Robust Multirobot Coordination in Dynamic Environments

M. Bernardine Dias, Marc Zinck, Robert Zlot, and Anthony (Tony) Stentz
The Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
{mbdias, mbz, robz, axs}@ri.cmu.edu

***Abstract*— Robustness is crucial for any robot team, especially when operating in dynamic environments. The physicality of robotic systems and their interactions with the environment make them highly prone to malfunctions of many kinds. Three principal categories in the possible space of robot malfunctions are communication failures, partial failure of robot resources necessary for task execution (or partial robot malfunction), and complete robot failure (or robot death). This paper addresses these three categories and explores means by which the TraderBots approach ensures robustness and promotes graceful degradation in team performance when faced with malfunctions.**

Keywords-multirobot coordination; robustness; dynamic environments; communication failures; partial malfunctions; robot death; market-based.

I. INTRODUCTION

Many multirobot applications demand some level of robustness to malfunctions. The requirement for robustness becomes increasingly important when the application domain requires the robots to interact within a highly dynamic environment, and where prior information about the environment is sparse. Applications such as urban reconnaissance, urban search and rescue, planetary exploration, and hazardous cleanup inherently include hazardous conditions that will cause robotic malfunctions with high probability. Key to the success of these applications is the team's ability to gracefully degrade their performance and maximize the efficiency with which the available resources are used to complete the task. Multirobot coordination approaches deal with malfunctions in different ways. The three main categories of malfunctions, and multirobot coordination approaches that account for these malfunctions are explored next.

A. Communication Failures

Communication failures are abundant in many application domains. These failures can vary from occasional loss of messages to complete loss of communication. Different approaches handle losses in communication using a variety of strategies. As described by Balch and Arkin [1], some approaches forego communication altogether and robots make action decisions entirely independent of decisions made by teammates. Other approaches forego explicit communication, but instead, coordinate team actions by basing action selection on observed environmental clues [1], anticipated actions of teammates [13], socially attentive monitoring of teammates [7], or pre-defined rules, triggered by environmental cues or observation of specific team formations or actions [11]. None of these coordination approaches are affected by failures in communication. However, they are also unable to effectively use information that can improve team performance if shared with teammates. Vail and Veloso [13] show that teams can perform more effectively if teammates coordinate and share information.

If the robots explicitly communicate with each other, there are still several methods to ensure graceful degradation in performance with communication failures and limitations. Stone and Veloso [11] present a set of techniques for dealing with communication-based coordination of robot teams in adversarial environments with unreliable, high-cost communication. However, not all domains are adversarial. Parker's ALLIANCE architecture [8] does not reason about hostile agents, but encourages fault tolerance in several ways. A method to ensure robustness to message loss in less stringent application domains is the use of acknowledgements, as in the original Contract Net Protocol by Smith [9]. While this approach adds a level of robustness to message loss, some limitations are evident. The acknowledgement can be lost as easily as the message, the acknowledgements add to the communication load, and the approach does not explicitly deal with the scenario of complete loss of communication.

B. Partial Robot Malfunction

Relatively little work has been done to investigate efficient use of partially malfunctioning robots. When a robot malfunctions partially, it loses the ability to effectively use some of its resources but retains the ability to use others. Inherent in this definition of partial malfunction is the robot's ability to plan for itself or the ability to communicate with a planning agent; if the robot loses this capability, it is considered dead. One of the difficulties in dealing with partial robot malfunctions is detecting the malfunction. A host of literature on fault detection and identification demonstrate different techniques that enable robots to detect and identify their own faults. However, relatively little work has been

done to address handling detected faults in a team. Techniques such as socially attentive monitoring [7] and regular monitoring of the task/environment state and adapting to it [6] allow teammates to discover faults that the robot cannot detect itself. Once a fault is detected, fewer techniques have been proposed to deal with them. Bererton and Khosla ([2] and [3]) analyze the merits and challenges of repairing robots when failures are detected.

Many reactive and behavioral approaches ([1] and [8]) are resilient to partial robot malfunctions because robots execute tasks independently of what other team members do, and hence all tasks with no specific time deadlines are accomplished as long as at least one capable robot remains active. Gerkey [6] demonstrates a fault-tolerant auction scheme that decomposes a cooperative box-pushing task into short-duration pusher and watcher/coordinator tasks. Since the tasks span only a short duration, the team re-evaluates the progress of the task frequently and thus recovers from faults by reassigning short duration tasks designed to adapt to the most current state. However, these approaches do not reason about efficient utilization of remaining active resources of the partially malfunctioning robots.

C. Robot Death

Robot death is similar to the case of partial robot malfunction, except that the affected robot cannot aid in the recovery from the malfunction in any way. Most of the research in fault tolerance ([6] and [8]) deals with robot death. As with partial robot malfunctions, many reactive and behavioral approaches are resilient to robot death.

