Sold!: Auction Methods for Multirobot Coordination

Brian P. Gerkey and Maja J. Matarić

***Abstract*—The key to utilizing the potential of multirobot systems is *cooperation*. How can we achieve cooperation in systems composed of failure-prone autonomous robots operating in noisy, dynamic environments? In this paper, we present a novel method of dynamic task allocation for groups of such robots. We implemented and tested an auction-based task allocation system which we call MURDOCH, built upon a principled, resource centric, *publish/subscribe* communication model. A variant of the Contract Net Protocol, MURDOCH produces a distributed approximation to a global optimum of resource usage. We validated MURDOCH in two very different domains: a tightly coupled multirobot physical manipulation task and a loosely coupled multirobot experiment in long-term autonomy. The primary contribution of this paper is to show empirically that distributed negotiation mechanisms such as MURDOCH are viable and effective for coordinating physical multirobot systems.**

***Index Terms*—Auctions, contract nets, coordination, multirobot systems, task allocation.**

I. Introduction

HOW CAN WE intelligently coordinate groups of robots? Effective robot teams must work together on tasks, efficiently sharing the workload in much the same way as a group of humans might. The key to exploiting the potential of multirobot systems is *cooperation*. Robots should, whenever possible, cooperate strongly in order to maximize their overall task performance. Modern robots are equipped with high-bandwidth communications and a diverse array of sensors and actuators; these resources can and should be exploited in order to achieve cooperative behavior at the group level. By sharing information and leveraging each others' skills, a group of robots can truly be more than the sum of its parts.

How can we achieve such cooperation in systems composed of failure-prone autonomous robots operating in noisy, dynamic environments?

One model, especially popular in the field of swarm robotics [1], [2], is *emergent* cooperation: the robots do not explicitly work together, but group-level cooperative behavior emerges from their interactions with each other and the world. This kind of cooperation is observed in nature and appears to be the evolutionary favorite for cooperation among nonhuman animals. An emergent system, when constructed skillfully, can be extremely effective and is likely the simplest and most elegant solution to a problem. However, it is a solution to a *specific* problem, and if robots are to be generally useful, they must be capable of solving a variety of problems. Furthermore, emergent cooperation solutions have traditionally been applied to homogeneous robot groups and have relied heavily on redundant skills across the group to achieve good overall performance.

For our general strategy, we choose instead an *intentional* model of cooperation [3]. In this model, robots cooperate explicitly and with purpose, often through task-related communication. As compared with emergent cooperation, intentional cooperation is better suited to the kinds of real-world tasks that we, as humans, might want robots to do. If the robots are deliberately cooperating with each other, then, intuitively, humans can deliberately cooperate with them, which is a long-term goal of multirobot research. In this paper, our use of intentional cooperation is at the level of *task allocation* and need not propagate to the level of task execution. Importantly, we do not prescribe or proscribe any particular method for implementing the details of a task. For example, if we assign a foraging task to a team of robots because they are the best fit for the job, they can execute the task in any way they wish, from probabilistic swarming to classical planning.

In deciding who should do what in a team, we take inspiration from the distributed artificial intelligence (DAI) community. Specifically, we employ fitness-based auctions and a simple negotiation protocol. The concept of negotiation necessarily entails some form of *task commitment*, which can complicate system design and might hinder performance or fault tolerance [4]. However, a large class of tasks, especially those involving physical state, can benefit from the robots' willingness to commit. Furthermore, as we demonstrate, task allocation based on explicit negotiation can be an effective and fault-tolerant method for controlling multirobot systems. Thus, this paper answers the challenge posed in [3]:

> "[Negotiation-based] solutions have not been adequately demonstrated in *situated* agent (i.e., robotic) teams, which have to live in and react to dynamic and uncertain environments amidst noisy sensors and effectors, frequent agent failures and a limited bandwidth, noisy communication mechanism."

To address the challenge, we present a novel method of dynamic task allocation for groups of robots. We have implemented and tested MURDOCH, a general task allocation system based on a principled, resource centric, *publish/subscribe* communication model that makes extensive (but efficient) use of explicit interrobot communication. MURDOCH is a variant of the well-known Contract Net Protocol (CNP) [5] and uses simple auctions to allocate tasks. Since our

Manuscript received March 16, 2001; revised March 14, 2002. This paper was recommended for publication by Associate Editor T. Arai and Editor S. Hutchinson upon evaluation of the reviewers' comments. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under Grant DABT63-99-1-0015 and Grant F30602-00-2-0573, in part by the Office of Naval Research under Grant N00014-00-1-0140 and Grant N0014-99-1-0162, and in part by the Jet Propulsion Laboratory under Contract 1216961.

The authors are with the Robotics Research Laboratory, Computer Science Department, University of Southern California, Los Angeles, CA 90089 USA (e-mail: bgerkey@cs.usc.edu, mataric@cs.usc.edu).

Digital Object Identifier 10.1109/TRA.2002.803462

goal is not to exploit resources in a globally optimal fashion, but rather to investigate practical and efficient methods for allocating tasks to groups of autonomous and heterogeneous physical robots, we have built MURDOCH as a completely distributed system. As such, it offers a distributed approximation to a global optimum of resource usage which is equivalent to an instantaneous greedy scheduler. We have tested this system in two very different domains: a tightly coupled multirobot physical manipulation task and a loosely coupled multirobot experiment in long-term autonomy. The primary contribution of this paper is to empirically demonstrate that distributed negotiation mechanisms, such as MURDOCH, are indeed viable and efficient for coordinating physical multirobot systems.

The rest of this paper is organized as follows. In the next section, we precisely define the problem with which we are concerned. In Section III, we position our research in the context of the relevant work on this problem. In Section IV, we explain and justify our approach. Sections V and VI give the details of our implementation of the approach, called MURDOCH, and the experiments we have used to validate it. We present and discuss results from these experiments in Section VII, and conclude with Section VIII.

II. The Problem

As stated above, we are concerned with the problem of multirobot coordination. In this paper, we attack one part of that problem: task allocation for groups of robots. We characterize our domain with the following assumptions:

- the system is composed of physically embodied robots;
- the robots are heterogeneous, possessing different skills;
- the robots can communicate, but messages may be lost;
- the robots are honest and cooperative;
- the robots (or parts of them) can fail at any time;
- a robot may not be aware of its own (partial) failure;
- the robots are multipurpose, not task specific;
- no model is available to describe the sequence in which tasks are generated;
- if a robot is to oversee a task, it can determine its own progress and completion of the task.

Together, these assumptions define a realistic environment in which task allocation can be explored. In fact, all but the last three assumptions arise directly from the unavoidable characteristics and constraints of working with physical robots. The three self-imposed constraints represent our view of what will be required of multirobot systems for them to be generally useful.

In order to reason clearly about multirobot task allocation, we cast it as a *dynamic resource-allocation problem*. Given a set of resources and a sequence of tasks, the goal is to assign resources to tasks in an efficient manner. In its most general form, this problem is equivalent to the NP-complete conjunctive planning problem [6]. Clearly, the traditional planning approach will not suffice for the physical robot domain with which we are currently concerned. Aside from the well-known problems of providing timely responses and reacting to unexpected contingencies, a planning system is especially unsuited to our task-allocation problem because, from the perspective of the robots, which are model free, the tasks appear to be *randomly* generated (in our validation they are, in fact, generated randomly). Without a model of the world, planning provides no benefit.

