Stable Task Load Balancing Strategies for Cooperative Control of Networked Autonomous Air Vehicles

Jorge Finke, *Member, IEEE*, Kevin M. Passino, *Fellow, IEEE*, and Andrew G. Sparks

***Abstract*—We introduce a mathematical model for the study of cooperative control problems for multiple autonomous air vehicles (AAVs) connected via a communication network. We propose a cooperative control strategy based on task-load balancing that seeks to ensure that no vehicle is underutilized and we show how to characterize task-load balancing as a stability property. Then, using Lyapunov stability analysis, we provide conditions under which task-load balancing is achieved even in the presence of communication delays. Finally, we investigate performance properties of the cooperative controller using Monte Carlo simulations. This shows the benefits of cooperation and the effects of network delays and communication topology on performance.**

***Index Terms*—Cooperative systems, distributed control, distributed decision making, load balancing, mobile robots.**

I. Introduction

THERE is a significant amount of current research activity focused on cooperative control of autonomous air vehicles (AAVs). In [1], the authors identify some research directions and highlight the need for detailed theoretical guidelines and proofs guaranteeing that decomposed hierarchical systems remain stable and solve the correct problems, particularly when uncertainties are considered. In [2] and [3], the authors address some of these challenges by presenting hierarchical distributed control schemes which use Voronoi diagrams for generating AAV trajectories while minimizing the risk of threats. The controller developed in [3] allows for rendezvous and target classification. In [4], a receding horizon approach to schedule a group of AAVs for several targets is considered. Trajectories are generated by computing solutions to a sequence of optimization problems. Information about the status of the targets and the vehicles, which make up the parameters of the cost function, serves as feedback and updates the cost function after every computation.

Manuscript received March 3, 2004; revised May 18, 2005. Manuscript received in final form March 22, 2006. Recommended by Associate Editor D. A. Schoenwald. This work was supported by the Air Force Research Library/Air Vehicles Directorate (AFRL/VA) and the Air Force Office of Scientific Research (AFOSR) Collaborative Center of Control Science under Grant F33615-01-2-3154.

J. Finke and K. M. Passino are with the Department of Electrical and Computer Engineering, Ohio State University, Columbus, OH 43210 USA (e-mail: passino@ece.osu.edu).

A. G. Sparks is with the Air Force Research Library/Air Vehicles Directorate (AFRL/VA), Control Design and Analysis Branch, Wright-Patterson AFB, OH 45433 USA.

Digital Object Identifier 10.1109/TCST.2006.876902

This allows for scenarios where high degrees of uncertainty are present. Their work, however, differs from ours in part because even though we consider high uncertainty cases, here once an AAV is assigned to a specific target, it will pursue it. Moreover, if network constraints dominate the problem, then optimal cooperative planning of multiple AAV activities far into the future in real-time is generally not feasible or even useful. In [5], the authors combine receding horizon approaches at a lower level with a high-level task selection, and consider the presence of uncertainties in future tasks, current tasks, and network information. Then, in [6], they study stability properties of a receding horizon controller. Other work based on receding horizon approaches can be found in [7].

Other methods that are being used in cooperative control include what one might call "map-based approaches" as in [8]–[15]. The more recent work in [13] and [15], defines "rate of return" (ROR) maps for search, task locations, and priorities (classification, engagement, verification), threats, and fuel use and uses these to define "payoffs." The work here assumes that there is limited prior knowledge about the terrain, not in a detailed form as the ROR map or other such maps (e.g., "threat maps"). We simply assume that we have certain regions that we are interested in searching called "search points." While we do not study the possibility here, probabilistic maps could be generated that define these search points.

More recent work focuses on cooperative control problems where there are communication imperfections and high scenario uncertainties. In this respect two notable studies are in [16], where the authors consider the problem of dynamic reassignment of tasks among a cooperative group that communicates relevant information asynchronously with arbitrary finite delays, and [17], where the authors study the synchronization of information for cooperative control.

When considering uncertain environments there are different types of uncertainties one can study. 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 online since there is high uncertainty *a priori* about whether any given task beyond search needs to be performed. Moreover, we consider the effects of communication imperfections via unknown but bounded delays involved in communicating the status of the targets from one AAV to another. We do this via a load balancing approach to cooperative control, something that has not been considered before. Other work that considers network delays in an analytical framework is in [18], where a distributed scheduling approach is used for

a specific class of cooperative control problems. A description of our earlier work in this area is provided in [19] and [20].

The main contributions of this paper are as follows. First, in Section II we introduce an analytical model for a wide class of cooperative control problems. It defines the problem that we solve here and provides the research community with a control-theoretic formulation of many other challenges in this research area. Second, in Section III we introduce the idea of using task-load balancing strategies from distributed computing [21], [22] for cooperative control. Indeed, our main theoretical result (Theorem 1) in Section IV is likely to be of interest in the area of distributed computing. In [23] and [24], the authors extended the theory in [21] in several ways including the nondelay discrete load case. Here, we extend the theory to the discrete load case with delays and where there can, in a certain way, be task load arrivals and departures (e.g., due to cooperative actions that have one AAV perform certain tasks for another AAV), something that has not been achieved before. Third, our results in Section V show when cooperation between AAVs is beneficial and how communication imperfections affect system-wide performance levels.

II. Plant Model and Closed-Loop Specifications

We begin by specifying a plant model that includes the vehicles, targets/threats ("objects"), sensors, actuators, and communication network. 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). It is nonlinear since we use a nonlinear kinematic vehicle model and due to the need to use automata. It is decentralized since the vehicles are separate entities that can be spatially distributed and connected over a communication network. Here, we keep the entire model in discrete time (even the vehicles) to get a finite-dimensional model even when we have communication delays, and to avoid the typical complications that arise in hybrid model definitions and simulations that are not essential features of the represented problem (e.g., simulating true continuous time and asynchronism on a digital computer, discontinuities that lead to existence and uniqueness issues in the ordinary differential equation, and "racing" of automata that can arise if events can occur an infinite number of times in a finite time period). We assume that there is a global clock so all the vehicles can use this time reference, be synchronized with it, and hence with each other (e.g., global positioning system (GPS) can provide such a time reference).

A. Vehicle and Object Models

Suppose that there are N_v AAVs and that the ith one obeys a continuous-time kinematic model given by $\dot{x}^i_{v1} = v\cos(\theta^i_v), \dot{x}^i_{v2} = v\sin(\theta^i_v)$, and $\dot{\theta}^i_v = wu^i_v$, where x^i_{v1} is its horizontal position, x^i_{v2} is its vertical position, v is its (constant) velocity, θ^i_v is its orientation, w is its maximum angular velocity, and $-1 \leq u^i_v \leq 1$ is the steering input. We assume that vehicles will either travel on the minimum turning 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) [25], [26]. 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 final desired location and heading, the code will generate trajectories between these two points, which are minimum time/distance trajectories.[1] For convenience, let

$$x^i_v = \left[x^i_{v1}, x^i_{v2}, \theta^i_v\right]^\mathsf{T}$$
$$x_v = \left[(x^1_v)^\mathsf{T}, \ldots, (x^{N_v}_v)^\mathsf{T}\right]^\mathsf{T}$$
$$u_v = \left[u^1_v, \ldots, u^{N_v}_v\right]^\mathsf{T}.$$

The environment is modeled as a two-dimensional (2-D) plane, the upper right quadrant of a Cartesian coordinate system with axes (x_1, x_2). The upper boundaries along both axes are denoted by P. In the environment we assume that there can be a variety of stationary 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. Similarly, a "priority," that is proportional to the object's importance, can be specified. It is assumed that at most N objects are in this square region, but initially, we know neither where they are nor the number N. We also assume that no objects lie on the boundaries of the region. The jth object has characteristics specified by its state which is $x^j_o = [x^j_1, x^j_2, \theta^j, o^j, d^j]^\mathsf{T}$, where x^j_1, x^j_2, and θ^j are the horizontal position, vertical position, and orientation of the jth 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 at exactly the same location, but with the same or different orientations) since then, we can number the objects and thereby uniquely identify them. Let o^j be a number representing the type of object for the jth 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 $x^j_p = [x^j_1, x^j_2, \theta^j]^\mathsf{T}$ and $x_o = [(x^1_o)^\mathsf{T}, \ldots, (x^N_o)^\mathsf{T}]^\mathsf{T}$.

B. Sensing, Tasks, and Actions

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

We assume that each AAV has a sensor that can be commanded to operate in different modes and we let $x^i_s = [(p^i_s)^\mathsf{T}, (p^i_c)^\mathsf{T}, (p^i_a)^\mathsf{T}, (p^i_v)^\mathsf{T}]^\mathsf{T}$ define the sensor state of AAV i, where $p^i_s = [p^{i1}_s, \ldots, p^{iN}_s]^\mathsf{T}$, $p^i_c = [p^{i1}_c, \ldots, p^{iN}_c]^\mathsf{T}$, $p^i_a = [p^{i1}_a, \ldots, p^{iN}_a]^\mathsf{T}$, and $p^i_v = [p^{i1}_v, \ldots, p^{iN}_v]^\mathsf{T}$. 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 ith AAV for the j_ith object. Initially, we use

[1]Here, we used the development by AFRL/VA Control Science Center of Excellence (CSCOE) and the code from the public release of their multi-AAV simulation to generate the optimal path trajectories in MATLAB.

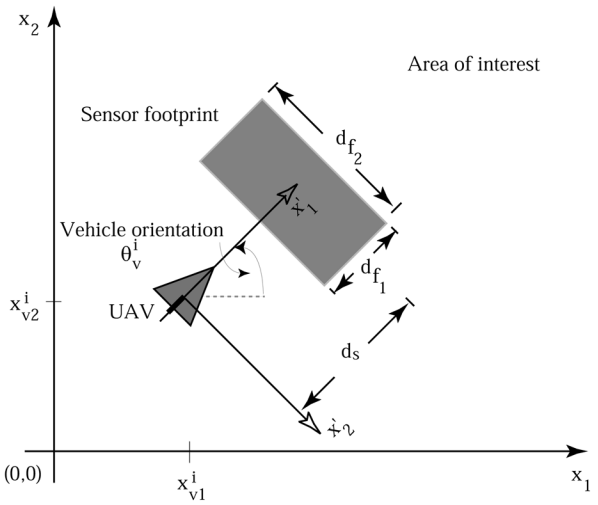


Fig. 1. Area of interest, vehicle, and sensor footprint.

"$-$" as a symbol for "not known" and any element of p_s^i, p_c^i, p_a^i, or p_v^i could hold such a value. Let $x_s = [(x_s^1)^\mathsf{T}, \ldots, (x_s^{N_v})^\mathsf{T}]^\mathsf{T}$. We assume AAV i has a "footprint" $S^i(x_v^i)$, as shown in Fig. 1, where d_{f_1} represents the depth of the footprint, d_{f_2} represents its width, and d_s is the distance from AAV i to the center of its footprint for $i = 1, 2, \ldots, N_v$. For convenience, we assume that $P = \bar{N} d_{f_1}$ for some positive integer $\bar{N}$, so we may tile the region of interest with sensor footprints.

There are four task types that the AAV can perform which are search (where the object is detected), classification (to determine the type of object that was found via search), attack, and verification (to evaluate the level of damage after an object is engaged). We will number the task types in the following order: 1) search; 2) classification; 3) attack; and 4) verification. In order to perform each of these tasks, the vehicle must be in an appropriate position and orientation as follows.