The detection problem is more difficult for robot death since the dead robot cannot detect its own death and reallocate its tasks. A common method of detecting robot death is to monitor a heartbeat (a periodic signal) from robots and assume the robot is dead if the heartbeat is not detected. Other methods of monitoring, such as socially attentive monitoring [7], can also be used to detect the death of teammates. Once a dead robot is discovered, any tasks assigned to that robot must be reassigned to other capable robots or the dead robot must be repaired. Bererton's and Khosla's work on robot repair ([2] and [3]) can be applicable to some cases of robot death. Note that in the cases where malfunctioning robots or dead robots can be repaired and return to the team, the coordination approach needs to be fluid in order to accommodate both the exit of the dead robots and the entrance of the repaired robots.

D. Contribution

The contribution of this paper is two-fold. First, this paper identifies three principal categories of robotic failures: communication failures, partial robot malfunctions, and robot death, and presents the most comprehensive study of robustness of robot teams executing cooperative tasks in dynamic environments. Second, this paper details an implementation of the TraderBots approach, capable of gracefully handling all three identified categories of robot malfunctions.

II. The TraderBots Approach

Dias and Stentz [4] report a detailed overview of the TraderBots approach; a market-based approach for multirobot coordination inspired by the contract net protocol by Smith [9]. A brief overview of the TraderBots philosophy is presented here. Consider a team of robots assembled to perform a particular set of tasks. Consider further, that each robot in the team is modeled as a self-interested agent, and the team of robots as an economy. The goal of the team is to complete the tasks successfully while minimizing overall costs. Each robot aims to maximize its individual profit; however, since all revenue is derived from satisfying team objectives, the robots' self-interest equates to doing global good. Moreover, all robots can increase their profit by eliminating unnecessary waste (i.e. excess cost). Hence, if the global cost is determined by the summation of individual robot costs, each deal made by a robot (note that robots will only make profitable deals) will result in global cost reduction. Furthermore, the individual aim to maximize profit (rather than to minimize cost) allows added flexibility in the approach to prioritize tasks that are of high cost but also high priority over tasks that incur low cost to execute but also provide lower value to the operation. The competitive element of the robots bidding for different tasks enables the systems to decipher the competing local information of each robot, while the currency exchange provides grounding for the competing local costs in terms of the global value of the tasks being performed. A detailed description of the current implementation of the TraderBots approach is published in the proceedings of the 2004 conference on Intelligent Autonomous Systems [5].

A. Handling Communication Failures

The TraderBots approach does not depend on communication to complete tasks. Communication mainly plays the role of enabling improved efficiency in the generated solutions. Zlot et al. [15] investigate the performance degradation of the team, in the TraderBots approach, given the absence of communication. Newer implementations of the TraderBots approach are made more robust to communication failures. Message loss is expected and often witnessed resulting in only minor degradations in solution efficiency. Strategies used to improve robustness are: frequent auctioning and bidding which help reallocate tasks among robots more efficiently, the absence of assumptions that all robots will participate in any auction, monitoring of communication connectivity to robots that have subcontracted tasks, and continuous scheduling of assigned tasks for execution as tasks are completed.

However, in a case where only the OpTrader (an interface agent responsible for trading on behalf of the operator) is aware of all tasks, and the tasks are divided among the robots, a scenario such as a combination of communication failure between the OpTrader and all robots, plus the death of a robot with assigned tasks can result in the task assigned to the dead robot remaining incomplete. Thus, domains where completion of the global task (i.e. all tasks assigned to the team) must be guaranteed (if resources are available) require a different strategy. A possible strategy for these domains is to disseminate knowledge of all tasks to all robots, as would be

the case in many reactive approaches. Note that this strategy is only required if specific tasks are assigned to the team. In the case where robots dynamically generate tasks (i.e. where the same tasks can be generated by other robots given sufficient time), as in work published by Zlot et al.[14], this strategy is unnecessary.

In the current implementation, it is possible for more than one robot to believe it is responsible for executing the same task if communications are imperfect. For example, when robot A awards a task to another (robot B), an acknowledgment is sent from B to A. If the acknowledgment is lost, then robot A does not know if B has accepted the task. In that case both A and B will maintain responsibility for completing the task. It is also possible that this duplication of tasks can be repaired: one of the robots may subsequently try to auction the task, in which case the other will be likely to win it as its marginal cost for the task is 0.

B. Handling Partial Robot Malfunctions

Detecting partial robot malfunctions in the TraderBots approach is achieved by monitoring the resources available to the robots. While specific algorithms for fault detection and identification are beyond the scope of this paper, in general, the TaskExec's (the module responsible for task execution) loss of access to a particular resource, the Trader's loss of access to its TaskExec, the discovery of an unforeseen depletion of a resource, or the discovery that the accrued cost in attempting to complete a task surpasses the estimated cost for that task, can indicate a partial robot malfunction. Once a Trader discovers a partial robot malfunction, it attempts to sell all tasks it cannot complete to other robots even if it has to take a loss for some of the trades. (The trader still attempts to maximize profit, so any losses will be minimized). If however trading becomes impossible due to a coupling with loss of communication, then the relevant strategy described in the previous section needs to be used for the case where a static set of tasks, all of which must be completed, is assigned to the team. Thus, graceful degradation with malfunctions of team performance is achieved. If the malfunction occurs with the Trader, then it falls into the category of robot death. In this implementation, robots were able to detect malfunctions caused by disconnection of the on-board SICK laser used for obstacle detection and mapping, and gyro errors caused by sudden drastic rotations of the robot due to a wheel getting stuck. Ongoing implementation efforts also include detection and appropriate handling of low battery conditions that require the robot to head back to a re-charging station.

