Robust Multi-Robot Coordination in Noisy and Dangerous Environments*

Sanem Sariel
Istanbul Technical University
Department of Computer Engineering
Istanbul, TURKEY, TR34496
sariel@cs.itu.edu.tr

Tucker Balch
Georgia Institute of Technology
College of Computing Department
Atlanta, GA, USA 30332
tucker.balch@cc.gatech.edu

***Abstract* - In this paper we present the design and implementation of a complete framework for multi-robot coordination in which robots collectively execute inter-dependent tasks of an overall complex mission requiring diverse capabilities. Given a heterogeneous team of robots and task dependencies, proposed framework provides a distributed, robust mechanism for assigning robots to tasks in an order that efficiently completes the mission. The approach is robust to unreliable communication and robot failures. It is market-based approach, and therefore scalable, but it does not provide guarantees of optimality. In order to obtain optimum allocations in noisy environments we introduce a *coalition maintenance scheme* for dynamic reconfiguration of the assigned tasks at run time. Additional routines, called *precautions* are added in the framework for addressing different types of failures common in robot systems and solving conflicts in cases of these failures. Framework has been tested in simulations that include variable message loss rates and robot failures. We expect to port the system to mobile robots in the future. Our experiments illustrate effectiveness of the proposed approach in realistic scenarios.**

Index Terms – distributed AI, robotics, multi-agent systems.

I. INTRODUCTION

In this work, we present a framework for multi-robot teams that must coordinate to complete complex missions including tightly coupled tasks that require diverse capabilities and collective work. Our approach combines *auctions*, *coalition maintenance* and recovery routines called *precautions* to provide an overall system that finds near optimal solutions in the face of noisy communication and robot failures. We do not formally establish the optimality of the proposed approach's solutions in this paper, but we do illustrate its capabilities in a series of simulation experiments that include message loss and robot failures. *Precaution* routines embedded in the framework enable the system to dynamically respond to these failures at run time and still complete the mission.

Our system relies on a set of task dependencies that are compiled *a priori* by a mission commander, or generated by a planner before the mission. The task dependencies, a specification of the mission, and a specification of the robot teams' capabilities are distributed (reliably) to the robots before mission execution begins. At that point, negotiation of task assignments and execution of the mission begins.

In some cases, task assignments are discovered to be invalid due to conflicts or environmental constraints. Our framework of *precaution* routines discovers such conflicts and resolves them.

Proposed framework strives to provide optimal solutions while simultaneously responding effectively to communication losses and robot failures. To our knowledge, the framework presented in this paper is the first coordination scheme to address such a broad range of failures for heterogeneous teams executing tightly coupled tasks requiring collective work.

II. BACKGROUND AND RELATED WORK

A number of researchers have investigated the dynamic task allocation problem in the face of robot failures (including partial and complete failures) or environmental changes. In most cases the test domain missions include independent sub-tasks that can be executed by a single robot. The market-based approach (borrowed from DAI research) shows great promise in these applications. We briefly review the existing work in multi-robot coordination for team tasks.

TraderBots is a market-based approach for multi robot coordination [1]. Optimality analysis of TraderBots is implemented on multi-robot exploration problem in which robots are homogeneous and tasks have same type of capability requirements. MURDOCH also uses auction methods for coordination of multi-robot systems for multiple tasks [2]. In MURDOCH, however, resources are not exploited in a globally optimal fashion. In L-ALLIANCE task-oriented missions are embedded in a behavior based system [3]. The ability for robots to respond to unexpected events such as robot failures is provided through the use of motivations. Task allocation is implemented by motivations by using behavior based approach. Experiments for this work are implemented on real robots. Dahl presents a differentiation among homogeneous robots based on their performances to improve performance [4]. Communication failure analysis is not implemented. Chaimowicz's work addresses the task dependence issue [5]. His role exchange mechanism is

* This work is supported by Siemens TURKEY, Tincel Kultur Vakfi TURKEY and NSF.

implemented for a cooperative transportation mission. A utility calculation is used to provide suggestions for role exchanges. It is assumed that the robots know their positions on the environment and there are no errors in the explicit communication.

