2003

Scalability and Schedulability in Large, Coordinated, Distributed Robot Systems

John D. Sweeney
University of Massachusetts Amherst

Scalability and Schedulability in Large, Coordinated, Distributed Robot Systems

John D. Sweeney, Huan Li, Roderic A. Grupen, and Krithi Ramamritham
Laboratory for Perceptual Robotics
Department of Computer Science
University of Massachusetts, Amherst
{sweeney, lihuan, grupen, krithi}@cs.umass.edu

Abstract

Multiple, independent robot platforms promise significant advantage with respect to robustness and flexibility. However, coordination between otherwise independent robots requires the exchange of information; either implicitly (as in gestural communication), or explicitly (as in message passing in a communication network.) In either case, control processes resident on all coordinated peers must participate in the collective behavior. This paper evaluates the potential to scale such a coupled control framework to many participating individuals, where scalability is evaluated in terms of the schedulability of coupled, distributed control processes.

We examine how schedulability affects the scalability of a robot system, and discuss an algorithm used for off-line schedulability analysis of a distributed task model. We present a distributed coordinated search task and analyze the schedulability of the designed task structure. We are able to analyze communication delays in the system that put upper bounds on the size of the robot teams. We show that hierarchical methods can be used to overcome the scalability problem. We propose that schedulability analysis should be an integrated part of a multi-robot team design process.

Keywords: *multi-robot systems, schedulability analysis, coordinated control*

1 Introduction

As robotic technology becomes more mature, implementing distributed robot systems with a large number of robots will be possible. The expense and mean-time to failure of component hardware limit the size of fieldable teams to tens of robots [3]. However, in the future we can expect that developing technology will allow, and applications will require, teams with an order of magnitude more robots. While swarms of robots are an attractive idea, they are also frequently assumed to be composed of independent platforms and control processes. To do useful work, we may have to coordinate the activity of several robots, which introduces processing and communication constraints among the team. In this paper, we propose a coordination model and evaluate the bounds on coordinated robots teams that arise due to those constraints.

We present a distributed, coupled control framework applied to a leader/follower search task. The distributed controller can address multiple, concurrent objectives while maintaining global behavior constraints. Robust, closed-loop controllers are defined as primitives within the control architecture. Multiple controllers are combined via the nullspace projection operator: subordinate control actions are projected onto the nullspace of superior controllers, so that incorrect interactions between controllers are avoided. This allows "best-effort" guarantees to be made about global behavior. In the task of leader/follower search, the leader must concurrently search while maintaining connectivity by remaining within line-of-sight (LOS) of the follower.

In any robot system, interaction with the world imposes a real-time constraint on computation, whose logical correctness depends on the correctness of its outputs as well as their timeliness. The robot must be able to process sensor input, respond to dynamic environments, and send messages to other robots. Real-time specifications for such systems are derived from time constraints in the control processes and in the environment. If the system is unable to perform distributed control tasks in a timely manner, then overall performance suffers.

In this paper, we look at the issue of scalability from the perspective of real-time schedulability. A distributed multi-robot system is viewed as a collection of homogeneous processors. Each robot has a set of tasks that run periodically, with data flow between tasks on a single robot and between tasks on different robots. In the context of a real-time multi-robot system, schedulability analysis determines whether all tasks in the system can be scheduled to some period and deadline. We propose that schedulability analysis should be an integral part of

This work was supported in part by NSF CDA-9703217, DARPA/IPTO DABT63-99-1-0022 and DABT63-99-1-0004.

the multi-robot system design process.

The paper is organized as follows. First we briefly present related work, then we give an overview of the distributed controller for concurrent, multi-objective tasks presented in [13]. This distributed controller is used to perform a leader/follower search task. Then we examine the controller using the algorithm developed in [8] for off-line schedulability analysis. We finish with conclusions and future work.