C. Handling Robot Death

Once a robot is incapable of trading, it is considered dead. In this case, the robot cannot aid in the detection or recovery process. Several methods can be employed to allow teammates to discover robot death as discussed above in section I.C. The TraderBots approach can deal with detected robot deaths by attempting to discover all trades that affected the dead robot. Each trader can keep track of awards it makes and receives. Thus, if a robot death is detected, each trader checks to see if it has awarded any tasks to the dead robot, or if it has won any tasks from the dead robot.

If a robot has awarded a task to the dead robot, it makes an announcement to the remaining robots to find out if they subcontracted that task from the dead robot. If such a trade is discovered, the two robots re-negotiate their deal with respect to that task. If a robot cannot discover any robot that is currently committed to executing that task, the task is added back to its commitment list. Note that this can result in the task being repeated due to communication limitations. The premise of such an implementation would be that it is better to repeat the execution of a task rather than leave any task incomplete, if available resources permit. Ongoing implementation efforts include enabling the TraderBots approach to gracefully and robustly accommodate robot death. Results in detecting and handling robot death will be added in final submission of this paper if it is accepted for publication. Finally, note that the TraderBots approach easily accommodates fluidity by allowing repaired robots or new robots to enter the team since any available robot can participate in the frequently conducted auctions. Initial experiments demonstrating this capability are reported in results submitted for publication to the 2004 conference on Intelligent Autonomous Systems [5]. A limitation in the current implementation is that detection of a robot death is indistinguishable from a severe communications failure since the only way robots detect one another is via communication. It would be possible to improve this if the robots additionally had some other mode of detecting/monitoring each other (for example, by using a camera or some other sensor).

III. EXPERIMENTS

An implementation of the TraderBots approach on a team of 3 Pioneer robots enables the reported results. The robot team (shown in Figure 1) consists of a homogenous set of off-the-shelf mobile robot platforms outfitted with additional sensing and computing.

Figure 1: Robot Team

Serving as the mobility platform is an ActivMedia Pioneer II DX indoor robot. An 802.11b wireless card on each robot allows ad-hoc communication between robots. Encoder data from the drive wheels is collected on-board from which dead reckoning position is calculated (x, y, θ). Alternate angle measurements using a fiber optic rate gyroscope (KVH E-Core 1000) allow improved localization. Sensing is accomplished using a 180˚ scanning laser range finder (SICK LMS 200). Horizontal scan-range-data is incorporated with position data to create a 2D map. In addition to providing information to the operator, the map is used for local navigation and cost estimation for trading. Further implementation details are

reported in our submission to the 2004 conference on Intelligent Autonomous Systems [5].

The chosen application is a distributed sensing problem where 3 robots are tasked with gathering sensory information from a number of designated locations of interest in a large dynamic environment. This translates into a version of the traveling salesman problem (TSP) with the robots being represented by multiple salesmen following paths instead of tours (i.e. without the requirement that robots need to return to their starting locations) and where all the robots can start from different base locations – this is known as the multi-depot traveling salesman path problem (MD-TSPP). The tasks can be considered as cities to be visited where the costs are computed as the time taken to traverse between cities. A task is completed when a robot arrives at a city. The global task is complete when all cities are visited by at least one robot. The global cost is computed as the summation of the individual robot cost, and the goal is to complete the global task while minimizing the number of robot-hours consumed. When the robots are not executing tasks, they remain stationary at their current locations.

Figure 2: Photograph of test-site

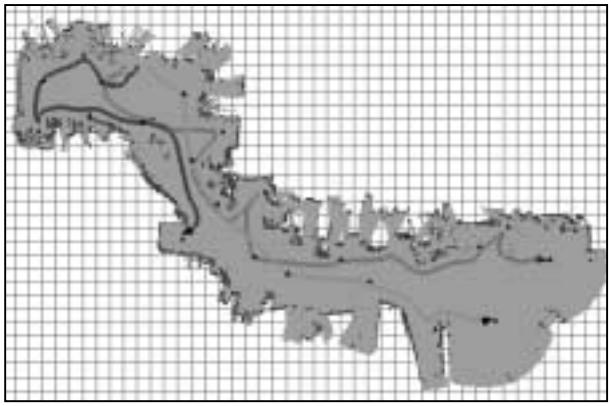

Figure 3: Map of environment showing assigned goals and robot paths (grid squares = 1m×1m)

Each robot is responsible for optimizing its own local schedule (i.e. given a set of tasks, the robots attempt to find the optimal TSPP solution to their local problem instance). In general, the TSPP is NP-hard, so approximation algorithms are often used when the problem instances encountered are large. In the implemented TSPP scenario, all valuations are derived from inter-point distance costs. These costs are estimated using a D* path planner [10] with the robot's local map as input. The experiments performed, using this implementation, are described in this section. The results obtained, and the relevant analysis is presented in section IV.