In a model-free environment, with no expectation of what future task allocation requirements might be, the goal is to allocate each incoming task as efficiently as possible based on what the system is trying to optimize. In our approach to task allocation, we strive to minimize three aspects of the system:

- resource usage;
- task completion time;
- communication overhead.

These criteria are not orthogonal; rather, they interact strongly. For instance, a task allocation that minimizes resource usage might take longer to complete than another assignment which minimizes completion time. Likewise, having found an inadequate task allocation, one might choose to expend additional bandwidth in search of a better allocation. The details of these tradeoffs are application specific, but the overall optimization goal remains the same.

III. Related Work

The problem of multirobot coordination touches on many established research areas. In this section, we discuss the relevant work in task allocation (both physically embodied and not) and box pushing (one of our experimental validation domains; see Section VI-B.II).

A. Unembodied Task Allocation

Beginning with the seminal CNP [5], automated coordination schemes for multiagent systems have been applied to a wide variety of problem domains. In fact, the CNP itself has been widely studied and extended [7]–[9]. A number of other coordination approaches have also been proposed; to date, two of the more mature systems are the Open Agent Architecture [10] and RETSINA [11]. Both architectures focus on providing a maximally general environment in which a diverse population of agents, such as user interfaces and legacy database management systems, can interact and coordinate. Since we are concerned specifically with task allocation for physical mobile robots, we do not require the overhead (such as ontology specifications) that allows these systems to be so general. There is a component of task allocation among their agents, although the tasks are purely informational, rather than physical. Both architectures accomplish task allocation through a broker (called *facilitator* and *matchmaker*, respectively) which matches new task requests with agents that have previously advertised relevant capabilities. In contrast, MURDOCH completely distributes the matchmaking process and thus, does away with the centralized broker [12].

A more situated (but still not *embodied*) distributed coordination system is the process scheduler *Condor* [13], which attempts to maximize processor usage in a network by remote execution of background jobs on idle workstations. More recently, the same process scheduling problem has been approached from an agent perspective with *Challenger* [14]. The scheduling mechanism in *Challenger* requires agents throughout the network to bid for available jobs in a manner not unlike the auctions used in MURDOCH. Auctions have been

demonstrated in the control of simulated robots performing an area-coverage task [15], and preliminary results have been reported with physical robots in a mapping task [16]. However, to our knowledge, MURDOCH is the first proven application of auction methods to the coordination of physical multirobot systems applied to multiple tasks.

B. Embodied Task Allocation

Compared to software agent systems, task allocation for physically embodied robots has received far less attention. A notable exception is ALLIANCE [3], an architecture for coordinating behavior-based robots [17], [18] that must cooperate to achieve some task. Every robot knows the entire task to be completed; the robots use *motivational behaviors* to compete for the various task components over time. Each robot tracks both its own and its teammates' fitness and progress, incorporating this performance information into local measures of *impatience* and *acquiescence*. If a robot becomes sufficiently impatient, it may seize a task from a sufficiently acquiescent robot, thereby providing fault tolerance. ALLIANCE has been demonstrated on multirobot foraging [3], box pushing [19], and target tracking (CMOMMT) [20].

A similar but more minimalist approach to robot task allocation is taken by BLE [21], a modern distributed derivative of the Subsumption Architecture [22] that has been demonstrated on a weighted version of the CMOMMT target-tracking task. In the BLE framework, robots have no commitment to their task; the relevant behaviors themselves continuously communicate to decide who is best fit for each job. Since the robots may switch tasks at any moment, the system has a good dynamic response to environmental changes, at the expense of added communication required for the continuous fitness broadcasts.

Importantly, both BLE and ALLIANCE assume behavior-based control of the robots in the team. They implement task allocation through behavior inhibition; each robot has some desire to execute each task and the robots suppress each others' activity directly at the behavior level. We make no such assumptions in MURDOCH, requiring only that each robot be able to start and stop a task–execution module (be it behavior based, hybrid, or deliberative) on demand.

C. Box Pushing

Box pushing has long been one of the canonical task domains for mobile robot researchers and a natural one for cooperative robotics. Several (but not many) systems have been demonstrated; we highlight the relevant ones here.

At one extreme, [23] and [24] describe a swarm-like method for moving a large box with many small, locally controlled robots; the system could fairly be described as *emergent*. In stark contrast is the planner-based master-slave pushing system described in [25]. A similarly deliberative approach is taken in [26]; the authors focus on the analysis of various two-robot pushing protocols with regard to information requirements. Some middle ground is found by the two-robot behavior-based approach presented in [27], with an emphasis on the robots' learning policies to enable effective cooperation. In [19], a fault-tolerant two-robot box-pushing system is developed and proof-of-concept demonstrations are given. More recently, a method for single-robot box pushing through an obstacle field (in the context of robot soccer) is given in [28].

With regard to the pushing control system itself, the work we present in this paper is most similar to the pusher-steerer protocol of [26] and, to a lesser extent, the master-slave system of [25]. However, neither of these architectures made any provision for robot failures. Of the other three multirobot systems, only [27] is goal directed, and in that case, both pushing robots could directly perceive the goal, somewhat reducing the need for cooperation.

IV. Our Approach

We have developed an approach to the resource-allocation problem given in Section II. In this section, we motivate our communication model, task representation, and auction model. The implementation details of the approach are given in the next section.

A. Anonymous Communication

Traditionally, communication in computer networks uses point-to-point unicast message delivery. That is, when Computer *A* wants to send a message to Computer *B*, *A* addresses the message *by name* to *B* and hands it off to the network. This communication model is problematic in robot systems for two reasons.

First, compared to broadcast messaging, unicast is extremely inefficient for delivery of messages to multiple recipients. Using broadcast, we can send a message once and it will be received by everyone, yielding a potentially large savings in bandwidth, which is an important consideration for multirobot systems. Second, the systems that we wish to control are fluid. Robots can move in and out of communication range, the communication system itself can drop messages, and the robots can experience many different failures. In order to tolerate these dynamics, the communication layer should never address robots by name. Instead, robots should communicate *anonymously* through broadcast means.

B. Hierarchical Task Structure

To effectively describe the tasks required for our allocation problem, we must choose a representation that is both powerful enough to handle a wide variety of tasks and simple enough for a human to use. For this purpose, we have developed a flexible hierarchical task structure. A task is simply a tree which can contain other tasks. That is, the root node of one task tree can be an intermediate node in another task tree. In these trees, a parent-child relationship means that the parent task is responsible for allocating (and subsequently monitoring) the child task. Since, in this paper, we are investigating task allocation, not task decomposition, we delegate to the designer the work of defining the task trees in the form accepted by MURDOCH. Alternatively, an offline planner, such as that described in [29], endowed with knowledge of the preconditions, postconditions, and interdependences of the tasks involved could be employed. Note that the planner would not be deriving specific motion sequences, but rather overall task structure; the robots themselves generate *in situ* trajectories.