2 Related Work

The control framework described in this paper is based on a bottom-up approach to control, similar to approaches such as the subsumption architecture [2], where robust, low-level control primitives are combined to produce high-level behaviors. Individual controllers are constructed using the control basis approach [5, 13].

There is a lot of work in the literature on cooperative multi-robot teams, such as [3], which presents an overview of cooperative robotic techniques. However, the issue of the scalability of the coordination scheme is not fully addressed. Carpin et al. [4] have presented an approach for the leader/follower application. Their system is designed to allow teams of any size, however, they do not address the effects of using broadcast communications on the effective team size. Yoshida et al. [14] have examined how information propagation within a team and team performance were affected by using a shared communication channel and team size, but they did not consider real-time computing constraints.

Real-time operating systems for robotics that allow communication among distributed resources have been developed [1, 12]. There are also tools for designing controllers for real-time robotic systems such as [10]. Schedulability analysis for distributed real-time systems has also received a lot of attention in recent years [9]. For tasks with temporal constraints, researchers have focused on generating task attributes (e.g., period, deadline and phase) with the objective of minimizing the utilization and/or maximizing system schedulability while satisfying all temporal constraints. However, schedulability is clearly affected by both temporal characteristics and allocation of real-time tasks. A more comprehensive approach that takes into consideration task temporal characteristics and allocations, in conjunction with schedulability analysis, is required.

3 Distributed, Coordinated Leader/Follower Controllers

We first give a brief overview of the architecture for reactive, coordinated controllers that address multiple, concurrent objectives in a mobile robot team, described in [13]. An example application of such a system is a multi-robot search task. Each robot is equipped with sensors specific to the search goal, in this case IR proximity sensors, and wireless communication. A team of robots R must search an unknown environment, while maintaining wireless connectivity throughout the team. The limited range of the wireless transmitters imposes a path constraint on the members of the team in that any pair of robots must ensure they are within line-of-sight (LOS) and within range specifications for the desired QoS or bandwidth in order to guarantee connectivity.

The control basis approach constructs a controller $\phi_{\mathcal{E}}^{\mathcal{S}}$ by associating a state estimator, $\mathcal{S}$, and effectors, $\mathcal{E}$, with an objective function, or artificial potential, ϕ. In this paper, our artificial potentials are harmonic functions represented by discrete occupancy maps [6]. For example, the controller that enforces the LOS constraint is $\phi_j^{LOS_i}, i, j \in R$, where robot i generates the LOS region (computed from its position and an obstacle map), and robot j tries to stay within the LOS region by descending the potential ϕ. The search controller is ϕ_i^S, where robot i achieves search goal states S by greedy action on ϕ. The leader's search task implicitly avoids obstacles since it computes trajectories that move the leader away from obstacles and toward unexplored space.

3.1 The "Pull" Controller

A pairwise, concurrent, coordinated controller (denoted a "pull" controller) is constructed that allows the leader to search as long as the follower is within the LOS region:

$$\phi_i^S \quad \triangleleft \quad \phi_j^{LOS_i}, \tag{1}$$

where i is the leader and j is the follower. The "subject-to" operator ($\triangleleft$) allows concurrency by projecting the trajectory from ϕ_i^S onto the nullspace of $\phi_j^{LOS_i}$ (using the Moore-Penrose pseudoinverse, for example); ensuring that the leader's search task does not interact destructively with the LOS task. Here, the nullspace of $\phi_j^{LOS_i}$ refers to the nullspace of the Jacobian that maps changes in wheel displacements of robot j onto changes in the value of the artificial potential defined by ϕ^{LOS_i}. In general, planar mobile robots are not redundant with respect to their configuration space. However, a planar mobile robot may be redundant with respect to some objective function.

In addition to two robots, multiple robots can form a serial, kinematic chain by combining pull controllers. The robot at the head of the chain executes the controller in equation (1), while a robot k within the chain is involved in a pairwise pull controller with its neighbors:

$$\phi_k^{LOS_k+1} \quad \triangleleft \quad \phi_{k-1}^{LOS_k}. \tag{2}$$

The robot at the base of the chain is assumed to be a stationary communications hub for the team. This pull chain allows the leader to explore a great distance from the hub.