A. Communication Failures

1) Experiment A1: Simulated Message Loss
Description: Communications between robots are blocked at different percentage levels (20%, 40%, 50%, 60%, 80%, and 100%) and the corresponding performance for 3 robots assigned 23 tasks is reported.

B. Partial Robot Failures

1) Experiment B1: Simulated Malfunctions
Description: Laser is turned off or gyro error is introduced at a specific point during execution and the resulting performance for 3 robots assigned 23 tasks is reported.

C. Robot Death

1) Experiment C1: Simulated Death Before Execution
Description: A robot is turned off just prior to task execution, simulating an unforeseen hardware error that prevents operation, and the resulting performance for 3 robots assigned 7 tasks is measured.

2) Experiment C2: Simulated Death During Execution
Description: One or more robots are turned off at random times during execution and the resulting performance for 3 robots assigned 7 tasks is measured.

D. Fluidity

1) Experiment D1: Re-Entering Dead Robots
Description: One or more robots are turned off at random times during execution and one robot is returned to the base station, turned back on, and re-inserted into the team at a later random time. The corresponding effect on performance for 3 robots assigned 7 tasks is measured.

IV. Results and Discussion

Experiments A1 through D1 were completed as described above. The corresponding results are reported in this section. All reported results in experiments A1 and B1 are averaged over three consecutive runs in the same environment under nearly identical conditions.

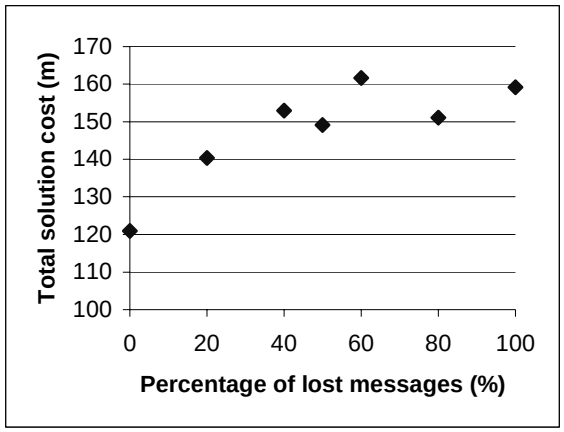

Figure 4: System Performance with Communication Failures

The first set of experiments investigates the effect of communication failures on the team performance. Inter-robot communication is blocked at different percentages and the resulting solution recorded. Figure 4 shows the variation of solution cost with the percentage of message loss. While the solution cost increases with loss of communication until approximately 60% loss, further communication loss has little added effect on performance. The reason for this is that when the loss rate is significant but not too large, it is often the case that tasks are subcontracted, but their award acceptance acknowledgement message does not reach the seller. When this happens, the task ends up being duplicated, thereby increasing the global cost. When the loss rate is high, the trades do not progress to this stage as often and this effect is not seen as frequently (e.g. the bids or the award are already lost, so the task is not awarded). Since our initial solution based on the initial OpTrader auctions is reasonably good, the result is that we sometimes can do better with 100% loss rate than with 60% loss. We hypothesize that if we start off with a worse solution (for example, an initial random allocation), then we would expect that this function would be more monotonic. Also, if we enable the robot death handling, then there would be more duplications of tasks at the high loss rates when a death was detected and robots try to make up for the tasks of the "dead" robots. Note that the different failure conditions were studied separately in these experiments and hence the death handling was disabled during the testing of communication failures and partial robot failures.

The results from experiments A and B are reported in Table 1. For each experiment, the mean cost, the standard deviation in the cost over the three runs of the experiment, the number of tasks that succeeded, and the number of tasks that failed are shown. Note that tasks can fail for several reasons due to the dynamic environment and conditions. Note further that tasks are sometimes duplicated and hence the addition of succeeded and failed tasks sometimes exceeds the number of assigned tasks (23). Future implementations will be able to better deal with duplicate tasks as follows. If a trader is selling task x and another robot already has committed to task x then that robot will bid very low for the task and win it (its marginal cost is 0). When the robot is awarded the task, it should check if it is a duplicate, and if so it should be discarded.