Lemarie's [6] and, separately, Kalra and Stentz's [7] work addresses the task dependency issue for tightly coupled missions. Kalra and Stentz's system was tested for a collective perimeter-sweeping mission. Task dependencies are considered in keeping a formation while simultaneously obeying some rules. Heterogeneity of the robot team is not explored. In Lemarie's work, the main goal is to keep the architecture distributed among the robots so as to support scalability, robustness and reactivity. However the solution's optimality is not guaranteed. Lemarie also takes advantage of a market-based approach. For controlling auction generation, a token-ring network approach is used. Only one auctioneer can initiate an auction. In these approaches, performance for combined tests of failures is not measured.

Proposed framework is distinct from previous work because it seeks optimal solutions, recovers from failures online, supports heterogeneous teams, and supports dependent tasks that require multiple robots. We assume that robots are not able to infer the state of others by observation; although such capabilities would only provide more reliability (e.g. [8]). The main objective is to provide a system for allocating tasks in an optimal manner in valid plans. However, failures most common in robot systems: communication failure and robot failure are also handled by the framework. These various failures, including the kinds of failures that can result from lossy communications during task negotiations are addressed by *precaution* routines. According to the classification given in Gerkey and Mataric's work [9], the proposed framework is for single-task robots, time-extended assignment and multi-robot tasks.

III. Framework Overview

In this work, a distributed coordination framework for multi-robot teams implementing complex missions of tightly coupled tasks requiring diverse capabilities and collective work is proposed. Optimal solutions (in terms of task execution cost) are sought using a cost optimal action selection method. While we do not prove it here, we believe the plans generated by the framework are cost optimal when there are no communication failures (proof of optimality is for later work). However, the framework also provides responding to communications and robot failures online. Online (i.e. during execution) failures are addressed using *precaution routines* that recognize and recover from various failure modes including conflicts. The *coalition maintenance scheme* also identifies sub-optimal task assignments, and can direct reassignments during execution if a lower cost solution would result. To our knowledge, this is the first complete framework for heterogeneous robots implementing tightly coupled tasks requiring diverse capabilities and collective work that addresses optimality and produces recovery solutions against real environmental constraints.

We assume we are provided a specification of the mission to include a set of tasks to be completed as well as the robot capabilities that are required for accomplishing each tasks. It is possible that some tasks may require several robots for completion. We also assume we are provided a specification of task dependencies that describes ordering constraints between tasks. This information is communicated reliably to the robots, whereupon they begin to negotiate for and execute mission tasks in a fully distributed manner without a central coordinator or leader.

The framework is robust to message losses and robot failures. To deal with message losses, each robot maintains a local model of the mission state. When messages are received from other robots, their mission model is updated accordingly. In some cases a new message implies a conflict with the robot's model of the task state. In this case, appropriate precaution routines are activated to either correct the model, or initiate a recovery. Recovery operations may include warning other robots of the problem or changing task allocations. These inconsistencies usually arise when robots are not informed about tasks that are completed, under execution or under auction.

A. Task Representation

One of the important mechanisms of the framework is the validity of the generated plans for executing tasks among robots. Although there is not a central planner, the robots execute tasks by considering dependencies among them. There are some models for task representations for robots such as TDL [10], however, to our knowledge, a TDL representation supporting multi-robot coordination has not been released or published yet.

We use a simple representation for tasks with ordering and dependency constraints, similar to but simpler than some representations in TAEMS [11]. We consider *Hard* and *Soft* dependencies. If task T_2 is hard dependent on task T_1, it cannot be executed before task T_1 is completed. In this case, there is a strict ordering between tasks. If task T_2 is soft dependent on task T_1, it can be executed while task T_1 is being executed or before its execution begins. However task T_2 cannot be completed until task T_1 is completed. Therefore some portion of the two tasks can be executed at the same time. A sample situation can be seen in Fig. 1. Tasks T_2 and T_3 are hard dependent on task T_1 while task T_3 is also soft dependent on task T_2. The dashed area represents the waiting time of the robot R_3 for task T_2 to be completed.

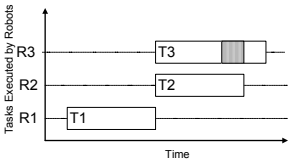


Fig. 1. Task Dependence Relationship

The required capabilities for executing a task are also encoded in the representation. Robots lack of necessary capabilities for tasks are not eligible for executing. They do not consider such tasks for execution. We also added the required number of robots to implement the task in the representation. The tasks can only be executed when necessary number of robots is ready for executing this task.

The heterogeneity of the system with robots, the task dependencies, required capabilities and number of robots restrict the allocation of tasks to the robots.

