Distributed Collaboration with Limited Communication using Mission State Estimates

Mark F. Godwin, Stephen Spry, and J. Karl Hedrick

Abstract— **This paper describes a distributed system for collaboration and control of a group of mobile agents, such as unmanned aerial vehicles (UAVs) or unmanned ground vehicles (UGVs). The system allows a group of vehicles to work together to accomplish a mission via an allocation mechanism that works with a limited communication range and is tolerant to agent failure. This system could be used in a number of applications including mapping, surveillance, search, and rescue operations.**

The user defines the objectives of the group in terms of a mission plan. The mission plan contains a set of tasks, and each task contains a number of subtasks. A subtask is one objective that can be executed by a single agent, while a task is a series of related objectives that can be executed by one or more agents. Each agent in the group maintains information on the status of the mission in the form of a mission state estimate. The mission state estimate contains an agent's current knowledge of the progress of each subtask in the mission. Whenever possible, agents share their mission state estimates and use an update protocol to produce updated estimates. This protocol is designed such that the most valid information available will propagate throughout the group.

Agent decision-making is based on a combination of information from both the mission state estimate and the internal state of the agent. After checking the subtasks for completion, fault, and redo conditions, the agent selects the task and subtask for which it is best suited, taking into account its knowledge of the assignments and expected performance of others in the group. An execution module produces lower-level commands for the agent based on a set of subtask parameters. A simulated example mission demonstrates operation of the system.

I. Introduction

THE use of unmanned vehicles (UVs) to accomplish both military and civilian missions is an active area of research. While many aspects of our system can be applied to any UV we will more specifically consider unmanned aerial vehicles (UAVs). Much work to date has focused on control of single UAVs, including trajectory tracking [1] and trajectory planning [2,3]. Coordinated motion of multiple UAVs has been studied and demonstrated with static and dynamic formations of unmanned aircraft [4, 5, 6].

This work was supported in part by the Office of Naval Research under contract N00014-03-C-0187.

Authors are with the Department of Mechanical Engineering, University of California, Berkeley, CA. 94720. {markfg@berkeley.edu,sspry@newton.berkeley.edu,khedrick@me.berkeley.edu }.

In order to expand the range of missions that UAVs can effectively perform, it is necessary to develop ways for multiple autonomous vehicles to work together in collaborative groups [7,8,14]. This involves applying ideas from the areas of multi-agent systems and distributed problem solving to networked multi-vehicle systems such as groups of UAVs.

Approaches to team organization and task allocation can be categorized by degree of centralization. On one end of the spectrum, which includes behavioral [9] or emergent [12] approaches, groups of autonomous agents are designed with individual behaviors that are intended to produce desired group actions and behaviors. On the other end of the spectrum, a group of agents may simply execute commands issued by a central planner. A number of approaches, including auction-based allocation [10,11] and hierarchical dispatching [13], fall between these extremes. The viability of these approaches for a given application is closely tied to the communications topology.

While the more centralized approaches can generally promise more optimal performance, they are also the least scalable and most sensitive to failures of agents or communication links. A centralized planner might work well in ideal cases but may not be feasible in the presence of real-world constraints on communication and computation. Furthermore, any central planner or allocation node represents a single point of failure for the system.

In our system, we consider a group of UAVs with limited communication. The communication topology varies with time as the aircraft move about, and does not generally form a connected graph. This rules out a centralized solution. Instead, we seek a distributed solution that does not rely on any fixed communication topology but exploits whatever communication links are present at a given time to coordinate activities and disseminate information throughout the group.

The distributed artificial intelligence community has studied distributed problem solving and task allocation problems for some time now, although mostly in the context of systems of software agents. Work with physical agents includes [9,11,13]. In the ALLIANCE architecture [9], distributed allocation of tasks between a group of robots emerges as a result of agent behavior parameters that describe the agents' tendency to seize tasks from or relinquish tasks to other agents. In this scheme, there is no

specific commitment of an agent to complete a task. In the Contract Net Protocol [10,11], tasks are allocated between agents through the use of auctions. The agent with the winning bid is awarded the task and commits to completing the task. The award may be subject to periodic renewal based on task progress.

In this paper, which contains an expansion and refinement of the concepts presented in [15], we describe a distributed task allocation technique based on opportunistic exchange of information. Whenever two agents are within communication range, they exchange estimates of the mission 'state.' Following the exchange, each agent merges its current mission state with a mission state received from the other agent.