The leader/follower pull controller described above is implemented with two of our UMASS UBot mobile robots, each one using a 206 MHz StrongARM CPU with the K-Team Kameleon motor driver board. The controllers are implemented using the Player/Stage robot control system [7]. In the next section we describe our method of real-time schedulability analysis in a distributed control system, and analyze the schedulability of the pull controller.

4 Our Approach for Real-time Schedulability Analysis

Equation (1) describes a coordinated controller that involves several processes: sensor processing to determine S and LOS_i, motor Jacobians that generate wheel velocities on platforms i and j, and processes that descend potential functions. From the real-time systems view, the scalability of such a scheme involves the ability to find feasible schedules as the task and processor sets increase, i.e., as the number of robots increases. The real-time constraint imposes a hard deadline on the amount of processing that can be completed in a given period of time. If robots are independent and do not collaborate in a coordinated control scheme, then scalability is not an interesting problem, since it becomes one of scheduling on an individual robot, which has been widely studied [11]. However, if the team members are cooperative, then, in addition to task constraints such as periods or deadlines, system level constraints are also introduced.

For instance, in order to achieve a common goal, robots may exchange messages. Therefore, communication costs and precedence constraints must be considered. The difficulty is that scheduling tasks with precedence constraints and individual deadlines for a multiprocessor system is an NP-complete problem even for unit processing time of each task. We propose a heuristic algorithm that takes into account all types of constraints to predict the scalability and schedulability for a large, recursive robot system [8]. In this context, recursive means that the task model of the system has a symmetric structure that can be easily generalized to accommodate additional robots. The pull controller is an example of a recursive, distributed system.

In the following sections, we will briefly review the system model and our approach to do real-time schedulability analysis for distributed coordinated robotics, which is discussed in [8].

4.1 System Model

The distributed robot system consists of m identical uniprocessor sites. In this paper, we use *site* and *robot* interchangeably. The sites are connected by a shared communication medium from one site to another. Communications must be scheduled at specific times to assure that no contention for the channel occurs at run time.

Tasks we study here are real-time tasks that have the following characteristics:

- **Period**. This defines the inter-release times of instances of the task. One instance of the task should be executed every period.
- **Relative deadline**. This specifies the time at which each task instance must be completed.
- **Computation time**. This is the worst case execution time of any instance of the task.
- **Precedence relationships**. These constrain the execution order of the tasks and the production and consumption relationships of the data flow.
- **Locality constraints**. These relationships are based on the nature of the environment required for tasks to execute. For example, actuator control tasks must run on the processor that connects to the actuator. In this distributed task model, tasks without locality constraints can be assigned to any available processor.
- **Communication constraints**. Communication between tasks that are on different sites requires a communication medium and time to send or receive the message. Messages sent between tasks must be scheduled with the communication medium as a resource requirement. If the communication is modeled as a special task, then this task must satisfy the precedence constraints with the two communicating tasks separately.

4.2 Algorithm Overview

In order to help understand the method, we now present a brief overview of the schedulability algorithm described in [8]. The first step of the algorithm assigns unallocated tasks to sites. A heuristic, which takes into account the trade-off between communication cost and processor workload, is used to assign tasks to sites. The basic idea of the heuristic is to cluster tasks with a high communication cost together on the same site while minimizing the utilization of each processor.

Next, we construct an extended task graph that includes communications represented as tasks. The algorithm uses communication tasks to model the communication cost and channel contention that occurs if the tasks are allocated to different sites. Then we build a comprehensive graph containing all instances of all tasks including communication tasks that will execute within the least common multiple (LCM) of all task periods, and preprocess precedence relations of tasks by setting up the relative earliest start time of consumers. Finally, a search is used to find a *feasible* schedule, if possible, mapping starting times to all tasks including communication, to determine if they can start and complete execution before their deadlines.