1) *Search for Object Detection*: Tile (partition) the 2-D environment with sensor footprints with search center points x_{sc}^m at (x_{sc1}^m, x_{sc2}^m), for $m = 1, 2, \ldots, \bar{M}$, and assume that to consider the mth cell of this tiled region to be searched some AAV must, at least at one time, k be at one of the four positions and orientations given by

$$x_v^i(k) \in \left\{ \begin{bmatrix} x_{\mathrm{sc1}}^m - d_s \\ x_{\mathrm{sc2}}^m \\ 0 \end{bmatrix}, \begin{bmatrix} x_{\mathrm{sc1}}^m \\ x_{\mathrm{sc2}}^m - d_s \\ \frac{\pi}{2} \end{bmatrix}, \begin{bmatrix} x_{\mathrm{sc1}}^m + d_s \\ x_{\mathrm{sc2}}^m \\ \pi \end{bmatrix}, \begin{bmatrix} x_{\mathrm{sc1}}^m \\ x_{\mathrm{sc2}}^m + d_s \\ \frac{3\pi}{2} \end{bmatrix} \right\}$$

which correspond to having the vehicle approach the x_{sc}^m, $m = 1, 2, \ldots, \bar{M}$, points from the left, bottom, right, and top (respectively) at a "standoff distance" d_s (see Fig. 1). When this position and orientation are achieved, if there is perfect sensing, the sensor of AAV i returns x_1^j, x_2^j, and θ^j for any object j in the sensor footprint S^i. AAV i numbers the objects according to the order it found them and then labels the object information with j_i so that it knows $x_1^{j_i}$, $x_2^{j_i}$, and θ^{j_i} for any object j_i in the sensor footprint, which coincides with the mth cell. At the same time, it returns $p_s^{ij_i}$ for each object found in the footprint by AAV i. To avoid pathological cases that arise when objects are placed right between cells of the tiled region, we assume that when a cell is searched the search includes the left and lower boundaries of the cell, but it does not include the two endpoints lying opposite to where the boundaries join. We assume that the same object cannot be found by searching two different cells.

2) *Classification*: After an object is found via search, it may need to be classified. Let θ_c denote a classification angle. In order to classify an object, it must be found and for some AAV i

$$x_v^i(k) = \begin{bmatrix} x_1^{j_i} + d_s \cos(\theta^{j_i} + \theta_c) \\ x_2^{j_i} + d_s \sin(\theta^{j_i} + \theta_c) \\ \theta^{j_i} + \theta_c \end{bmatrix}$$

so that the object is in the center of the footprint and the AAV is approaching from $\theta^{j_i} + \theta_c$. A typical choice for θ_c might be $\theta_c = (\pi/2)$ so that the AAVs will approach from the side of the object. When classification occurs for object j_i the sensor provides o^{j_i} and $p_c^{ij_i}$.

3) *Attack*: Let θ_a be used to define which angle to attack the object at and let d_a be the standoff distance from the object in order to attack it (in some applications $d_a = 0$). To attack object j_i at time k, AAV i must have

$$x_v^i(k) = \begin{bmatrix} x_1^{j_i} + d_a \cos(\theta^{j_i} + \theta_a) \\ x_2^{j_i} + d_a \sin(\theta^{j_i} + \theta_a) \\ \theta^{j_i} + \theta_a \end{bmatrix}.$$

Here, we let $\theta_a = \pi$ so that the AAV approaches from the head of the object. Upon attack, nothing about the environment is sensed, but the AAV computes the level of attack certainty $p_a^{ij_i}$ for target j_i.

4) *Verification*: Let θ_b denote a verification angle. To verify object j_i at time k, AAV i must have

$$x_v^i(k) = \begin{bmatrix} x_1^{j_i} + d_s \cos(\theta^{j_i} + \theta_b) \\ x_2^{j_i} + d_s \sin(\theta^{j_i} + \theta_b) \\ \theta^{j_i} + \theta_b \end{bmatrix}.$$

A typical choice is $\theta_b = \theta_a$. When verification occurs, the sensor provides d^{j_i} and $p_v^{ij_i}$.

C. Communication Network

Here, we assume that communication links and the overall topology are fixed, there is sufficient link capacity to transmit the required information, and that the only imperfection on a link is a possible delay in transmitting/sensing information. These delays should not be thought of as arising 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 constraints, or temporary loss of a communication link on vehicles (e.g., via jamming or noise). The communication network topology is defined via a directed graph $G = (L, A)$, where

$L = \{1, 2, \ldots, N_v\}$ is the set of nodes (the vehicles) and $A = \{(i, i') : i, i' \in L\}$ 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. We assume that $(i, i) \notin A$ since any local information on a AAV is known to that AAV. We assume that for all $i \in L$ there exists $i' \in L$, such that $(i', i) \in A$ so all vehicles are connected to the network. Also, we assume that for any i there exists a path along arcs in A to any i. Let $A_i = \{i' : (i, i') \in A\}$ be the set of AAVs that AAV i can send messages to and receive messages from.

Next, we must represent how information is passed from one vehicle to another. We assume that the maximum delay between two vehicles i and i' is given by $D^{ii'} - 1$, $D^{ii'} > 1$ (this notation is used for convenience later on). For representation simplicity, we will simply view each communication link (i, i') as a memory that holds the last $D^{ii'} - 1$ values of the vectors of information transmitted between AAVs i and i'. Denote the vector of information that AAV i transmits AAV i' at time k as $x_n^{ii'}(k), (i, i') \in A$. A delay of up to $D^{ii'} - 1$ units on a link can be represented by having each link have a state vector $x_L^{ii'}(k) = [x_n^{ii'}(k-1), \ldots, x_n^{ii'}(k-(D^{ii'}-1))]^\mathsf{T}$. Let $x_L^i(k)$ be a vector of $x_L^{i'i}(k)$ for all $(i', i) \in A$ so that it represents all information received by AAV i via communication links. Note that the first element in $x_L^{ii'}(k)$ is based on the $k-1$ index because we assume it takes at least one time step to transmit information. Let $x_L = [(x_L^1)^\mathsf{T}, \ldots, (x_L^{N_v})^\mathsf{T}]^\mathsf{T}$. The specific form of $x_n^{i'i}(k)$ for $i' \neq i$ depends on the design of the cooperative controller. We will specify x_L in Section III.

To model a deterministic fixed delay $\tau^{ii'}, 1 \leq \tau^{ii'} \leq D^{ii'} - 1$ between AAV i and AAV i', we will only allow the receiving node i' to pull the delayed value $x_n^{ii'}(k - \tau^{ii'})$ off the link at time k. 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 $\tau^{ii'}, 1 \leq \tau^{ii'} \leq D^{ii'} - 1$); Here, we assume that $1 \leq \tau^{i'i}(k+1) \leq \tau^{i'i}(k) + 1 \leq D^{i'i} - 1$, so that messages are not received out of order.

D. Observation and State Equations

Next, we define the sensor modes, attack mode, and the information that is gathered from sensing and communications. Let $u^i \in \{s, c, a, v\}$ indicate that AAV i should search, classify, attack, verify, or do nothing, and let $u = [u^1, \ldots, u^{N_v}]^\mathsf{T}$. The AAV controllers will choose these u^i values, and then move to the appropriate location to perform the task. Next, we define the observation map for AAV i, that is the sensed information at time k for AAV i, by

$$y^i(k) = \left[\left(x_v^i(k)\right)^\mathsf{T}, \left(\hat{x}^i(k)\right)^\mathsf{T}, \left(x_s^i(k)\right)^\mathsf{T}, \left(\hat{x}_L^i(k)\right)^\mathsf{T}\right]^\mathsf{T} \tag{1}$$

and let $y = [(y^1)^\mathsf{T}, \ldots, (y^{N_v})^\mathsf{T}]^\mathsf{T}$. Hence, each AAV knows its own position and orientation $x_v^i(k)$ (e.g., it may obtain this via onboard sensors and GPS), and sensed information (e.g., levels of certainty) about objects via its own sensors $x_s^i(k)$. Also, we use $\hat{x}^{ij_i}(k)$ to represent what AAV i measures at time k about object j_i and $\hat{x}^i = [(\hat{x}^{i1})^\mathsf{T}, \ldots, (\hat{x}^{iN})^\mathsf{T}]^\mathsf{T}$. If AAV i is in the correct position and orientation (as define above) to sense object j_i, then $\hat{x}^{ij_i}(k) = [x_1^{j_i}, x_2^{j_i}, \theta^{j_i}, -, -]^\mathsf{T}$ so that it senses the object position and orientation. Finally, notice that when $u^i(k) = s$, a value is not returned for neither the object type nor the damage level. We represent this with the "$-$" entry in $\hat{x}^{ij_i}(k)$. When $u^i = c$ (respectively, v) and AAV i is in the correct position and orientation, we get to classify (verify) object j_i, then $y^i(k)$ holds $\hat{x}^{ij_i}(k) = [-, -, -, o^{j_i}, -]^\mathsf{T}$ $(= [-, -, -, -, d^{j_i}]^\mathsf{T}$, respectively). Notice also that $\hat{x}_L^i(k)$, the vector of information received by AAV i from any AAV 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 AAV i. This means that information in $\hat{x}_L^i(k)$ is always delayed information received at time k (at least by one time step). For example, for delays $1 \leq \tau^{i'i} \leq D^{i'i} - 1$ on link $(i', i) \in A$, $\hat{x}_L^i(k)$ is a vector of $x_n^{i'i}(k - \tau^{i'i})$, i.e., received values from all AAVs i' connected on the network to AAV i, but delayed by $\tau^{i'i}$.

Next, we define how the plant state evolves. The state has four parts: $x_v(k), x_o(k), x_s(k)$, and $x_L(k)$. We have already explained how to generate $x_v(k+1)$. Next, we will, in turn, explain how to generate $x_o(k+1), x_s(k+1)$, and $x_L(k+1)$. Suppose $x_o(k+1) = f(x_v(k), x_o(k), x_s(k), x_L(k), u(k), u_v(k))$. To define $x_o(k+1)$, first note that since objects are stationary, the positions and orientations of all objects stay the same. We also assume that the objects types do not change. All that remains are the damage levels of the objects $d^j(k+1)$ and f must represent this. Here, when any AAV i is in the correct position and orientation it may attack object j at time k and to model simultaneous attacks we let $d^j(k+1) = d^j(k) + \sum_{i=1}^{N_v} d^{ij}(k)$, where $d^{ij}(k)$ is the amount of damage inflicted at time k on object j by AAV i (defined so $d^j(k) \in [0, 1]$ for all k).

Next, consider how to define $x_s^i(k+1)$. Other ways to define this for different levels of uncertainty are discussed in [20]. Here, we consider the case where there is a high scenario uncertainty, but AAVs have perfect sensor capabilities. In this case, 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 so that upon completion of a task an AAV only knows whether another task needs to be completed for the object, and what task is needed. Hence, upon sensing object j_i AAV 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. 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 known successful attack so there is no need for a verification, or $p_a^{ij_i} = 0$ representing that there is uncertainty as to the attack's effectiveness, 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 postmission 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.

