CONF-9409306--1

ALLIANCE: An Architecture for Fault Tolerant, Cooperative Control of Heterogeneous Mobile Robots*

Lynne E. Parker
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, TN 37831-6364

To be published in the Proceedings of the IROS '94 (IEEE/RSJ/GI International Conference on Intelligent Robots and Systems), Munich, Germany, September 12–16, 1994

* Research provided in part by the University Research Initiative under the Office of Naval Research contract N00014-86-K-0685, in part by the Advanced Research Projects Agency under the Office of Naval Research contract N00014-85-K-0124, and in part by the Mazda Coportation. Additional support has been provided by the Office of Engineering Research Program, Office of Basic Energy Sciences of the U.S. Department of Energy, under contract No. DE-AC05-84OR 21400 with Martin Marietta Energy Systems, Inc.

RWR

ALLIANCE: An Architecture for Fault Tolerant, Cooperative Control of Heterogeneous Mobile Robots

Lynne E. Parker
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P. O. Box 2008
Oak Ridge, TN 37831-6364 USA
internet: epk@ornl.gov or ParkerLE@ornl.gov

Abstract

This research addresses the problem of achieving fault tolerant cooperation within small- to medium-sized teams of heterogeneous mobile robots. We describe a novel behavior-based, fully distributed architecture, called ALLIANCE, that utilizes adaptive action selection to achieve fault tolerant cooperative control in robot missions involving loosely coupled, largely independent tasks. The robots in this architecture possess a variety of high-level functions that they can perform during a mission, and must at all times select an appropriate action based on the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. Since such cooperative teams often work in dynamic and unpredictable environments, the software architecture allows the team members to respond robustly and reliably to unexpected environmental changes and modifications in the robot team that may occur due to mechanical failure, the learning of new skills, or the addition or removal of robots from the team by human intervention. After presenting ALLIANCE, we describe in detail our experimental results of an implementation of this architecture on a team of physical mobile robots performing a cooperative box pushing demonstration. These experiments illustrate the ability of ALLIANCE to achieve adaptive, fault-tolerant cooperative control amidst dynamic changes in the capabilities of the robot team.

1 Introduction

Achieving cooperative robotics is desirable for a number of reasons. First, many robotic applications are inherently distributed in space, time, or functionality, thus requiring a distributed solution. Second, it is quite possible that many applications could be solved much more quickly if the mission could be divided across a number of robots operating in parallel. Third, by duplicating capabilities across robot team members, one has the potential of increasing the robustness and reliability of the automated solution through redundancy. Finally, it may actually be much cheaper and more practical in many applications to build a number of less capable robots that can work together at a mission, rather than trying to build one robot which can perform the entire mission with adequate reliability.

Achieving cooperative robotics, however, is quite challenging. Many issues must be addressed in order to develop a working cooperative team, such as action selection, coherence, conflict resolution, and communication. Furthermore, these cooperative teams often work in dynamic and unpredictable environments, requiring the robot team members to respond robustly, reliably, and adaptively to unexpected environmental changes, failures in the inter-robot communication system, and modifications in the robot team that may occur due to mechanical failure, the learning of new skills, or the addition or removal of robots from the team by human intervention.

Previous research in heterogeneous mobile robot cooperation includes: [8], which proposes a three-layered control architecture that includes a planner level, a control level, and a functional level; [5], which describes an architecture that includes a task planner, a task allocator, a motion planner, and an execution monitor; [1], which describes an architecture called ACTRESS that utilizes a negotiation framework to allow robots to recruit help when needed; and [6], which uses a hierarchical division of authority to address the problem of cooperative fire-fighting. However, these approaches deal primarily with the task selection problem and largely ignore the difficult issues for physical robot teams, such as robot failure, communication noise, and dynamic environments. In contrast, our research emphasizes the need for fault tolerant and adaptive cooperative control as a principal characteristic of the cooperative control architecture.

This paper describes an architecture that we have built for heterogeneous mobile robot control that emphasizes fault tolerant, adaptive cooperation. This architecture, called ALLIANCE, is designed for small- to medium-sized teams of robots performing missions composed of loosely coupled subtasks that are largely independent of each other. By "largely" independent, we mean that tasks can have fixed ordering dependencies, but they cannot be of the type of "brother clobbers brother" [12], where the execution of one task undoes the effects of another task. Even with this restriction, however, this research covers a very large range of missions for which cooperative robots are useful. In [11], we report on a wide variety of applications that have been implemented on both physical and simulated robot teams that fall into this domain of loosely coupled, largely independent subtasks; we present one of these implementations in this paper.

Section 2 describes our cooperative architecture, first giving an overview for how we achieve fault tolerant, adaptive control, and then providing details on the operation of our primary control mechanism — the motivational behav-

ior. We then describe, in section 3, the implementation of ALLIANCE on a physical robot team performing a cooperative box pushing mission. In section 4, we offer some concluding remarks.

2 The ALLIANCE Architecture

ALLIANCE is a fully distributed architecture for fault tolerant, heterogeneous robot cooperation that utilizes adaptive action selection to achieve cooperative control. Under this architecture, the robots possess a variety of high-level task-achieving functions that they can perform during a mission, and must at all times select an appropriate action based on the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. In ALLIANCE, individual robots are designed using a behavior-based approach [2]. Under the behavior-based construction, a number of task-achieving behaviors are active simultaneously, each receiving sensory input and controlling some aspect of the actuator output. The lower-level behaviors, or competences, correspond to primitive survival behaviors such as obstacle avoidance, while the higher-level behaviors correspond to higher goals such as map building and exploring. The output of the lower-level behaviors can be *suppressed* or *inhibited* by the upper layers when the upper layers deem it necessary. This approach has been used successfully in a number of robotic applications, several of which are described in [4].