This approach is similar to the ALLIANCE approach in that tasks are allocated and possibly reallocated between agents without the presence of any third party such as an auctioneer. It is similar to the Contract Net approach in that when allocation or reallocation occurs, it is based on qualification.

The paper is organized as follows: In section II we discuss the description of a mission in terms of a mission plan, the mission state estimates that are maintained by each agent in the system, and the distinction between the local perception of mission state and a global mission limit state. Section III describes how an agent uses its internal state and mission state estimate to make decisions. In section IV, we explain the simulation environment that was developed to test the system, and in section V, we present and discuss simulation results from an example mission scenario. Finally, section VI draws conclusions and looks ahead to future work.

II. Mission Plan and Mission State Estimates

A. Mission Plan

The set of objectives that we would like a group of agents to accomplish is specified in a mission plan. The mission plan consists of a finite set of a distinct tasks:

$$M=\{T_1, T_2, \ldots T_a\}$$

Each task, T_i, may describe a wide variety of objectives, such as searching a specified area, patrolling a boundary, or tracking a convoy of vehicles. T_i consists of a set of subtasks, S_i, and a set of task transition rules, R_i:

$$T_i=\{S_i, R_i\}$$

where a subtask is defined as one distinct objective that can be accomplished by a single agent and the transition rules are Boolean tests used to determine if and when an agent will switch out of one task and into another.

The set S_i consists of the b_i subtasks of task T_i:

$$S_i=\{S_{i\,1}, S_{i\,2}, S_{i\,3}, S_{i\,4}, \ldots\ldots S_{i\,bi}\}$$

with each subtask S_{ij} consisting of a set of subtask parameters P_{ij} and a set of subtask transition rules R_{ij}:

$$S_{ij}=\{P_{ij}, R_{ij}\}$$

The parameter set P_{ij} specifies the information required to execute subtask, S_{ij}. If, for instance, subtask S_{ij} consisted of visiting a known location and taking a photograph of the ground, then P_{ij} would contain the coordinates of the location, required camera type, etc. The subtask transition rules R_{ij} are a set of Boolean tests used to define subtask conditions such as *complete, redo* or *fault*. These transition rules will be discussed more in Section III.

B. Mission State Estimates

During operation, each agent maintains a mission state estimate, which contains information from the mission plan as well as additional information regarding the state of the tasks and subtasks as known by the agent. The mission state estimate is designed to encapsulate high-level information that is needed for the agent to make effective decisions. It does not contain lower-level information such as the positions and velocities of other agents. We will denote the mission state estimate of agent k by $\hat{M}^k$.

The estimate $\hat{M}^k$ consists of a set of task state estimates, one for each task in the mission plan:

$$\hat{M}^k=\{\hat{T}_1, \hat{T}_2, \ldots \hat{T}_a\}$$

where we will drop the k superscript on the components of $\hat{M}^k$ in order to simplify notation.

For each task T_i in the mission, the task state estimate $\hat{T}_i$ contains the information:

$$\hat{T}_i=\{\hat{S}_i, \tau_i, R_i\}$$

where $\hat{S}_i$ is a set of subtask state estimates, R_i is the set of transition rules, and τ_i is a timestamp that contains the saved start or end time of task T_i. The set $\hat{S}_i$ consists of the b_i subtask state estimates of task T_i:

$$\hat{S}_i=\{\hat{S}_{i1}, \hat{S}_{i2}, \hat{S}_{i3} \ldots \hat{S}_{ib_i}\}$$

For each subtask S_{ij}, the subtask state estimate $\hat{S}_{ij}$ contains the information:

$$\hat{S}_{ij}=\{U_{ij}, A_{ij}, C_{ij}, \tau_{ij}, I_{ij}, R_{ij}, P_{ij}\}$$

where P_{ij} and R_{ij} are the subtask parameters and transition rules as specified in the mission plan, and U_{ij}, A_{ij}, C_{ij}, τ_{ij}, and I_{ij} contain additional information regarding the state of the subtask as known by the agent. This additional information is described in more detail below.

U_{ij}: Status. The status of subtask S_{ij}. The value of U_{ij} can be either *todo*, *assigned*, or *done*, and is determined as follows: If S_{ij} is in progress by an agent, then U_{ij} = {*assigned*}; if S_{ij} has been completed by any agent, then U_{ij} = {*done*}; otherwise, U_{ij} = {*todo*}.