$IR_{1,2}$	$POS_{1,2}$	$M_{1,2}$	ϕ_2^S	LOS_2	$\triangleleft$	$\phi_1^{LOS_2}$
20	20	20	25	1	10	25

Table 1: Worst-case execution time (in milliseconds) for the tasks in Figure 1.

4.3 Pull Controller Analysis

The task model for the pull controller with two robots is shown in Figure 1. Every robot must execute IR obstacle detection tasks, denoted by IR_i, odometry tasks, POS_i, and motor control tasks, M_i. The three tasks IR_i, POS_i, and M_i, which are drawn with solid ellipses, are specific to the hardware of each robot, so they are all preallocated to run locally on each robot. The control tasks ϕ_2^S and $\phi_1^{LOS_2}$, the nullspace projection $\triangleleft_2$, and the sensor processing task LOS_2 may reside on a single robot, or be distributed between the pair, if necessary, to optimize processor utilization or communication costs. They are denoted with dotted ellipses. The functionality of the team is not affected by altering the allocations of the control tasks. However, a good allocation strategy does improve schedulability.

The communication cost between tasks, if they are distributed, is given in milliseconds on the corresponding arc. The computation times of the tasks are given in Table 1. Computation times and communication costs were determined experimentally on the platform.

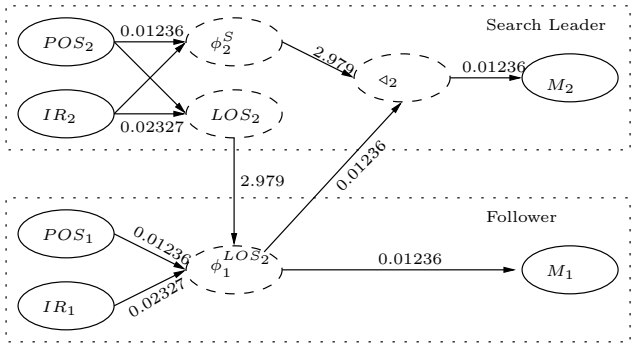

Figure 1: The pull controller task model for two robots. Preallocated tasks are in solid ellipses, and dotted ellipses are tasks that may be distributed across the pair. The communication cost between tasks (in milliseconds) is shown on the arcs.

The sensor and motor tasks IR_i, POS_i, and M_i are designed to be updated periodically, and the control tasks must execute periodically in order to consume the new sensor data and give new motor commands. Consequently, the periods of the control tasks should be based on the periods of the sensor and motor tasks. We determined experimentally that to achieve satisfactory performance, the sensor and motor tasks should run at least every 200 ms. The period of execution of all the tasks in the pull model determines the robot's responsiveness, and also affects the scalability of the system. After applying the scheduling algorithm using the "aggressive" allocation heuristic described in [8], we found that ϕ_2^S, LOS_2, and $\triangleleft_2$ are assigned to site 2, and $\phi_1^{LOS_2}$ is assigned to site 1. With this allocation of tasks, the only communications that require the wireless channel are $LOS_2 \rightarrow \phi_1^{LOS_2}$ and $\phi_1^{LOS_2} \rightarrow \triangleleft_2$.