B. Roles

The framework is designed for missions consisting of sub-tasks having dependencies and requiring either one or more than one robot to execute. Therefore the tasks may be executed by a group of robots. We have chosen the *coalition* organizational paradigm to organize the task execution for our framework [12, 13]. These coalitions can contain one or more robots according to the required number of robots to execute the tasks. Auction based selection of the coalition members are implemented. The auctioneers are also active robots in the system.

The overall objective is completing a mission (M) consisting of T_i s $(0 \leq i < ||M||)$ with a multi-robot team ($r_j \in R$, $0 \leq j < ||R||$) in a cost optimal manner. Each coalition (C_i) is formed to execute a task T_i of overall mission M. Sizes of coalitions vary according to the required number of robots ($reqno_i$) to execute the task. The capabilities (cap_j) of robots R_j in a coalition should be a superset of the required capability set for T_i ($reqcap_i$).

A robot (r_j) may be in different roles for task T_i such as auctioneer, bidder (B_{ij}), coalition leader (CL_i) and coalition member (CM_i) in different time steps.

- An *Auctioneer* robot is responsible for managing auction negotiation steps and selecting $reqno_i$ suitable members of a coalition.
- A *Bidder* robot is a candidate to become a member of a coalition executing a task.
- A *Coalition Leader* is the robot responsible for maintaining the coalition and providing synchronization. It executes a portion of the task.
- A *Coalition Member* is one of the members of the coalition, and it executes a portion of the task.

A is the auctioneer set in a time step. B_{ij} is the bidder robot r_j for task T_i. A robot r_j may be in more than one B_{ij} roles for different tasks. However it is not allowed that a robot r_j may be in more than one of roles A_i, CM_i, or CL_i.

We assume that auctioneer is informed about the number of necessary robots to execute that task. Therefore it is responsible to select necessary number of candidates for task execution. After the auctioneer selects the coalition leader and members, it finishes the auction. The coalition leader forms the coalition and keeps track of the coalition members' situations and their updated information. After the execution of the task is completed, the coalition ends. Each robot is allowed to be in only one coalition until it leaves the coalition. The auctioneer may either be in the coalition of which the auction it offered, or not.

Coordination of coalition members are implemented by synchronization messages. Therefore each coalition member has responsibilities for the effective coordination. To ensure maintaining optimality against environmental changes, a dynamic reconfiguration approach is proposed.

C. Precautions

For dealing uncertainties because of the message losses, each robot keeps track of the models of known tasks and other robots in their world knowledge. The up to date situations of the known tasks are kept track by representing each known situation in a FSM. The transitions among the states for the tasks can be seen in Fig. 2. The state transitions are implemented either by own motivations or incoming information from other robots. The state transition details are explained in corresponding sections.

When the robots get information from the others they update their world knowledge accordingly. Whenever there are message loses in the system, the world knowledge of each robot may be inconsistent. However the framework ensures an update mechanism when conflicts are detected to reduce inconsistency. When the robots receive inconsistent messages, they either warn others or correct themselves. These inconsistencies occur when robots are not informed about the tasks that are completed, under execution or under auction. It is assumed that robots are trusted and benevolent.

Execution conflicts:

For parallel execution of the same task, all the robots cancel their execution. This is for ensuring optimality. A new auction is generated to consider all the robots' costs. The robots are allowed to generate different auctions for the same task. However one of them continues the auction negotiation process the details of which are explained in section *E*.

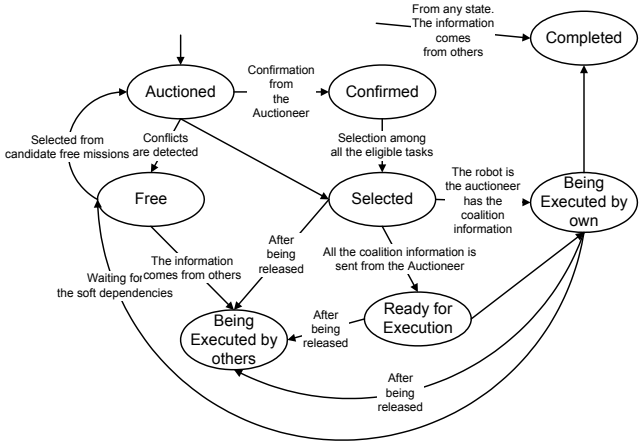


Fig. 2. FSM of a Known Task

To keep consistency, each robot executing a task is responsible for broadcasting an *execution message* for informing others that the task is under execution and the

robot is alive. The robots getting this execution message are informed about the task and they assume that the task is under execution for a period of time. If a robot cannot get any message related to a task for a period of time, the task is assumed to be free. The *execution message* is also a clue that the executer robot is still alive and the task is being executed. The same update is implemented when robots offer auctions for tasks. In this case the world knowledge is updated as the robot is not executing a task and the task is under consideration.