Extensions to this approach are necessary, however, when a robot must select among a number of competing actions — actions which cannot be pursued in parallel. Unlike typical behavior-based approaches, ALLIANCE delineates several "behavior sets" that are either active as a group or hibernating. Figure 1 shows the general architecture of ALLIANCE and illustrates three such behavior sets. The jth behavior set, a_{ij}, of a robot r_i corresponds to those levels of competence required to perform some high-level task-achieving function, such as finding a toxic waste spill, moving a toxic waste spill, or reporting the progress of the robot team to a human monitor. When a robot activates a behavior set, we say that it has selected the task corresponding to that behavior set. Since different robots may have different ways of performing the same task, and will therefore activate different behavior sets to perform that task, we define the function $h_i(a_{ij})$, for all robots, r_i, on the team, to refer to the task that robot r_i is working on when it activates its j-th behavior set.

Because of the alternative goals that may be pursued by the robots, the robots must have some means of selecting the appropriate behavior set to activate. Thus, controlling the activation of each of these behavior sets is a *motivational behavior*. Due to conflicting goals, only one behavior set per robot should be active at any point in time. This restriction is implemented via cross-inhibition of motivational behaviors, represented by the arcs at the top of figure 1, in which the activation of one behavior set suppresses the activation of all other behavior sets. However, other lower-level competences such as collision avoidance may be continually active regardless of the high-level goal the robot is currently pursuing. Examples of this type of continually active competence are shown in figure 1 as layer 0, layer 1, and layer 2.

2.1 Motivational Behaviors: Overview

The primary mechanism for achieving adaptive action selection in this architecture is the *motivational behavior*.

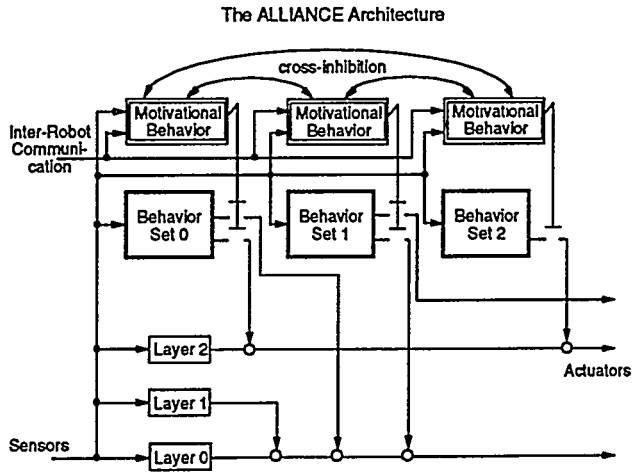

Figure 1: The ALLIANCE architecture. The symbols in this figure that connect the output of each motivational behavior with the output of its corresponding behavior set (vertical lines with short horizontal bars) indicate that a motivational behavior either allows all or none of the outputs of its behavior set to pass through to the robot's actuators.

At all times during the mission, each motivational behavior receives input from a number of sources, including sensory feedback, inter-robot communication, inhibitory feedback from other active behaviors, and internal motivations called *robot impatience* and *robot acquiescence*. The output of a motivational behavior is the activation level of its corresponding behavior set, represented as a non-negative number. When this activation level exceeds a given threshold, the corresponding behavior set becomes active.

Intuitively, a motivational behavior works as follows. Robot r_i's motivation to activate any given behavior set a_{ij} is initialized to 0. Then, over time, robot r_i's motivation $m_{ij}(t)$ to activate behavior set a_{ij} increases at a fast rate as long as the task corresponding to that behavior set (i.e. $h_i(a_{ij})$) is not being accomplished, as determined from sensory feedback. However, we want the robots to be responsive to the actions of other robots, adapting their task selection to the activities of team members. Thus, if a robot r_i is aware that another robot r_k is working on task $h_i(a_{ij})$, then r_i should be satisfied for some period of time that the task is going to be accomplished even without its own participation, and thus go on to some other applicable action. Its motivation to activate behavior set a_{ij} still increases, but at a slower rate. This characteristic prevents robots from replicating each other's actions and thus wasting needless energy. Of course, detecting and interpreting the actions of other robots (often called *action recognition*) is not a trivial problem, and often requires perceptual abilities that are not yet possible with current sensing technology. As it stands today, the sensory capabilities of even the lower animals far exceed present robotic capabilities. Thus, to enhance the robots' perceptual abilities, ALLIANCE utilizes a simple form of broadcast communication to allow robots to inform other team members of their current activities, rather than relying totally on sensory capabilities. At some pre-specified rate, each robot r_i broadcasts a statement of

its current action, which other robots may listen to or ignore as they wish. No two-way conversations are employed in this architecture.

Each robot is designed to be somewhat impatient, however, in that a robot r_i is only willing for a certain period of time to allow the communicated messages of another robot to affect its own motivation to activate a given behavior set. Continued sensory feedback indicating that a task is not getting accomplished thus overrides the statements of another robot that it is performing that task. This characteristic allows robots to adapt to failures of other robots, causing them to ignore the activities of a robot that is not successfully completing its task.