C. Auctions

Auctions, in one form or another, have been used in societies throughout history to allocate scarce resources among individuals and groups. The auction, as used by humans, has proven a powerful tool for achieving efficient resource allocations, especially in large-scale environments in which the acquisition of consistent global state information is difficult or impossible. As a consequence, auctions have garnered a great deal of attention from economists, and a rich body of theory exists regarding their properties, especially their ability to deal with uncertainty. For an overview of the common auction types, see [30]. Auctions have also been studied theoretically by researchers in the field of multiagent systems. Many algorithms for holding and deciding auctions have been proposed and analyzed, both from the perspectives of resource allocation and task allocation. We decided upon auctions because they are particularly well suited to our distributed robotic domain: auctions are scalable, allow for significant compartmentalization of data and control, and are efficient in computation and communication.[1]

It may seem counterintuitive to employ a competitive mechanism (i.e., the auction) in order to elicit cooperative behavior. In our case the difference between competition and cooperation reduces to a question of motivation: are the robots selfish? Philosophical arguments aside, we believe that the answer is no. Having programmed them ourselves, we can state unequivocally that the robots' most basic motivation is to do useful work (i.e., accomplish tasks). They are neither greedy nor dishonest and are only self-interested in so far as their limited resources impact their ability to do work. For example, as described in Section VI-B.I, when its battery level is sufficiently low, a robot will refuse to take on future tasks and will seek a charging station. Except in such special circumstances, the robots cooperate with each other to the greatest possible extent.

V. MURDOCH

A. Publish/Subscribe Messaging

In implementing distributed control systems for teams of robots, researchers typically resort to *ad hoc* communication strategies. These specialized strategies are often implemented as hand crafted, task-specific communication graphs. For example, in [3] and [21], communication channels are explicitly created among individual behaviors on the different robots. While this model is well suited to the design of special purpose, single-task systems, it has not been demonstrated for the general case of controlling a large group, in which the members dynamically form teams to accomplish different tasks as they are presented to the system. As an alternative to such special-purpose systems, we propose a principled communication model based on *publish/subscribe messaging*, an instance of anonymous communication.

1) Background: The unifying concept of publish/subscribe systems is that *messages are addressed by content rather than by destination*. This idea, often called *subject-based addressing*, is used to divide the network into a loosely coupled association of anonymous data producers and data consumers. A data producer simply tags a message with a subject describing its content and "publishes" it onto the network; any data consumers who have registered interest in that subject by "subscribing" will automatically receive the message. Data producers need not have any knowledge of which consumers, if any, are receiving their messages and vice versa. This kind of communication represents a fundamental departure from the traditional unicast model. We have tailored this idea slightly so that when a message is published, it is addressed to a set of subjects, rather than just one. A data consumer will receive a message if the subjects in the message comprise any subset of the consumer's current subscription list. Although we have not optimized the subset matching algorithm in our implementation, others have investigated the topic [31], [32].

2) Subject Namespace: The first step in creating a publish/subscribe system is designating the semantics of the subject namespace. Analogous to deciding the layout of a database, the interpretation of subjects will heavily influence the rest of the system. In MURDOCH, since we are allocating tasks among a group of potentially heterogeneous robots, we use subjects to represent their "resources."[2] Resources can be physical devices (e.g., `camera`, `gripper`, `sonar`), higher level capabilities (e.g., `mobile`, `door-opener`), or abstracted notions of current state (e.g., `idle`, `have-puck`, `currently-pushing-box`). Thus, if we have a task that involves going to some location and observing it, we can reach the capable robots (who are otherwise unengaged) by addressing a message to the set {`mobile camera idle`}. Since messages are addressed to subjects and subjects represent resources, all interrobot communication will necessarily be resource centric, which we believe to be fundamental in achieving our goals. The robots never interact with each other by name and, in fact, have no explicit knowledge of each others' existence; rather they only communicate about tasks and all messages are addressed in terms of the resources required to perform those tasks.

B. Auction Protocol

At the heart of MURDOCH lies a simple distributed negotiation protocol that allocates tasks via a sequence of first-price one-round auctions [30]. The process is triggered by the introduction of a task to the system. The task could be introduced in many ways, including a human user, a `cron`-style alarm, or an already ongoing task. In every case, the auction proceeds in five steps.

1) **Task announcement.** An agent (working on behalf of a user, alarm, or task) acts as "auctioneer" by publishing an `announcement` message for the task. The message contains details about the task, such as its name, length, and a new subject on which to negotiate it. The message

[1]Admittedly, the auctioneer somewhat centralizes the system. However, because we use a hierarchical task structure and because any individual can act as auctioneer for a particular task, MURDOCH is primarily a distributed mechanism.

[2]We could have used *tasks* instead of *resources* in constructing the subject namespace. Since the robots must agree on a common task vocabulary, few changes would be required in MURDOCH. Further, the vocabulary of tasks is smaller than the vocabulary of resources, and by adhering to a task-level abstraction, we could easily accommodate more heterogeneous systems in which different robots execute the same task using different resources.

is addressed to the set of subjects which represent the resources required to execute the task; thus, only those robots currently capable of performing the task will receive the message.

2) **Metric evaluation.** Since there may be more than one capable robot available for a given task, we need a method for deciding among them. This decision process is the very basis of achieving effective task allocation; the goal is to allocate each task to the most fit robot. Thus, the `announcement` also includes *metric(s)* with which each candidate robot can determine its fitness (see Section V-B.I for details on metrics).
3) **Bid submission.** After evaluating the appropriate metric(s), each candidate robot publishes its resulting task-specific fitness "score" as a `bid` message.
4) **Close of auction.** After sufficient time has passed, [3] the auctioneer processes the `bids`, determines the winner, and notifies the bidders with a `close` message. The winner is awarded a time-limited contract to execute the task and the losers return to listening for new tasks.
5) **Progress monitoring/contract renewal.** While the winner executes the task, the auctioneer monitors task progress. Assuming sufficient progress is being made, the auctioneer periodically sends a `renewal` message to the winner, which responds with an `acknowledgment` message, until the task is completed.

The fact that the winner's contract is time limited is an essential part of MURDOCH's fault-tolerant capabilities. Recalling the argument for "soft state" [33], time-limited contracts provide a built-in timeout that can trigger fault detection and recovery. For example, if the auctioneer fails to receive an acknowledgment after sending a renewal, it can assume that the robot previously executing the task has failed, and so it can reassign the task. Similarly, the task can be reassigned if the auctioneer finds that insufficient progress has been made. In either case, the previous winner will terminate task execution when its contract has expired without renewal.

Since each task will always be claimed by the most capable robot available at the time, MURDOCH acts as an instantaneous greedy task scheduler. Thus, it suffers from the well-known problems of greedy algorithms; they are manifested in our domain as situations in which, although sufficient resources exist to achieve a given set of tasks, the order in which the tasks are presented causes resources to be exploited in a nonoptimal manner, such that not all the tasks are actually achieved. Centralized broker and matchmaker systems sometimes avoid this pitfall by analyzing concurrent tasks before allocating them. That kind of planning, however, will not help in the class of domains where individual tasks are input stochastically from some outside source, such as a human user.