Auction conflicts:

Whenever a robot gets an auction message, it first checks the validity of the auction by considering the task information. If it is a valid auction it always sends bid information to the auctioneer. Therefore a more suitable task for the robot may be selected in future, whether it is currently executing a task or not. If the robots get an offer for an auction of a task that has hard dependencies, then it is assumed that these dependent tasks are completed and the world knowledge is updated as so. The same update is implemented when execution, cancellation messages are received.

Synchronization for Coordination:

For synchronized coordination, CL_i sends synchronization message to all CM_is. If either the CM_is or the CL_i believes that there is a problem with the coalition, they terminate the coalition processes.

Robot Failures:

The robot failures in a coalition can be recognized by keeping track of the models of the members. Since synchronization message is sent periodically by the CL_i, CM_i s know that the leader is alive. The CL_i also receives updated cost information from CM_i s. If members cannot get these messages for a defined period of time they believe that there is a problem and then they finish coalition processes and stop execution of the task.

Since the world knowledge is updated based on the incoming information from others, if a robot executing a task fails, after a period of time other robots not receiving messages from the failed robot mark the related task as left/free to consider it in the action selection process.

D. Action Selection

Each robot initially updates its world knowledge based on the incoming messages. Then considering the new updated world knowledge and the situations of the processes for different roles, it selects the action. The action may be joining a coalition executing a task or becoming an auctioneer auctioning for a task. Before selection of the action, the robot should perform its routine duties for different roles. These duties are given in Fig. 3. If it is leading an auction, it performs the necessary negotiation process details of which are explained in section *E*. Duties of roles CM_i and CL_i are performed for ensuring synchronization while coordinating in a coalition. If the robot is a CL_i, it checks the current coalition members' situations and can decide to reconfigure the coalition when new robots join or some members are not reachable. This coalition maintenance and dynamic configuration mechanism is explained in Section *F*.

While selecting the action, the robot considers the tasks confirmed by an auctioneer to be a coalition member, tasks under execution but with a higher value of costs than the robot has, the soft dependencies of the current task, and the free tasks which are not executed or under auction. The robot should be capable of executing all the tasks under consideration. The hard dependencies of them should be completed, and the robot should be able to execute these tasks with its capabilities. Before the robot decides on one of these tasks, it ranks these tasks based on the costs for them and selects the minimum cost task. This ranking procedure provides the optimum cost selection of the tasks for the given and known situation. The tasks confirmed by an auctioneer to make the robot one of the coalition members have higher priority than free tasks, and the tasks of coalitions with higher maximum cost value than the robot's cost have the highest priority of all. If there are only free tasks, the robot selects the one with the lowest cost. However if there are more than one free task with the same lowest cost, the robot selects the one with the lowest number of soft dependencies and lowest number of robot requirements for executing. Therefore if the costs of the tasks are the same, initially the tasks requiring low coordination is selected.

```
Routine Duties
if r_j ∈ A
        AuctionNegotitationProcess

if r_j is a CM_i
    if the "synchronization message" has been arrived
        Set coalition count down
    else if the coalition count down is 0
        Cancel execution of the task; break
        Send leave message to the CL_i
    else
        if the last communication with CL_i is more than τ (threshold)
            Send "coalition leader query" message
        Decrease coalition count down
    Send updated cost/bid value to the CL_i

if r_j is a CL_i and ||C||>1
    if each CM_i has already sent the execution/updated cost message
        Send "synchronization message" to all CM_is
        Set coalition count down
        Broadcast max cost(CM_i)
    else if the coalition count down is 0
        Terminate the coalition; break
    else
        Decrease coalition count down
        Broadcast max cost(CM_i)
    Begin CoalitionMaintenanceProcess

if the robot is not executing a task || the task under execution is released
    Begin SelectAction
Send query messages for the tasks which are free
If there is a task selected, execute the own portion
```

Fig. 3. Routine Duties of a Robot

Canceling of the current task when a better task appears can only be performed if the robot is released from the coalition of the task. Decision on releasing a robot from a coalition is only made by the coalition leader. The details are discussed in Section *F*. The Action selection algorithm is given in Fig. 4.