***A_{ij}*:** Agent ID. The agent ID, A_{ij}, contains the identifier of the agent believed to be assigned to or to have completed a subtask. If, for example, agent 3 has chosen subtask S_{24}, then U_{24} ={*assigned*}, and A_{24}=3. An agent identifier is assumed to be a unique positive integer.

***C_{ij}*:** Cost. If the status U_{ij} equals *assigned*, then the variable C_{ij} contains the reported cost for agent A_{ij} to accomplish S_{ij}. If, for example, subtask S_{ij} consisted of visiting a point, then C_{ij} might be an estimate of the time, distance, or energy required to reach that point. In this paper we will assume C_{ij} is the estimated time required to complete S_{ij}. This will simplify the comparison of cost values, which may have been computed at different times.

***τ_{ij}*:** Timestamp. The time at which the cost estimate C_{ij} was calculated by agent A_{ij}.

***I_{ij}*:** Initialization time. The time at which S_{ij} was initially created or the time at which its status U_{ij} was most recently reset from *done* to *todo* via the *redo* transition, which will be described in section III.

Note that the status, agent identifier, and task cost information are not specifically stored in the task state estimate $\hat{T}_i$, as they can be determined directly from the set of subtask state estimates $\hat{S}_i$. On the other hand, the start and end timestamp τ_i of task T_i cannot be determined from the set of $\hat{S}_i$ and is therefore stored explicitly in $\hat{T}_i$.

C. Updating Mission State Estimates

Due to communications limitations, it is generally not possible for all agents in a group to have access to complete and current information regarding the progress of the mission and the status of other agents. Therefore, at any given time, the mission state estimates of two different agents *A* and *B*, $\hat{M}^A$ and $\hat{M}^B$, will likely be different.

Whenever possible, we would like to use communications to spread the most current mission information throughout the group. We now describe a mission state estimate update protocol that is designed to achieve this. Whenever communication is available between any two agents, they will share their knowledge of the mission by exchanging mission state estimates. Following this exchange, each agent will use the update protocol described below to update its mission state estimate, based on its own previous estimate and the estimate received from the other agent.

Consider an agent A, with mission state estimate $\hat{M}^A$, which has just received an estimate $\hat{M}^B$, from agent B. As a preliminary step, any subtasks that are found in one estimate but not in the other are copied over, so that both $\hat{M}^A$ and $\hat{M}^B$ have the same set of subtasks. The update procedure for agent A then proceeds as follows. The *ij* subscripts have been dropped to simplify notation:

For each subtask *S*:

1) Compare subtask initialization times. If ($I^B > I^A$), then overwrite S^A with S^B, otherwise keep S^A.

2) Given the status U^A and U^B check the appropriate condition in Table 1. If the condition is true then overwrite S^A with S^B, otherwise keep S^A.

TABLE I
SYNCHRONIZATION LOGIC TABLE

Subtask, *S*		*Agent A*		
		Todo	Assigned	Done
Agent B	Todo	$\tau^B > \tau^{A*}$	$\tau^B > \tau^A$	FALSE
	Assigned	$\tau^B \geq \tau^A$	See Below	FALSE
	Done	TRUE	TRUE	$\tau^B < \tau^{A*}$

*or if (($\tau^B = \tau^A$) & ($A^B > A^A$)) {TRUE, Overwrite S^A with S^B }

Assigned-Assigned

{($A^B = A^A$) & ($\tau^B > \tau^A$)}
OR {($A^B \neq A^A$) & ($\tau^B + C^B < \tau^A + C^A$)}
OR {($\tau^B + C^B = \tau^A + C^A$) & ($A^B > A^A$)}

This procedure steps through all subtasks in the mission. For each subtask agent A either keeps S^A or completely replaces S^A with S^B. The overwrite based on initialization times enables agents to share new tasks/plans initialized by the user or another agent.

D. Global vs. Local Information

As discussed above, $\hat{M}^k$ is the state of the mission as known by a specific agent *k*. It is interesting to consider the existence of a limiting mission state estimate. Suppose that time is stopped at time *t*, that communication between agents is unlimited, and that the communication algorithm between agents is allowed to run for some finite number of iterations *n*. Denote the resulting mission state estimate of agent *k* as $\hat{M}_n^k$. If given a set of *K* agents, there exists an *N* such that for all $n \geq N$,

$$\hat{M}_n^1 = \hat{M}_n^2 = \cdots = \hat{M}_n^{K-1} = \hat{M}_n^K := \overline{M}$$

then we call $\overline{M}$ the limit state of the mission at time *t*. Depending on the level of access to information, at any given time *t*, $\hat{M}^k$ may be very different from the limit state of the mission $\overline{M}$. We are currently investigating the existence of a limit state $\overline{M}$ as well as a number or upper bound on the number of iterations *N* required for the information to converge to $\overline{M}$.

