks for multi-robot systems," in *Proc. of the 18 IFAC World Congress*, Milan, Italy, 2011, pp. 13 570–13 575.

[20] ——, "Multirobot communication network topology via centralized pairwise games," in *(to appear) IEEE Int. Conf. on Robotics and Automation*, 2013.

[21] Y. Ho and K. Chu, "Team decision theory and information structures in optimal control problems-Part I," *IEEE T. Automat. Contr.*, vol. 17, no. 1, pp. 15–22, 1972.

[22] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[23] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," *Int. Journal of Network Management*, vol. 15, no. 5, pp. 297–310, 2005.

[24] C. S. Karagöz, H. Bozma, and D. E. Koditschek, "Coordinated navigation of multiple independent disk-shaped robots," The Univ. of Michigan, Comp. Sci. Eng. Div., Dept. of Elec.Eng. & Comp. Sci., Tech. Rep., 2004.