E. Auction Negotiation Process

Each robot offers an auction for a free task that is selected in action selection step. When a robot is an auctioneer it should maintain the auction negotiation process. In this process the coalition members and the leader are selected for the task to be executed.

SelectAction
For each known task T_i
  if the $cap_i \subseteq reqcap_i \wedge$ hard dep. of T_i are completed
    if the task is not in one of the following types skip
      The tasks for which the robot is accepted for joining (with higher max cost value than the robot's)
      Confirmed tasks by the auctioneers
      Free tasks which are not under consideration
    else add the task in a priority queue ordered by cost with the priorities in given order
Select the minimum cost task in the priority queue
if Rj is in a coalition
  cancel executing the task, send "leave" message to the CL_i

if the selected task is a free task
  begin **AuctionNegotitationProcess**
else
  set the selected task as the current task
  if it is a confirmed task
    send "accept become a member" message

Fig.4. Action Selection

Initially the auctioneer offers the auction. The robots can get the necessary task details from the auctions and first check the validity of the auction. If the auction is invalid, a warning message is sent to the auctioneer. This invalidity may occur if the auctioneer has incomplete knowledge about the mission status. Possible situations may be that the task is completed or it has already been executing. If the auction passes the validity check, the candidate robot becomes a bidder of the task (B_{ij}), calculates the cost and sends the cost value as a bid. The other candidate robots behave as so. The auctioneer robot is also a bidder and generates a bid for the task at hand. It waits until the end of the deadline. If the auctioneer cannot get the necessary number of bids from the other robots (including cost of the own) until the deadline, it cancels the auction. Otherwise it ranks all the bids. It selects the robot with the minimum cost as the CL_i of the coalition. The remaining robots are selected among the other bidders in the ranking. If the necessary number of robots to execute the task is one, the selected leader is the only member of the coalition. In this ranking process, the auctioneer may also select itself either as a CL_i or a CM_i. In the selection, the current situations of the robots are also considered. If they are recently marked as executing a task, they are not selected. That means after they sent their bids they began execution of another task.

A bidder robot may get confirmation from different auctioneers. However in the action selection step, it selects the optimum cost task for it. Therefore it only sends a message to become a CM_i to only one of these auctioneers. In the current implementation, each robot involves in only one coalition executing one task.

It is allowed that more than one robot offer an auction for the same task. In this case, when the robots detect this conflict, the robot having higher number of robot models is the winner of this race. We assume that a robot having higher number of robot models communicated with fewer robots is and it is likely to be communicated with more in future. If the number of robot models is the same, the robot having the smaller identification number is the winner. This decision only provides to recover the conflict. There may be other solutions but we avoid adding extra complexity to the negotiation process. Besides this, since these robots are in the reliable communication range (they can detect that both them auctioned for the same task), the bid of each of them is considered in either case not affecting the optimality.

Cost Calculation:

Cost function calculated by robots highly affects the overall performance. Since there may be different constraints for different domains, optimality analysis is under consideration. In some domains, using penalized cost functions for forwarding more expensive robots to more complex tasks may improve overall system's performance. A research work on time optimality versus cost optimality should be implemented. Currently in the bid/cost calculation, the energy constraint for the task and the current situations of the robots are considered and a task exchange cost is added. However analysis of this calculation is left as a future work.

F. Coalition Maintenance Process

We added a releasing and locking mechanism to prevent the coalition members leave the coalition until a new more suitable robot joined to the coalition. The coalition leader is responsible to broadcast the maximum cost value of the coalition members in each execution step. Since in the decision stage, each robot taking these kinds of messages considers the maximum cost value of each coalition, if they detect that their cost is lower than the maximum cost of the coalition and they are released, it sends a *join request* message to the coalition leader. The leader getting *join request* message, directly adds the robot to the coalition. If the coalition leader detects that the size of the coalition is more than required, it can release coalition members having the maximum cost value for the current mission. If a robot gets *released* message, it can select another more suitable task after then. When the coalition leader considers the size of the current coalition, it also checks the failures. Since each robot in the coalition broadcasts *under execution* and updated cost messages, their failure can be detected by the coalition leader. The failed robots are also released if there is enough number of members. If there is not enough number of members to execute the task for a period of time, the coalition leader cancels the coalition.