A complementary characteristic in these robots is that of acquiescence. Just as the impatience characteristic reflects the fact that other robots may fail, the acquiescence characteristic indicates the recognition that a robot itself may fail. This feature operates as follows. As a robot r_i performs a task, its willingness to give up that task increases over time as long as the sensory feedback indicates the task is not being accomplished. As soon as some other robot r_k indicates it has begun that same task and r_i feels it (i.e. r_i) has attempted the task for an adequate period of time, the unsuccessful robot r_i gives up its task in an attempt to find an action at which it is more productive. Additionally, even if another robot r_k has not taken over the task, robot r_i may give up its task anyway if it is not completed in an acceptable period of time. This allows r_i the possibility of working on another task that may prove to be more productive rather than becoming stuck performing the unproductive task forever. With this acquiescence characteristic, therefore, a robot is able to adapt its actions to its own failures.

The behavior-based design of the motivational behaviors also allows the robots to adapt to unexpected environmental changes which alter the sensory feedback. The need for additional tasks can suddenly occur, requiring the robots to perform additional work, or existing environmental conditions can disappear and thus relieve the robots of certain tasks. In either case, the motivations fluidly adapt to these situations, causing robots to respond appropriately to the current environmental circumstances.

2.2 Motivational Behaviors: Formal Model

Having presented the basic philosophy behind the ALLIANCE architecture, we now look in detail at how this philosophy is incorporated into the motivational behavior mechanism by presenting a formal model of the motivational behavior. As we discuss these details, we introduce a number of parameters that are incorporated into ALLIANCE. An extension to the ALLIANCE architecture, called L-ALLIANCE (for *L*earning ALLIANCE), allows these parameters to be updated automatically through the use of knowledge learned about team member capabilities. This dynamic parameter update mechanism relieves the human of the tedium of parameter adjustments and allows the heterogeneous robot team members to select their tasks quite efficiently. Refer to [11] for details on the L-ALLIANCE mechanism. We also note that all of our implementations of this model have used the features of the Behavior Language [3], for both physical robot teams and simulated robot teams.

In the following subsections, we first discuss the threshold of activation of a behavior set, and then describe the five primary inputs to the motivational behavior. We conclude this section by showing how these inputs are combined to determine the current level of motivation of a given behavior set in a given robot.

2.2.1 Threshold of activation

The threshold of activation is given by one parameter, θ. This parameter determines the level of motivation beyond which a given behavior set will become active. Although different thresholds of activation could be used for different behavior sets and for different robots, in ALLIANCE we assume that one threshold is sufficient since the rates of impatience and acquiescence can vary across behavior sets and across robots.

2.2.2 Sensory feedback

The sensory feedback provides the motivational behavior with the information necessary to determine whether its corresponding behavior set needs to be activated at a given point during the current mission. Although this sensory feedback usually comes from physical robot sensors, in realistic robot applications it is not always possible to have a robot sense the applicability of tasks through the world — that is, through its sensors. Often, tasks are information-gathering types of activities whose need is indicated by the values of programmed state variables. These state variables, therefore, act as a type of *virtual* sensor which serves some of the same purposes as a physical sensor.

We define a simple function to capture the notion of sensory feedback as follows:

$$sensory\_feedback_{i,j}(t) = \begin{cases} 1 & \text{if the sensory feedback in robot } r_i \text{ at time } t \\ & \text{indicates that behavior set } a_{ij} \text{ is applicable} \\ 0 & \text{otherwise} \end{cases}$$

2.2.3 Inter-robot communication

The inter-robot broadcast communication mechanism utilized in ALLIANCE serves a key role in allowing robots to determine the current actions of their teammates. As we noted previously, the broadcast messages in ALLIANCE substitute for passive action recognition, which is quite difficult to achieve. Two parameters control the broadcast communication among robots: ρ_i and τ_i. The first parameter, ρ_i, gives the rate at which robot r_i broadcasts its current activity. The second parameter, τ_i, provides an additional level of fault tolerance by giving the period of time robot r_i allows to pass without receiving a communication message from a specific teammate before deciding that that teammate has ceased to function. While monitoring the communication messages, each motivational behavior of the robot must also note when a team member is pursuing a task corresponding to that motivational behavior's behavior set. To refer to this type of monitoring in our formal model, we define the function *comm_received* as follows:

$$comm\_received(i,k,j,t_1,t_2) = \begin{cases} 1 & \text{if robot } r_i \text{ has received message from} \\ & \text{robot } r_k \text{ related to behavior set } a_{ij} \text{ in the} \\ & \text{time span } (t_1,t_2), \text{ where } t_1 < t_2 \\ 0 & \text{otherwise} \end{cases}$$

2.2.4 Suppression from active behavior sets

When a motivational behavior activates its behavior set, it simultaneously begins inhibiting other motivational behaviors from activating their respective behavior sets. At

this point, a robot has effectively "selected an action". The first motivational behavior then continues to monitor the sensory feedback, the communication from other robots, and the levels of impatience and acquiescence to determine the continued need for the activated behavior set. At some point in time, either the robot completes its current task, thus causing the sensory feedback to no longer indicate the need for that action, or the robot acquiesces its task. In either case, the need for the activated behavior set eventually goes away, causing the corresponding motivational behavior to inactivate this behavior set. This, in turn, allows another motivational behavior within that robot the opportunity to activate its behavior set.

We refer to the suppression from action behavior sets with the following function:

$$activity\_suppression_{ij}(t) = \begin{cases} 0 & \text{if another behavior set } a_{ik} \text{ is active, } k \neq j, \\ & \text{on robot } r_i \text{ at time } t \\ 1 & \text{otherwise} \end{cases}$$

This function says that behavior set a_{ij} is being suppressed at time t on robot r_i if some other behavior set a_{ik} is currently active on robot r_i at time t.