III. AGENT DECISIONS

With an understanding of the information that is contained in a mission plan and a mission state estimate, the contents and functionality of an individual agent can be discussed. For each agent k, we define an internal state $\underline{X}^k$ as seen below where $\hat{M}^k$ is the mission state estimate as described previously. We also define the message sent by the k^{th} agent to all other agents as $\underline{Y}^k$.

$$\underline{X}^k = \begin{Bmatrix} AgentID \\ t = time \\ Dynamic\_State \\ Vehicle\_Type \\ \text{Re}\, sources\_Available \\ \hat{M}^k \end{Bmatrix} \quad \underline{Y}^k = \begin{Bmatrix} AgentID \\ \hat{M}^k \end{Bmatrix}$$

We now describe how an agent uses its internal state information to choose and execute subtasks.

A. Subtask Transition Execution

As a prelude to choosing a subtask, the status transition conditions (*complete, fault, redo*) are tested for each subtask in the mission plan. Which conditions are checked depends on the current status of the subtask, as shown in Figure 1. Whenever a condition is true, the corresponding transition is made, resulting in a change in the status of the subtask.

This is accomplished using the functions *Complete*($\underline{X}^k$), *Fault*($\underline{X}^k$), and *Redo*($\underline{X}^k$), which are described below.

1) Complete($\underline{X}^k$)

This function applies to any subtask S_{ij} that is currently assigned to the agent, that is when U_{ij}={assigned} and A_{ij}=*AgentID*. The following transition occurs when S_{ij} is complete according to the user defined transitions rules R_{ij}:

if *(complete(S_{ij}) & U_{ij}={assigned} & A_{ij}=AgentID)*
then *{U_{ij}={done}, A_{ij}=AgentID, τ_{ij}=t}*

One simple way to test completeness is by comparing the current cost estimate C_{ij} to a value C_{tol}, specified in R_{ij}:

$$C_{ij} \leq C_{tol}$$

This says that if the cost has been reduced to below the user specified value C_{tol}, then the subtask is complete.

2) Fault($\underline{X}^k$)

Similarly, for subtasks S_{ij} that are assigned to any agent (U_{ij}={assigned}), the fault transition occurs when S_{ij} is faulted according to the transition rules R_{ij}. For example, a user may specify the condition:

if($\tau_{ij} + \tau_{timeout} > t$)
then *{U_{ij}=todo, A_{ij}=AgentID, τ_{ij}=t}*

This says if the agent assigned to S_{ij} has not been heard from for a user defined τ_{timeout} seconds then we assume that agent has faulted and the status of the subtask must be set *todo*. The "lost" agent can reestablish control of the subtask in question if it is still the best agent for the subtask.

3) Redo($\underline{X}^k$)

For subtasks S_{ij} that are done (U_{ij}={done}), this transition occurs when the subtask should be redone according to the transition rules R_{ij}. For example a user may specify the condition:

if($\tau_{ij} + \tau_{cycle} > t$ & U_{ij}={done})
then *{U_{ij}=todo, A_{ij}=AgentID, τ_{ij}=t, I_{ij}=t}*

This says if a task has been in the done state for some user defined time, τ_{cycle}, then set the task *todo* so it will be executed again. This transition would be applicable to subtask such as a periodic surveillance of a point.

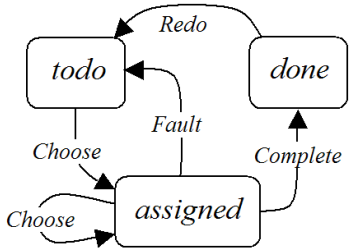

Figure 1. Subtask status transition diagram.

B. Subtask Choice

Given the internal state of an agent and an updated mission state estimate, the agent decision function Choose($\underline{X}^k$) is used to select a subtask for execution.

The agent first chooses a task, and then based on cost analysis chooses a feasible subtask within that task. A subtask is considered feasible if the agent has the ability to complete it. This may involve being a certain vehicle type, having a certain sensor type, having a sufficient amount of fuel, etc. The selection procedure is outlined below.