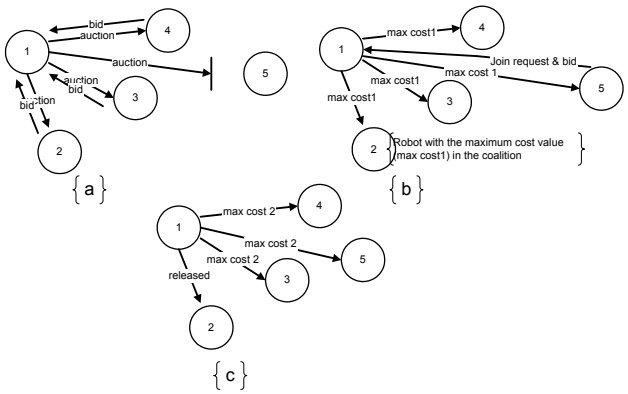


Fig. 5. Dynamic Coalition Reconfiguration

An illustrative example of coalition reconfiguration is given in Fig. 5**.** This kind of situation may occur when a robot is not reachable when the auction announcement is made {a}. When the situation is changed {b}, the robot may take a role instead of the member having the maximum cost value in this coalition {c}. Therefore the cost optimality is maintained as much as possible.

IV. EXPERIMENTAL DESIGN

We have conducted two stages of experiments to test the performance of the framework in terms of time to complete the overall mission and active execution time of the robots. In the first stage, we measured the performance against message losses for a complex mission consisting of tasks having hard and soft dependencies among each other. In the second stage, we measured the effectiveness of the framework and recovery solutions when the robot failures are added to the system. The conducted experiments are run on a simulator simulating the message exchange and execution of the missions. The results are from 100 independent runs with random seeds. The maximum simulation step number is selected as 200. Field experiments on real robots are left as a future work.

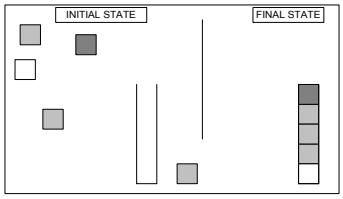


Fig. 6. Initial and Final States of the Test Domain

Test Problem Specification

We have selected collective building construction domain. In the designed experiment, there are three types of objects: *A*, *B* and *C*. The overall mission for the robots contains tasks of finding the necessary objects and making the construction while obeying the specified ordering restrictions. In the construction, initially the object *A* should be pushed into the destination and then object *B* and *C* respectively. The pushing requirements of these objects are also different. The initial and final stages can be seen in Fig. 6. The objects A, B and C are colored based on their weights: white, gray and black respectively. This mission both contains dependent tasks and requires cooperative work of the robots.

The sub-tasks and dependencies of them are given in Table 1. The third and fourth columns represent the hard and soft dependencies respectively. The required capabilities for the task and the required number of robots to execute the tasks are given in last two columns. The capabilities of having blob finder, sonar sensor, prods for objects A and C and prods for object B are listed as 0, 1, 2 and 3 respectively. Based on the specifications of the overall mission, the task dependencies can be represented graphically as in Fig. 7 in which solid lines represent the hard dependencies and the dashed lines represent the soft dependencies.

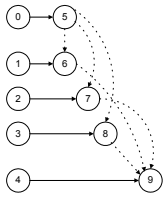
TABLE I

TASK SPECIFICATIONS

ID	Description	Hard Dep.	Soft Dep.	Req. Cap.	Req. Rob. Num
0	Find-object-A	-	-	0,1	1
1	Find-object-B	-	-	0,1	1
2	Find-object-B	-	-	0,1	1
3	Find-object-B	-	-	0,1	1
4	Find-object-C	-	-	0,1	1
5	Push-object-A-to-destination	0	-	0, 1, 2	1
6	Push-object-B-to-destination	1	5	0, 1, 3	2
7	Push-object-B-to-destination	2	5	0, 1, 3	2
8	Push-object-B-to-destination	3	5	0, 1, 3	2
9	Push-object-C-to-destination	4	6, 7, 8	0, 1, 2	3

The robot specifications are given in Table 2. We selected the number of robots suitable for tasks that needs coordination as one more than the required number of robots to execute, to see if they can reconfigure themselves against robot failures. The robots R_4 and R_7 and are added for measuring this performance.