Description	Cost (m) Avg.	+/-	Tasks (#) Succ	Fail
Nominal	121	12	21.0	2.0
Partial Failure	140	5	22.0	1.0
20% msg. loss	140	5	24.0	0.3
40% msg. loss	153	3	24.7	2.0
50% msg. loss	149	10	24.0	0.7
60% msg. loss	162	9	25.3	0.7
80% msg. loss	151	3	22.3	0.7
100% msg. loss	159	5	21.0	2.0

Table 1: Performance Statistics

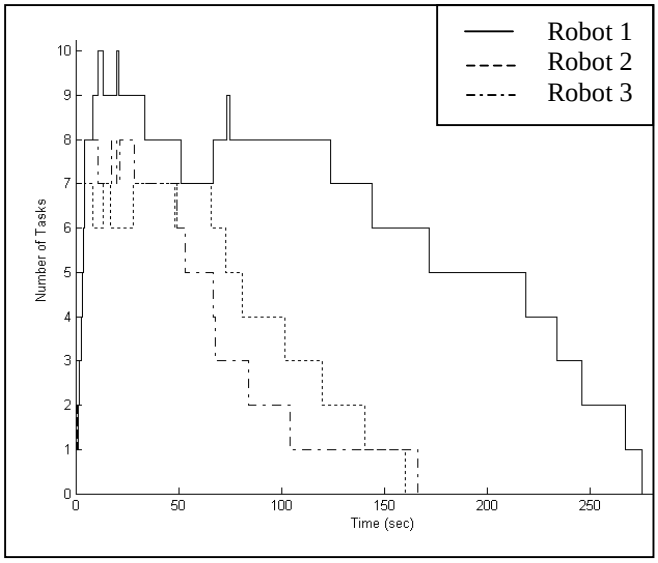

Figure 5: Nominal Performance with 3 Robots and 23 Tasks

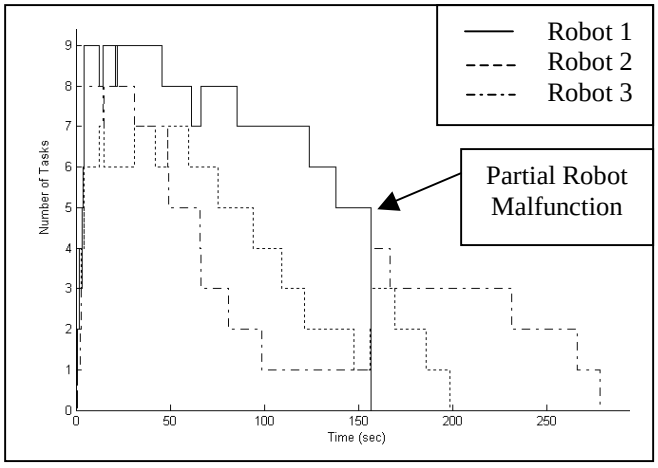

Figure 6: Partial Robot Malfunction

Figure 5 and Figure 6 show the variation in the number of tasks assigned to each robot over time for a nominal run and a partial robot failure run respectively. While the number of tasks gradually decreases with time as tasks are executed in a nominal run, when a partial failure occurs, that robot

immediately trades away all of its tasks attempting to minimize its losses, and hence the malfunctioning robot has a sudden loss in the number of assigned tasks. The other two robots have a sudden gain since they are assigned the unfinished tasks of the malfunctioning robot.

Figure 7 shows the nominal performance of a 3 robot 7 task scenario where robot 1 executes 3 tasks, and robots 2 and 3 execute 2 tasks each. This scenario corresponds to experiments C and D.

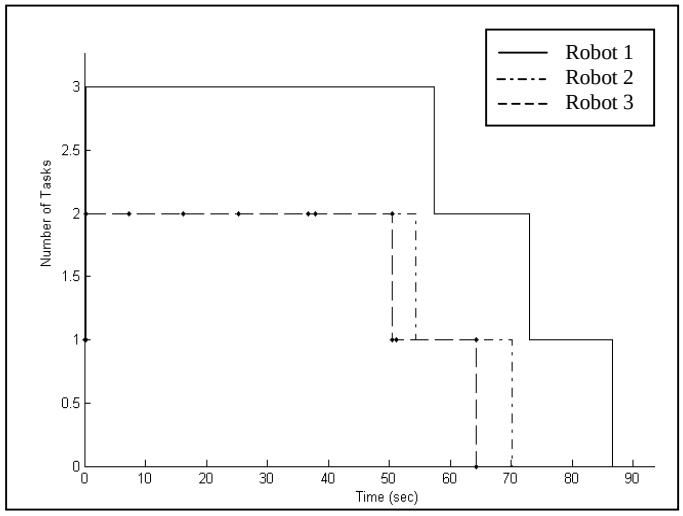

Figure 7: Nominal Performance with 3 Robots and 7 Tasks

In accordance with experiment C1, when robot 3 is killed after initial trading with the OpTrader has been completed, and before execution has commenced, the 2 tasks assigned to robot 3 are later discovered and executed (one each) by robots 1 and 2 as shown in Figure 8.