2.2.5 Robot impatience

Three parameters are used to implement the robot impatience feature of ALLIANCE: $\phi_{ij}(k,t)$, $\delta\_slow_{ij}(k,t)$, and $\delta\_fast_{ij}(t)$. The first parameter, $\phi_{ij}(k,t)$, gives the time during which robot r_i is willing to allow robot r_k's communication message to affect the motivation of behavior set a_{ij}. Note that robot r_i is allowed to have different ϕ parameters for each robot on its team, and that the parameters can change during the mission (indicated by the dependence on t). This allows r_i to be influenced more by some robots than others, if necessary, and for this influence to be updated as robot capabilities change.

The next two parameters, $\delta\_slow_{ij}(k,t)$ and $\delta\_fast_{ij}(t)$, give the rates of impatience of robot r_i concerning behavior set a_{ij} either while robot r_k is performing task $h_i(a_{ij})$ or in the absence of other robots performing this task, respectively. We assume that the fast impatience parameter corresponds to a higher rate of impatience than the slow impatience parameter for a given behavior set in a given robot. The reasoning for this assumption should be clear — a robot r_i should allow another robot r_k the opportunity to accomplish its task before becoming impatient with r_k; however, there is no reason for r_i to remain idle if a task remains undone and no other robot is attempting that task.

The question that now arises is: what slow rate of impatience does a motivational behavior controlling behavior set a_{ij} use when more than one other robot is performing task $h_i(a_{ij})$? The method used in ALLIANCE is to increase the motivation at a rate that allows the slowest robot still under its allowable time $\phi_{ij}(k,t)$ to continue its attempt.

The specification of the current impatience rate for a behavior set a_{ij} is given by the following function:

$$impatience_{ij}(t) = \begin{cases} \min_k(\delta\_slow_{ij}(k,t)) & \text{if } (comm\_received(i,k,j, \\ & t-\tau_i,t)=1) \\ & \text{and} \\ & (comm\_received(i,k,j,0, \\ & t-\phi_{ij}(k,t))=0) \\ \delta\_fast_{ij}(t) & \text{otherwise} \end{cases}$$

This function says that the impatience rate will be the minimum slow rate, $\delta\_slow_{ij}(k,t)$, if robot r_i has received communication indicating that robot r_k is performing task $h_i(a_{ij})$ in the last τ_i time units, but not for longer than $\phi_{ij}(k,t)$ time units. Otherwise, the impatience rate is set to $\delta\_fast_{ij}(t)$.

The final detail to be addressed is to cause a robot's motivation to activate behavior set a_{ij} to go to 0 the first time it hears about another robot performing task $h_i(a_{ij})$. This is accomplished through the following:

$$impatience\_reset_{ij}(t) = \begin{cases} 0 & \text{if } \exists k.((comm\_received(i,k,j,t-\delta t,t)=1) \text{ and} \\ & (comm\_received(i,k,j,0,t-\delta t)=0)), where \\ & \delta t = \text{time since last communication check} \\ 1 & \text{otherwise} \end{cases}$$

This reset function causes the motivation to be reset to 0 if robot r_i has just received its first message from robot r_k indicating that r_k is performing task $h_i(a_{ij})$. This function allows the motivation to be reset no more than once for every robot team member that attempts that task. Allowing the motivation to be reset repeatedly would allow a persistent, yet failing robot to jeopardize the completion of the mission.

2.2.6 Robot acquiescence

Two parameters are used to implement the robot acquiescence characteristic of ALLIANCE: $\psi_{ij}(t)$ and $\lambda_{ij}(t)$. The first parameter, $\psi_{ij}(t)$, gives the time that robot r_i wants to maintain behavior set a_{ij} activation before yielding to another robot. The second parameter, $\lambda_{ij}(t)$, gives the time robot r_i wants to maintain behavior set a_{ij} activation before giving up to possibly try another behavior set.

The following *acquiescence* function indicates when a robot has decided to acquiesce its task:

$$acquiescence_{ij}(t) = \begin{cases} 0 & \text{if ((behavior set } a_{ij} \text{ of robot } r_i \text{ has been active} \\ & \text{for more than } \psi_{ij}(t) \text{ time units at time } t) \text{ and} \\ & (\exists x.comm\_received(i,x,j,t-\tau_i,t)=1)) \\ & \text{or} \\ & \text{(behavior set } a_{ij} \text{ of robot } r_i \text{ has been active} \\ & \text{for more than } \lambda_{ij}(t) \text{ time units at time } t) \\ 1 & \text{otherwise} \end{cases}$$

This function says that a robot r_i will not acquiesce behavior set a_{ij} until one of the following conditions is met:

- r_i has worked on task $h_i(a_{ij})$ for a length of time $\psi_{ij}(t)$ and some other robot has taken over task $h_i(a_{ij})$
- r_i has worked on task $h_i(a_{ij})$ for a length of time $\lambda_{ij}(t)$

2.2.7 Motivation calculation

We now combine all of the inputs described above into the calculation of the levels of motivation as follows:

$$\begin{aligned} m_{ij}(0) &= 0 \\ m_{ij}(t) &= [m_{ij}(t-1) + impatience_{ij}(t)] \\ &\quad \times activity\_suppression_{ij}(t) \\ &\quad \times sensory\_feedback_{ij}(t) \\ &\quad \times acquiescence_{ij}(t) \\ &\quad \times impatience\_reset_{ij}(t) \end{aligned}$$

Initially, the motivation to perform behavior set a_{ij} in robot r_i is set to 0. This motivation then increases at some positive rate $impatience_{ij}(t)$ unless one of four situations occurs: (1) another behavior set in r_i activates, (2) the sensory feedback indicates that the behavior set is no longer needed, (3) the robot has decided to acquiesce the task, or (4) some other robot has just taken over task $h_i(a_{ij})$ for the first time. In any of these four situations, the motivation returns to 0. Otherwise, the motivation grows until it crosses the threshold θ, at which time the behavior set is activated, and the robot can be said to have selected an action. Whenever some behavior set a_{ij} is active in robot r_i, r_i broadcasts its current activity to other robots at a rate of ρ_i.

