Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

Cooperative Control via Task Load Balancing for Networked Uninhabited Autonomous Vehicles

Jorge Finke
Dept. Electrical Engineering
The Ohio State University
Columbus, OH 43210
finkej@ee.eng.ohio-state.edu

Kevin M. Passino*
Dept. Electrical Engineering
The Ohio State University
Columbus, OH 43210
k.passino@osu.edu

Andrew Sparks
AFRL/VACA
Wright Patterson AFB, OH 45433
Andrew.Sparks@wpafb.af.mil

***Abstract*— In this paper we first define a mathematical model of the "plant" for the cooperative control problem for multiple uninhabited autonomous vehicles (UAVs). This includes a representation for the vehicles, environment, and communication network. Next, we define an approach to cooperative control that uses local UAV task planning and multi-UAV coordination via task load balancing over the communication network. Our approach is motivated by our desire to cope with significant imperfections in the communication network and uncertainty in the environment, and yet provide a scalable strategy which can be implemented in real time across a network of UAVs, each of which only has relatively low processing power. Our focus in this paper is on defining the cooperative controller problem in a mathematical framework that is familiar to a control engineer, studying properties of a load balancing strategy for cooperation, and then via simulations to identify key challenges when network influences dominate the problem.**

I. INTRODUCTION

Given multiple uninhabited autonomous vehicles connected via a communication network there is a need to develop decentralized decision-making methods so that the group can cooperatively decide "who should do what and when they should do it." For instance, without human intervention they must coordinate the best way to search for targets by optimally exploiting a priori target location information, and as targets are found, dynamically agree on which UAVs to send to perform the subsequent tasks of classification, engagement, and verification [1]-[4]. This must be done with as little communication between the vehicles as possible and considering all major network effects such as topology variations and delays.

In [5]-[7] receding horizon approaches are considered for cooperative control. For some cooperative control problems such methods can be very useful. The approach, however, differs from ours in part because our focus is on the case where network constraints and uncertainty dominate the problem to the extend that optimal cooperative planning of multiple UAV activities far into the future in real time is not feasible or even useful.

Other methods that are being used in cooperative control include what one might call the "map-based approaches" like in [8]-[12] that provide some advantages since they can be used for probabilistic frameworks and convenient incorporation of a priori knowledge. Our work assumes that there is some prior knowledge about the terrain, but not in a detailed form like the "ROR map" [9] or other such maps (e.g., "threat maps" [12]). We simply assume that we have certain regions we are interested in searching called "search-points."

When considering uncertain environments there are different types of uncertainties one can study. We could for example account for uncertainties in the environment by considering (i) the loss of part of the UAV fleet as suggested in [13] (e.g., UAVs might crash or be destroyed), (ii) communication network imperfections, (iii) the uncertainty about which search-points the UAVs need to visit, (vi) uncertainty arising from the presence of "false targets" [14], and (v) the uncertainty about what tasks to perform once these search-points are reached (e.g., we may not need to attack after a classification). This task uncertainty determines how accurately we can predict ahead in an uncertain environment. Our approach focuses on the type of uncertainty encountered when search-points are reached and new tasks like classification or engagement need to be scheduled on-line since there is high uncertainty a priori about whether any given task beyond search needs to be performed. We consider the effects of the communication channel via unknown but bounded delays involved in communicating the status of the targets from one UAV to another. At this time, we are not considering communication topology influences beyond what could be modeled by such delays. The delays we consider should not be thought of as rising only from delays on network links, but also from processing delays (e.g., from image processing or coordinated operation with a human), occlusions and sensing/communication range constrains, or temporary loss of a communication link. The delays we use can model each of these.

In summary, our focus is on trying to exploit distributed load balancing algorithms [15], [16] to achieve inter-vehicle cooperation. We take a conventional control-theoretic modeling and analysis approach and use Monte Carlo simulations to give insight into design trade-offs.

II. PLANT MODEL

We include the vehicles, targets/threats ("objects"), sensors, actuators, and communication network in the plant model. The resulting model is "hybrid" and stochastic since the vehicle dynamics result from a continuous time system, and yet other aspects are best represented by nondeterministic automata-type representations (e.g., tasks, task orderings/status, and sensor modes).

A. Vehicle and Object Models

Suppose that there are N_v UAVs and that the i^{th} one obeys a continuous time kinematic model given by

$$\begin{aligned} \dot{x}_{v1}^i &= v\cos(\theta_v^i) \\ \dot{x}_{v2}^i &= v\sin(\theta_v^i) \\ \dot{\theta}_v^i &= w u_v^i \end{aligned} \tag{1}$$

where x_{v1}^i is its horizontal position, x_{v2}^i is its vertical position, v is its (constant) velocity, θ_v^i is its orientation, w is its maximum angular velocity, and $-1 \leq u_v^i \leq 1$ is the steering input. Hence, $u_v^i = +1$ stands for the sharpest possible turn to the right, and analogously $u_v^i = -1$ represents the maximum possible left turn.

*This work was supported by the AFRL/VA and AFOSR Collaborative Center of Control Science (Grant F33615-01-2-3154). More information about CCCS activities can be found at http://eewww.eng.ohio-state.edu/~passino. Also, the authors would like to acknowledge the inputs of P. Chandler, J. Layne, S. Rasmussen, and C. Schumacher of AFRL.