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

$$x_L^{ii'}(k+1) = \left[\left(x_n^{ii'}(k)\right)^{\mathsf{T}}, \ldots, \left(x_n^{ii'}(k-D^{ii'})\right)^{\mathsf{T}}\right]^{\mathsf{T}}$$

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

III. Cooperative Control Strategy Based on Load Balancing

According to the model, as the mission progresses AAVs sense and react to the environment and thereby obtain new tasks and complete others. One of the main challenges in cooperative control is how to manage this information that becomes available as the environment is explored, so that the benefits of feedback can be exploited. In this section, we introduce a cooperative controller that uses task-load balancing. For it, AAVs cooperate by sharing the work (load) to complete the tasks by passing tasks and other relevant information over the network.

Load balancing provides a distributed control scheme that evenly allocates load over the network of vehicles and is a flexible and scalable approach applicable to small or large groups of AAVs. As the mission progresses load balancing exploits feedback information about the status of objects and vehicles to persistently try to maintain a balanced task load, thereby forcing cooperation. The definition of "load" normally depends on the mission objective. We assume that the task load on vehicles can be partitioned into discrete "blocks." The largest block has size B and the smallest block has size b, so that $B \geq b > 0$. We also assume that all vehicles are trying to balance their load within $M \geq 1$, which means that the load on neighboring AAVs only differs by at most M. Under these assumptions the best we can guarantee is that neighboring vehicles (i.e., ones within one communication link) will balance to within $M = B$ if we use a distributed balancing policy. Some definitions of load are as follows.

1) *Number of Tasks*: In some scenarios vehicles may want to balance the total number of tasks in order to: a) avoid underutilization of some vehicles due to them not having tasks to perform while others have many to perform and b) to guarantee that all vehicles will start heading back to their home base within a certain number of tasks (time) from one another (which forces cooperation since there is pressure for no AAV to go home until every other AAV has only one more task to complete).
2) *Path Lengths/Travel Time*: By balancing path lengths (or equivalently travel time if velocity is constant) needed to perform the tasks, we also avoid underutilization in the group of vehicles and force cooperation. Furthermore, we can guarantee that all vehicles will start heading back to their home base within a certain number of time steps from one another (e.g., assuming we know the longest time it could take to travel between any two objects or search points).

To define our cooperative controller we need to specify what information is passed between AAVs, how tasks are scheduled on individual AAVs, and who to pass to. We assume that information that is received can be stored by a AAV so that retransmission of the same information is not necessary. For convenience in the notation we will, however, indicate retransmission of an entire vector when only some or no information in that vector changes. The local controller for the ith AAV can be modeled as a dynamical system

$$x_c^i(k+1) = f_c^i\left(x_c^i(k), u_c^i(k)\right)$$
$$y_c^i(k) = h_c^i\left(x_c^i(k)\right).$$

A. Local AAV Controller State and Command Input

First, let $u_c^i(k) = y^i(k)$ from (1) so all measured local information is available for making decisions. Let $t_m^i(k) = (\ell, x_p^{j_i})$ represent that task type ℓ is scheduled to be performed on search-point or object j_i at step m by AAV i. We define the state of the local controller $x_c^i(k) = [T_s^i(k), T_u^i(k), x_a^i(k)]^{\mathsf{T}}$ which is composed of the schedule for AAV i which is an ordered list $T_s^i(k) = (t_1^i(k), \ldots, t_{N_t}^i(k))$ 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 AAV i or ones that other AAVs communicated to AAV i. We assume that tasks belonging to $T_s^i(k)$ cannot be passed to neighboring AAVs, whereas tasks in $T_u^i(k)$ can. 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 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 as discussed earlier. This set S is fixed so it is not part of the state. It is simply used in $t_m^i = (1, x_p^{j_i})$, the search task, to specify where it needs to do a search task. It is assumed that all N_v AAVs hold the same S. In some cases, $T_u^i(0) \neq \emptyset$ which would represent an initial task distribution such that tasks in $T_u^i(k)$ can be passed to neighboring vehicles for $k > 0$. Basically, $T_s^i(0)$ and $T_u^i(0)$ are defined by *a priori* intelligence about the area of interest. The initial state $x_c^i(0)$ also defines $x_a^i(0) = [- \cdots -]^{\mathsf{T}}$ so that no information on object location has been sensed.

For the task types of classification, attack, and verification ($\ell = 2, 3, 4$), to specify the meaning of $t_m^i = (\ell, x_p^{j_i})$, 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 AAV i has gathered about each object j_i. Again, $x_a^i(k)$ denotes this accumulated information at time k that is stored in the controller for all j_i. Moreover, other AAVs may send tasks and the associated object information to AAV i and this will also be accumulated in $x_a^i(k)$. 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

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

where $\emptyset$ is the null task, and we assume that the task ordering is appropriately renumbered. Task $t_1^i(k) = (\ell, x_p^{j_i})$ 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

$$x_{Lr}^i(k) = \left[x_{Lr}^{i'i}(k), \ldots, x_{Lr}^{i''i}(k)\right]^{\mathsf{T}} = \left[x_n^{i'i}\left(k - \tau^{i'i}(k)\right), \ldots, x_n^{i''i}\left(k - \tau^{i''i}(k)\right)\right]^{\mathsf{T}}$$

where $A_i = \{i', \ldots, i''\}$. Let $x_{Ls}^i(k)$ be the set of all the information sent by AAV i at time k defined by

$$x_{Ls}^i(k) = \left[x_{Ls}^{ii'}(k), \ldots, x_{Ls}^{ii''}(k)\right]^{\mathsf{T}} = \left[x_n^{ii'}(k), \ldots, x_n^{ii^*}(k), \ldots, x_n^{ii''}(k)\right]^{\mathsf{T}}$$

where $A_i = \{i', \ldots, i''\}$, and we assume that at any time k an AAV may pass at most one task to any of its neighboring AAVs (note that it may still pass different tasks to several AAVs at the same time). This task will be put in $x_n^{ii^*}(k)$ for all AAVs $i^* \in A_i^*(k)$, where $A_i^*(k)$ represents the set of vehicles AAV i passes load to at time k. Suppose that the task being passed is denoted by $t^{ii^*}(k) = (\ell, x_p^{j_i})$ which is the task passed from AAV i to AAV i^* such that $(i, i^*) \in A$. Notice that at time $k = 0$, and $x_{Lr}^i(0) = x_{Ls}^i(0) = [-, \ldots, -]^{\mathsf{T}}$ so that no messages have been received or sent. Moreover, $x_n^{ii'}(k - \tau^{ii'}) = ''-''$ if $k - \tau^{ii'} < 0$ for all $(i, i') \in A$. Notice also that $x_{Lr}^i(k)$ and $x_{Ls}^i(k)$ are not part of the controller state $x_c^i(k)$. They are used to define the set of unscheduled tasks [which is part of $x_c^i(k)$] and the controller output $y_c^i(k)$, respectively.

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 AAVs via AAV i at time k, which we denote $T_{\rm nu}^i(k)$. 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 $T_u^i(k)$. Define it as

$$\begin{aligned} T_{\rm nu}^i(k) &= T_u^i(k) \bigcup \left\{\left(2, x_p^{j_i}\right) : \text{if } p_s^{ij_i}(k) = 1\right\} \\ &\quad \bigcup \left\{\left(3, x_p^{j_i}\right) : \text{if } p_c^{ij_i}(k) = 1\right\} \\ &\quad \bigcup \left\{\left(4, x_p^{j_i}\right) : \text{if } p_a^{ij_i}(k) = 0\right\} \\ &\quad \bigcup \left\{\left(\ell, x_p^{j_i}\right) : \quad \forall i' \in A_i \text{ if } t^{i'i}(k - \tau^{i'i}) \right. \\ &= \left.\left(\ell, x_p^{j_{i'}}\right) \neq \emptyset, x_p^{j_{i'}} = x_p^{j_i} \quad \text{for some } j_i\right\} \\ &\quad \bigcup \left\{\left(\ell, x_p^{\max\{j_i\}+1}\right) : \quad \forall i' \in A_i \quad \text{if } t^{i'i}(k - \tau^{i'i}) \right. \\ &= \left.\left(\ell, x_p^{j_{i'}}\right) \neq \emptyset, x_p^{j_{i'}} \neq x_p^{j_i} \quad \text{for any } j_i\right\}. \end{aligned}$$

Here, the first set is the set of unscheduled tasks, and the next three sets in the union defining $T_{\rm nu}^i(k)$ are the set of classification, attack, and verify tasks that are added to AAV i due to its own sensors or actions, respectively. The fourth set in $T_{\rm nu}^i(k)$ collects all tasks on objects that are known to AAV i, that are received through the network (e.g., AAV i may have found the object and given the classification task to another AAV and then that AAV could have given the attack task back to AAV i). The fifth set in the $T_{\rm nu}^i(k)$ definition is the set of tasks on objects AAV i did not have any information about (this forces its local object index to increase by 1). Hence, we think of new tasks that arrive over the network in the same way as we do of new tasks arising from a AAV using its own sensors and taking actions. In this way, the AAVs are cooperatively sharing sensed information.

AAV i also passes the accumulated information about object j_i, its current schedule $T_{\rm sc}^i(k)$, and its set of new unscheduled tasks $T_{\rm nu}^i(k)$. Hence, the messages sent over the network, for $i^* \in A_i^*(k)$, has the form

$$x_n^{ii^*}(k) = \left[x_v^i(k), t^{ii^*}(k), x_a^{ij_i}(k), T_{\rm sc}^i(k) \bigcup T_{\rm nu}^i(k)\right]^{\mathsf{T}}.$$

Also, for $i' \notin A_i^*(k)$, but $i' \in A_i$

$$x_n^{ii'}(k) = \left[x_v^i(k), \emptyset, x_a^{ij_i}(k), T_{\rm sc}^i(k) \bigcup T_{\rm nu}^i(k)\right]^{\mathsf{T}}$$

so that at each time instant, each AAV transmits to its neighbors its current tasks since this information is used by the other vehicles. Basically, $x_n^{ii'}(k)$ contains information that is required for the vehicles to cooperate. Here, $y_c^i(k) = h_c^i(x_c^i(k))$. In particular, the output $y_c^i(k) = [u_v^i(k), u^i(k), x_{Ls}^i(k)]^{\mathsf{T}}$, 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 AAVs at time k. Here, if we have $t_1^i(k)$ then $u_v^i(k)$ and $u^i(k)$ are specified.

B. Local Optimal Task Scheduling

Each AAV i can compute an optimal task sequence to perform all tasks in $\{T_{\rm sc}^i(k) \bigcup T_{\rm nu}^i(k)\}$ (we use union to form a set from the elements in a list and a set). Note that any $t_m^i(k) = (\ell, x_p^{j_i}) \in \{T_{\rm sc}^i(k) \bigcup T_{\rm nu}^i(k)\}$ holds the location and orientation of the j_ith object to perform task ℓ on. These locations and orientations will then define the distance (cost) to travel from one object to another and perform the corresponding tasks. Also, the first element in the list $T_{\rm sc}^i(k)$ remains the first element in the optimal sequence (since this is the object AAV i is approaching and we do not allow any task $t_m^i \in T_{\rm sc}^i(k)$ to be passed over the network). When computing this optimal sequence, we let the cost from any node (object) to the first element be zero, to represent the fact that AAV i does not care about going back to where it is currently located.