3 Robot Box Pushing Experiments

The ALLIANCE architecture has been successfully implemented in a variety of proof of concept applications on both physical and simulated mobile robots. The applications implemented on physical robots include a hazardous waste cleanup mission, reported in [11] and [10], and a cooperative box pushing mission, which is described below. Over 50 logged physical robot runs of the hazardous waste cleanup mission and over 30 physical robot runs of the box pushing mission were completed to elucidate the important issues in heterogeneous robot cooperation. Many runs of each of these physical robot applications are available on videotape. The applications using simulated mobile robots include a janitorial service mission and a bounding overwatch mission (reminiscent of military surveillance), which involved dozens of runs each. Details of these implementations are reported in [11] and [9].

The cooperative box pushing mission offers a simple and straight-forward illustration of a key characteristic of the ALLIANCE architecture: fault tolerant and adaptive control due to dynamic changes in the robot team. This box pushing mission requires a long box to be pushed across a room; the box is sufficiently heavy and long that one robot cannot push in the middle of the box to move it across the room. Thus, the box must be pushed at both ends in order to accomplish this mission. To synchronize the pushing at the two ends, the mission is defined in terms of two recurring tasks — (1) push a little on the left end, and (2) push a little on the right end — neither of which can be activated (except for the first time) unless the opposite side has just been pushed. We use this mission to demonstrate how the ALLIANCE architecture endows robot team members with fault tolerant action selection due to the failure of robot team members, and with adaptive action selection due to the heterogeneity of the robot team. Note that our emphasis in these experiments is on issues of fault tolerant cooperation rather than the design of the ultimate box pusher. Thus, we are not concerned at present with issues such as robots pushing the box into a corner, obstacles interfering with the robots, how robots detect box alignment, and so forth.

In the next subsection, we describe the robots used in these experiments and the design of the robot control software. We then present and discuss the results of this proof of concept implementation.

3.1 Physical Robot Team

The box pushing application was implemented using three mobile robots of two types — two R-2 robots and one Genghis-II. All of these robots were designed and manufactured at IS Robotics, Inc., of Cambridge, Massachusetts. The first type of robot, the R-2, has two drive wheels arranged as a differential pair, and a two-degree-of-freedom gripper for grasping objects. Its sensor suite includes eight infrared sensors and seven bump sensors evenly distributed around the front, sides, and back of the robot. In addition, a break-beam infrared sensor between the gripper and a bump sensor lining the inside of the fingers facilitate the grasping of small objects. The second type of robot, Genghis-II, is a legged robot with six two-degree-of-freedom legs. Its sensor suite includes two whiskers, force detectors on each leg, a passive array of infrared heat sensors, three tactile sensors along the robot belly, four near-infrared sensors, and an inclinometer for measuring the pitch of the robot.

A radio communication system [7] is used in our physical robot implementations to allow the robots to communicate their current actions to each other. This system consists of a radio modem attached to each robot, plus a base station that is responsible for preventing message interference by time-slicing the radio channel among robots. The design of the radio system limits the frequency of messages between robots to only one message every three seconds. All of the results described below, therefore, involve communication between robots at no more than about $\frac{1}{3}$ Hertz.

3.2 Robot Software Design

Since the capabilities of the R-2 and Genghis-II robots differ, the software design of the box pushing mission for these robots varies somewhat. We therefore describe the ALLIANCE box pushing software of these robots separately.

3.2.1 R-2 Control

Figure 2 shows the ALLIANCE implementation of the box pushing mission for the R-2 robots. As shown in this figure, the R-2 is controlled by two behavior sets — one for pushing a little on the left end of the box (called *push-left*), and one for pushing a little on the right end of the box (called *push-right*). As specified by the ALLIANCE architecture, the activation of each of these behavior sets is controlled by a motivational behavior. Let us now examine the design of the *push-left* motivational behavior and the *push-left* behavior set of a robot r_i in more detail; the *push-right* design is symmetric to that of *push-left*.

The sensory feedback required before the *push-left* motivational behavior within r_i can activate its behavior set is an indication that the right end of the box has just been pushed. This requirement is indicated in figure 2 by the *pushed-at-right* arrow entering the *push-left* motivational behavior. The right end of the box can be pushed either by some robot other than r_i, or it can be pushed by r_i itself. If r_i is the robot doing the pushing, then the *pushed-at-right* feedback comes from an internal message from r_i's *push-right* motivational behavior. However, if some robot other than r_i is pushing, then r_i must detect when that other robot has completed its push. Since this detection is impossible for the R-2s with their current sensory suites, the robots are provided with this capability by having the team members broadcast a message after each push that indicates the completion of their current push. The pushing is initiated at the beginning of the mission by programming the control code so that each robot "thinks" that the opposite end of the box has just been pushed.