The minimum turn radius for the vehicles can then be defined as $R = \frac{w}{v}$. Rather than use this continuous time representation we assume that vehicles will either travel on the minimum turn radius or on straight lines. It is then possible to analytically write down the formulas for the vehicle trajectories (e.g., in terms of arc segments on circles and line segments). Next, we quantize these trajectories with a sampling interval T to obtain discrete time sequences that we denote by $x^i_{v1}(k), x^i_{v2}(k), \theta^i_v(k)$, and $u^i_v(k)$ for $k = 0, 1, 2, \ldots$. Then, for a given initial $x^i_{v1}(k), x^i_{v2}(k)$, and $\theta^i_v(k)$ and a desired location and heading, the code will generate trajectories between these two points, which are minimum time/distance trajectories.[1] For convenience, let $x^i_{vp} = \left[x^i_{v1}, x^i_{v2}\right]^\top$, $x^i_v = \left[(x^i_{vp})^\top, \theta^i_v\right]^\top$, $x_v = \left[(x^1_v)^\top, \ldots, (x^{N_v}_v)^\top\right]^\top$, and $u_v = \left[u^1_v, \ldots, u^{N_v}_v\right]^\top$.

The environment is modeled as a two-dimensional plane. It is assumed that at most N objects are in this square region, but initially we do not know where they are at. In the environment we assume that there can be a variety of targets, threats, entities that are both targets and threats, and other entities that may be neither targets nor threats. For convenience we will refer to all these as "objects" of different types. Whether an object is a target or a threat depends on mission objectives and this can be specified in the controller (e.g., along with a "priority" for a target) or perhaps by a human operator. The j^{th} object has characteristics specified by its state which is

$$x^j = \left[x^j_1, x^j_2, \theta^j, o^j, d^j\right]^\top$$

where x^j_1, x^j_2, and θ^j are the horizontal position, vertical position, and orientation of the j^{th} object in (x_1, x_2) coordinates. It is assumed that the objects are at distinct points (i.e., there are no two objects that are exactly at the same location, but with the same or different orientations since then we can number the objects and thereby uniquely identify them). The object type is represented by o^j, with o^j a number representing the type of object for the j^{th} object. Finally, $d^j \in [0, 1]$ represents the amount of damage to an object due to an attack, with $d^j = 1$ representing that an object is completely destroyed. For convenience, let $x^j_p = \left[x^j_1, x^j_2, \theta^j\right]^\top$ and $x = \left[(x^1)^\top, \ldots, (x^N)^\top\right]^\top$.

B. Sensing, Tasks, and Actions

UAV sensors to be defined next return the position, orientation, and other aspects of the objects but do not know the indexing $j = 1, 2, \ldots, N$ of the last subsection. Hence, UAV i will number the objects that it finds, in the order it finds them, by $j_i = 1, 2, \ldots, N$. If more than one object is found by a sensor at the same time then the UAV just orders them randomly.

Each UAV has a sensor that can be commanded to operate in different modes. Let

$$x^i_s = \left[(p^i_s)^\top, (p^i_c)^\top, (p^i_a)^\top, (p^i_v)^\top\right]^\top$$

define the sensor state of UAV i, where $p^i_s = \left[p^{i1}_s, \ldots, p^{iN}_s\right]^\top$ and similar for p^i_c, p^i_a, and p^i_v. The values of $p^{ij_i}_s, p^{ij_i}_c, p^{ij_i}_a, p^{ij_i}_v \in [0, 1]$ are the levels of search, classification, attack, and verification certainty (probability), respectively, by the i^{th} UAV for the j_i^{th} object. There is a type of classification that is done for search so that upon searching an area we will gain information about both $p^{ij_i}_s$ and $p^{ij_i}_c$. The value of $p^{ij_i}_c$ is further modified by a UAV that revisits an object for classification. The value of $p^{ij_i}_a$ and $p^{ij_i}_v$ are set during attack and verification of an object, respectively. Hence, $p^{ij_i}_s = p^{ij_i}_c = p^{ij_i}_a = p^{ij_i}_v = 0$ would correspond to the sensor on UAV i saying that the probability that object j_i was found is zero, and the classification, attack and verification was unsuccessful, respectively. If $p^{ij_i}_s = p^{ij_i}_c = p^{ij_i}_a = p^{ij_i}_v = 1$ it means that for UAV i object j_i was definitely found, object classification is certain, attack is certain, and that verification is certain. Finally, note that initially we use "$-$" as a symbol for "not known" and any element of p^i_s, p^i_c, p^i_a, or p^i_v could hold such a value. For instance, at $k = 0$, $p^{ij_i}_s = p^{ij_i}_c = p^{ij_i}_a = p^{ij_i}_v =$"$-$" since nothing has been sensed. Let $x_s = \left[(x^1_s)^\top, \ldots, (x^N_s)^\top\right]^\top$. We also model the sensor footprint and how tasks are completed for each task type, which includes consideration of uncertainty.