The optimal sequence can then be found by solving an asymmetric traveling salesman problem (TSP) [27] or, more generally, a mixed-integer linear programming (MILP) problem (which allows multiple constraints to be taken into consideration [28]). Objects are "cities" and costs are distances along feasible AAV trajectories. For both definitions of load, as the number of tasks or as the path length to perform tasks, AAVs use the TSP algorithm to obtain an optimal sequence that is then used to define $t_1^i \in T_{\rm sc}^i(k+1)$, if $T_{\rm sc}^i(k) = \emptyset$ and $T_{\rm nu}^i(k) \neq \emptyset$.

In words, TSP specifies the next locally available task (those in $T^i_{\rm nu}(k)$) if all scheduled tasks have been performed. Note that this local optimization does not affect the proof in Section IV.

C. 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 $T^i_u(k+1)$. This will then precisely define the distributed decision making strategy since it chooses what task any AAV i should pass (if any), and how its local scheduled and unscheduled tasks evolve over time.

Let $|\cdot|$ denote the length of a list, the size of a set, or for scalars, the absolute value. If $T^i_{\rm nu}(k) \neq \emptyset$, there is at least one task in the unscheduled set of AAV i that may stay unscheduled in $T^i_u(k+1)$, or be sent to any of its neighboring AAVs (since any task in $T^i_{\rm nu}(k)$ can be considered for cooperation). If $|T^i_{\rm nu}(k)| \geq 1$, we will choose element $t^i_{\rm nu} \in T^i_{\rm nu}$ to be such a task. Recall that $A^*_i(k)$ represents the set of vehicles AAV i passes load to at time k. Then, if $A^*_i(k) \neq \emptyset$, for $i^* \in A^*_i(k)$

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

otherwise, if $A^*_i(k) = \emptyset$

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

where the element in $T^i_s(k+1)$ is determined by having the TSP algorithm, which we denote by F, return the first element of an optimal sequence of performing all unscheduled tasks, as we discussed above.

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

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

which arises when nothing has been sensed or attacked, and no new tasks have been received over the network. If $T^i_s(k) = T^i_u(k) = \emptyset$, then AAV i starts returning to the home base at time k.

Let $x^i(k)$ be the measure of the load on AAV i

$$x^i(k) = \Omega\left(T^i_{\rm sc}(k) \bigcup T^i_{\rm nu}(k)\right)$$

where Ω is a measure of the length of the task schedule of AAV i (e.g., it could be the number of tasks on $T^i_{\rm sc} \bigcup T^i_{\rm nu}$ or the path length to execute the tasks on $T^i_{\rm sc} \bigcup T^i_{\rm nu}$). It is important to notice that the dynamics of the model defined in Section III drive the dynamics of $x^i(k)$. It will be the key variable in our analysis below. Let $x^{i'}_i(k)$ be the load "perception" AAV i has about AAV i' at time k (i.e., it is the amount of load it thinks i' holds). Let $\alpha^{i\to i'}_i \geq 0$ denote the load AAV i passes to AAV i'; it is the amount of load removed from AAV i when i passes to AAV i'. We also define $\alpha^{i\to i'}_{i'} \geq 0$ as the amount of load AAV i' will receive due to AAV i sending load to i' at time k. Then, the following conditions define a class of load passing policies for AAV i at time k:

1) $\alpha^{i\to i'}_i = 0$, if $x^i(k) - x^{i'}_i(k) \leq M$;
2) $x^i(k) - \sum_{\{i':(i,i')\in A\}} \alpha^{i\to i'}_i > \min\left\{x^{i'}_i(k) : i' \in A_i\right\}$;
3) If $\alpha^{i\to i'}_i > 0$ for some i', then $\alpha^{i\to i^*}_i > 0$ where $i^* = \arg\min_{i''}\left\{x^{i''}_i(k) : i'' \in A_i\right\}$;
4) If $\alpha^{i\to i'}_i > 0$ for some i', then $x^{i'}_i(k) + \alpha^{i\to i'}_{i'} \leq \max_{i''}\left\{x^{i''}_i(k) : i'' \in A_i\right\}$.

Any load passing controller that satisfies conditions 1)–4) belongs to the class of controllers of interest. Condition 1) says that AAV i may only pass load to AAV i' if its load perception about AAV i' at time k is lower than its own load by more than M. Condition 2) limits the amount of load that AAV i can pass, so that the remaining load on AAV i after passing some load to some neighboring AAVs is still larger than the minimum perception of load of all its neighboring AAVs before passing the load. Condition 3) means that if AAV i does not perceive itself as being balanced within M with all of its neighbors, then it must pass some load to the least loaded neighboring vehicle i^*. If $\alpha^{i\to i'}_i = \alpha^{i\to i'}_{i'} \forall i, i' \in L$, condition 2) implies condition 4) (i.e., if vehicles are trying to balance total number of tasks). However, if this is not the case, then approaching a given object or search point means a different load size for different vehicles depending on their current positions and the location of other objects that are scheduled to be visited (e.g., if vehicles are trying to balance total path length, then to visit different objects means different tours for different vehicles). If $\alpha^{i\to i'}_i > 0$,, then $i' \in A^*_i(k)$. This completes the definition of $x^i_c(k+1)$.

IV. Theoretical Analysis of Load Balancing

So far, we have modeled the plant (vehicles, objects, sensors, and the communication network) and the cooperative control strategy based on load balancing. Next, we analyze closed-loop system properties. For convenience, we assume $D^{ii'} = D$ for all $(i, i') \in A$ and that each AAV 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.

Here, we are only considering properties of the evolution of a subset of the state variables of the closed-loop system, i.e., state variables that represent task load passing and sensing, since we only seek to verify properties of the load balancing algorithm. Let this subset of the state be $x(k) \in \mathbb{R}^{(2N_v+A)\times D}$ and suppose it is composed of three "substates." Let $x_{n0}(k) \in \mathbb{R}^{N_v \times D}$ represent the loads of the N_v AAVs at times $k, k-1, \ldots, k-D+1$ (the current load of all vehicles and all the past load values up to the maximum delay $D-1$). The first column represents the loads of the N_v AAVs at time k, that is $x^1(k), \ldots, x^{N_v}(k)$, the second

column represents the loads of the AAVs at time $k-1$, and so on. Let $x_{n1}(k) \in \mathbb{R}^{N_v \times D}$ represent the loads of the N_v AAVs at times $k-D, k-D-1, \ldots, k-2D+1$. The first column represents the loads of the AAVs at time $k-D$, the second column represents the loads of the vehicles at time $k-D-1$, and so on. Let $x_t(k) \in \mathbb{R}^{A \times D}$ represent all of the $|A|$ loads in transit between the vehicles at times $k, k-1, \ldots, k-D+1$. The first column represents the loads in transit at time k, the second column represents the loads in transit at time $k-1$, and so on. Then, the state $x(k)$ may be represented as

$$x(k) = \begin{bmatrix} x_{n0}(k) \\ x_{n1}(k) \\ x_t(k) \end{bmatrix} = \begin{bmatrix} x_n(k) \\ x_t(k) \end{bmatrix}, \quad x_n = \begin{bmatrix} x_{n0}(k) \\ x_{n1}(k) \end{bmatrix}.$$

Notice, that since we assume a maximum delay of $D-1$ in both transmitting and receiving information, x_n must hold the load of all vehicles from time k back to $k-2(D-1)$.

For convenience, we define some notation. If y is a matrix, then $\sum y$ is equal to the sum of all of the elements of y. Further, let $x_e(k) = [x_{n0}(k), x_t(k)]^{\mathsf{T}}$ so that the sum of the elements of column k' of $x_e(k)$ is equal to the total load of the system at time $k-k'+1$. Note that the total load of the system is not constant as it is in [21] and [23] since load is measured differently by different AAVs. Therefore, it is possible for AAV i to pass load to some neighboring AAV i' and increase or reduce the total amount of the load of the network. However, when AAV i passes some load to AAV i', it will always be the case that for $\alpha_i^{i \to i'} > 0, 0 < \alpha_{i'}^{i \to i'} \leq B$, because we assume that only one task is passed at a time and we assume there exists a maximum load size B. With this in mind, note that

$$\frac{1}{D}\sum x_e(k) = \frac{1}{D}\left(\sum x_{n0}(k) + \sum x_t(k)\right) \leq \frac{1}{D}\sum x_{n0}(k) + |A|B \leq B(NN_v + |A|) \tag{2}$$

since there is only a finite amount of load on the vehicles, and there is at most one task per object on the network (defined by how new tasks arrive at the network of AAVs in Section III-A) at any time k, so that $(1/D)\sum x_{n0}(k) \leq NN_vB$. Therefore, we know that $(1/D)\sum x_e(k)$ remains bounded at all times.

A. Achievement of Task Load Balancing

Next, we want to define an invariant set, such that any state $x(k)$ that is in the invariant set exhibits the following properties at time k: 1) the load between any two neighboring AAVs is balanced within M and 2) there is no load in transit between vehicles. Let $L' = \{1, 2, \ldots, 2N_v\}$, $G = \{1, 2, \ldots, D\}$, and $H = \{1, 2, \ldots, |A|\}$. If y is a matrix, let $(y)_{pq}$ denote the element in row p and column q of y. Choose the set as

$$\mathcal{X}_b = \Big\{ x(k) \in \mathbb{R}^{(2N_v+|A|)\times D} : |(x_n(k))_{ij} - (x_n(k))_{pq}| \leq M \forall i, p \in L' \text{ and } j, q \in G; \; (x_t(k))_{ij} = 0 \forall i \in H \text{ and } j \in G \Big\}.$$

The proof of the following theorems are in the Appendix.

Theorem 1: For a network of N_v AAVs with delays and vehicles persistently trying to balance their load the invariant set $\mathcal{X}_b$ is exponentially stable in the large.

Notice that load balancing condition 1) in Section II is necessary. If condition 1) is removed, then it is possible that vehicles may pass load to their more heavily loaded neighbors. In this case, the least loaded vehicle may pass load to some of its neighbors. Hence, the lightest load in the network may decrease and exponential stability will not hold. If condition 3) is removed, it is no longer true that the least loaded vehicle on the network must increase its load after a finite number of time steps. Hence, $\mathcal{X}_b$ is no longer exponentially stable. Condition 4) in Section III only sets a limit on the rate the total network load may grow at each time step. This condition can be used to determine a specific upper bound for the exponential decay, but is not required for the proof in the Appendix (since assuming a maximum load size B is enough).

B. Vehicle Network Topology Effects

In the last subsection, we did not assume any characteristics about the topology of the communication network other than there exists $i' \in L$, such that $(i', i) \in A$ for all $i \in L$. Let $R = \max\{|A_i|\}$ be the maximum numbers of neighbors of any AAV $i \in L$ and let S be the maximum number of links that must be spanned to reach any vehicle $i' \in L$ from any other vehicle $i \in L$.

Theorem 2: For any network of N_v AAVs with delays.

1) The least loaded vehicle increases its load by at least $(\delta\epsilon_0^2)/(16BN_vD^3M^2(NN_v + |A|)[1 + (D|A|)/(N_v)])$ every DSR time steps, where δ and ϵ_0^2 must be computed for a particular discrete load network.
2) If we balance the total number of tasks, all vehicles are guaranteed to be balanced after $(BN + (|A|)/(N_v))/(B)$DSR time steps.