When the sensory feedback is satisfied, the *push-left* motivational behavior grows impatient at either a rate δ_fast_R

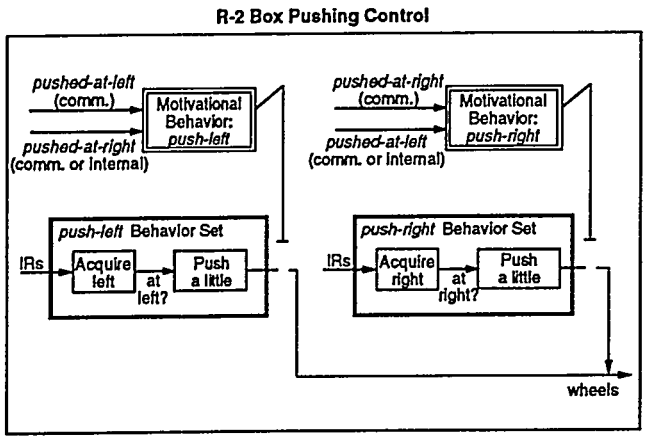

Figure 2: The ALLIANCE design of the R-2 software for the box pushing mission.

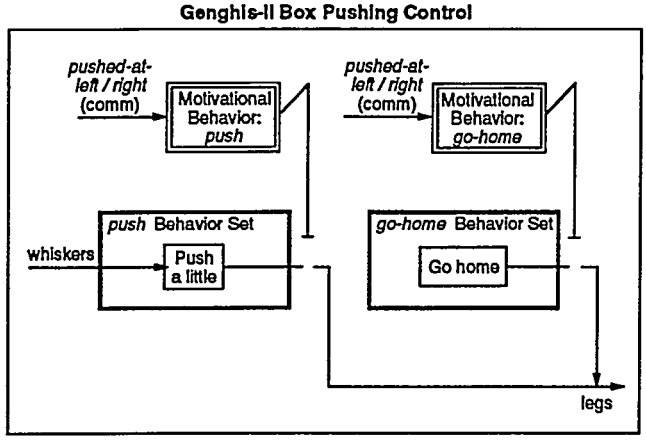

Figure 3: The ALLIANCE design of the Genghis-II software for the box pushing mission.

(the R subscript stands for any R-2 robot) if no other robot is performing the *push-left* task, or at a rate $\delta\_slow_R(robot\text{-}id)$ when robot *robot-id* is performing the *push-left* task.[1] When the *push-left* motivation grows above threshold, the *push-left* behavior set is activated. The *push-left* behavior set involves first acquiring the left end of the box and then pushing a little on that end. If the robot is already at the left end of the box, then no acquiring has to take place. Otherwise, the R-2 assumes it is at the right end of the box, and moves to the left end of the box by using the infrared sensors on its right side to follow the box to the end, and then backing up and turning into the box. As we shall see below, this ability to acquire the opposite end of the box during the mission is important in achieving fault tolerant cooperative control. At the beginning of the mission, we would ideally like the R-2 to be able to locate one end of the box on its own. However, since this is beyond the scope of these proof of concept experiments, an implicit assumption is made in the R-2 control that at the beginning of the mission, the R-2 is facing into a known end of the box.

As the R-2 pushes, it uses the infrared sensors at the ends of its gripper fingers to remain in contact with the box. The current push is considered to be complete when the R-2 has pushed for a prescribed period of time. After the *push-left* task is completed, the motivation to perform that task temporarily returns to 0. However, the motivation begins growing again as soon as the sensory feedback indicates the task is needed.

3.2.2 Genghis-II Control

Genghis-II and the R-2s are different in two primary ways. First, Genghis-II cannot acquire the opposite end of the box, due to a lack of sensory capabilities, and second, Genghis-II cannot push the box as quickly as an R-2, due to less powerful effectors. The first difference means that Genghis-II can only push at its current location. Thus, implicit in the control of Genghis-II is the assumption that it is located at a known end of the box at the beginning of the mission. The second difference with the R-2s implies that if an R-2 pushes with the same duration, speed, and frequency when teamed with Genghis-II as it does when teamed with another R-2, the robot team will have problems accomplishing its mission due to severe box misalignment.

Figure 3 shows the organization of Genghis-II's box pushing software. As this figure shows, Genghis-II is controlled by two behavior sets, each of which is under the control of a motivational behavior. Genghis-II's pushing at its current location is controlled by the *push* behavior set. The only sensory feedback which satisfies the *push* motivational behavior is that which indicates that some other robot is pushing the opposite end of the box. This requirement is shown in figure 3 as the *pushed-at-left/right* arrow going into the *push* motivational behavior. Once the sensory feedback is satisfied, Genghis-II becomes impatient to perform the *push* behavior at a rate δ_fast_{GP} (the G subscript refers to *Genghis-II*; the P subscript refers to the *push* behavior set). Once the motivation crosses the threshold of activation, the *push* behavior set is activated, causing Genghis-II to push the box by walking into it while using its whiskers to maintain contact with the box. Once Genghis-II has pushed a given length of time, the motivation to perform *push* returns to 0, growing again whenever the sensory feedback is satisfied.

The sensory feedback required for the *go-home* behavior set to be activated is the opposite of that required for the *push* behavior set — namely, that no other robot is pushing at the opposite end of the box. When the sensory feedback for *go-home* is satisfied, the motivation to activate *go-home* grows at the rate δ_fast_{GH} (the H subscript refers to the *go-home* behavior set), with the behavior set being activated as soon as the motivation crosses the threshold. The *go-home* behavior set causes Genghis-II to walk away from the box.

3.3 Experiments and Results

To demonstrate the fault tolerant, adaptive nature of the ALLIANCE architecture due to changes in the robot team capabilities, we undertook two basic experiments using the box pushing mission. Both of these experiments began with two R-2s pushing the box — one at each end of the box — as illustrated in figure 4. We note that the fast rates of

[1] To simplify the notation, we omit the j subscript of the fast and slow impatience rates (see section 2.2.5) since the fast rates of impatience are the same for all behavior sets in all R-2s, and the slow rates of impatience are the same functions of *robot-id* for all R-2s. We also omit the dependence upon t of these impatience rates, since we do not deal here with updating these parameters during the mission.