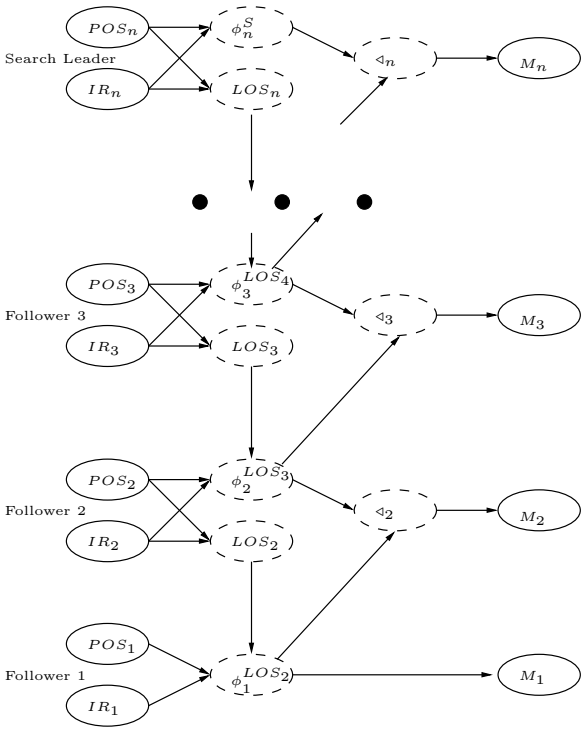

Figure 2: The task model for a chain of pull controllers for n robots.

The generalizability of the pull task model allows a robot to join the chain with only a minimal change in communication structure. Only the leader and its immediate follower execute the controller from equation (1), while the rest of the robots execute the controller in equation (2). A combined task model for a chain of n robots is shown in Figure 2. Since the controllers are designed to have the same real-time task characteristics and communication patterns within the chain, and our approach captures such recursive properties in advance, the result is a predictable allocation for the additional tasks. A larger team can be scheduled simply by generalizing the results from the smaller team. For example, if the n^{th} robot joins the head of a chain with $n-1$ members, tasks ϕ_n^S, LOS_n, and $\triangleleft_n$ are known to be allocated to robot n. Also, schedulability can be ensured by checking if the laxity time is larger than the sum of the additional communication cost and computation time introduced by the n member.

With the allocation of tasks shown in Figure 2, we see that the only communication tasks that require the wireless channel are $LOS_i \rightarrow \phi_{i-1}^{LOS_i}$ and $\phi_{i-1}^{LOS_i} \rightarrow \triangleleft_i$. Thus, during every 200 ms period, both of those communication

tasks must be scheduled on the wireless channel for every pair of robots in the team. This channel contention, along with the length of the period and precedence constraints, imposes a limit on the total number of robots that can be involved in a team.

How can the shared communication channel solution scale for larger teams? This question is answered by the analysis of the largest computation time within the team, which is the sum of execution times of any tasks, including communication tasks, along any path from an input task, or set of input tasks (tasks without incoming edges), to an output task (tasks without outgoing edges). We assume that communication occurring within a site can be ignored.

For a team with n robots as in our pull model, with the addition of a new robot, two communications need to be considered: $LOS_i \rightarrow \phi_{i-1}^{LOS_i}$ and $\phi_{i-1}^{LOS_i} \rightarrow \triangleleft_i$, $i \geq 2$. Because the robots are autonomous except for communication constraints between different members, the tasks scheduled locally need only to satisfy precedence constraints. For instance, POS_i and IR_i must be scheduled before $\phi_i^{LOS_{i+1}}$ (ϕ_i^S for the leader) and $\triangleleft_i$ can only be scheduled to start after the completion of $\phi_i^{LOS_{i+1}}$ (ϕ_i^S for the leader) and $\phi_{i-1}^{LOS_i} \rightarrow \triangleleft_i$.

Now let us consider the schedule of a leader/follower team. Initially, we start with a pair of robots. By using the earliest deadline plus the earliest start time first strategy [8], the leader has the longest execution time, since it must 1) transmit data from LOS_2 to $\phi_1^{LOS_2}$ using the communication channel, and 2) wait for data from $\phi_1^{LOS_2}$. Because of the parallel task execution on two robots, the total execution time of the leader is $T_L + C_{L\rightarrow F}$, where T_L is the sum of execution times of tasks allocated to the leader and $C_{L\rightarrow F}$ is the total communication cost. T_L and $C_{L\rightarrow F}$ are computed as:

$$\begin{aligned} T_L &= POS_2 + IR_2 + LOS_2 + \phi_2^S + \triangleleft_2 + M_2 \\ &= 20 + 20 + 1 + 25 + 10 + 20 = 96 \text{ ms} \\ C_{L\rightarrow F} &= (LOS_2 \rightarrow \phi_1^{LOS_2}) + (\phi_1^{LOS_2} \rightarrow \triangleleft_2) \\ &= 2.979 + 0.01236 \approx 2.99 \text{ ms.} \end{aligned}$$