Notice that the number of objects N, the number of vehicles N_v, and the maximum communication delay D, clearly slow down the rate of convergence (e.g., in a worst case analysis as we do here). Intuitively, this is what we would expect. Other parameters like $|A|$ that appear in the denominator in Theorem 2-1 may be misleading in the sense that a decrease in $|A|$ appears to accelerate convergence. However, this is not the case since Theorem 2-1 is just a lower bound. Moreover, some variables are affected by other parameters of the model. Here, the maximum number of links that must be spanned to reach any vehicle S, and the maximum number of neighbors of any AAV R, seem to have a bigger effect on the load dynamics, than the total number of network links $|A|$, itself. More discussion on this will follow later.

V. Simulations

To measure how well the mission objective (here, mission time to complete all tasks) is met by the fleet of vehicles, we first need to choose a particular load-balancing controller. Here, we assume that at time k, AAV i passes load to some vehicle i^* only if $i^* = \arg\min_{i'}\{x_i^{i'}(k) : (i, i') \in A\}$, so that vehicle i only passes load to the vehicle perceived as least loaded (if $x^i(k) - x_i^{i^*}(k) > M$). If several vehicles are perceived as least loaded, vehicle i chooses which one to pass to arbitrarily.

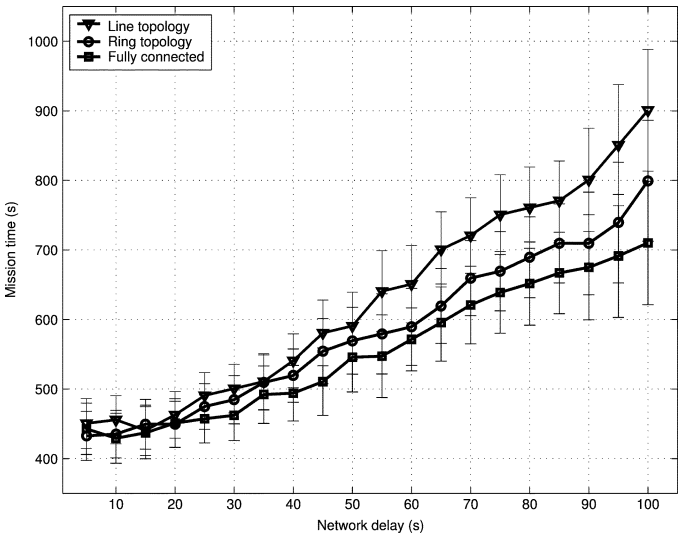


Fig. 2. Performance degrades as network delays increase. The error bars show the standard deviation of every data point which corresponds to the average mission time of 60 independent mission runs with varying target location.

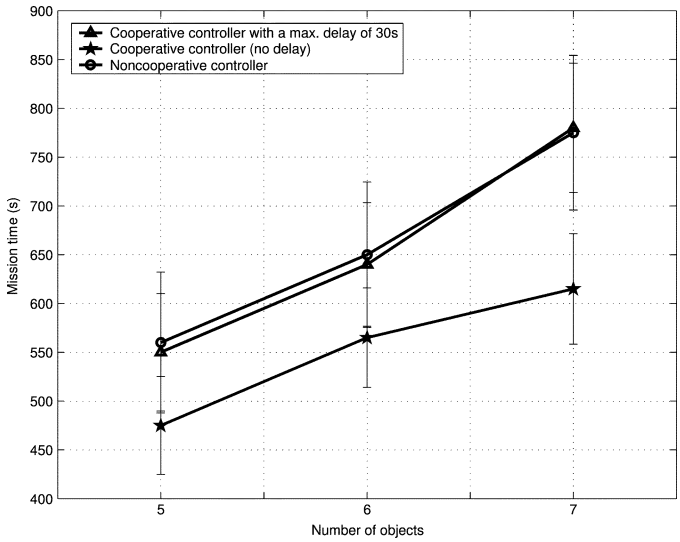


Fig. 3. Cooperative versus noncooperative case.

If the load perception of all neighboring vehicles is higher than its own load, vehicle i does not send any tasks over the network. To determine the initial task distribution among vehicles, we solve in MATLAB for an optimal task distribution via MILP that minimizes the mission time if every object were to be visited once (i.e., not considering scenario uncertainty by tasks that arise during mission execution).

We will first study the effects of network topology and random but bounded communication delays between vehicles. We choose a scenario with five objects, a sampling time $T = 0.1$ s and perform Monte Carlo runs by letting the maximum communication delay between vehicles vary from 5 to 100 s. The results are shown in Fig. 2. The bottom curve shows the case where AAVs are fully connected, so they can all cooperate with each other. The middle curve represents a ring topology case. For a four vehicle mission this means that two pairs of vehicles are not connected to each other, and thus, cannot "directly" cooperate, meaning that a task cannot be passed directly between these two vehicle pairs. Note that a task could still be passed between these vehicles via another AAV, but larger network delays make this less likely. The top curve represents the case where AAVs use a line topology, so that there are three pairs of vehicles that cannot directly cooperate with each other. Clearly, in all cases the average mission time goes up as the network delay increases as one would expect. Note, however, that the delays have a more significant impact for more sparsely connected communication topologies. This is due to two reasons. First, if tasks are transferred over a path of several vehicles, then delays accumulate. Second, when direct cooperation is not possible, poorer coordination between subgroups of AAVs occur and mission time increases. Next, note that network topology seems to have little effect when small communication delays are present, but becomes more relevant as they increase. This is due to the fact that when delays are small travel times have the dominant effect on lengthening the mission. As the spatial distribution of tasks increases, the effects of travel times will become even more significant. However, if the spatial distribution decreases the communication delays will have a relatively large impact.

Next, we compare the performance of the cooperative strategy to a noncooperative strategy. In the noncooperative case, vehicles distribute their tasks initially in the same way as for the cooperative controller. As the mission progresses, however, AAVs do not balance their tasks since there is no communication network. Due to the lack of communication each vehicle independently schedules new tasks that arise as the mission progresses. Fig. 3 shows the performance of both strategies. The bottom curve shows the average total time to complete a mission with five, six, and seven objects using load balancing, and assuming no communication delays. The middle curve shows how performance degrades when we introduce a random but bounded delay (30 s maximum) between the vehicles. The top curve shows the mission time for the noncooperative case. Note that communication delays significantly lower the benefits of cooperation, and can reduce performance of the cooperative controller to the noncooperative one. Note also that the data point for a specific number of objects is computed as an average of 60 mission runs. For some particular missions scenarios, it might be possible that the noncooperative strategy performs equal to or better than the cooperative one, so that cooperation does not show any benefits. The results here show when we can, on average, expect an improvement in performance by using load balancing and, thus, be able to justify the cost of implementing a communication network.

Next, for a given object distribution, let the average time between objects be the time it takes on average to travel from one object to another and perform any task. Fig. 4 shows that as the average time between objects increases from 32 to 72 s local path planning (e.g., via TSP) becomes more important. Each bar represents the average of the average mission time for scenarios with 8, 12, 16, and 20 objects. While all vehicles try to balance path length, the darker bars shows the average performance

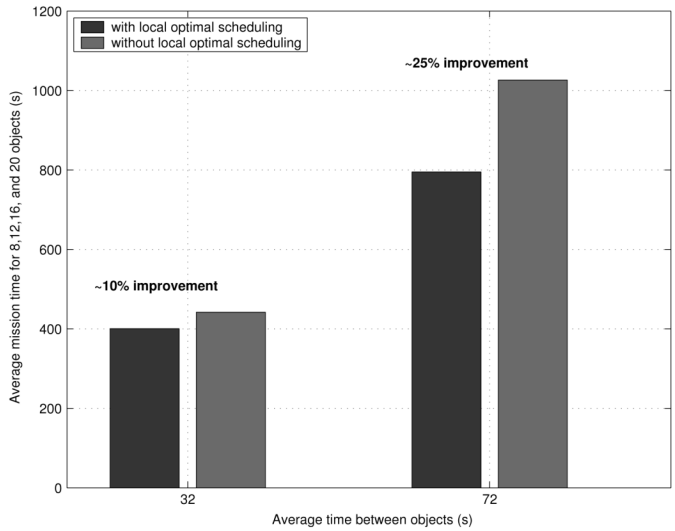


Fig. 4. Performance measure with and without local optimal scheduling. The worst standard deviation in any of the two cases is 101 s.

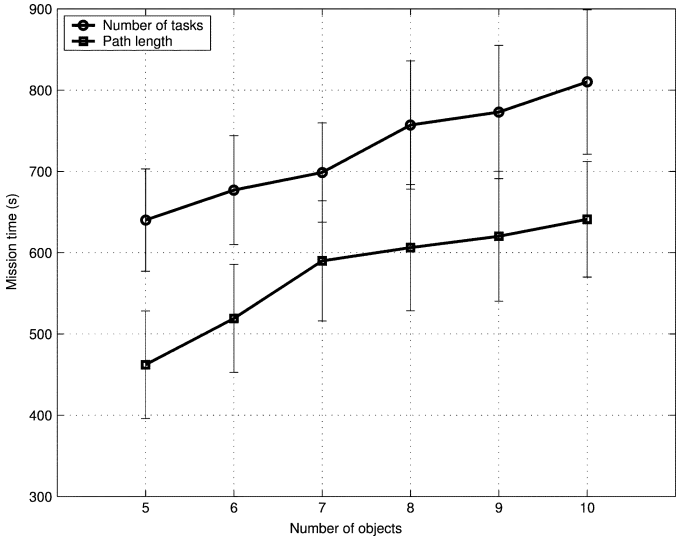


Fig. 5. Balancing number of tasks versus balancing path length.

when local optimal scheduling is used (e.g., locally, AAVs compute TSP-algorithms as they balance their load). As the average time between objects increases, we see more benefits in using local optimal scheduling as one would expect. Next, Fig. 5 compares the performance of a cooperative controller that balances path length with one balancing the total number of tasks. This shows that balancing path length requires more computations, but results in better performance. This is due to gains from the load balancing algorithm trying to minimize mission time by passing in order to reduce individual AAV travel time.

Finally, we consider the effects of not satisfying all the load balancing conditions described in Section III-C. We consider a cooperative strategy that uses a fully connected communication network and vehicles try to balance their tasks in the sense that vehicles with higher load pass tasks to vehicles less loaded. However, this strategy will satisfy condition 1) only, while the other conditions are not taken into account. Fig. 6

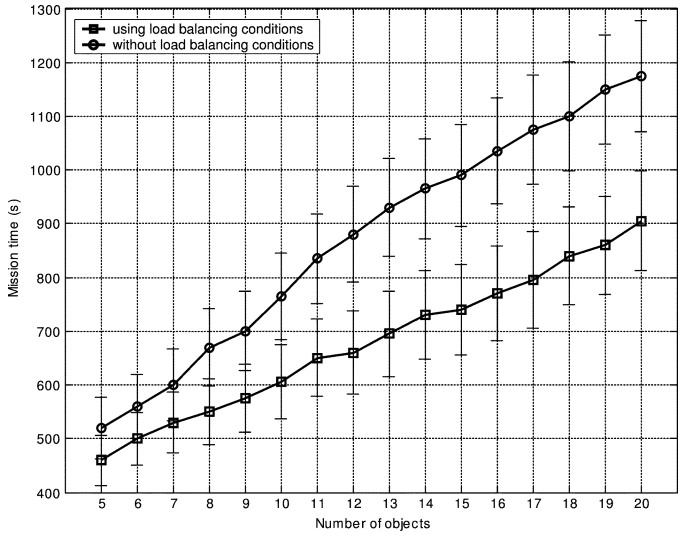


Fig. 6. Performance degrades when all load balancing conditions are not satisfied.

shows how the strategy degrades when load balancing conditions are not met. A closer look at the load dynamics (not shown here) shows that the total load of the system does not reach a steady state, but keeps on moving between vehicles until the mission is completed (even when no new tasks arise). Thus, if conditions 1)–3) in Section III-C are not met, vehicles fail to cooperatively decide which vehicle should schedule which tasks and a type of oscillation in task assignments across the network degrades performance.