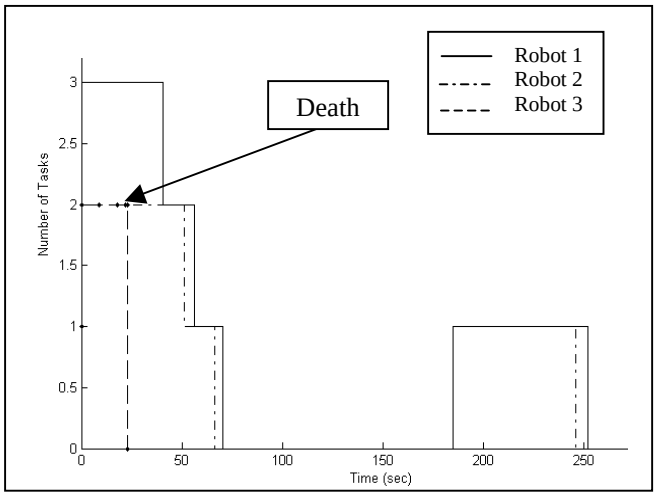

Figure 8: Performance with 3 Robots and 7 Tasks with one robot death prior to execution

Note that there is a delay after the completion of the initial tasks assigned to robots 1 and 2 while the death of robot 3 is discovered and the tasks assigned to robot 3 prior to its death are reassigned by the OpTrader.

A similar scenario occurs in the case of Figure 9. Here robot 1 is killed prior to execution having won 3 tasks. These 3 tasks are later executed by robot 2.

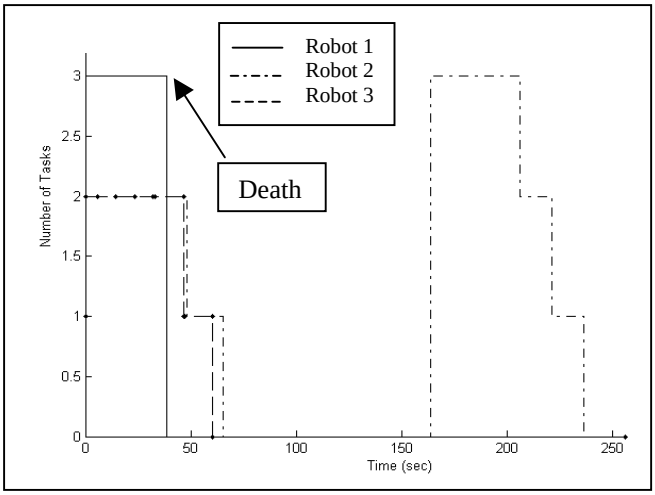

Figure 9: Performance with 3 Robots and 7 Tasks with one robot death prior to execution

Experiment C2 investigates the robustness of the TraderBots approach to multiple deaths at random times during execution. Figure 10 shows the scenario where both robot 2 and 3 are killed after executing one task each. These two tasks are later completed by robot 1.

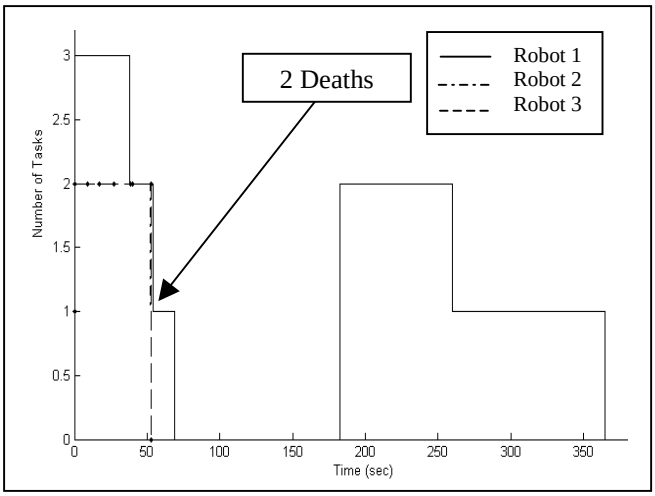

Figure 10: Performance with 3 Robots and 7 Tasks with two robot deaths during execution

Two similar scenarios are shown below in Figure 11 and Figure 12. In the first case, robot 1 and robot 2 are killed after executing 1 task each. The remaining 3 tasks assigned to these two robots are later completed by robot 3. In the second case, robot 1 and robot 3 are killed after executing 1 task each. Robot 2 completes the 2 tasks assigned to it and also the 3 tasks assigned to the robots that were killed. Note that in all these

scenarios, the tasks assigned to robots that are killed are reassigned once the OpTrader discovers the death of a particular robot. This discovery can happen because the OpTrader loses a connection to the robot for a sufficiently long period of time such that a death is suspected and that suspicion is reinforced by any remaining "live" robots, or because a live robot detects and reports the death of a robot to the OpTrader. Note further that in general tasks will be robustly completed as long as the generator of the task is alive and detects the death of a robot, or if the task was assigned to one of the live robots at some point in time.