Fig. 7. Task dependency graph of the overall mission



V. EXPERIMENTAL RESULTS

The experiments are conducted for the given complex mission and by using the heterogeneous robot set with different robot failure numbers. The step number in which the failure occurs is selected as the most important time step to complete the mission for all message loss rates. Based on the results of no failure case, we observed that 25^{th} time step is a time step that the robots have already finished executing tasks not requiring coordination, trying to coordinate for the other tasks for most of the runs. However the failed robot is selected randomly. The mission completion ratio over all runs can be seen in Table 3. In this table, the main columns represent robot systems with no failure, 1 robot failure and 2 robot failure results respectively. An additional column is added for systems with failures to report the number of runs in which one or two of randomly failed robot is a CL_i.

TABLE II
ROBOT SET USED IN THE EXPERIMENTS

Robot Set	
RobotID	**Cap.**
0, 8	0, 1
1, 2, 3, 4	0, 1, 2
5, 6, 7	0, 1, 3

It can be seen from these results, the completion of the mission is difficult when the message loss rate is greater than 50%. Since the synchronization is highly required for multi-robot coordination of tasks 6, 7, 8 and 9, the message losses cause the synchronization to be lost and also the auction negotiation steps are not completed reliably. However we can also observe that the framework can easily handle message loss rates smaller than 50% which is a very high message loss rate for real experiments. It can also be seen that although there are robot failures even the failed robots are CL_is, the mission is completed as if there is no robot failures in the system. In the third column, there are some runs in which the mission is not completed. This is because the randomly selected robots are from the same type and when they are failed, it is not possible to complete the mission. The number of runs in which the failed robot is a CL_i is 100 for message loss rate is 100. This is because, in this case each robot is the coalition leader of its own task not requiring coordination, and since the robots cannot communicate, the execution of the tasks not requiring coordination are completed in later steps and all robots try to execute all the tasks by themselves. The mission completion time results can be seen in Fig. 8. The mission completion time increases for increasing message loss rates logarithmically. However even for message loss rate is 70% there are some runs in which the mission is completed. It can be seen that when there is not a robot failure in the system the result is better. However when the failures are added, the results are very close to the no failure case. The framework can easily handle robot failures.

TABLE III
MISSION COMPLETION RATIOS

Message loss rate (%)	**# of runs in which the mission is completed/all runs**				
	no failure	1 robot failure		2 robot failure	
	# runs	# runs	#*CL i*	# runs	#*CL i*
0	100	100	22	83	39
10	100	100	22	79	45
20	100	100	27	81	33
30	100	100	27	78	42
40	100	100	20	77	31
50	98	95	19	72	27
60	59	48	15	30	27
70	4	3	13	2	18
80	0	0	5	0	19
90	0	0	4	0	25
100	0	0	100	0	100

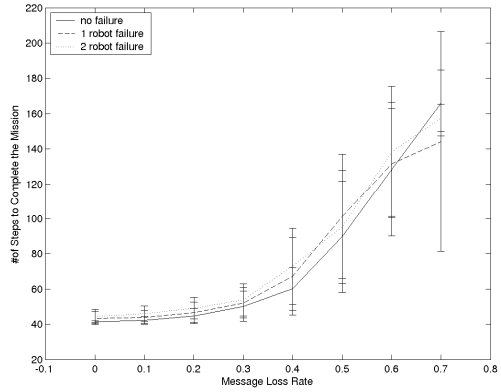


Fig. 8. Number of Steps to Complete the Mission Analysis
Averaged over 100 runs given with standard deviation

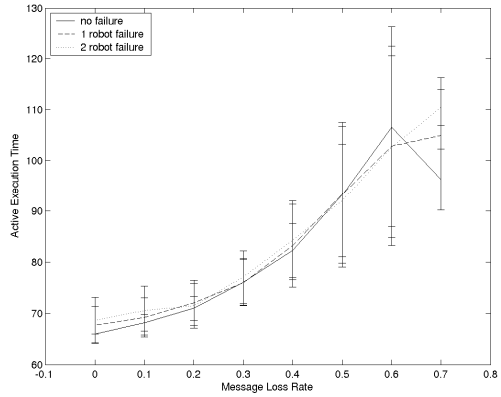


Fig. 9. Total Active Execution Time Analysis
Averaged over 100 runs given with standard deviation

In Fig. 9, the total execution time analysis can be seen. Here active execution time is counted even the robots cannot synchronize and waits for the execution. The gradient of the active execution time curve is greater than the completion time curve because of the waiting time for synchronization and some parallel executions of the tasks not requiring coordination. These parallel executions can only be eliminated by implicit communication when the message loss rates are higher. Because of that, the active execution time for 70% message loss is decreased because the executions are performed with less waiting time by chance and also the sample space for this loss rate is small to decide. Active execution time for no failure case is greater than the failure cases for some message loss rates because the number of robots in the system is greater than the failure cases.