The computation time of $\phi_1^{LOS_2}$ is not included since it is equal to that of ϕ_2^S; otherwise, if it is larger, it needs to be taken into account. Thus the total execution time of the leader is $T_L + C_{L\rightarrow F} = 96 + 2.99 = 98.99$ ms.

The laxity within one period is $200 - 98.99 = 101.01$ ms. If a third robot joins the group and keeps the same control pattern, then even though the other tasks are running concurrently on different processors, the communication channel must be shared. Hence, the communication delay for each new member i comes from the accumulation of $LOS_i \rightarrow \phi_{i-1}^{LOS_i}$ and $\phi_{i-1}^{LOS_i} \rightarrow \triangleleft_i$, a delay of $2.979 + 0.01236 \approx 2.99$ ms. For a chain of size n, the leader will always have the longest execution time of

$$96 + 2.99(n-1) \text{ ms.}$$

Based on a period of 200 ms, the bound on the size of the pull chain is

$$\begin{aligned} 200 &= 96 + 2.99(n-1) \\ n &= \lfloor \frac{200 - 96}{2.99} + 1 \rfloor = 35. \end{aligned}$$

For the maximal chain size, the total execution time required for the 35^{th} robot is $96 + 2.99 \cdot 34 = 197.66$ ms. Based on this analysis, we know that every 200 ms, each robot can successfully complete their work and achieve coordination. If more than 35 robots are involved, then there is no guarantee that messages will be be able to reach their destination before the task executes. This results in tasks executing with old data, and the system performance drops due to this time lag. A time-line showing the schedule for a team of four robots is shown in Figure 3.

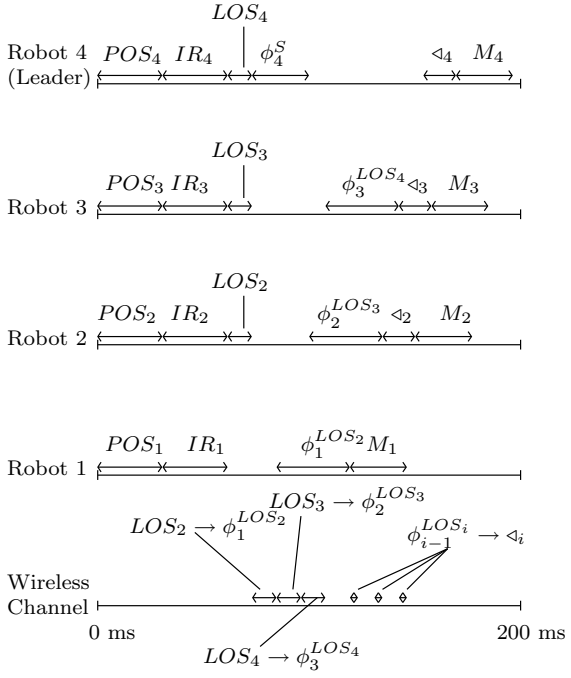

Figure 3: A feasible schedule for a four-robot pull chain. Note that the schedule is not drawn to scale.

If, during the design phase, we find that that we need to coordinate more robots than the upper bound, we can split the team into small groups geographically at runtime, where groups communicate with each other through one specialized element of the team to share information and/or decisions. Even though the communication resource is limited, we can predict in advance the available resources for a small group given the pre-analysis done by our algorithm. At run time a dedicated hierarchical communication model can be built just by looking up the grouping of the robots. Another solution is to redesign

the system to reduce resource contention. This could entail reducing the amount of communication needed between members, by making the messages smaller, or eliminating communication by restructuring the flow of messages.