<u>*Choose Task*</u> The agent transitions into a suitable task, T_i, based on the user specified task transition rules, R_i. The main purpose of this step it to assign a capable team into the same task. *An example task transition rule might be: Switch into T_1 if 3 or less agents exist in T_1 and switch out of T_1 if more than 3 agents exist in T_1. In this example a capable team consists of 3 agents.*

<u>*Evaluate Feasible Subtasks*</u> For each S_{ij} within T_i, such that S_{ij} is feasible & $U_{ij} \neq$ done, determine the modified expected completion cost C'_{ij} according to the following rules:

- if U_{ij}={*todo*}, $C'_{ij} = C_{ij}$, where C_{ij} is the expected completion cost.
- if (U_{ij}={*assigned*} & A_{ij}=*AgentID*), $C'_{ij} = C_{ij}$ - H, where H > 0
- if (U_{ij}={*assigned*} & $A_{ij} \neq$*AgentID*), $C'_{ij} = C_{ij}$ + L, where L > 0

Note that when a task is already assigned, the cost values are modified using the values H and L in order to make either leaving a subtask or "stealing" one from another agent less attractive.

Select Subtask Find min(C'_{ij}) and denote the associated subtask as S_c. Then:

- If the agent was previously assigned to any subtask S_{ij}, then set $\{U_{ij}=\{\text{todo}\}, A_{ij}=AgentID, \tau_{ij}=t\}$.
- Set the values in S_c: $\{U_c=\{assigned\}, A_c=AgentID, \tau_c=t, C_c=C_{ij}\}$

C. Subtask Execution

Once an agent has an assigned subtask, that subtask still needs to be executed. This is done by additional functions that take the parameters of a chosen subtask and produce the desired behavior. If taking a picture of a position on the ground were one of these subtasks then a "take_picture" function would use the parameters to produce a series of waypoints to the objective and command a picture to be taken within some distance of the given position.

Each type of subtask has an associated planner function such that a subtask's parameters, P_{ij}, and the internal state of an agent contain all the information needed for completion of the subtask.

IV. Simulation Environment

Although the ideas above may be applied to any group of mobile agents, we will now consider an application to a group of unmanned aerial vehicles (UAVs). To test and validate our distributed system a MATLAB® simulation and visualization environment was developed. The simulation environment includes a kinematic aircraft model, a grid-based obstacle map and limited communication. The grid based obstacle map is used to represent a operation area with obstacles which then must be avoided by a path-planning algorithm. The simulation parameters are chosen to approximate the capability of the test aircraft we have at UC Berkeley [16].

A. Low Level Controller & Kinematic Model

If an aircraft operates in a flat plane, at constant speed and is considered to be small relative to the operating environment then a constrained 2D kinematic model is a reasonable assumption. The governing kinematic equations are:

$$\begin{aligned} \dot{x} &= V_{aircraft} * \cos(\psi) + V_{windx} \\ \dot{y} &= V_{aircraft} * \sin(\psi) + V_{windy} \\ \dot{\psi} &= u_1 \end{aligned} \tag{1}$$

where $V_{aircarft}$ is the constant velocity of the aircraft, ψ is the yaw angle and the control action u_1 is equal to the yaw rate (Figure 2). For the simulations the aircraft has a fixed velocity of 20 m/s, wind velocity is set to zero, and $\dot{\psi} \in [-0.2, 0.2] rad/\sec$.

The trajectory-tracking controller tracks waypoints given by the planner. The waypoint tracker controls toward a point, such as (x_2,y_2) in Figure 2, using proportional feedback:

$$u_1 = K_p * (\psi_{desired} - \psi_{actual}) \tag{2}$$

where $\psi_{desired}$ and ψ_{actual} are the angles as specified in Figure 2, K_p is the gain of the P-controller, and u_1 is the control action.

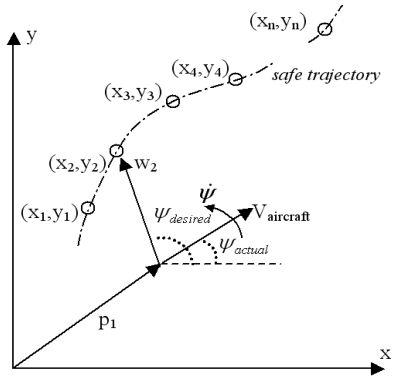

Figure 2. Waypoint tracker and kinematic model variable definitions

When the aircraft comes within some specified radius of the current waypoint the waypoint tracker switches to the next waypoint, such as (x_3,y_3) above, until it has reached its final waypoint.