C. Communication Network

Here, we assume that communication links and the overall topology are fixed and that the only imperfection on a link is a possible delay in transmitting information (but as mentioned earlier we view this delay as also modeling other sources of communication imperfections). Note that we consider the communication network to be part of the plant since we view communication as sensing (receiving data) and taking actions (transmitting data) and earlier we considered similar characteristics to be part of the plant. The communication network topology is defined via a directed graph $G = (V, A)$, where $V = \{1, 2, \ldots, N_v\}$ is the set of nodes (the vehicles) and $A = \{(i, i') : i, i' \in V\}$ is a set of directed arcs representing the communication links. If $(i, i') \in A$ this represents that vehicle i can send vehicle i' information (which sometimes may be thought of as vehicle i' being able to sense information from vehicle i). For all $(i, i') \in A$, we assume that $i \neq i'$ since we assume that any local information on a UAV is known to the UAV and hence does not need to be transmitted. Let $A^s_i = \{i' : (i, i') \in A\}$ and $A^r_i = \{i' : (i', i) \in A\}$ be the set of UAVs that UAV i can send messages to (receive messages from, respectively).

Next, we must represent how information is passed from one node to another. For representation simplicity we will simply view each communication link (i, i') as a memory that holds the last $D^{ii'}$ values of the vectors of information transmitted between UAVs i and i'. Denote the vector of information that UAV i transmits UAV i' at time k as $x^{ii'}_n(k)$, $(i, i') \in A$. A delay of up to $D^{ii'}$ units on a link can be represented by having each link have a state vector

$$x^{ii'}_L(k) = \begin{bmatrix} x^{ii'}_n(k-1) \\ \vdots \\ x^{ii'}_n(k - D^{ii'}) \end{bmatrix}$$

Let $x^i_L(k)$ be a vector of $x^{i'i}_L(k)$ for all $(i', i) \in A$ so that it represents all information received by UAV i via communication links. Let $x_L = \left[(x^1_L)^\top, \ldots, (x^{N_v}_L)^\top\right]^\top$. The specific form of $x^{i'i}_n(k)$ depends on the design of the cooperative controller. It can hold $x^{i'}_v(k), x^{i'}_s(k), \hat{x}^{i'm}(k), u^{i'}(k)$, and $u^{i'}_v(k)$ for $i' \neq i$ and for the m^{th} object, where $\hat{x}^{i'm}$ and $u^{i'}$ will be defined in the next section. For convenience in the notation we indicate retransmission of an entire vector when only some or no information in that vector changes.

To model a deterministic fixed delay $\tau^{ii'}, 1 \leq \tau^{ii'} \leq D^{ii'}$ between UAV i and UAV i' we will only allow the receiving node i' to pull the delayed value $x^{ii'}_n(k - \tau^{ii'})$ off the link at time k.

[1]Here, we used the development by AFRL/VA COECS and the code from the public release of their multi-UAV simulation to generate the optimal path trajectories in MATLAB.

An unknown but bounded delay could be represented via a random choice of which element to pick from the past vectors that are currently stored (i.e., random choice of $\tau^{ii'}, 1 \leq \tau^{ii'} \leq D^{ii'}$); this then represents that the messages that are sent can be delayed and out of order upon arrival.

D. Observation and State Equations

Next, we define the sensor modes, attack mode, and the information that is gathered from sensing and communications. To do this we first define the observation map for UAV i

$$y^i(k) = h^i\left(x_v^i(k), x(k), x_s^i(k), x_L^i(k), u^i(k), w_o(k)\right) \quad (2)$$

where $y^i(k)$ is the sensed information at time k for UAV i, $y = \left[\left(y^1\right)^\top, \ldots, \left(y^{N_v}\right)^\top\right]^\top$, and $u = \left[u^1, \ldots, u^{N_v}\right]^\top$.

Here, $u^i \in \{s, c, a, v, n\}$ indicates that UAV i should search, classify, attack, verify, or do nothing. The UAV controllers will choose these u^i values, and then move to the appropriate location to perform the task. The case where $u^i(k) = n$ typically only arises at the end of a mission when there are no remaining tasks so a UAV may choose to do nothing and move back to its home (which we consider to be at $(0,0)$). Let

$$y^i(k) = \left[\left(x_v^i(k)\right)^\top, \left(\hat{x}^i(k)\right)^\top, \left(x_s^i(k)\right)^\top, \left(\hat{x}_L^i(k)\right)^\top\right]^\top \quad (3)$$

so that each UAV knows its own position and orientation $x_v^i(k)$ (e.g., it may obtain this via onboard sensors and GPS), and sensed information about objects via its own sensors $x_s^i(k)$. Also, we use $\hat{x}^{ij_i}(k)$ to represent what UAV i measures at time k about object j_i and $\hat{x}^i = \left[\left(\hat{x}^{i1}\right)^\top, \ldots, \left(\hat{x}^{iN}\right)^\top\right]^\top$. If there is uncertainty in the sensing, the $w_o(k)$ can be used to represent sensor inaccuracies in search (e.g., measurement errors in object position and orientation and errors in classification during detection), classification (errors in measuring object type), and verification (errors in measuring damage level). Notice also that $\hat{x}_L^i(k)$, the vector of information received from any UAV i such that $(i', i) \in A$ at time k is the measured value of $x_L^i(k)$, and is part of the measured output for UAV i. It is via the output map h^i and $w_o(k)$ that we represent network characteristics discussed above. For example, random but bounded delays $1 \leq \tau^{i'i} \leq D^{i'i}$ on link $(i, i') \in A$ would be held in $w_o(k)$ and $\hat{x}_L^i(k)$ is a vector of $x_n^{i'i}(k - \tau^{i'i})$, i.e., received values from all UAVs i' connected on the network to UAV i, but delayed by $\tau^{i'i}$.