These simulation results suggest the following principle in decentralized decision making in highly uncertain environments. If vehicles schedule new tasks that arise while the mission progresses, the "selfishness" of the noncooperative strategy described above does not perform very well because underutilized vehicles finish their tasks earlier and return to the home base without cooperating with the remaining vehicles. If vehicles pass to the least loaded vehicle without precisely considering all load balancing conditions (e.g., if they are too "generous"), then the asynchronicity of the problem creates large loads of tasks that can cycle over the network. The lack of agreement of where to schedule all tasks does not allow the system to settle down, also degrading its performance. When all load balancing conditions are met, it takes a finite amount of time for decentralized vehicles to decide how to distribute tasks that were not considered at the beginning of the mission. Since the system task loads are guaranteed to converge, we know all tasks will be performed, while trying to minimize underutilization of the fleet.

VI. Conclusion

We first introduced a mathematical model for the cooperative control problem for multiple AAVs. We then introduced a class of cooperative strategies based on task-load balancing, and provided Monte Carlo simulations to compare performance of different strategies (e.g., to a noncooperative strategy) in order to uncover system-level tradeoffs and principles of design and

operation. Simulations showed that strategies persistently balancing the vehicles load perform better than strategies where load was passed over the network, but load balancing conditions were not strictly met, or where load was not shared at all. Other aspects, like tradeoffs between achieving perfect balancing and scheduling tasks far into the future remain future research directions, as well as the further study of task arrivals/departures and its relation to the task throughput of the system (e.g., to determine capacity conditions, for task arrivals/departures, etc.).

We also performed a theoretical analysis of the load balancing properties of the system. We presented sufficient conditions for a distributed cooperative strategy based on sharing the task load to achieve a balanced state of the total system load (e.g., by showing the load distribution is exponentially stable even in the presence of communication delays). For particular networks, we then determined the specific time the system takes to settle down after the last task was completed or found. It would be of interest to be able to obtain similar results for a time-varying communication topology.

APPENDIX

A. *Proof of Theorem 1*

To study the ability of the system to automatically redistribute load to achieve balancing, we employ a Lyapunov stability theoretic approach. For convenience, we first define some notation. If y is a matrix, then $\min\{y\}$ is equal to the minimum of all of the elements of y, $\max\{y\}$ is equal to the maximum of all of the elements of y, and $\sum y$ is equal to the sum of all of the elements of y, as defined earlier. Further, let $(y)^j$ be column j of y recall $(y)_{ij}$ is the ith element in the jth column. Let $T = \{1, 2, \ldots, (2N_v + |A|)\}$. Choose

$$\begin{aligned}&\rho(x(k), \mathcal{X}_b)\\ &\quad = \inf\{\max\{|(x(k))_{ij} - (\bar{x})_{ij}| :\\ &\qquad\qquad \text{for all } i \in T, j \in G\} : \bar{x} \in \mathcal{X}_b\} \quad (3)\\ &V(x(k))\\ &\quad = \begin{cases} \frac{1}{N_v D}\sum x_e(k) - \min\{x_n(k)\}, & x(k) \notin \mathcal{X}_b \\ 0, & x(k) \in \mathcal{X}_b. \end{cases} \quad (4)\end{aligned}$$

Notice that $V(x(k))$ is the average load (total network load divided by N_v) minus the minimum load, taken over times $k - 2D + 1, \ldots, k - 1, k$, at any vehicle $i \in L$. Notice also that for $x(k) \notin \mathcal{X}_b$, at least one of the two cases hold: 1) there must be two vehicles i and i', $(i, i') \in A$, such that $x^i(k') - x^{i'}(k'') > M$ for some $k - 2D < k' \leq k$ and $k - 2D < k'' \leq k$ or 2) there is some load in transit between at least two vehicles.

We first demonstrate that $V(x(k))$ is bounded from above and from below by our choice of $\rho(x(k), \mathcal{X}_b)$ and $V(x(k))$. We will find a constant $\eta \in (0, \infty)$ such that $\eta\rho^2(x(k), \mathcal{X}_b) \geq V(x(k))$ for all $x(k) \in \mathbb{R}^{(2N_v+|A|)\times D}$. But notice that according to (3) and (4) if $x(k) \in \mathcal{X}_b$, then $V(x(k)) = \rho(x(k), \mathcal{X}_b) = 0$. Therefore, any value of η will suffice for $x(k) \in \mathcal{X}_b$, and we need only to be concerned with $x(k) \notin \mathcal{X}_b$.

From (3) and (4)

$$\begin{aligned}&\rho(x(k), \mathcal{X}_b)\\ &\quad \geq \max\Big\{\frac{1}{2}\max\{(x(k))_{ip} - (x(k))_{jq} - M :\\ &\qquad\qquad i, j \in L' \text{ and } q, p \in G\}, \max\{x_t(k)\}\Big\}\\ &\quad \geq \frac{1}{2}\max\Big\{(\max\{x_n(k)\} - \min\{x_n(k)\} - M),\\ &\qquad\qquad \max\{x_t(k)\}\Big\}. \quad (5)\end{aligned}$$

Also

$$\max\{x(k)\} = \max\{\max\{x_n(k)\}, \max\{x_t(k)\}\}. \quad (6)$$

We must consider two cases. If $\max\{x_n(k)\} - \min\{x_n(k)\} - M \geq \max\{x_t(k)\}$, then $\max\{x_n(k)\} \geq \max\{x_t(k)\}$ and $\max\{x_n(k)\} = \max\{x(k)\}$. It follows then from (5), that $2\rho(x(k), \mathcal{X}_b) \geq \max\{x(k)\} - \min\{x_n(k)\} - M$.

On the other hand, if $\max\{x_t(k)\} \geq \max\{x_n(k)\} - \min\{x_n(k)\} - M$, then according to (6)

$$\max\{x_t(k)\} \geq \max\{x_t(k)\} - \min\{x_n(k)\} - M$$

it must be the case that

$$\max\{x_t(k)\} \geq \max\{x(k)\} - \min\{x_n(k)\} - M.$$

Once again, (5) implies that

$$2\rho(x(k), \mathcal{X}_b) \geq \max\{x(k)\} - \min\{x_n(k)\} - M.$$

Thus, we can conclude that for all $x(k) \notin \mathcal{X}_b$

$$2\rho(x(k), \mathcal{X}_b) \geq \max\{x(k)\} - \min\{x_n(k)\} - M. \quad (7)$$

Next, let $\psi_1(x(k)) = \max\{x(k)\} - \min\{x_n(k)\} - M$ and $\psi_2(x(k)) = \max\{x(k)\} - \min\{x_n(k)\}$. We will find a constant $\phi \in (0, \infty)$, such that $\phi\psi_1^2(x(k)) \geq \psi_2(x(k))$. Later, we will relate $\psi_2(x(k))$ and $V(x(k))$, and using (7), we can then relate $V(x(k))$ to $\rho(x(k), \mathcal{X}_b)$. Notice that $\psi_1(x(k)) > -M$ for all $x(k) \notin \mathcal{X}_b$ so since $\psi_1(x(k)) + M = \psi_2(x(k)), \psi_2(x(k)) > 0$. As we mentioned before, if $x(k) \notin \mathcal{X}_b$ at least one of the following cases must be true. If there is an imbalance greater than M between neighboring AAVs, then $\max\{x(k)\} - \min\{x_n(k)\} > M$, and, therefore, $\psi_1(x(k)) > 0$. On the other hand, it could be the case that at time k all neighboring AAVs are balanced within M, but $(x_t(k))_{ij} \neq 0$ for some $i \in H$ and $j \in G$. Then we can only guarantee that $\max\{x(k)\} - \min\{x_n(k)\} > 0$ and, therefore, $\psi_1(x(k)) > -M$. We conclude that $\psi_1(x(k))$ is always bounded from below by the lesser of the two bounds, i.e., that $\psi_1(x(k)) > -M$.

Next, because the network contains a finite number of blocks, each of finite size, there must be some constant, ϵ_0, $-M < \epsilon_0 \leq M$, such that for $x(k) \notin \mathcal{X}_b$, $\psi_1(x(k)) \geq \epsilon_0$. Recall that $M \geq 1$. Thus, if we choose $\phi = 2M^2/\epsilon_0^2$, it is clear that if $-M < \psi_1(x(k)) < M$, then

$$\begin{aligned}\phi\psi_1^2(x(k)) &= \frac{2M^2}{\epsilon_0^2}\psi_1^2(x(k)) \geq 2M^2 \geq 2M \\ &\geq \psi_1(x(k)) + M = \psi_2(x(k))\end{aligned}$$

and if $\psi_1(x(k)) \geq M$, then

$$\begin{aligned}\phi\psi_1^2(x(k)) &= \frac{2M^2}{\epsilon_0^2}\psi_1^2(x(k)) \geq 2\psi_1^2(x(k)) \geq 2\psi_1(x(k)) \\ &\geq \psi_1(x(k)) + M = \psi_2(x(k)).\end{aligned}$$

It follows that, for all $\psi_1(x(k)) > -M$

$$\phi\psi_1^2(x(k)) \geq \psi_2(x(k)). \tag{8}$$

It is clearly true for all k', $k - D < k' \leq k$, that the total load in transit at time k' is equal to the total system load at time k' minus the load at the vehicles at time k'. Hence, if q is the column of $x(k)$ that contains the load at the vehicles and in transit at time k', then for all $q \in G$

$$\begin{aligned}|A|B &\geq \sum (x_t(k))^q \\ &= \sum (x_e(k))^q - \sum (x_{n0}(k))^q \\ &\geq \sum (x_e(k))^q - N_v \max\{(x_n(k))^q\} \\ &\geq \sum (x_e(k))^q - N_v \max\{x_n(k)\}.\end{aligned}$$

Since the total network load is bounded by (2), there must exist a column $q^* = \arg\max\{\sum (x_e(k))^q : q \in G\}$. Note that column q^* may not be unique. In fact if the total network load is constant (e.g., if we balance the total number of tasks), then $\sum (x_e(k))^{q'} = \sum (x_e(k))^{q''} \forall q', q'' \in G$, so that q^* represents any column. Either way $\sum (x_e(k))^{q^*} \geq \frac{1}{D}\sum x_e(k)$, therefore,

$$\sum (x_t(k))^{q^*} \geq \frac{1}{D}\sum x_e(k) - N_v \max\{x_n(k)\}.$$

However, for all $q \in G$

$$|A|B \geq |A| \max\{x_t(k)\} \geq \sum (x_t(k))^q$$

for all k', $k - D < k' \leq k$. Therefore, $|A| \max\{x_t(k)\} \geq \sum (x_t(k))^{q^*}$. It follows that:

$$\begin{aligned}\max\{x_t(k)\} &\geq \frac{1}{|A|}\left[\frac{1}{D}\sum x_e(k) - N_v \max\{x_n(k)\}\right] \\ &\geq \frac{1}{|A|}\left[\frac{1}{D}\sum x_e(k) - N_v \max\{x(k)\}\right]. \end{aligned} \tag{9}$$