1) Metrics: Metrics can take many forms, with the restriction that each one, when evaluated in the context of a specific robot, should return a scalar "score" representing that robot's fitness for the task. A metric is usually defined as some function of the robot's current state, although in general, a metric could perform any arbitrary computation, including interrobot communication. As an example, if the task under consideration is to go to a certain location and pick up an object, one possible metric is to compute the Cartesian distance from the robot's current position to the goal position, with a shorter distance scoring better.[4] Multiple metrics can be defined for a single task, with the final score being some combination of the individual scores; we have experimented with combining metrics through both simple sums and weighted sums that allow for a prioritization of metrics. It is important to note that metrics, being functions of each robot's own sensor data, may not accurately represent the current state of the robots, possibly resulting in a nonoptimal allocation of the task. Since finding an optimal allocation would require gathering global data, guaranteeing its accuracy and centralizing control, we find our metric-based distributed approximation to be a parsimonious alternative.

[3]In the current implementation, the timeout for an auction is 1.5 s.

Fig. 1. Typical Pioneer 2-DX robot. Visible in this image are the two drive wheels, front sonar ring, compass (small cube), Ethernet modem (antennae protruding vertically) and laser range finder (looks like a coffee maker). At the rear of the robot is another ring of sonars.

VI. Experimental Validation

We have validated MURDOCH through a series of experiments designed to test the task allocation and fault-tolerant capabilities of the system. In this section, we detail the robot platform and experiments, then present and discuss the results.

A. Platform

We used as our experimental test bed a team of ActivMedia Pioneer 2-DX mobile robots (see Fig. 1). The Pioneer 2-DX is a 44 cm $\times$ 38 cm $\times$ 22 cm nonholonomic two-wheeled base that

[4]By using such metrics to compare different robots, we assume that the scalar fitness values produced by the metrics are indeed comparable. For example, if distance is a metric, then although they may use different methods for measuring distance [e.g., odometry versus intertial meausrement unit (IMU)], the robots must all report distance in the same units.

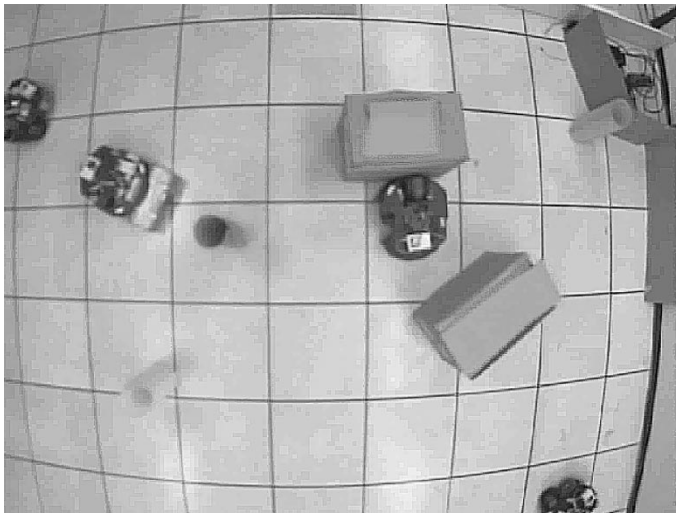

Fig. 2. Overhead snapshot of the long-term loosely coupled scenario. The robot in the center of the image is performing the **cleanup** task while the robot to its immediate left is performing the **object-tracking** task. In the upper right corner is the charging station.

is differentially steered (a passive caster provides balance). This versatile research robot can be configured with many different peripherals, including front and rear sonar arrays, compasses, pan-tilt-zoom cameras and laser range finders.

Each robot houses a Pentium-based computer running Linux, which executes the robot's control program. Also onboard is the device server Player,[5] [34] which handles low-level sensor and actuator control. MURDOCH is implemented on the Pioneer through the Player interface. Interrobot communication is provided by way of wireless Ethernet (802.11), with an effective shared bandwidth of approximately 1.9 Mb/s. The topology is such that every machine on the network can communicate freely with every other machine.

B. Experimental Design

We have claimed that MURDOCH's negotiation-style task allocation is applicable to a variety of problems and that it is robust to environmental dynamics, such as robot failures. To substantiate the first claim, we have evaluated MURDOCH in two very different domains: a long-term scenario consisting of many loosely coupled single-robot tasks, and a cooperative box-pushing task requiring tight coordination among the robots. To demonstrate fault tolerance, we explored several robot failure modes in the box-pushing task. Video footage of these experiments is available at: http://robotics.usc.edu/~agents/projects/auctions.html.

1) Loosely Coupled Task Allocation: In the pursuit of long-term autonomy, we designed a scenario (see Fig. 2) in which a large heterogeneous group was required to self-organize to accomplish a randomly generated sequence of tasks.

a) Task description: The team comprised the following machines:

- three robots with cameras;
- three robots with cameras and laser range finders;
- one robot with a camera and a tactile bumper array;
- one desktop computer with an overhead camera.

We exercised this team with four different single-robot tasks:

- **object-tracking;**
- **sentry-duty;**
- **cleanup;**
- **monitor-object.**

The first, **object-tracking**, requires the `camera` and `mobile` resources. The task is to find and track from a safe distance a certain colored object. The second, **sentry-duty**, requires `camera`, `laser`, and `mobile`. The task is to go to a target location (marked by a colored object), then turn about and remain still, watching for any motion with the laser and setting off an intruder alarm if motion is detected. Our third task, **cleanup**, requires `camera`, `bumpers`, and `mobile`. The task is to find each small box of a certain color and use tactile bumpers to push the box to the edge of the room, thereby cleaning the room. The final task, **monitor-object**, requires only `overhead-camera`. The task is to monitor the positions of various colored objects, such as boxes and robots, from an overhead view and log the information for later review.

Each robot also runs a battery-monitoring behavior that checks the current charge whenever the robot is idle, between tasks. At that time, if the battery is low enough, the robot will unsubscribe from all subjects (thereby removing itself from consideration for future tasks) and go to a clearly marked charging station. After charging for some time, the robot is freed to reenter the experiment.

b) Experiment: We ran the team over a long period of time (approximately three hours), periodically injecting random tasks of random lengths into the system. Each new task was auctioned (and subsequently monitored) by an automated tasking agent that was executing on a separate desktop computer. Each machine was controlled by a copy of the same program; this program simply queried its host for the list of currently available devices, then made the proper resource subscriptions. For example, the robots equipped with both cameras and lasers subscribed to {`camera laser mobile`}, while the desktop computer only subscribed to `overhead-camera`.

2) Box Pushing: While the previous experimental domain demonstrates MURDOCH's ability to handle heterogeneous multirobot systems, the tasks themselves require little coordination beyond the initial task assignment. To be generally useful as a tool for control of robot teams, a system must also be capable of allocating and coordinating tasks that require tightly coupled cooperation among the robots and it must do so in a fault-tolerant manner. We chose as an example of this tightly coupled task domain the long-studied *box-pushing* problem.

a) Task description: Our experimental box-pushing setup is shown in Fig. 3. The task we address is to cooperatively move a box from some initial location to some observable goal location. In solving this problem, we take inspiration from human coordination commonly observed when people move large pieces of furniture. If the people who are pushing or carrying cannot see where they are going, another person stands between the carried object and the goal and periodically

[5]Player was developed jointly at the USC Robotics Research Lab and HRL Labs and is freely available under the GNU General Public License from http://playerstage.sourceforge.net.