Next, we define how the plant state evolves. The state has four parts, $x_v(k), x(k), x_s(k)$, and $x_L(k)$. We have already explained how to generate $x_v(k+1)$. Next, we will, in turn, explain how to generate $x(k+1), x_s(k+1)$, and $x_L(k+1)$. Suppose that

$$x(k+1) = f(x_v(k), x(k), x_s(k), x_L(k), u(k), u_v(k), w_s(k)) \quad (4)$$

where $w_s(k)$ is a vector that will have components that will be used to represent uncertainty in sensing and attacks (no uncertainty is used for the communication links in the state update; uncertainty in communications is represented in the observation equation above). To define $x(k+1)$ first note that if we assume that the objects are stationary the positions and orientations of all objects stay the same (of course "pop-up" or moving objects could be modeled). We also assume that the objects types do not change (but they could if you consider, e.g., current vehicle configuration to be part of the object type). All that remains are the damage levels of the objects $d^j(k+1)$ and f must represent this. Here, when any UAV i is in the correct position and orientation (defined earlier) it may attack object j at time k and

$$d^j(k+1) = d^j(k) + \sum_{i=1}^{N_v} d^{ij}(k)$$

where $d^{ij}(k)$ is the amount of damage inflicted at time k on object j by UAV i (the $d^{ij}(k)$ must be defined so that $d^j(k) \in [0,1]$ for all k). The d^{ij} may be a fixed percentage of $d^j(k)$ or may contain a random component that we can represent via a component of $w_s(k)$. Alternatively it may be that $d^j(k) = 1$ after a fixed number of attacks. Notice that we model the possibility of simultaneous attacks.

Next, consider how to define $x_s^i(k+1)$, the update of $p_s^i(k), p_c^i(k), p_a^i(k)$ and $p_v^i(k)$. There are several ways to represent how these values evolve dynamically depending on the level of uncertainty that is present in the problem that is being considered. Here we consider the case where there is a small enough amount of uncertainty with respect to task completion, so that we do not have to repeat tasks but high enough uncertainty so that upon completion of a task a UAV only knows whether another task needs to be completed for the object, and what task is needed. Hence, upon sensing object j_i UAV i will have $p_s^{ij_i} = 1$ or $p_s^{ij_i} = 0$ representing its confidence in whether or not it found object j_i. Regardless, we consider the task of searching the region in which object j_i was found to be completed. We consider search to have given a classification during the search process $p_c^{ij_i} = 0$ for object type o^{j_i} indicating detection of object type, but that a classification must be done since it is completely uncertain about object type. *Upon classification* we assume that either $p_c^{ij_i} = 0$ or $p_c^{ij_i} = 1$ depending on the object type so the classification always succeeds. Classification is assumed to perfectly distinguish between the true and false targets and appropriately indicate whether to attack ($p_c^{ij_i} = 0$ for a false target and it is not attacked, and $p_c^{ij_i} = 1$ for a true target and it is attacked). Upon attack we have with equal probability $p_a^{ij_i} = 1$ representing a very successful attack so there is no need for a verification, or $p_a^{ij_i} = 0$ representing that there is uncertainty if the attack was good, so there is a need for verification. A verification is performed with the result that $p_v^{ij_i} = 1$ and $d^j(k)$ is sensed and stored for post-mission analysis. Hence, once an object is found by search and classified as a true target it is attacked. If there is uncertainty in how good the attack was, the target is then verified but then ignored for the remainder of the mission. The initial object type distribution and the effectiveness of the attacks will determine how many tasks must be completed to finish the mission.

Note that there needs to be next state functions for $x_s^i(k+1)$ for each UAV appropriately defined depending on sensor uncertainty characteristics. Then, for example, two UAVs may sense the same object and they may have different levels of uncertainty about each object.

Next we must define how $x_L(k+1)$ is generated. Here, we simply assume that given $x_L^{ii'}(k)$,

$$x_L^{ii'}(k+1) = \left[\left(x_n^{ii'}(k)\right)^\top, \ldots, \left(x_n^{ii'}(k - D^{ii'} + 1)\right)^\top\right]^\top$$

so that we simply shift the values at each time step. This represents that at time k we can get new transmitted information, and hence each UAV can get an updated received value from each UAV it communicates with.