B. Limited Communication & Broadcast Simulation

Our broadcast simulation is meant to represent a lower bound on performance we could obtain from a more refined system. There are much more extensive solutions available from the networking community; however, we believe that if our system works well with this representation, then it is likely to work with many other systems. However for a significantly large number of UAVs a more elaborate communication methodology would be required.

If a UAV broadcasts, it is not guaranteed that other UAVs will receive that broadcast. From an inspection of the capabilities of our experimental platform, we have developed equation 3. This is not meant to be a completely realistic simulation of a data network and is instead meant to limit information, vary the rate of information exchange, and change the order of communication.

The probability of a successful communication is given by

$$P(r) = \begin{Bmatrix} 1 & r < b * r_{\max} \\ \dfrac{1}{(4r/r_{\max})^2} & r \geq b * r_{\max} \end{Bmatrix} \tag{3}$$

where r is the Euclidian distance between UAVs, $r_{\max}$ is the

maximum communication radius and *b* is the percentage of the max communication radius that will result in guaranteed communication.

V. Example Mission

We will use an example simulation of mission M_1 to highlight some of the capabilities of our system. Figures 3-5 show the paths of six UAVs with their current positions represented by a circle. Clusters of small circles represent obstacles and when two UAVs communicate, a line is drawn between them. Finally, subtasks, which in this example consist of geographical locations, are represented by dots.

A 2000 second simulation of M_1 was run with no simulated UAV faults. A maximum communication radius of 2000 meters and operation area of 5000 meters x 5000 meters was specified for the simulation. The aircraft and communication model are as described in section IV. T_1 of M_1 can be seen in Figure 3.

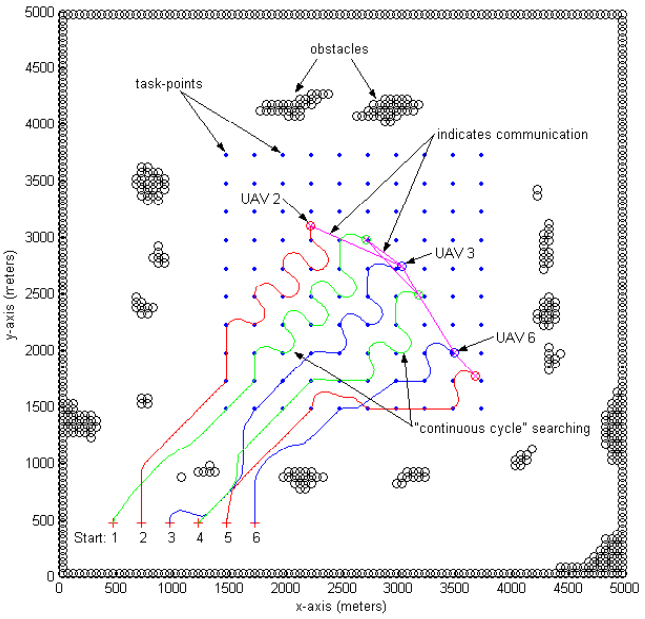

Figure 3. Six UAVs during M1, executing T1 @ t=250 seconds

The UAVs begin at their start positions and then disperse over the grid of subtasks represented by S_{1i} to complete T_1. T_1 results in a sweeping search of an area defined by $\{P_{11}, P_{12}, P_{13}, \ldots P_{1\,b1}\}$ and a set of rules defined by $\{R_{11}, R_{12}, R_{13}, \ldots R_{1\,b1}\}$. In this case the greater task T_1 contains a set of subtasks to visit point locations. One UAV is required to visit each point of a subtask at least once. After $\{S_{11}, S_{12}, S_{13}, \ldots S_{1\,b1}\}$ are all set as *done*, T_1 switches to T_2 as denoted by R_1.

The initial stages of T_2 can be seen in Figure 4 and show the UAVs after they have already dispersed toward one of the 7 subtasks of T_2. The rules applied to the subtasks of T_2 are the same as those applied to T_1 except for the introduction of time-based fault tolerance.