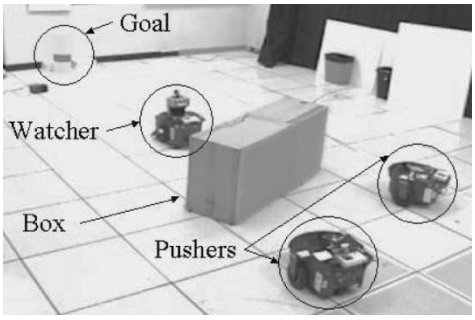

Fig. 3. Our experimental box-pushing setup. The task is for the `pushers` to move the box to the goal with the help of the `watcher`. (Image 1 of 3 taken from an experimental trial; see Figs. 4 and 5).

directs them. This "watcher" can see both the current position of the object and the goal, and thus, can compute an error signal, perhaps in the form of a correction angle, that can be communicated to the "pushers." [6]

We formally define this problem with a set of constraints. First, both the box and the goal are observable and there is an obstacle-free path between them that is wide enough for the box and robots to pass (i.e., we do not consider negotiating obstacles in a coordinated fashion, only as part of individual low-level control). Second, the box is large compared to the size of the robots.[7] Third, the robots can only move the box by pushing through frictional contact. Finally, the pushing robots cannot directly perceive the goal due to the size of the box.

We solve this problem with MURDOCH by following the division of labor used by humans. We construct a natural task hierarchy in which high-level watching tasks are auctioned to watcher robots, who then direct the actions of pusher robots by auctioning appropriate low-level pushing tasks. In these experiments, as shown in Fig. 3, two robots act as `pushers` and a third performs the `watcher` role. The `pushers` can see the box, and the `watcher` can see the goal. In addition, the `watcher`, while servoing on the goal, can accurately perceive (using a laser range finder) the angular error of the box's orientation with respect to the path from the box to the goal. Our aim, then, is to rotate the box until that angular error is zero (i.e., the box is orthogonal to the path to the goal), while simultaneously translating it toward the goal. We call this approach to box pushing `pusher-watcher`; for details of the control law used, see [35]. We describe here only aspects related to task structure and allocation.

Our box-pushing team is composed of the following machines:

- one robot with a camera and laser range finder;
- two robots with cameras.

A user poses a `relocate-box` task to MURDOCH; this task is hierarchical and is composed of a `watch-box` task that has two children `push-box` tasks. The `watch-box` task is auctioned first, with the resource list {`mobile laser camera`}. The one robot with those resources claims the task, becoming the `watcher`. It begins executing the `watch-box` task,[8] which consists of the following simple algorithm:

1) servo toward the goal (with the camera);
2) determine the angular error of the box (with the laser range finder) and calculate appropriate left and right pushing velocities;
3) if new velocities are approximately equal to the previously assigned velocities, then renew existing contracts and go to Step 1;
4) sequentially auction new left and right `push − box` tasks with resource lists (mobile camera), in order of decreasing velocity;
5) go to Step 1.

The metric for the `push-box` tasks is based on the color camera input and is a measure of how well the robot is positioned for pushing on a particular end of the box. For example, when the task is to push on the right end, the metric reflects whether the box is offset to the left in the robot's camera image. Because of the dynamic nature of the system, each `push-box` task is 3 s in length.

In a typical run of `pusher-watcher`, the `watcher` initially auctions left and right `push-box` tasks with proper velocities and lets them push until the box's orientation changes sufficiently to warrant different pushing velocities and thus, new tasks. At that point, the current contracts are allowed to expire and new ones are formed. This reactivity to world conditions is the feature that enables MURDOCH to dynamically reassign tasks in the face of robot failure. For example, when only a single robot is available, the `watcher` will actually try to allocate two pushing tasks as usual, but only one (the one that has the higher velocity and is auctioned first) will be claimed. That single contract is renewed and the robot pushes on one end of the box until the orientation changes enough that it is more important to push the other end, at which point the robot will simply switch sides. When another robot is introduced, it will claim the next available pushing task, and the two robots will cooperate at pushing the box.

b) Experiments: In order to evaluate MURDOCH in the box-pushing domain, we performed five sets of experiments on our group of Pioneer robots, as described below. During the experiments, we measured two quantities: success/failure and elapsed time. We define success as the situation in which the `watcher` declares that the task is terminated and the center of the box is positioned within 0.5 m of the target location; we do not specify a target orientation for the box. Conversely, a trial is a failure if either the `watcher` declares termination when the box is not close enough to the goal, or the box is rotated so far that the `watcher` can no longer perceive it using its laser range finder (this threshold is approximately 70°). For ground truth we used an external metrology system consisting of multiple laser range finders and beacons to track the positions of the box and robots throughout the experiments (plots are based on that data).

As a control, Experiment Set 1 involved the simplest scenario. Two `pusher` robots had to move the box along a straight-line

[6]Actually, the watcher likely will not communicate the raw angle, but rather some higher level command, such as "push more on the right." We do the same.

[7]Specifically, the intended contact surfaces should allow at least two robots to be pushing simultaneously.

[8]As with all tasks, the `watch-box` task is time limited and is monitored, in this case, by an agent on behalf of the user. If the `watcher` fails in some way, then the `watch-box` task will be automatically reallocated if a new `watcher` can be found; otherwise the user is notified of the problem.

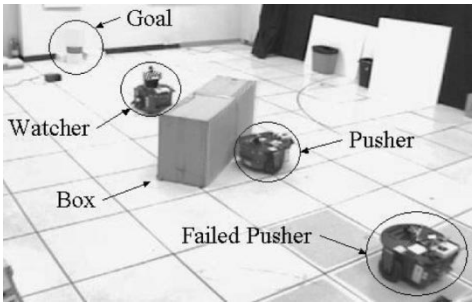

Fig. 4. Fault-tolerance in action: After we induced a single robot failure, the remaining robot is left to push on its own. (Image 2 of 3 taken from an experimental trial; see Figs. 3 and 5).

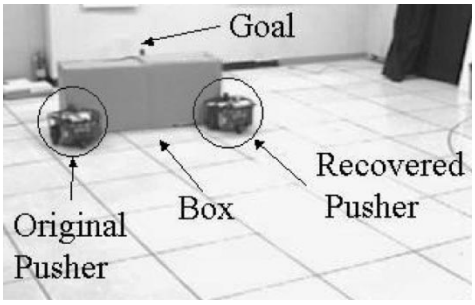

Fig. 5. We "revived" the failed robot, it was reintegrated into the team and all completed the task together. (Image 3 of 3 taken from an experimental trial; see Figs. 3 and 4).

path for approximately 3 m (90% of the length of our lab) and no faults were introduced. In Experiment Set 2, we tested the system's tolerance to an individual robot failure. The setup is the same as in Experiment Set 1, with two `pushers`, but, after they pushed the box approximately 1.2 m, we simulated a robot failure by seizing one `pusher` and shutting it off. As a result, the remaining `pusher` was left to push the box by itself, alternating sides under the direction of the `watcher` (see Fig. 4). In Experiment Set 3, we tested MURDOCH's progress monitoring capability by introducing a partial robot fault. As with Experiment Set 2, two `pushers` began the task together; we then simulated a robot becoming stuck (e.g., in sand) by disabling its motor power. This failed robot was still in communication with the team and claimed to be fit for pushing tasks, but was immobile. In order to complete the task, the `watcher` had to detect the lack of progress on the part of the stuck robot and exclude it from future task offerings. In Experiment Set 4, we tested the system's dynamic response by inducing both failure and recovery. We first let them both push approximately 0.6 m, then seized one `pusher` to simulate complete failure. After the remaining `pusher` had single-handedly pushed the box another 1.2 m, we reintroduced the failed `pusher`, at which point the two had to finish the task together (see Fig. 5).