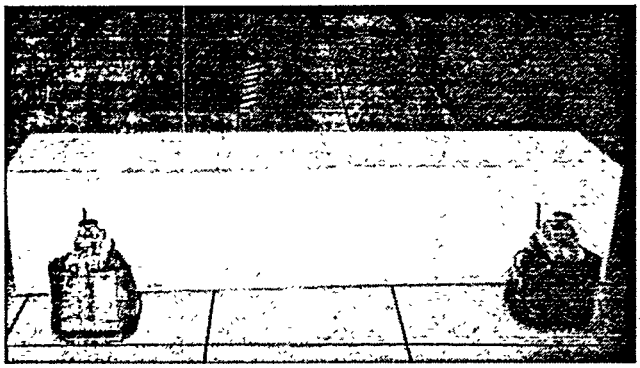

Figure 4: The beginning of the box pushing mission. Two R-2s are pushing the box across the room.

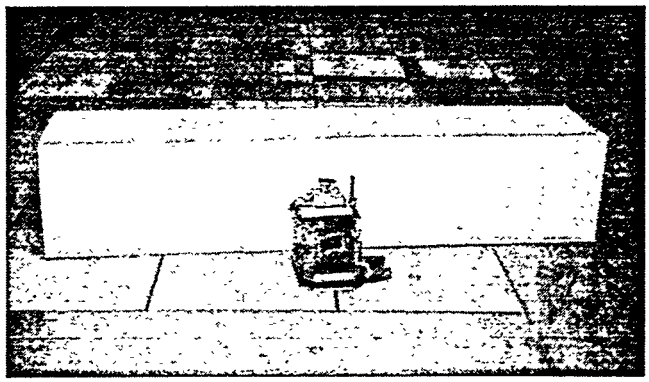

Figure 5: Fault tolerant action selection. In the first experiment, we seize one of the R-2 robots and turn it off. This causes the remaining R-2 robot to have to perform both tasks of the box pushing mission: pushing at the right end of the box, and pushing at the left end of the box. Here, the R-2 is acquiring the right end of the box.

impatience were set such that the delay between individual pushes by each robot is quite small — from imperceptible to about 2 to 3 seconds, depending upon when the $\frac{1}{3}$ Hz communication messages actually get transmitted.

After the two R-2s push the box for a while we dynamically altered the capabilities of the robot team in two ways. In the first experiment, we altered the team by seizing one of the R-2 robots during the mission and turning it off, mimicking a robot failure; we then later added it back into the team. In the second experiment, we again seized one of the R-2 robots, but this time we replaced it with Genghis-II, thus making the team much more heterogeneous; we then later seized the remaining R-2 robot, leaving Genghis-II as the sole team member. The following subsections describe the results of these two experiments.

3.3.1 Experiment 1: Robot "failure"

As we have emphasized, a primary goal of our architecture is to allow robots to recover from failures of robot team members. Thus, by seizing an R-2 and turning it off, we test the ability of the remaining R-2 to respond to that "failure" and adapt its action selection accordingly. In this experiment, what we observe after the seizure is that after a brief pause of about 5 to 8 seconds (which is dependent upon the setting of the δ_slow_R(R-2) parameter), the remaining R-2 begins acquiring the opposite end of the box, as shown in figure 5, and then pushes at its new end of the box. This R-2 continues its back and forth pushing, executing both tasks of pushing the left end of the box and pushing the right end of the box as long as it fails to "hear" through the broadcast communication mechanism that another robot is performing the push at the opposite end of the box. When we add back in the second R-2, however, the still-working robot adapts its actions again, now just pushing one side of the box, since it is satisfied that the other end of the box is also getting pushed. Thus, the robot team demonstrates its ability to recover from the failure of a robot team member.

3.3.2 Experiment 2: Increased heterogeneity

Another goal of our architecture is to allow *heterogeneous* robot teams to work together efficiently. Robots can be heterogeneous in two obvious ways. First, robots may differ in which tasks they are able to accomplish, and second, robots may differ in how well they perform the same task. In this experiment, we deal primarily with the second type of heterogeneity, in which Genghis-II and the R-2 use quite different mechanisms for pushing the box. By substituting

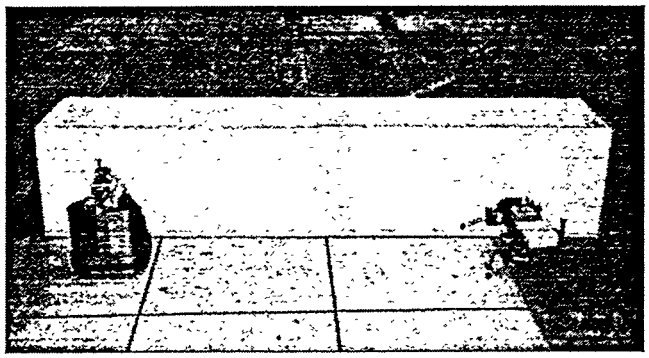

Figure 6: Adaptivity due to heterogeneity. In the second experiment, we again seize one of the R-2 robots, but this time we replace it with Genghis-II. Since Genghis-II cannot push as powerfully as an R-2, the remaining R-2 robot adapts its actions by pushing less frequently.

robots during the middle of a mission, we test the ability of the remaining team member to respond to the dynamic change in the heterogeneity of the team.