III. Task Load Balancing for Cooperative Control

According to the model, as the mission progresses UAVs sense and react to the environment and thereby obtain new tasks and complete others. Here we assume that the UAVs cooperate by sharing the work to complete the tasks, by passing tasks over the network and other relevant information. Each UAV schedules its own tasks and the schedule and any unscheduled tasks are considered to be the task "load" of the UAV. Via the network links UAVs may exchange tasks to "balance" the load to try to make sure that all UAVs have tasks to perform (i.e., so there is no "underutilization" of any UAV) and so that the UAVs finish their mission at nearly the same time (which forces cooperation since there is pressure for no UAV to go home until every other UAV has only one more task to complete). Why use this type of cooperative controller? Our design decisions were driven by the need to (i) cope with uncertainty arising from both an imperfect communication network and the uncertain results of task completion (here we use the "uncertainty reduction" characterization in Section 2), (ii) keep the computational complexity of local UAV scheduling algorithms low so that they do not demand too much processing power on a UAV, and (iii) obtain a flexible and scalable approach applicable to small or large groups of UAVs.

The local controller for the i^{th} UAV can be modeled as a dynamical system

$$\begin{aligned} x_c^i(k+1) &= f_c^i\left(x_c^i(k), u_c^i(k)\right) \\ y_c^i(k) &= h_c^i\left(x_c^i(k)\right) \end{aligned} \qquad (5)$$

Recall that we assumed the existence of a global clock via GPS; hence all UAVs use the same time indices. Next, we define each variable of this model.

A. Local UAV Controller State and Command Input

First, let $u_c^i = y^i$ from Equation (3) so all measured local information is available for making decisions. Let $t_m^i(k) = (\ell, j_i)$ represent that task type ℓ is scheduled to be performed on search-point or object j_i at step m by UAV i. We define the state of the local controller

$$x_c^i(k) = \begin{bmatrix} T_s^i(k) \\ T_u^i(k) \\ x_a^i(k) \end{bmatrix}$$

which is composed of the schedule for UAV i which is an ordered list $T_s^i = (t_1^i, \ldots, t_{N_t}^i)$ at time k (e.g., $t_1^i(k)$ is the task on the schedule at time k that is currently being performed using x_v^i from u_c^i), $T_u^i(k)$ that is the *set* of unscheduled tasks, and $x_a^i(k)$ (with components $x_a^{ij_i}$) that is the accumulated information on the objects sensed by UAV i or ones that other UAVs communicated to UAV i. Next, we define the initial state $x_c^i(0)$. The list $T_s^i(0)$ is the initial schedule and for convenience here we assume that the only task type that is scheduled initially is a search task. It is assumed that the search tasks are initially set via N_v-TSP to define all the components of $T_s^i(0)$ and $T_u^i(0)$, given a priori specified search-points. Clearly this could be generalized to allow specification of other task types if more a priori information is known (e.g., an object location is known but a classification needs to be done). It is assumed that the set of search-points is given by S, where $|S| = \bar{M}$ and an element in S holds search-point position and orientation for search (e.g., by gridding the area via sensor footprints we can define S). This set S is fixed so it is not part of the state. It is simply used in $t_m^i = (1, j_i)$, the search task, to specify where it needs to do a search task. It is assumed that all N_v UAVs hold the same S. The initial state $x_c^i(0)$ also defines $x_a^i(0) = [-, \ldots, -]^\top$ so that no information on object location has been sensed. We assume that all UAVs know the location of their "home base."

For the tasks of classification, attack, and verification ($\ell = 2, 3, 4$), to specify the meaning of $t_m^i = (\ell, j_i)$ we need information that is sensed during the mission about object j_i (i.e., we need $\hat{x}^{ij_i}(k)$ information from y_i). The information $\hat{x}^{ij_i}(k)$ comes in asynchronously as information is sensed about the environment. This needs to be stored in the controller to keep track of all information that UAV i has gathered about each object j_i. Let x_a^i denote this accumulated information at time k that is stored in the controller for all j_i. Moreover, other UAVs may send tasks and the associated object information to UAV i and this will also be accumulated in x_a^i. More discussion on this point follows below.

Next, we change $T_s^i(k) = (t_1^i(k), \ldots, t_{N_t}^i(k))$ if $t_1^i(k)$ was completed at time k. In particular, we let

$$T_{sc}^i(k) = \begin{cases} (t_2^i(k), \ldots, t_{N_t}^i(k), \emptyset) & \text{if } t_1^i(k) \text{ is completed} \\ T_s^i(k) & \text{otherwise} \end{cases} \qquad (6)$$

where $\emptyset$ is the null task and we assume that the task ordering is appropriately renumbered. When is $t_1^i(k) = (\ell, j_i)$ completed? It is completed at time k if for $\ell = 1$, $p_s^{ij_i}(k) \in \{0,1\}$; for $\ell = 2$, $p_c^{ij_i}(k) \in \{0,1\}$; for $\ell = 3$, $p_a^{ij_i}(k) \in \{0,1\}$; or for $\ell = 4$, $p_v^{ij_i}(k) \in \{0,1\}$. The $p_s^{ij_i}, p_c^{ij_i}, p_a^{ij_i}$, and $p_v^{ij_i}$ information comes from $x_s^i(k)$ in the $u_c^i = y^i$ vector.