While Experiment Sets 1–4 showcase the ability of the `pusher-watcher` system to cooperatively move a box along a straight line, it is important to be able to follow more general paths. In Experiment Set 5, we tested the ability of the system to execute curved trajectories by placing the goal marker approximately 35° off the center line and 2.75 m away (see Fig. 6). In order to follow this path, the robots had to behave in

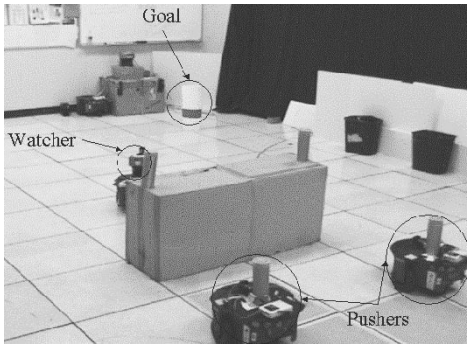

Fig. 6. Setup for Experiment Set 5. Goal is approximately 35° off the center line, requiring the robots to push the box along a curved trajectory.

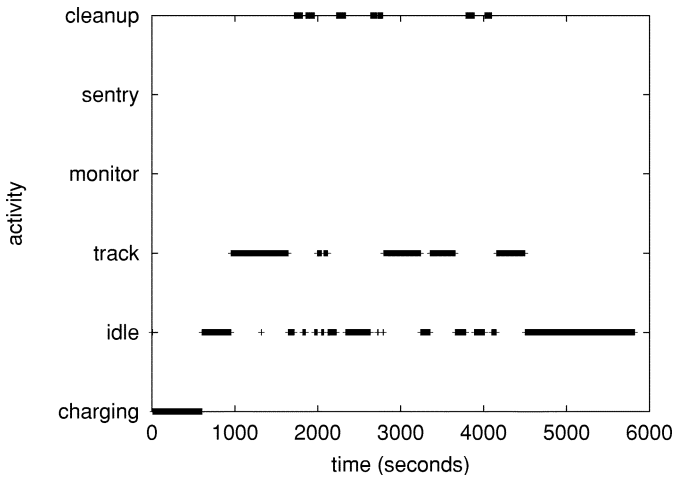

Fig. 7. An example plot of task execution during the long-term experiment. Shown here is the activity, over the first half of the experiment, of the robot equipped with a camera, laser range finder and bumpers.

a tightly coordinated fashion, making a series of rotational and translational adjustments.

VII. Results

We now present and discuss the results from experiments performed with MURDOCH in our two task domains.

A. Loosely Coupled Task Allocation

In the loosely coupled scenario, we observed the following system behavior. Over the course of three hours, the group (seven robots and one desktop computer) successfully executed 49 tasks. They returned to charge their batteries twelve times. Some of the randomly generated tasks were unachievable due to a lack of resources, because all the capable robots were either charging or otherwise engaged. In these situations, an error was returned, suggesting that the task be reintroduced later. We have not automated the task reintroduction process, but a variety of solutions are immediately apparent (e.g., the exponential back-off algorithm used by Ethernet to resolve bus contention).

The same control program executed on each robot for the length of the experiment, with robots periodically idle (only executing passive collision avoidance), executing some task, or charging. Shown in Fig. 7 is a plot of the task execution over

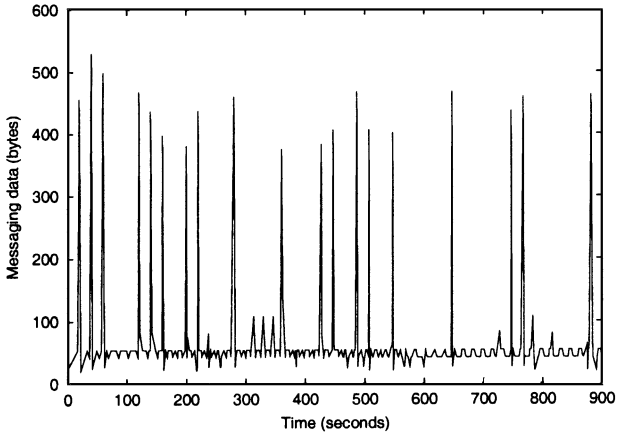

Fig. 8. Communication overhead incurred by MURDOCH. Shown here is the total amount of data transferred among the robots during the first 15 minutes of the long-term experiment. Each spike corresponds to an auction, which causes a brief flurry of activity.

time of one of the robots. The plot is from the robot equipped with a camera and tactile bumper array. The robot begins at the charging station, then is periodically idle and engaged in various tasks. Given its resources, this robot was only capable of the **track-object** and **cleanup** tasks, and so those are the tasks that it accepted and executed.

Whenever sufficient resources were available for a task, the task was allocated and executed. Further, each task was allocated optimally, up to the accuracy of the metrics and the sensor data on which it relied. Given our assumption of truly random task injection, with no look ahead, the robot assigned to each task was the best available robot for that task at the time of assignment. Finally, MURDOCH is efficient with regard to communication. Over the entire experiment, 3791 messages were sent, with an average size of 36.06 B (see Fig. 8). The average total bandwidth used by MURDOCH was 0.66 kb/s, with a maximum burst of 4.74 kb/s. This usage, which is the entire communication overhead incurred by MURDOCH, is a tiny fraction of the bandwidth available on modern radio networks (1–11 Mb/s), suggesting that the system will scale well with larger numbers of robots and/or tasks.

B. Box Pushing

We performed 10 trials each from Experiments Sets 1–4. In the 40 trials, there were a total of four failures, one occurring in each set. Three failures were due to over-rotation of the box, and the fourth was due to premature termination on the part of the `watcher`, presumably because of sensor noise. With 36 successes in 40 trials, the two-sided 95% binomial confidence interval for the overall success rate of the system is: $p \in (0.76, 0.97)$. We also analyzed the time elapsed during the successful trials, as a measure of relative efficiency among the different experiments. The results are shown in Table I. In Experiment Set 1, with no failure, the two `pushers` almost always executed the task in one continuous movement, yielding extremely similar (and short) completion times across trials. When one robot failed, the completion time rose considerably because the remaining robot was left to push either side of the box in turn, which is rather inefficient compared to the two-robot cooperative case.