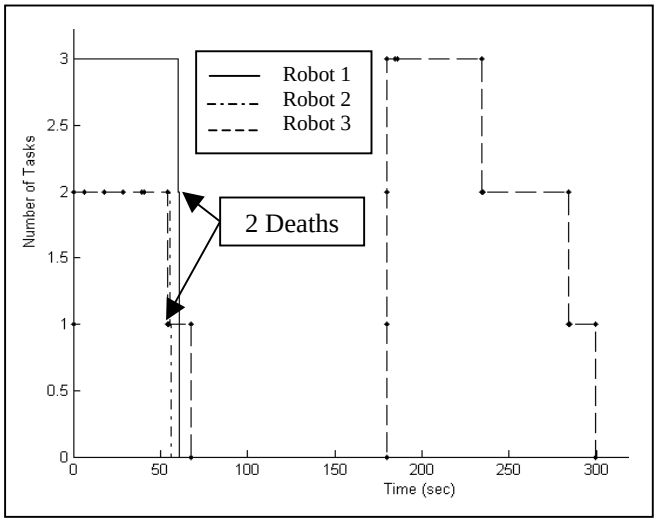

Figure 11: Performance with 3 Robots and 7 Tasks with two robot deaths during execution

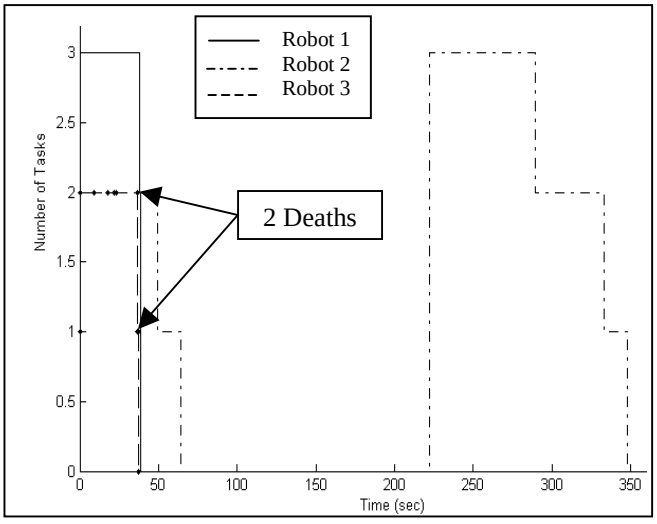

Figure 12: Performance with 3 Robots and 7 Tasks with two robot deaths during execution

A final experiment conducted was to demonstrate the fluidity of the TraderBots approach. Previous work [5] demonstrated the ability to insert a robot into the team during execution and allow it to participate in executing the team mission. Experiment D1 demonstrates the ability to kill a robot during task execution, have its tasks reassigned, repair the dead robot, re-enter it into the team, and allow it to execute any incomplete tasks.

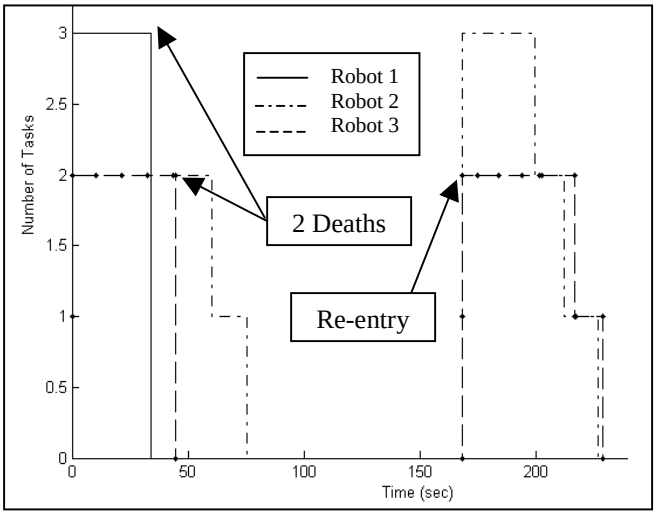

Figure 13: Performance with 3 Robots and 7 Tasks with two robot deaths and one robot re-entry during execution

Figure 13 shows robot 1 and robot 3 being killed prior to execution. Robot 2 completes the 2 tasks assigned to it. The remaining 5 tasks are then split between robot 2 and robot 3 who is re-entered into the team from its start location (thus simulating a repair and re-launch from a base station).

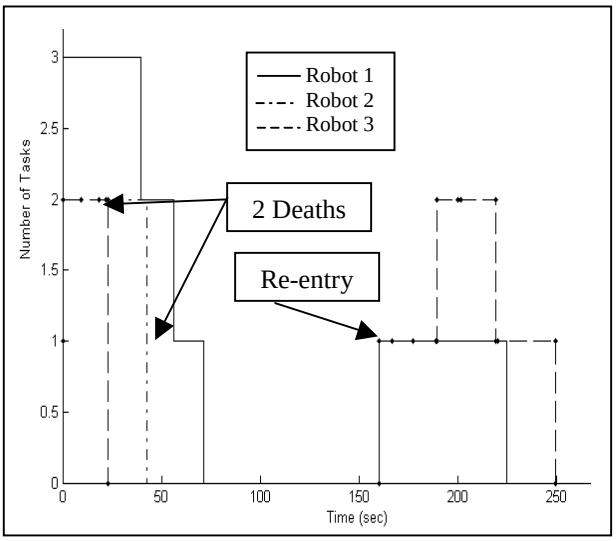

Figure 14: Performance with 3 Robots and 7 Tasks with two robot deaths and one robot re-entry during execution

A similar scenario is demonstrated in Figure 14. Robot 2 and robot 3 are killed soon after initial trading is complete in this scenario. Thus only robot 1 completes all of its 3 assigned tasks. Robot 2 completes 1 task before being killed, and robot 3 is killed before it can complete any of its tasks. Robot 3 is

later repaired and re-entered into the team. Thus, robot 1 and robot 3 complete the remaining 3 tasks, and thus, the team mission.