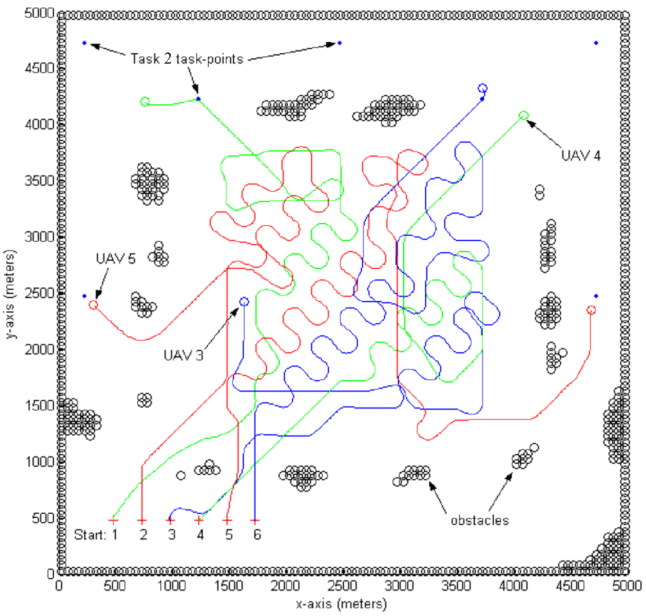

Figure 4. Six UAVs during M_1, executing T_2 @ t =620 seconds

With time-based fault tolerance, even if a failure were to occur out of range of any other functioning UAV, the subtask would still be completed.

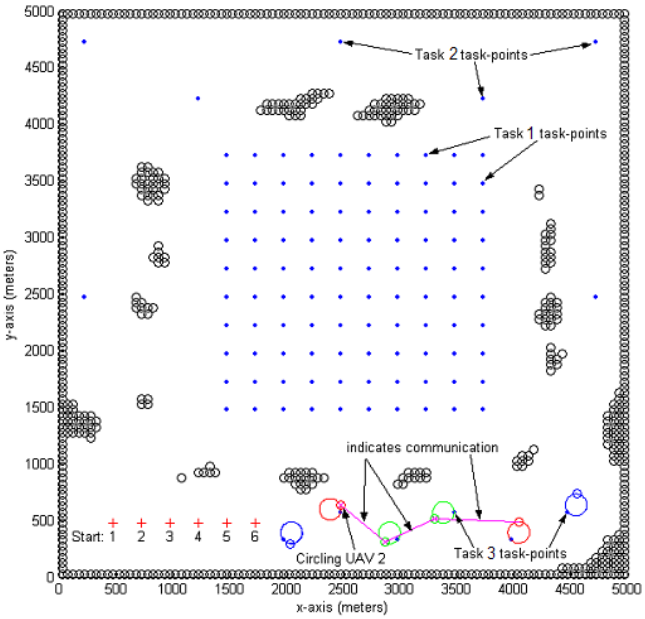

Figure 5. Six UAVs during M_1, and executing T_3 @ t =1199 seconds

Through the mission state a UAV can estimate the time at which a subtask should be completed. If this estimated time passes without confirmation that the subtask has been accomplished then the subtask is switched from *assigned→todo* by the fault decision function.

In addition, communication-based fault tolerance is applied to all subtasks of each task. If a UAV can communicate the knowledge of its failure to at least one other functional UAV then this knowledge would be integrated into the mission state estimate by the receiving

UAV. That is, any subtasks currently assigned to the faulted UAV would be reset from *assigned→todo* by the fault decision function.

Finally, Figure 5 displays all six UAVs circling at subtask points of the final task, T_3, in default mode. The visualization no longer displays the trail of each UAV, but it does display all the subtasks of T_1, T_2 and T_3 as their associate points. The default mode is actually the natural behavior of the system and is the result of no rules applied to the subtasks of T_3, that is $\{R_{31}, R_{32}, R_{33}, \ldots R_{3\,b3}\}$=NULL.

VI. Conclusions & Future Work

In this paper, we have presented a distributed approach to multi-agent collaboration. We first established a description of a mission plan in terms of tasks and subtasks. As a superset of the mission plan, we established a language in which to describe the estimated state of a mission at any time. This involved augmenting the information contained in the mission plan with the subtask status, agent identifier, times, and cost. Using this information we identified a methodology to synchronize different mission state estimates into one unique mission state estimate that represents the most valid information available. Finally, a functioning multi-agent system was implemented in a simulation environment and the results of an example mission were discussed.

In simulation, our system produces robust and consistent behavior despite uncertain communication links. It can execute different types of missions/tasks in a fault-tolerant way. In simulation, the number of collaborating UAVs is limited only by computer memory; in a real implementation, communication resources would likely be the limiting factor. With reasonable communication bandwidth, however, we believe that this system can be successfully implemented in real-time on a group of real aircraft.