TABLE I
MEAN (μ) AND STANDARD DEVIATION (σ) OF THE ELAPSED TIME (IN SECONDS) FOR SUCCESSFUL PUSHING TRIALS IN EACH OF FOUR BOX PUSHING EXPERIMENT SETS

Set	Description	μ	σ
1	No failure (straight path)	31.22	0.44
2	Pusher failure	132.75	26.94
3	Partial pusher failure	260.89	37.79
4	Pusher failure & recovery	116.44	37.72

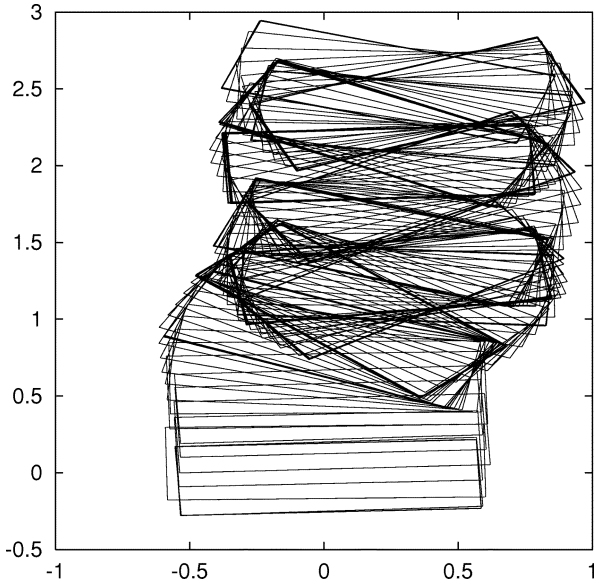

Fig. 9. Path of the box in an example trial from Experiment Set 3 (units are meters). The right `pusher` fails, leaving the other `pusher` to push either end of the box in turn.

Experiment Set 3 (see Fig. 9) took longer still because, before the healthy `pusher` could take over the task on its own, the `watcher` had to recognize that the other robot was "stuck" and declare it unfit to participate. Exactly how long the `watcher` should wait is a system parameter that represents a tradeoff between good dynamic response and the chance of falsely declaring a fault. The average completion time for Experiment Set 4 was less than for the other two failure modes, which suggests that the overhead required for coordination is outweighed by the improvement in performance from the robot's reintegration to the team. However, the large standard deviations preclude stronger comparisons. The magnitude of the standard deviations in the failure cases is explained intuitively by the complexity of the system; as the situation becomes more complicated, the exact behavior of the robots is less repeatable, due to the numerous interacting dynamic processes (e.g., variable torque output from motors, friction between the box and the floor, etc.).

We note that, in Experiment Set 4, after the failed robot recovered and was reintroduced, the `pushers` switched sides when appropriate, which was in half the trials. The appropriateness of switching was determined by the configuration of the box and the remaining `pusher` at the time of reintroduction, and this

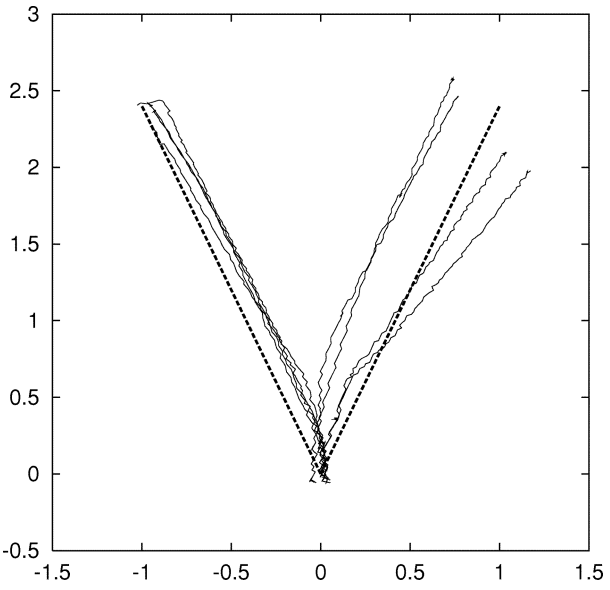

Fig. 10. Path of the box in trials from Experiment Set 5 (units are meters). Shown here is the trajectory of the center of the box for four successful trials in either direction. For comparison, the dashed lines represent the "ideal" paths.

configuration was, in turn, a result of the complex system dynamics mentioned above. However, the fact that the pushers automatically switched sides at the right times, with no detriment to performance, demonstrates that our task-allocation system performs as specified.

In addition to verifying the *validity* of the task assignments made by MURDOCH, we also want to know their *quality*. We evaluate the `pusher-watcher` system in this respect by comparing the measured trajectories of the box to an "ideal" trajectory. For this comparison, we define the ideal trajectory as the one which causes the center of the box to follow the shortest path to the goal. This path is simply the straight line from the initial location of the box to the goal. However, this trajectory is unrealistic because it would require that the box be rotated in place at the start, a feat of which the physical robots are not capable. Shown in Fig. 10 are measured and ideal box trajectories for trials in Experiment Set 5. We can see from this data that the `pusher-watcher` system generates efficient paths that are close to the (unachievable) ideal.

VIII. Conclusion

We have presented a novel method of dynamic task allocation for multirobot systems, based on the CNP [5]. To evaluate our approach, we have implemented the task-allocation system MURDOCH, based on a principled publish/subscribe messaging model. In this model, all interrobot communication is necessarily anonymous and resource centric. We tested MURDOCH on physical robots in both a long-term loosely coupled task domain and a short-term tightly coupled box-pushing task. We demonstrated that the system is extremely reactive to changes in the environment, including abrupt failures of robots and random introduction of new tasks. The primary contribution of this paper is the empirical demonstration that distributed negotiation mechanisms such as MURDOCH are effective in coordinating physical multirobot systems. Such systems are, as a rule, complex and difficult to coordinate. MURDOCH simplifies this problem by automating task allocation in a resource-efficient manner. The system is distributed, with no single point of congestion or failure, making it particularly well suited to multirobot coordination.

We are continuing the development of this task-allocation system. In addition to applying MURDOCH to other domains, we are exploring algorithms for allocating tasks in environments in which there is not a robot-level resource abstraction. For example, if we want to track some phenomenon (such as a person walking) throughout a building instrumented with sensors, the intuitive solution is to pose a single `track` task to the network. The sensors should then dynamically form teams and make collective bids for the task. We are interested in methods for guiding the formation of such teams.

Acknowledgment

The authors thank R. Vaughan, A. Fod, D. Goldberg, A. Howard, and the anonymous reviewers for insightful comments.

References

[1] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss, "Self organizing robots based on cell structures—CEBOT," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Nov. 1988, pp. 145–150.

[2] J.-L. Deneubourg, G. Theraulaz, and R. Beckers, "Swarm-made architectures," in *Proc. Euro. Conf. Artificial Life (ECAL)*, Paris, France, 1991, pp. 123–133.

[3] L. E. Parker, "ALLIANCE: An architecture for fault-tolerant multirobot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 220–240, Apr. 1998.

[4] E. H. Østergård, M. J. Matarić, and G. S. Sukhatme, "Distributed multirobot task allocation for emergency handling," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Wailea, HI, Oct. 2001, pp. 821–826.

[5] R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed problem solving," *Artif. Intell.*, vol. 20, pp. 63–109, 1983.

[6] D. Chapman, "Planning for conjunctive goals," *Artif. Intell.*, vol. 32, pp. 333–377, 1987.

[7] T. Sandholm, "An implementation of the contract net protocol based on marginal cost calculations," in *Proc. Nat. Conf. Artificial Intelligence (AAAI)*, Washington, DC, 1993, pp. 256–262.

[8] C. Ramos, "A holonic approach for task scheduling in manufacturing systems," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Minneapolis, MN, Apr. 1996, pp. 2511–2516.