Next, let $x_{Lr}^i(k) = \hat{x}_L^i(k)$ hold the set of all received information at time k,

$$x_{Lr}^i(k) = \begin{bmatrix} x_{Lr}^{i'i}(k) \\ \vdots \\ x_{Lr}^{i''i}(k) \end{bmatrix} = \begin{bmatrix} x_n^{i'i}\left(k - \tau^{i'i}(k)\right) \\ \vdots \\ x_n^{i''i}\left(k - \tau^{i''i}(k)\right) \end{bmatrix}$$

where $A_i^r = \{i', \ldots, i''\}$ and $1 \leq \tau^{i'i}(k+1) \leq \tau^{i'i}(k) + 1 \leq D^{i'i}$ so that messages are not received out of order. Notice that the delays $\tau^{i'i}(k)$ can be different for different communication links and that information can arrive from different UAVs at the same time. Let $x_{Ls}^i(k)$ be the set of all the information sent by UAV i at time k defined by,

$$x_{Ls}^i(k) = \begin{bmatrix} x_{Ls}^{ii'}(k) \\ \vdots \\ x_{Ls}^{ii''}(k) \end{bmatrix} = \begin{bmatrix} x_n^{ii'}(k) \\ \vdots \\ x_n^{ii^*}(k) \\ \vdots \\ x_n^{ii''}(k) \end{bmatrix}$$

where $A_i^s = \{i', \ldots, i''\}$ and we assume that at any time k a UAV may pass at most one task to only one of its neighboring UAVs, and it will be put in $x_n^{ii^*}(k)$. Suppose that we denote such a task by $t^{ii^*}(k) = (\ell, j_i)$ which is the task passed from UAV i to UAV i^* such that $(i, i^*) \in A$.

Next, given the information from $y^i(k)$, in particular $x_s^i(k)$, we can define the new tasks that arrive at the network of UAVs via UAV i at time k, which we denote $T_{nu}^i(k)$. Here, let the "new" set of unscheduled tasks be ones that arrive via sensing and actions (information from $x_s^i(k)$), or the network via $x_{Lr}^i(k)$, combined with unscheduled tasks from the last step $T_u^i(k)$. Hence, we think of new tasks that arrive over the network in the same way as we do of new tasks arising from a UAV using its own sensors and taking actions. In this way the UAVs are cooperatively sharing sensed information. UAV i will also pass the location information about object j_i, its current schedule $T^i_{sc}(k)$, and its set of new unscheduled tasks $T^i_{nu}(k)$. Hence the messages sent over the network all have the form

$$x_n^{ii^*}(k) = \begin{bmatrix} t^{ii^*}(k) \\ x_a^{ij_i}(k) \\ T^i_{sc}(k) \bigcup T^i_{nu}(k) \end{bmatrix}$$

Also, for $i' \neq i^*$, $i' \in A^s_i$

$$x_n^{ii'}(k) = \begin{bmatrix} \emptyset \\ - \\ T^i_{sc}(k) \bigcup T^i_{nu}(k) \end{bmatrix}$$

so that at each time instant each UAV transmits to its neighbors its current task load since this is used in load balancing.

Here,

$$y^i_c(k) = h^i_c(x^i_c(k))$$

In particular, the output

$$y^i_c(k) = \left[u^i_v(k), u^i(k), x^i_{Ls}(k)\right]$$

so it holds the commands on how to move the vehicle, what task it should perform at each step, and the information that should be transmitted to its neighboring UAVs at time k. Here, if we have $t^i_1(k)$ then $u^i_v(k)$ and $u^i(k)$ are specified.

B. Task Scheduling, Balancing, and Passing

To complete the definition of $x^i_c(k+1)$ we need to define $t^{ii^*}(k)$, $T^i_s(k+1)$, and $T^i_u(k+1)$. Let $|\cdot|$ denote the length of a list, the size of a set, or for scalars, the absolute value.

We will first assume that each UAV will have only one locally scheduled task, that is $|T^i_s(k)| = 1$ for all k and for all i if there is a task available. Below we will also consider the case where $|T^i_s(k)| \geq 1$. If $T^i_{nu}(k) \neq \emptyset$, there is at least one task in the unscheduled set of UAV i that may stay unscheduled in $T^i_u(k+1)$, or be sent to any of its neighboring UAVs. If $|T^i_{nu}(k)| \geq 1$ we will choose element $t^i_{nu} \in T^i_{nu}$ to be this one task. UAV i^* represents the UAV with the smallest task load and is given by

$$i^* = \arg\min \left\{ \Omega \left(T^{i'}_{sc}(k - \tau^{i'i}) \bigcup T^{i'}_{nu}(k - \tau^{i'i}) \right) \right\}$$

for $i' = 1, \ldots, N_v$ (we use union to form a set from the elements in a list and a set) where Ω is a measure of the length of the task schedule of UAV i' (e.g., it could be the number of tasks on $T^{i'}_{sc} \bigcup T^{i'}_{nu}$ or the path length to execute the tasks on $T^{i'}_{sc} \bigcup T^{i'}_{nu}$). Then, if $i \neq i^*$