Due to condition 2 in the load balancing policy and the bounded delay, each of the load passes between neighboring vehicles at time k', $k - D + 1 \leq k' \leq k$, must have been smaller than $\max\{x_n\} - \min\{x_n\}$ (since the perception of AAV i about the load of AAV i', $x_i^{i'}(k') \in x_{n0}(k)(k')$, may be a delayed value by at most $D - 1$). Therefore, the load in transfer between any neighboring vehicles at time k resulted from at most $D-1$ load passes. Hence

$$\begin{aligned}\max\{x_t(k)\} &\leq (D-1)(\max\{x_n(k)\} - \min\{x_n(k)\}) \\ &\leq D(\max\{x(k)\} - \min\{x_n(k)\}).\end{aligned} \tag{10}$$

Equations (8), (9), and (10) imply that

$$\begin{aligned}&\phi(\max\{x(k)\} - \min\{x_n(k)\} - M)^2 \geq \psi_2(x(k)) \\ &\quad = \max\{x(k)\} - \min\{x_n(k)\} \\ &\quad \geq \frac{1}{D|A|}\left[\frac{1}{D}\sum x_e(k) - N_v \max\{x(k)\}\right]\end{aligned} \tag{11}$$

and (7) and (11) imply that

$$\frac{4\phi D|A|}{N_v}\rho^2(x(k), \mathcal{X}_b) \geq \frac{1}{DN_v}\sum x_e(k) - \max\{x(k)\}. \tag{12}$$

Because $4\phi\rho^2(x(k), \mathcal{X}_b) \geq \phi\psi_1^2(x(k)) \geq \psi_2(x(k)) = \max\{x(k)\} - \min\{x_n(k)\}$, it is clear that $4\phi\rho^2(x(k), \mathcal{X}_b) + \min\{x_n(k)\} \geq \max\{x(k)\}$. Hence, using (12) and rearranging some terms, we get

$$\phi\left[\frac{4D|A|}{N_v} + 4\right]\rho^2(x(k), \mathcal{X}_b) \geq \frac{1}{DN_v}\sum x_e(k) - \min\{x_n(k)\}.$$

Therefore, it is always true that

$$V(x(k)) \leq \phi\left[4 + \frac{4D|A|}{N_v}\right]\rho^2(x(k), \mathcal{X}_b) = \eta\rho^2(x(k), \mathcal{X}_b). \tag{13}$$

We now show that there exists a bound from below for $V(x(k))$. Notice that

$$\sum x_e(k) \geq \max\left\{\sum x_{no}(k), \sum x_{n1}(k)\right\}.$$

Because $\sum x_e(k) \geq (\sum x_{no}(k) + \sum x_{n1}(k))/(2) = (\sum x_n(k))/(2)$, it follows then, from (4), that

$$V(x(k)) \geq \frac{1}{2N_vD}\sum x_n(k) - \min\{x_n(k)\}. \tag{14}$$

Then, $\sum x_n(k)$ can be bounded from below in terms of $\max\{x_n(k)\}$ and $\min\{x_n(k)\}$ by a sum of the elements where

exactly one element of $x_n(k)$ is equal to $\max\{x_n(k)\}$ and the remaining elements of $x_n(k)$ are equal to $\min\{x_n(k)\}$. From this analysis and (14), we have that

$$\begin{aligned} V(x(k)) &\geq \frac{1}{2N_vD}[\max\{x_n(k)\} \\ &\quad + (2N_vD-1)\min\{x_n(k)\}] - \min\{x_n(k)\} \\ &= \frac{1}{2N_vD}[\max\{x_n(k)\} - \min\{x_n(k)\}]. \end{aligned} \tag{15}$$

It is clear from the definition of $\mathcal{X}_b$ and (3) that

$$\begin{aligned} \rho(x(k),\mathcal{X}_b) \leq \max\{&(\max\{x_n(k)\} - \min\{x_n(k)\} - M) \\ &\times \max\{x_t(k)\}\}. \end{aligned} \tag{16}$$

We must consider two cases. First, consider $\max\{x_n(k)\} - \min\{x_n(k)\} - M \geq \max\{x_t(k)\}$. According to (16)

$$\begin{aligned} \rho(x(k),\mathcal{X}_b) &\leq \max\{x_n(k)\} - \min\{x_n(k)\} - M \\ &\leq \max\{x_n(k)\} - \min\{x_n(k)\}. \end{aligned} \tag{17}$$

Equations (15) and (17) yield

$$V(x(k)) \geq \frac{1}{2N_vD}\rho(x(k),\mathcal{X}_b). \tag{18}$$

Now, consider $\max\{x_t(k)\} > \max\{x_n(k)\} - \min\{x_n(k)\} - M$, so that

$$\rho(x(k),\mathcal{X}_b) \leq \max\{x_t(k)\}. \tag{19}$$

As before, the maximum load in transit at times $k - D + 1,\ldots,k-1,k$, is the sum of at most $D-1$ load passes, each of which must have been smaller than $\max\{x_n(k)\} - \min\{x_n(k)\}$. Hence

$$\begin{aligned} \max\{x_t(k)\} &\leq (D-1)(\max\{x_n(k)\} - \min\{x_n(k)\}) \\ &\leq D(\max\{x_n(k)\} - \min\{x_n(k)\}). \end{aligned} \tag{20}$$

Using (19) and (20), along with (15) yields

$$\begin{aligned} \frac{1}{D}\rho(x(k),\mathcal{X}_b) &\leq \max\{x_n(k)\} - \min\{x_n(k)\} \\ \frac{1}{2N_vD^2}\rho(x(k),\mathcal{X}_b) &\leq \frac{1}{2N_vD}[\max\{x_n(k)\} - \min\{x_n(k)\}] \\ &\leq V(x(k)). \end{aligned} \tag{21}$$

Because (18) and (21) both bound $V(x(k))$ from below, we can claim that $V(x(k))$ is always bounded from below by the lesser of the two bounds. Therefore, it is always true that

$$V(x(k)) \geq \frac{1}{2N_vD^2}\rho(x(k),\mathcal{X}_b). \tag{22}$$

Next, we define a constant δ on which the rest of the proof will depend. For a given discrete load network, there is a constant $\delta_1 > 0$, such that if AAV i sends some load to AAV i', then $x^i(k+1) \geq x^{i^*}(k) + \delta_1$, where $(i,i^*) \in A$ and $x^{i^*}(k) \leq x^{i'}(k)$ for all $(i,i') \in A$. In words, δ_1 measures how much the load on a vehicle passing some load at time k will differ from the load of its minimum loaded neighbor. For the same discrete load network, there is also a constant $\delta_2 > 0$, such that if $(i,i') \in A$ and $x^i(k) \neq x^{i'}(k)$, then $|x^i(k) - x^{i'}(k)| \geq \delta_2$. Let $\delta = \min\{\delta_1,\delta_2,b\}$. Below, we will see that we will be able to guarantee that the minimum load will increase by at least δ within a finite number of steps of any time k until the invariant set is reached.

Note that $\min\{x_n(k)\}$ is a nondecreasing function. To see this, notice that AAV i may pass some load to AAV i' only if the perception it has about the load of AAV i' is less than its own load (condition (i)). Furthermore, the balancing policy allows AAV i to pass only a certain amount of load, so that $x^i(k+1)$ may drop below its prepass load perception of AAV i', but not below its minimum prepass load perception of all its neighboring AAVs (condition 2). It is clear that any load perception about AAV i' must be larger or equal than the minimum perception of all its neighboring vehicles. Mathematically it is true that, $x^i(k+1) \geq \min_j\{x_i^j(k) : (i,j) \in A\}$, where $\min_j\{x_i^j(k) : (i,j) \in A\}$ is the minimum load perception AAV i has about its neighboring AAVs at time k. Note also, that $j^* = \arg\min_j\{x_i^j(k) : (i,j) \in A\}$ is an element in row j^* of $x_{n0}(k)$. Since $x^i(k+1) \geq \min_j\{x_i^j(k) : (i,j) \in A\} \geq \min\{x_{n0}(k)\} \geq \min\{x_n(k)\}$ for all $i \in L$, even after passing some load to some vehicle i', we conclude that $x^i(k+1) \geq \min\{x_n(k)\}$ for all $i \in L$.

Next, fix k and let $L_* \subset L$ be the set of all i, such that $x^i(k) = \min\{x_n(k)\}$. First, we consider the case where $|L_*| > 0$. Because $x^i(k') \geq \min\{x(k)\}$ for all $k' > k$, if $x^i(k) = \min\{x_n(k)\}$, then $x_i(k-m) = \min\{x_n(k)\}$ for all $m \in \{1,2,\ldots,2D-1\}$. In words, this means that if at time k the minimum value in $x_n(k)$ is in the first column of $x_n(k)$, then the entire row must have the same value since the minimum is nondecreasing, which will guarantee an accurate perception about the minimum load of the system. Thus, for any two vehicles i and i' such that $(i,i') \in A$ and $i' \in L_*$, $x_i^{i'}(k) = \min\{x(k)\}$. There must be time k', $k \leq k' < k+D$, such that if AAV i sends some load to AAV i' at time k, AAV i' receives this load by time k'. However, it might be the case that for some $i' \in L_*$ there does not exist a neighbor i such that $i \notin L_*$ and $(i,i') \in A$. However, note that since $x(k) \notin \mathcal{X}_b$, there must exist at least one vehicle $i \notin L_*$ such that $(i,i') \in A$ for some $i' \in L_*$ and note that every vehicle in the network can be reached by AAV i by spanning fewer than N_v links. Therefore, after $D(N_v-1)$, either the number of least loaded vehicles decreases by at least one, or the smallest load increases by at least δ. Because $|L_*| < N_v$, the above passing and receiving scenario may take place $N_v - 1$ times, to ensure that $x^{i'}(k_1) > \min\{x_n(k)\}$ for all $i' \in L_*$ and for some $k_1 \geq k$. It is apparent that for $k_1 = k + D(N_v-1)^2$, $x^{i'}(k_1) > \min\{x_n(k)\}$ for all $i' \in L_*$.

What happens if $|L_*| = 0$? $|L_*| = 0$ means that no vehicle is guaranteed to have an accurate perception about the least loaded neighbor. This case arises when tasks are passed at a very high rate, so that the minimum load changes at least every $2D-1$ time steps. However, since $|L_*| = 0$ this means that at least every $2D-1$ steps one vehicle must strictly increase its load,

thereby not letting $|L_*| \neq 0$. Therefore, $|L_*| = 0$ for at most $2D$ time steps without increasing the minimum load of the system.

Note that in the worst case the two cases described above can follow one another. In order to guarantee that the minimum load of the system will strictly increase, we need to consider both. Thus, the minimum load is guaranteed to increase by δ after $D(N_v-1)^2+2D \leq DN_v^2$ time steps for $N_v \geq 2$.