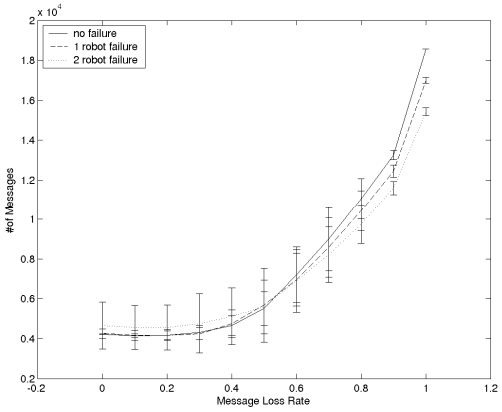


Fig. 10. Number of Messages Analysis
Averaged over 100 runs given with standard deviation

The messages sent for coordination also increase logarithmically as in Fig. 10. The robots continuously query the tasks that are free and try to coordinate for the tasks which need to be executed by more than one robot. The number of messages for no failure case is also greater when the message loss rate increases this is also because of the number of robots and their efforts to coordinate.

It can be observed from the results that the framework ensures to handle message losses as much as possible. The graphs increase logarithmically.

VI. Discussion

We propose a framework for multi-robot coordination suitable even for complex missions containing inter-dependent tasks requiring heterogeneity and coordination. The generated plans are always valid because of the task representation. The recovery solutions provided by *precaution* routines for different kinds of failures ensure the approach is complete. In this framework, close to optimal solutions are generated with available resources at hand. Experiments to validate the approach were conducted in a construction domain. The experiments tested the framework in the face of message losses and robot failures. Even for very high message loss rates, robot teams can still complete their mission using the proposed framework for coordination. If there were a way to consider implicit communication in the system, the overall performance may be increased. Analysis of the effects of the cost calculation, and proofs of optimality are left as future work. The next step in this research is real field experiments on real robots and optimality analysis for different domains.

References

[1] M. B. Dias, M. Zink, R. Zlot, A. Stenz, "Robust Multirobot Coordination in Dynamic Environments", *CMU Technical Report,* 2004.

[2] B. Gerkey and M. J. Matric, "Sold!: Auction Methods for Multirobot Coordination", *IEEE Transactions on Robotics and Automation,* vol. 18 no.5, pp. 758-768, 2002.

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

[4] T. S. Dahl, M. J. Mataric, and G. S. Sukhatme, "Emergent Robot Differentiation for Distributed Multi-Robot Task Allocation", *Distributed Autonomous Robotic Systems*, 2004.

[5] L. Chaimowicz, M. Campos, and V. Kumar, "Dynamic Role Assignment for Cooperative Robots", *International Conference on Robotics and Automation*, 2002.

[6] T. Lemarie, R. Alami, and S. Lacroix, "A Distributed Task Allocation Scheme in Multi-UAV Context", *ICRA,* 2004.

[7] N. Kalra and A. Stenz, "A Market Approach to Tightly-Coupled Multi-Robot Coordination: First Results", *ARL Collaborative Tech. Alliance Symposium*, 2003.

[8] T. Balch and R. C. Arkin, "Communication in Reactive Multiagent Robotic Systems", *Autonomous Robots*, pp. 1-25, 1994.

[9] B. Gerkey and M. J Matric, "A Formal Analysis and Taxonomy of Task Allocation", *Intl. Journal of Robotic Research* vol. 23 no.9 pp. 939-954, 2004.

[10] R. Simmons and D. ApfelbaumD, "A Task Description Language for Robot Control", *Conference on Intelligent Robotics and Systems*, 1998.

[11] K. Decker, "TAEMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms", *Foundations of Distributed Artificial Intelligence*, Chapter 16, pp. 429 – 448, 1996.

[12] H. Bryan and V. Lesser, "A Survey of Multi-Agent Organizational Paradigms", *UMass Computer Science Technical Report,* pp. 04-45, 2004.

[13] L. Soh, C. Tsatsoulis and H. Sevay, "A Satisficing, Learning, and Negotiated Coalition Formation Architecture", *Distributed Sensor Networks: A Multiagent Perspective. Ed.V. Lesser, M. Tambe, C. Ortiz. Kluwer*, 2003.