In a distributed multi-agent system information is key. Clearly, an agent's performance is limited by the information it has. If all agents had complete information, then the group could theoretically achieve a globally optimal solution. However this is difficult to achieve in any real distributed multi-agent system. Therefore, the important information must be identified, described in a concise manner, and widely disseminated as efficiently as possible. With this information available, good decisions can be made.

Our current and future work is largely committed to further refinement and formalization of the mission description concepts and the development of formal proofs. We wish to find a minimum set of rules/conditions that are required to guarantee that a mission will be completed if the resources exist.

We also wish to prove the existence of the limiting mission state described in section II(D). We would also like to determine an upper bound on the number of communications required for convergence as it relates to the number of agents in a group. We can conclude from inspection that for a group of two agents only one communication is required to reach the limiting mission state.

Finally, while each agent only plans one-step ahead in the current system, we hope to introduce an algorithm that would allow each agent to plan multiple steps ahead.

References

[1] W. Ren, R. W. Beard, "Trajectory Tracking for Unmanned Air Vehicles with Velocity and Heading Rate Constraints," IEEE Transactions on Control Systems Technology, In Press.

[2] Y. Kuwata, Real-time Trajectory Design for Unmanned Aerial Vehicles using Receding Horizon Control, Masters Thesis, MIT, June 2003.

[3] I. M. Mitchell and S. Sastry, "Continuous Path Planning with Multiple Constraints," IEEE Conference on Decision and Control, Hawaii, USA, December 2003

[4] S. Spry and J. K. Hedrick, "Formation Control Using Generalized Coordinates," IEEE Conference on Decision and Control, Bahamas, 2004.

[5] R. O. Saber and R. M. Murray, "Flocking with Obstacle Avoidance: Cooperation with Limited Communication in Mobile Networks," IEEE Conference on Decision and Control, Hawaii, USA, December 2003.

[6] H. Tanner, A. Jadbabaie, and G. J. Pappas., "Coordination of multiple autonomous vehicles," IEEE Mediterranean Conference on Control and Automation, Rhodes, Greece, June 2003.

[7] S. G. Breheny, R. D'Andrea and J. C. Miller, "Using Airborne Vehicle-Based Antenna Arrays to Improve Communications with UAV Clusters," IEEE Conference on Decision and Control, Hawaii USA, December 2003.

[8] A. R. Girard, A. S. Howell, and J. K. Hedrick, "Border Patrol and Surveillance Missions using Multiple Unmanned Air Vehicles", Submitted to IEEE Control Systems Technology, 2004.

[9] L.E. Parker, "ALLIANCE: An architecture for fault-tolerant multi-robot cooperation," IEEE Transactions on Robotics and Automation, 14(2), pp. 220-240, April 1998.

[10] R. Davis and R.G. Smith, "Negotiation as a metaphor for distributed problem solving," Artificial Intelligence, Vol. 20, pp.63-109, 1983.

[11] B.P. Gerkey and M.J. Mataric, "Sold! Auction Methods for Multirobot Coordination," IEEE Transactions on Robotics and Automation, 18(5), pp. 758-768, Oct. 2002.

[12] J. Kennedy and R. C. Eberhart, Swarm Intelligence, Academic Press, 2001.

[13] K. Konolige, D. Fox, C. Ortiz, et al., "Centibots: Very large scale distributed robotic teams," Proc. of the Intl. Symposium on Experimental Robotics, ISER 2004.

[14] S.C. Spry, A.R. Girard, and J.K. Hedrick, "Convoy Protection using Multiple Unmanned Aerial Vehicles: Organization and Coordination," Proc. of the 24th American Control Conference, Portland, OR., June 2005

[15] M. Godwin, S. Spry, and J.K. Hedrick, "A Distributed System for Collaboration and Control of UAV Groups: Experiments and Analysis," Proc. of the 5th International Conference on Cooperative Control and Optimization, Gainesville, FL, Jan. 2005.

[16] E. Frew, X. Xiao, S. Spry, T. McGee, Z. Kim, J. Tisdale, R. Sengupta, and J.K. Hedrick, "Flight Demonstrations of Self-Directed Collaborative Navigation of Small Unmanned Aircraft," Proc. of the 3rd AIAA Unmanned Unlimited Technical Conference, Workshop, & Exhibit, Chicago, IL, Sept. 2004.