One final observation worthy of discussion is the delay in detecting and responding to the death of a robot. This delay is evident in Figures 8-14. Note that there is a tradeoff between the response time to a suspected death and the number of false positives that might be detected. For example, a perceived death could simply be the result of a temporary communications failure. If this perceived death is immediately treated as a death, several tasks could be repeated unnecessarily. In the current implementation of TraderBots, the robot team experiences a delay on the order of ~100s before re-assigning the tasks of a dead robot. This delay could be made smaller or larger depending on the requirements of the application domain.

V. Conclusion and Future Work

This paper presents a comprehensive study of how the TraderBots approach is robust to failures. Three categories of failure are identified and explored in this study: communication failures, partial robot malfunctions, and robot death. All three categories of failure are studied for a team of 3 Pioneer robots assigned a distributed sensing task. Some robots are also re-introduced into the team following a simulated revival from death.

Ongoing work introduces random combination of failures at random times during the experiment, to gauge the effect on the overall performance. Introduced failures include communication failures, partial robot malfunctions, and robot deaths.

Disallowing robots to recover from malfunctions and not investigating cooperative means of robot repair thus far limits this study. Adversarial domains are not addressed either. Future work includes developing techniques for more efficient use of partially malfunctioning robots, examining strategies for cooperative recovery from failures, and more rigorous testing of the robustness of TraderBots in different scenarios.

Acknowledgment

This paper presents results based on a multirobot architecture, implementation, and test bed used for conducting research on two active projects. This work was sponsored in part by the U.S. Army Research Laboratory, under contract "Robotics Collaborative Technology Alliance" (contract number DAAD19-01-2-0012) and in part by NASA, under contract "Heterogeneous Multi-Rover Coordination for Planetary Exploration" (contract number NCC2-1243). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, NASA, or the U.S. Government. The authors wish to acknowledge the contributions of the members of the Multirobot research group: Juan P. Gonzalez, Bart Nabbe, Nidhi Kalra, Dave Ferguson, and Andres S. Perez-Bergquist, which enabled this implementation.

References

[1] Balch, T. and Arkin, R.C., "Communication in reactive multiagent robotic systems", Autonomous Robots, 1(1): pp.27-52, 1995.

[2] Bererton, C., and Khosla, P., "An Analysis of Cooperative Repair Capabilities in a Team of Robots", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2002.

[3] Bereton, C., "Repairable Robot Teams: Modeling, Planning, and Construction", Thesis Proposal, Carnegie Mellon University, http://www-2.cs.cmu.edu/~curt/Thesis_Proposal/Thesis_Proposal.pdf, 2003.

[4] Dias, M. B., and Stentz, A., "A Market Approach to Multirobot Coordination", Technical Report, CMU-RI -TR-01-26, Robotics Institute, Carnegie Mellon University, August 2001.

[5] Dias, M. B., Zlot, R., Zinck, M., Gonzalez, J. P., and Stentz, A., "A Versatile Implementation of the TraderBots Approach for Multirobot Coordination", Proceedings of the 8th Conference on Intelligent Autonomous Systems IAS 2004.

[6] Gerkey, B. P., "On Multi-Robot Task Allocation", Ph. D. Thesis, University of Southern California, 2003.

[7] Kaminka, G. and Tambe, M., "Robust agent teams via socially attentive monitoring", Journal of Artificial Intelligence Research (JAIR), (to appear, submitted in 2000).

[8] Parker, L. E., "ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation", IEEE Transactions on Robotics and Automation, Vol. 14, No.2, pp. 220-240, 1998.

[9] Smith, R., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", IEEE Transactions on Computers, Vol. C-29, No. 12, 1980.

[10] Stentz, A., "Optimal and Efficient Path Planning for Partially-Known Environments", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 1994.

[11] Stone, P., and Veloso, M., "Communication in Domains with Unreliable, Single-Channel, Low-Bandwidth Communication", Collective Robotics, pp. 85–97, Springer Verlag, 1998.

[12] Stone, P., and Veloso, M., "Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork", Artificial Intelligence, 110(2), pp.241-273, 1999.

[13] Vail, D., and Veloso, M., "Dynamic multi-robot coordination", Multi-Robot Systems: From Swarms to Intelligent Automata, Volume II, pp.87-100, 2003.

[14] Veloso, M., Stone, P., and Bowling, M., "Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer", Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II, volume 3839, 1999.

[15] Zlot, R., Stentz, A., Dias, M. B., and Thayer, S., "Multi-Robot Exploration Controlled By A Market Economy", Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2002.