The task model for the pull controller shown in Figure 1 is one possible way of designing the controller. We can design multiple task models for a given controller and evaluate them using the schedulability analysis algorithm to determine how well they scale. Although many task models achieve the same behavior, some are preferred because they allow more robots to coordinate or they may preserve more run-time flexibility when new or additional tasks are introduced.

5 Conclusions and Future Work

We have presented a distributed implementation of a coordinated controller for leader/follower behavior maintaining a line-of-sight constraint. We have analyzed the task model for the controller and derived an upper bound on the size of a feasibly scheduled, coordinated team. In general, a distributed, real-time, multi-robot system has an inherent scale that is a function of the hardware limitations of the robots and the higher-level design of the system. Resource limitations such as a shared communication channel and limited bandwidth create an upper bound on the number of robots that are able to coordinate within the team with a feasible schedule. Schedulability of the overall system should be taken into account so that the value of coordination outweighs the costs. When designing a robot system, a large group can be split into several smaller teams at run-time, whose size is based on the results of the schedulability analysis.

In the future, we plan on extending this work to teams using a mixture of different controllers. The pull controller has a symmetric pair controller (denoted a "push" controller) that allows the follower to specify the LOS region to the leader. In addition, push and pull controllers can be combined along a chain in various combinations to achieve a goal. We want to analyze the combinations of controllers that a robot could have, so that at run-time the robot may lookup the schedule from a precomputed table when it joins a push/pull chain.

The work in this paper examines only one small part of the structure that makes a scalable robot team. A complete scalability analysis of a robot system would examine many factors other than schedulability; reliability and ease of maintenance are hardware-related factors that are particularly important in robotic systems. All of these issues should be addressed during the design process of the robot team.

References

[1] J. Albus, R. Lumia, and A. Wavering. NASREM: The NASA/NBS standard reference model for telerobot control system architecture. In *Proceedings of the 20th Internation Symposium on Industrial Robots*, October 1989.

[2] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[3] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomuos Robots*, 4:1–23, 1997.

[4] S. Carpin and L. E. Parker. Cooperative leader following in a distributed multi-robot system. In *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2002.

[5] J. Coelho and R. Grupen. A control basis for learning multifingered grasps. *Journal of Robotic Systems*, 14(7):545–557, 1997.

[6] C. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotics Systems*, 10(7):931–946, 1993.

[7] B. P. Gerkey, R. T. Vaughan, K. Stoy, A. Howard, M. J. Mataric, and G. S. Sukhatme. Most valuable player: A robot device server for distributed control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1226–1231. IEEE/RSJ, 2001.

[8] H. Li, J. Sweeney, K. Ramamritham, R. Grupen, and P. Shenoy. Real-time support for mobile robotics. Technical Report 03–08, University of Massachusetts, Amherst, 2003.

[9] J. C. Palencia and M. G. Harbour. Exploiting preceding relations in the schedulability analysis of distributed real-time systems. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, December 1999.

[10] S. Schneider, V. Chen, J. Steele, and G. Pardo-Castellote. The ControlShell component-based real-time programming system, and its application to the Marsokhod Martian Rover. In *Proceedings of the ACM SIGPLAN 1995 workshop on Languages, compilers, & tools for real-time systems*, pages 146–155. ACM, 1995.

[11] J. A. Stankovic and K. Ramamritham. *Advances in Real-time systems*. IEEE Computer Society, 1993.

[12] D. B. Stewart, D. E. Schmitz, and P. K. Khosla. Implementing real-time robotics systems using CHIMERA II. In *Proceedings of IEEE International Conference on Systems Engineering*. IEEE, 1990.

[13] J. Sweeney, T. Brunette, Y. Yang, and R. A. Grupen. Coordinated teams of reactive mobile platforms. In *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2002.

[14] E. Yoshida, T. Arai, J. Ota, and T. Miki. Effect of grouping in local communication system of multiple mobile robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 808–815. IEEE/RSJ, 1994.