[9] M. Golfarelli, D. Maio, and S. Rizzi, "A task-swap negotiation protocol based on the Contract Net Paradigm," DEIS, CSITE—Universitá di Bologna, Italy, Tech. Rep. 005-97, 1997.

[10] D. L. Martin, A. J. Cheyer, and D. B. Moran, "The open agent architecture: A framework for building distributed software systems," *Appl. Artif. Intell.*, vol. 13, no. 1, pp. 91–128, Jan.–Mar. 1999.

[11] K. Sycara, K. Decker, A. Pannu, and M. Williamson *et al.*, "Distributed intelligent agents," *IEEE Expert*, vol. 11, pp. 36–46, Dec. 1996.

[12] B. P. Gerkey and M. J. Matarić, "Principled communication for dynamic multi-robot task allocation," in *Experimental Robotics VII, LNCIS 271*, D. Rus and S. Singh, Eds. Berlin, Germany: Springer-Verlag, 2001, pp. 353–362.

[13] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor—A hunter of idle workstations," in *Proc. Int. Conf. Distributed Computing Systems*, Washington, DC, 1988, pp. 104–111.

[14] A. Chavez, A. Moukas, and P. Maes, "Challenger: A multi-agent system for distributed resource allocation," in *Proc. Autonomous Agents*, Marina del Rey, CA, Feb. 1997, pp. 323–331.

[15] M. B. Dias and A. Stentz, "A free market architecture for distributed control of a multirobot system," in *Proc. 6th Int. Conf. Intelligent Autonomous Systems*, Venice, Italy, July 2000, pp. 115–122.

[16] S. Thayer, B. Digney, M. B. Dias, and A. Stentz *et al.*, "Distributed robotic mapping of extreme environments," in *Proc. SPIE*, vol. 4195, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII, Nov. 2000.
[17] R. C. Arkin, *Behavior-Based Robotics*. Cambridge, MA: MIT Press, 1998.
[18] M. J. Matarić, "Behavior-based control: Examples from navigation, learning and group behavior," *J. Experim. Theoret. Artif. Intell.*, vol. 9, no. 2–3, pp. 323–336, 1997.
[19] L. E. Parker, "ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Munich, Germany, Sept. 1994, pp. 776–783.
[20] ——, "A case study for life-long learning and adaptation in cooperative robot teams," in *Proc. SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, vol. 3839, 1999, pp. 92–101.
[21] B. B. Werger and M. J. Matarić, "Broadcast of local eligibility for multi-target observation," in *Distributed Autonomous Robotic Systems 4*, L. E. Parker, G. Bekey, and J. Barhen, Eds. New York: Springer-Verlag, 2000, pp. 347–356.
[22] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Automat.*, vol. RA–2, pp. 14–23, Mar. 1986.
[23] C. R. Kube and H. Zhang, "Collective robotic intelligence," in *Proc. Int. Conf. Simulation of Adaptive Behavior (SAB)*, 1992, pp. 460–468.
[24] ——, "The use of perceptual cues in multirobot box-pushing," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Minneapolis, MN, Apr. 1996, pp. 2085–2090.
[25] F. R. Noreils, "Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment," *Int. J. Robot. Res.*, vol. 12, no. 1, pp. 79–98, 1993.
[26] B. Donald, J. Jennings, and D. Rus, "Information invariants for distributed manipulation," *Int. J. Robot. Res.*, vol. 16, no. 5, pp. 673–702, Oct. 1997.
[27] M. J. Matarić, M. Nilsson, and K. T. Simsarian, "Cooperative multirobot box-pushing," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Pittsburgh, PA, Aug. 1995, pp. 556–561.
[28] R. Emery and T. Balch, "Behavior-based control of a nonholonomic robot in pushing tasks," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Seoul, South Korea, 2001, pp. 2381–2388.
[29] K. Erol, J. Hendler, and D. S. Nau, "UCMP: A sound and complete procedure for hierarchical task-network planning," in *Proc. Int. Conf. Artificial Intelligence Planning Systems*, Chicago, IL, June 1994, pp. 249–254.
[30] R. P. McAfee and J. McMillan, "Auctions and bidding," *J. Econ. Lit.*, vol. 25, no. 2, pp. 699–738, June 1987.
[31] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching events in a content-based subscription system," in *Proc. ACM Symp. Principles of Distributed Computing*, Atlanta, GA, May 1999, pp. 53–61.
[32] J. Pereira, F. Fabret, F. Llirbat, and D. Shasha, "Efficient matching for web-based publish/subscribe systems," in *Proc. Int. Conf. Cooperative Information Systems*, Eiat, Israel, Sept. 2000, pp. 162–173.
[33] D. D. Clark, "The design philosophy of the DARPA internet protocols," *Comput. Commun. Rev.*, vol. 18, no. 4, pp. 106–114, Aug. 1988.
[34] B. P. Gerkey, R. T. Vaughan, K. Støy, A. Howard, G. S. Sukhtame, and M. J. Matarić, "Most valuable player: A robot device server for distributed control," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Wailea, HI, Oct. 2001, pp. 1226–1231.
[35] B. P. Gerkey and M. J. Matarić, "Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, Washington, DC, May 2002, pp. 464–469.

Brian P. Gerkey received the B.S.E. degree in computer engineering (*Magna Cum Laude*, departmental honors) from Tulane University, New Orleans, LA, in 1998. He received the M.S. degree in computer science from the University of Southern California (USC), Los Angeles, in 2000. He is currently working toward the Ph.D. degree at USC.

He has been a Research Assistant in the USC Robotics Research Lab since 1998. In addition to his doctoral studies, he has worked in the Artificial Intelligence group at the NASA Jet Propulsion Lab. He is also a founding and lead developer on the open-source Player/Stage project, which produces Player, a networked robot device server, and Stage, a scalable multirobot simulator. His current research focuses on analyzing fundamental aspects of multirobot task allocation and developing novel approaches to the problem.

Maja J. Matarić received the Ph.D. degree in computer science and artificial intelligence from Massachusetts Institute of Technology (MIT), Cambridge, in 1994, the M.S. degree in computer science from MIT in 1990, and the B.S. degree in computer science from the University of Kansas, Lawrence, in 1987.

She is an Associate Professor in the Computer Science Department and the Neuroscience Program at the University of Southern California (USC), Los Angeles, the Director of the USC Robotics Research Lab and the Interaction Lab, and an Associate Director of Institute for Robotics and Intelligent Systems (IRIS). She has published over 30 journal articles, 13 book chapters, 66 conference papers, and 20 workshop papers, and has two books in the works with MIT Press. She has worked with NASA's Jet Propulsion Lab, the Free University of Brussels AI Lab, LEGO Cambridge Research Labs, GTE Research Labs, the Swedish Institute of Computer Science, and ATR Human Information Processing Laboratories. Her research is in the areas of control and learning in behavior-based multirobot systems and humanoids, and skill learning by imitation.

Dr. Matarić is a recipient of the NSF Career Award, the IEEE Robotics and Automation Society Early Career Award, the MIT TR100 Innovation Award, and the USC School of Engineering Junior Research Award. She is on the editorial board of three journals: the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, the *International Journal of Autonomous Agents and Multi-Agent Systems*, and *Adaptive Behavior*. She is on the AAAI Executive Council.