$$\begin{aligned} t^{ii^*}(k) &= t^i_{nu}(k) \\ T^i_s(k+1) &= \begin{cases} T^i_{sc}(k) & \text{if } T^i_{sc}(k) \neq \emptyset \\ F_{tsp}\left(T^i_{nu}(k) - \{t^i_{nu}(k)\}\right) & \text{otherwise} \end{cases} \\ T^i_u(k+1) &= T^i_{nu}(k) - \left\{t^i_{nu}(k)\right\} \end{aligned}$$

otherwise, if $i = i^*$

$$\begin{aligned} t^{ii^*}(k) &= \emptyset \\ T^i_s(k+1) &= \begin{cases} T^i_{sc}(k) & \text{if } T^i_{sc}(k) \neq \emptyset \\ F_{tsp}\left(T^i_{nu}(k)\right) & \text{otherwise} \end{cases} \\ T^i_u(k+1) &= T^i_{nu}(k) \end{aligned}$$

where the element in $T^i_s(k+1)$ is determined by having the TSP algorithm, which we denote by F_{tsp}, return the first element of an optimal sequence of performing all unscheduled tasks. Also, in these equations $t^i_{nu}(k)$ is the last scheduled task given by $F_{tsp}\left(T^i_{nu}(k)\right)$.

If $T^i_{nu}(k) = \emptyset$, then

$$\begin{aligned} t^{ii^*}(k) &= \emptyset \\ T^i_s(k+1) &= T^i_{sc}(k) \\ T^i_u(k+1) &= T^i_{nu}(k) \end{aligned}$$

which arises when nothing has been sensed or attacked, and no new tasks have been received over the network. If $T^i_s = T^i_u = \emptyset$ UAV i returns to the home base. This completes the definition of $x^i_c(k+1)$.

IV. Closed-Loop System Properties

We assume a fully connected network of N_v UAVs. Our analysis will not consider the arrival or departure of tasks from the system (e.g., UAVs finding targets that need to be classified or tasks that are completed). Therefore, in this section we assume that $T^i_s(k)$ and $T^i_u(k)$ are constant for all k; hence, we consider only the balancing that occurs after a task arrival or departure. This is valid if balancing occurs relatively fast compared to the task encounter/completion rate. However, this analysis applies to the scenario of UAVs completing a mission in the sense that each UAV is persistently trying to balance its load and therefore the entire system will persistently balance (even when tasks are found or completed). We define $T^i(k)$ to be the set of tasks formed from the tasks in the list $T^i_s(k)$ and the tasks in the set $T^i_u(k)$. If we assume that there are no delays, the total number of tasks of the system at time k is given by

$$\sum_{i=1}^{N_v} |T^i(k)| = \sum_{i=1}^{N_v} |T^i_s(k)| + \sum_{i=1}^{N_v} |T^i_u(k)| \tag{7}$$

which is the total amount of "fixed load" $T^i_s(k)$ and the tasks $T^i_u(k)$ that have not been assigned to a particular UAV yet. Below assume Ω measures load by the number of tasks.

Theorem 1: For a network of N_v UAVs persistently trying to balance, their load is stable in the sense of Lyapunov.

Stability in this sense means that for any maximum load imbalance $M \geq 1$ we want to have between any two neighboring UAVs, that is $||T^i(k)| - |T^{i'}(k)|| \leq M$ for all $(i, i') \in A$, and any initial load conditions, their load will remain close to the desired maximum load imbalance.

Theorem 2: For a network of N_v UAVs persistently trying to balance, if $|T^i_s(k)| = 1\ \forall i, k$, their load is asymptotically stable in the large.

Both these results hold if we allow for delays in the communication network as described in the model. The delays slow the balancing but do not make it impossible. Of course, both results involve analysis of properties of a subset of the state of the closed-loop system. The proof for these theorems can be found in [15]. The next theorems are an extension of some of the theorems found there.

Next, we focus our attention on the set of UAVs whose load distribution will not only remain close to a desired state, but is guaranteed to approach the desired state and remain within the desired maximum imbalance even when $\exists i, k : |T^i_s(k)| > 1$ (which is an important case since problem constraints such as vehicle capacity may dictate the need to plan its activities further ahead). Clearly, in this case balancing to within M is not possible for all UAVs; it is however still possible that a subset of the UAVs ("the balanced set") achieves a balanced condition as we show next.

Theorem 3: For a network of N_v UAVs persistently trying to balance, there exists a subset of balanced UAVs with tasks that are asymptotically stable in the large.

We have proven this (and Theorem 4) for the no-delay case. The size of the balanced set is determined by $|T_s^i(k)|$ and the total number of unscheduled tasks on all UAVs (e.g., the task encounter/completion rate).

Theorem 4: For the balanced set of UAVs persistently trying to balance, if $|T_s^i(k)| \geq 1$, their load is guaranteed to have balanced in a finite number of time steps.

For detailed information about these theorems and their proofs the reader should contact one of the authors. We are now working to extend Theorems 3 and 4 to the delay case and to find for Theorem 4 a tight bound on the amount of time it takes to balance. Obtaining such a bound is important since it helps to quantify effects of communication imperfections on cooperation.