It is also clear that there is some $\zeta > 0$, such that

$$\zeta\delta \geq \sum x_e(k) > \rho(x(k), \mathcal{X}_b).$$

Therefore, it follows that

$$V(x(k)) - V(x(k+DN_v^2)) > \frac{1}{\zeta}\rho(x(k), \mathcal{X}_b) \qquad (23)$$

which implies that $\mathcal{X}_b$ is exponentially stable in the large [24]. If the load blocks are of different sizes the ζ must be calculated from a worst case analysis.

B. Proof of Theorem 2

Let L_*^i be the set of all vehicles i' such that $(i', i) \in A$ and $x^{i'}(k) = \min\{x_n(k)\}$. Similar as argued above, if $i' \in L_*^i$, then $x_i^{i'}(k) = \min\{x_n(k)\}$ and vehicle i is guaranteed to have an accurate perception about the load on vehicle i' since $\min\{x_n(k)\}$ is nondecreasing. Then, notice that for $x(k) \notin \mathcal{X}_b$, there must exist at least one vehicle $i \notin L_*^{i'}$ for all $i' \in L$ so that every vehicle in the network can be reached by AAV i by spanning fewer than S links. Therefore, after DS, either the number of least loaded vehicles decreases by at least one, or the smallest load increases by at least δ. Since $\max\{L_*^i\} \leq R$ the above passing and receiving scenario can take place at most R times. Again, to ensure that $x^{i'}(k_1) > \min\{x_n(k)\}$ for some $k_1 \geq k$ let $k_1 = k + DSR$. Equation (23) can then be rewritten as

$$V(x(k)) - V(x(k+\mathrm{DSR})) > \frac{1}{\zeta}\rho(x(k), \mathcal{X}_b).$$

Recall from (2) that $\sum x_e(k) \leq DB(NN_v+|A|)$. Since $\delta > 0$, we let $\zeta = (DB(NN_v+|A|))/(\delta)$, it follows that:

$$V(x(k)) - V(x(k+DSR)) > \frac{\delta}{DB(NN_v+|A|)}\rho(x(k), \mathcal{X}_b).$$

On the other hand, recall from (13) and (22) that

$$\frac{1}{2N_vD^2}\rho(x(k), \mathcal{X}_b) \leq V(x(k)) \leq \frac{2M^2}{\epsilon_0^2}\left[4 + \frac{4D|A|}{N_v}\right]\rho^2(x(k), \mathcal{X}_b).$$

We conclude that

$$\rho(x(k), \mathcal{X}_b) \geq \frac{\epsilon_0^2}{4N_vD^2M^2\left[4+\frac{4D|A|}{N_v}\right]}.$$

Therefore,

$$V(x(k)) - V(x(k+\mathrm{DSR})) > \frac{\delta\epsilon_0^2}{16BN_vD^3M^2(NN_v+|A|)\left[1+\frac{D|A|}{N_v}\right]}. \qquad (24)$$

This completes the first part of the proof. The importance of (24) is to have an estimate of how rapidly the load on all AAVs converges within M. In particular, it shows the tradeoff between plant parameters and balancing rate.

Next, let us consider the case where AAVs try to balance the total number of tasks and $b = B$, so that each task represents B units of load. Then, $\delta = B$ and, therefore, $V(x(k)) - V(x(k+\mathrm{DSR})) \geq B$ for $x(k) \notin \mathcal{X}_b$. Using (2) and (4), we can bound $V(x(k))$ by $V(x(k)) \leq BN + (|A|)/(N_v)$. Then, we conclude that $V(x(k')) = 0$ for some k' such that $k \leq k' \leq k + (BN + (|A|)/(N_v))/(B)$ DSR.

REFERENCES

[1] P. Chandler and M. Pachter, "Research issues in autonomous control of tactical UAV," in *Proc. Amer. Contr. Conf.*, 1998, pp. 394–398.

[2] R. Beard, T. McLain, and M. Goodrich, "Coordinated target assignment and intercept for unmanned air vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 2581–2586.

[3] P. R. Chandler, M. Pachter, and S. Rasmussen, "UAV cooperative control," in *Proc. Amer. Contr. Conf.*, 2001, pp. 50–55.

[4] C. G. Cassandras and W. Li, "A receding horizon approach for solving some cooperative control problems," in *Proc. 41st IEEE Conf. Dec. Contr.*, 2002, pp. 3760–3765.

[5] D. A. C. Non and C. G. Cassandras, "Cooperative mission control for unmanned air vehicles," in *Proc. AFOSR Workshop Dyn. Syst. Contr.*, 2002, pp. 57–60.

[6] W. Li and C. G. Cassandras, "Stability properties of a cooperative receding horizon controller," in *Proc. 42nd IEEE Conf. Dec. Contr.*, 2003, pp. 492–497.

[7] J. Bellingham, A. Richards, and J. How, "Receding horizon control of autonomous aerial vehicles," in *Proc. Amer. Contr. Conf.*, 2002, pp. 3741–3746.

[8] K. M. Passino, "Cooperative control for autonomous air vehicles," in *Cooperative Control and Optimization*. Norwell, MA: Kluwer, 2002, vol. 66, pp. 233–271.

[9] M. Baum and K. M. Passino, "A search-theoretic approach to cooperative control for uninhabited air vehicles," in *Proc. AIAA GNC Conf.*, 2001, pp. 898–908.

[10] J. Hespanha and H. Kizilocak, "Efficient computation of dynamic probabilistic maps," presented at the Proc. 10th Mediterranean Conf. Contr. Autom., Lisbon, Portugal, Jul. 9–12, 2002.

[11] J. Hespanha, H. Kizilocak, and Y. Ateskan, "Probabilistic map building for aircraft-tracking radars," in *Proc. Amer. Contr. Conf.*, 2001, pp. 4381–4386.

[12] A. Mahajan, J. Ko, and R. Sengupta, "Distributed probabilistic map service," in *Proc. 41st IEEE Conf. Dec. Contr.*, Dec. 2002, pp. 130–135.

[13] S. Ganapathy and K. M. Passino, "Agreement strategies for cooperative control of uninhabited autonomous vehicles," in *Proc. 2003 Amer. Contr. Conf.*, 2003, pp. 1026–1031.

[14] M. Jun, A. I. Chaudhly, and R. D'Andrea, "The navigation of autonomous vehicles in uncertain dynamic environments: A case study," in *Proc. 41st IEEE Conf. Dec. Contr.*, 2002, pp. 3770–3775.

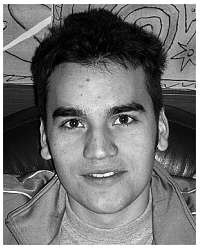
[15] S. Ganapathy and K. M. Passino, "Distributed agreement strategies for cooperative control: Modeling and scalability analysis," in *Proc. Conf. Cooperative Contr. Optimization*, 2002, pp. 127–147.

[16] D. A. Castañon and C. Wu, "Distributed algorithms for dynamic reassigment," in *Proc. 42nd IEEE Conf. Dec. Contr.*, 2003, pp. 13–18.

[17] R. W. Beard and T. W. McLain, "Multiple UAV cooperative search under collision avoidance and limited range communication constraints," in *Proc. 42nd IEEE Conf. Dec. Contr.*, 2003, pp. 25–30.

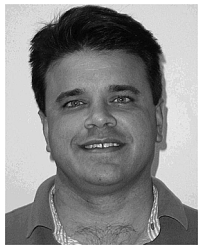
[18] A. Gil, S. Ganapathy, K. M. Passino, and A. Sparks, "Cooperative scheduling of tasks for networked autonomous vehicles," in *Proc. IEEE Conf. Dec. Contr.*, 2003, pp. 522–527.

[19] J. Finke, K. M. Passino, and A. Sparks, "Cooperative control via task-load balancing for networked uninhabited autonomous vehicles," in *Proc. IEEE Conf. Dec. Contr.*, 2003.
[20] J. Finke, K. M. Passino, S. Ganapathy, and A. Sparks, "Modeling and analysis of cooperative control systems for uninhabited autonomous vehicles," in *Cooperative Control*. New York: Springer-Verlag, 2004.
[21] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.
[22] N. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann, 1996.
[23] K. L. Burgess and K. M. Passino, "Stability analysis of load balancing systems," *Int. J. Control*, vol. 61, no. 2, pp. 357–393, Feb. 1995.
[24] K. M. Passino and K. Burgess, *Stability Analysis of Discrete Event Systems*. New York: Wiley, 1998.
[25] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal position," *Amer. J. Math.*, vol. 79, no. 3, pp. 497–516, Jul. 1957.
[26] H. Sussmann and G. Tang, "Shortest paths for reeds-sheep car: a worked out example of the use of geometric techniques in nonlinear optimal control," Sep. 1991, SYCON Rep.
[27] G. Laporte, "The vehicle routing problem: An overview of the exact and approximate algorithms," *Euro. J. Oper. Res.*, vol. 59, pp. 345–358, 1992.
[28] A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proc. Amer. Contr. Conf.*, 2002, pp. 1936–1941.



Jorge Finke (M'02) received the B.S. and M.S. degrees in electrical and computer engineering from Ohio State University, Columbus, in 2003 and 2004, respectively. He is currently working towards the Ph.D. degree in electrical engineering.

His research interests include cooperative task scheduling and resource allocation, coordinated motion of multi-agent systems, and bio-inspired methods for cooperative control systems.



Kevin M. Passino (S'79–M'90–SM'96–F'04) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, in 1989.

He is currently a Professor of Electrical and Computer Engineering at Ohio State University, Columbus, and Director of the Collaborative Center of Control Science that is funded by the Air Force Office of Scientific Research (AFOSR) and Air Force Research Library/Air Vehicles Directorate (AFRL/VA). He has served as the Vice President of Technical Activities of the IEEE Control Systems Society (CSS); was an elected member of the IEEE Control Systems Society Board of Governors; was the Program Chair of the 2001 IEEE Conference on Decision and Control; and is currently a Distinguished Lecturer for the IEEE Control Systems Society. He is co-editor (with P. J. Antsaklis) of the book *An Introduction to Intelligent and Autonomous Control,* (Kluwer, 1993); co-author (with S. Yurkovich) of the book *Fuzzy Control,* (Addison-Wesley, 1998); co-author (with K. L. Burgess) of the book *Stability Analysis of Discrete Event Systems,* (Wiley, 1998); co-author (with V. Gazi, M. L. Moore, W. Shackleford, F. Proctor, and J. S. Albus) of the book *The RCS Handbook: Tools for Real Time Control Systems Software Development,* (Wiley, 2001); co-author (with J. T. Spooner, M. Maggiore, R. Ordoñez) of the book *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques,* (Wiley, 2002); and author of *Biomimicry for Optimization, Control, and Automation,* (Springer-Verlag, 2005). For more information, see: http://www.ece.osu.edu/~passino/

Andrew G. Sparks received the B.S. and M.S. degrees in mechanical engineering from Massachuchettes Institute of Technology (MIT), Cambridge, in 1986 and 1988, respectively, and the Ph.D. degree in aerospace engineering from the University of Michigan, Ann Arbor, in 1995.

Since 1988, he has been with the Air Force Research Laboratory and its predecessor organizations at Wright Patterson Air Force Base, Dayton, OH. His research interests include the application of multivariable control theory to Air Force air and space systems, most recently the cooperative control of groups of unmanned air vehicles.