What we observe in this experiment is that the remaining R-2 begins pushing much less frequently as soon as it "hears" that Genghis-II, rather than an R-2, is the robot pushing the opposite end of the box. Thus, the robots remain more or less aligned during their pushing. Figure 6 illustrates the R-2 and Genghis-II pushing together.

The reduced rate of pushing in the R-2 when Genghis-II is added is caused by the following. First of all, the R-2's δ_slow_R(R-2) and δ_slow_R(Genghis-II) parameters differ quite a bit since Genghis-II is much slower at pushing the box than the R-2. Note that as described in [11], these parameter differences can be easily learned by these robots using the features of the L-ALLIANCE architecture which allow the robots to monitor and learn from the performance of robot team members. However, since we have not explained this mechanism in this paper, let us just assume that these impatience rates were assigned by the human designer so

that δ_slow_R(Genghis-II) is less than δ_slow_R(R-2). With this in mind, let us now assume that the R-2 is pushing on the left of the box, and that Genghis-II is swapped into the team on the right end of the box. Since Genghis-II takes longer to complete its pushing than the old R-2 did, the sensory feedback of the remaining R-2's *push-left* motivational behavior is not satisfied as frequently, and thus R-2's *push-left* behavior set cannot be activated as frequently. In the meantime, the *push-right* motivational behavior of the remaining R-2 is becoming more impatient to activate the *push-right* behavior set since it is not "hearing" that any other robot is accomplishing its task. However, since the *push-right* motivation is now growing at a reduced rate of impatience, δ_slow_R(Genghis-II), the motivation to activate the *push-right* behavior set does not cross the threshold of activation before Genghis-II announces its completion of the task. This in turn prevents the remaining R-2 from taking over the push of the right side of the box as long as Genghis-II continues to push. In this manner, the R-2 demonstrates its ability to adapt to a dynamic change in team heterogeneity.

We complete this experiment by removing the remaining R-2 from the team. This causes Genghis-II to activate its *go-home* behavior, since it cannot complete the box pushing task on its own. Thus, Genghis-II also demonstrates its adaptive action selection due to the actions and failures of robot team members.

4 Conclusions

In this paper, we have described ALLIANCE — a novel, fault tolerant cooperative control architecture for small- to medium-sized heterogeneous mobile robot teams applied to missions involving loosely-coupled, largely independent tasks. This architecture is fully distributed at both the individual robot level and at the team level. At the robot level, a number of interacting *motivational behaviors* control the activation of the appropriate sets of behaviors which allow the robot to execute any given task. At the team level, control is distributed equally to each robot team member, allowing each robot to select its own tasks independently and without any centralized control. These two levels of distribution allow the ALLIANCE architecture to scale easily to missions involving larger numbers of tasks. The architecture utilizes no form of negotiation or two-way conversations; instead, it uses a simple form of broadcast communication that allows robots to be aware of the actions of their teammates. The control mechanism of ALLIANCE is designed to facilitate fault tolerant cooperation; thus, it allows robots to recover from failures in individual robots or in the communication system, or to adapt their action selection due to changes in the robot team membership or the changes of a dynamic environment. We have demonstrated these abilities through the implementation of ALLIANCE on both physical and simulated robot teams. In this paper, we reported the results of a physical robot team performing a box pushing demonstration.

Not reported in this paper are a number of additional studies we have undertaken on many issues of multi-robot cooperation, including the effect of the lack of awareness of robot team member actions and numerous efficiency considerations. Refer to [11] for more details on these studies, plus descriptions of additional, more complex, implementations of the ALLIANCE architecture in both physical and simulated mobile robot teams.

Acknowledgements

This research was performed while the author was a graduate student at the MIT Artificial Intelligence Laboratory. Support for this research was provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124, and in part by the Mazda Corporation. Additional support has been provided by the Office of Engineering Research Program, Basic Energy Sciences, of the U.S. Department of Energy, under contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

References

[1] H. Asama, K. Ozaki, A. Matsumoto, Y. Ishida, and I. Endo. Development of task assignment system using communication for multiple autonomous robots. *Journal of Robotics and Mechatronics*, 4(2):122–127, 1992.

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

[3] Rodney A. Brooks. The behavior language: User's guide. Memo 1227, MIT A.I. Lab, Cambridge, MA, April 1990.

[4] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[5] Philippe Caloud, Wonyun Choi, Jean-Claude Latombe, Claude Le Pape, and Mark Yim. Indoor automation with many mobile robots. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 67–72, Tsuchiura, Japan, 1990.

[6] Paul Cohen, Michael Greenberg, David Hart, and Adele Howe. Real-time problem solving in the Phoenix environment. COINS Technical Report 90-28, University of Massachusetts at Amherst, 1990.

[7] IS Robotics, Inc., Somerville, Massachusetts. *ISR Radio Communication and Positioning System*, October 1993.

[8] Fabrice R. Noreils. Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment. *The International Journal of Robotics Research*, 12(1):79–98, February 1993.

[9] Lynne E. Parker. Adaptive action selection for cooperative agent teams. In Jean-Arcady Meyer, Herbert Roitblat, and Stewart Wilson, editors, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 442–450. MIT Press, 1992.

[10] Lynne E. Parker. An experiment in mobile robotic cooperation. In *Proceedings of the ASCE Specialty Conference on Robotics for Challenging Environments*, Albuquerque, NM, February 1994.

[11] Lynne E. Parker. *Heterogeneous Multi-Robot Cooperation.* PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA, February 1994. MIT-AI-TR 1465 (1994).

[12] Gerald J. Sussman. *A Computer Model of Skill Acquisition.* PhD thesis, Massachusetts Institute of Technology, 1973.