V. Simulations

Here, we compare the performance of a noncooperative strategy and a cooperative strategy that uses balancing as described above. The results for $N_v = 4$ are shown in Figure 1. The bottom curve shows the average total time to complete a mission with 5, 6 and 7 objects using balancing, and assuming perfect communication. The middle curve shows how performance degrades when we introduce a random but bounded delay (200s max.) between the vehicles. The top curve shows the mission time for the noncooperative case. In the noncooperative as in the cooperative case we use N_v-TSP to initially schedule objects, but the noncooperative does not balance its tasks as the mission progresses (e.g., tasks are not passed over the network). In summary, communication delays significantly lower the benefits of cooperation, and can reduce the cooperative case to the noncooperative case if we increase the delay more.

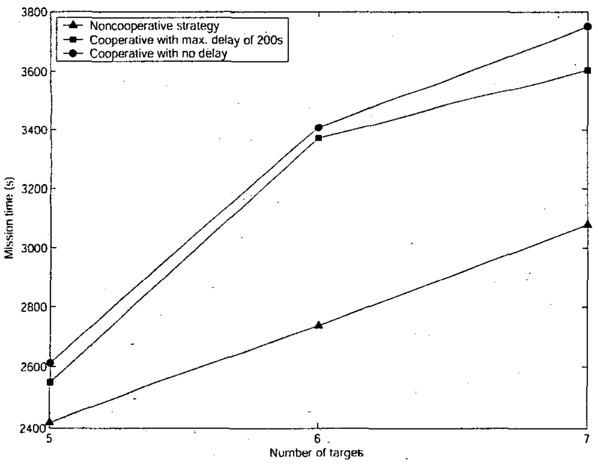

Fig. 1. Performance measure of Monte Carlo simulations.

VI. REFERENCES

[1] P. Chandler and M. Pachter, "Research issues in autonomous control of tactical UAV," in *Proceedings of the American Control Conference*, (Philadelphia, Pennsylvania), pp. 394–398, June 24–26 1998.

[2] R. Beard, T. McLain, and M. Goodrich, "Coordinated target assignment and intercept for unmanned air vehicles," in *Proceedings of IEEE International Conference on Robotics and Automation*, (Washington, DC), pp. 2581–2586, May 2002.

[3] P. R. Chandler, M. Pachter, and S. Rasmussen, "UAV cooperative control," in *Proceedings of the American Control Conference*, (Arlington, Virginia), pp. 50–55, June 2001.

[4] P. Chandler *et al.*, "Complexity in UAV cooperative control," in *Proceedings of the American Control Conference*, (Anchorage, Alaska), pp. 1831–1836, May 2002.

[5] C. G. Cassandras and W. Li, "A receding horizon approach for solving some cooperative control problems," in *Proceedings of the 41st IEEE Conference on Decision and Control*, (Las Vegas, Nevada), pp. 3760–3765, Dec. 2002.

[6] D. A. Castanon and C. G. Cassandras, "Cooperative mission control for unmanned air vehicles," in *Proceedings of the AFOSR Workshop on Dynamic Systems and Control*, (Pasadena, California), pp. 57–60, August 2002.

[7] J. Bellingham, A. Richards, and J. How, "Receding horizon control of autonomous aerial vehicles," in *Proceedings of the American Control Conference*, (Anchorage, Alaska), pp. 3741–3746, May 2002.

[8] K. M. Passino, M. Polycarpou, *et al.*, "Cooperative control for autonomous air vehicles," in *Cooperative Contorl and Optimization* (R. Murphey and P. M. Pardalos, eds.), vol. 66, pp. 233–271, Kluwer Academic Publishers, 2002.

[9] M. Baum and K. Passino, "A search-theoretic approach to cooperative control for uninhabited air vehicles," in *AIAA GNC Conference*, 2001.

[10] J. Hespanha, H. Kizilocak, and Y. Ateskan, "Probabilistic map building for aircraft-tracking radars," in *Proceedings of the American Control Conference*, (Arlington, Virginia), pp. 4381–4386, June 2001.

[11] M. Jun, A. I. Chaudhly, and R. D'Andrea, "The navigation of autonomous vehicles in uncertain dynamic environments: A case study," in *Proceedings of the 41st IEEE Conference on Decision and Control*, (Las Vegas, Nevada), pp. 3770–3775, Dec. 2002.

[12] S. Ganapathy and K. Passino, "Distributed agreement strategies for cooperative control: Modeling and scalability analysis," in *Proceedings of the Conference on cooperative control and optimization*, (Gainsville, FL), pp. 127–147, Dec. 2002.

[13] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Cooperative path planning for multiple UAVs in dynamic and uncertain environments," in *Proceedings of the 41st IEEE Conference on Decision and Control*, (Las Vegas, Nevada), pp. 2816–2822, Dec. 2002.

[14] D. R. Jacques and D. P. Gillen, "Cooperation behavior schemes for improving the effectiveness of autonomous wide area search munitions," in *Cooperative Contorl and Optimization* (R. Murphey and P. M. Pardalos, eds.), vol. 66, pp. 95–120, Kluwer Academic Publishers, 2002.

[15] K. Passino and K. Burgess, *Stability Analysis of Discrete Event Systems*. John Wiley and Sons, Inc., NY, 1998.

[16] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, MA, 1997.