Concurrent Control of Mobility and Communication in Multi-Robot Systems

James Stephan, Jonathan Fink, Vijay Kumar, and Alejandro Ribeiro

***Abstract*—We develop a hybrid system architecture that enables a team of mobile robots to complete a task in a complex environment by self-organizing into a multi-hop ad hoc network and solving the concurrent communication and mobility problem. The proposed system consist of a two-layer feedback loop. An outer layer performs infrequent global coordination of the team and operates at a centralized coordination unit. The inner layer operates in a decentralized manner and is responsible for continuous determination of motion control and data communication variables. This two-layered architecture allows for the lightweight coordination and responsiveness that is typical of decentralized systems without the characteristic drawback of convergence to local minima, which are avoided by the operation of the outer loop. This results in a system that allows a team of robots to complete a task in complex environments while maintaining desired end-to-end data rates and retaining the light coordination and responsiveness of decentralized systems. The behavior of the system is evaluated in simulations and experiments. In particular, we demonstrate: (i) Successful task completion in complex environments by avoiding local minima. (ii) Efficient operation for large team sizes and environments. (iii) The achievement of equal or greater end-to-end data rates as compared to a centralized system. (iv) Robustness to unexpected events such as motion restriction. We conclude by exemplifying the efficacy of the proposed system to complete a high-level task, by considering hallway patrolling while maintaining a target end-to-end data rate for the lead member of the patrol.**

I. Introduction

Cooperative task completion for teams of autonomous mobile robots has seen an increase in interest over the past few years. This higher-level interest is due to both, the decrease in the cost of robotic components, as well as the increase in computational abilities of such robots. These factors have lead to advancements in the field of multi-robot systems, ranging from flock behavior, to surveillance, to exploration. While these robots have become cheaper and more ubiquitous, the underlying assumption in most systems is the existence of a wireless network over which the robots can communicate. This assumption, while valid in some scenarios, prevents the current algorithms from operating in environments where they are of the most use. For instance, when performing search and rescue in a collapsed building after an earthquake, the assumption of the existence of a wireless network is most likely invalid. This requires that the multi-robot team not only execute the search algorithm, but also create and maintain the ad-hoc network that such an algorithm relies on.

Two main factors complicate the creation of such a network. The first being the dynamic motion of the robots in the network; as a robot moves through the environment, the link between it and every other robot in the network changes. These changes at the link level, no matter how small, can have large effects on the network, and as such the system needs to take this into account as the robots move. The second complication arises from the random nature of wireless links. In environments with few obstructions inter-robot distances dominates the change in the wireless links. As the environments become more complex, the effects of obstacles, in the form of shadowing shadowing, and multi-path propagation, in the form of fading, begin to dominate.

Many systems attempt to control the effect of motion on the underlying communication network. These systems rely on graph-theoretic metrics to measure and maintain the desired communication network properties. We can organize the control laws used in these systems into two distinct classes, global and local. When implementing a global control law, the system seeks to command each robot at the same time to perform an action based on global information, in order to reach a globally-optimal solution. These systems have demonstrated that it is feasible to control the global properties of the underlying communication graph, such as the second eigenvalue of the Laplacian [1], [2] or k-connectedness [3]. In contrast to global, a local control law utilizes local information and therefore cannot guarantee a globally optimal solution. However, using only local information, one can develop systems that are able to maintain connectivity through either distributed estimation of the Laplacian's second eigenvalue [4], [5], hysteresis [6], switching network theory [7], [8], or dual gradient descent algorithms [9].

To mitigate the random nature of the wireless links, these systems use one of three categories of link models, depending on the assumed environment. The first category is a binary disc model, in which two robots are able to communicate if they are within some nominal distance of each other, [6]–[8], [10]. While this model is effective in simple environments, [11] shows that small changes in the nominal distance used can have dramatic effects on the network topology,. The second category models the reliability of a link as a function of the inter-robot distance, [1]–[5], [9], [12]. This class of model performs well in complex environments but loses accuracy as the number of obstacles increases. The third, and most recent, category is a probabilistic model in which not only is the expected value of the channel used but also the variance to

This work is supported by the ARL MAST-CTA under Grant W911NF-08-2-0004. J. Stephan, V. Kumar, and A. Ribeiro are with the Department of Electrical & Systems Engineering, University of Pennsylvania, Philadelphia, PA. Email {jstephan, kumar, aribeiro}@seas.upenn.edu. J. Fink is with the US Army Research Laboratory, Adelphi, MD. Email at jonathan.r.fink3.civ@mail.mil

capture the effects of shadowing and fading, [13]–[17].

Most of the previously mentioned systems do not consider network routing, only two [9], [16] incorporate network routing into the problem formulation. Incorporating routing is advantageous because it focus on maintaining reliability of the links that are actually being used for communication and it further adds the ability to reroute information to more reliable links. However, incorporating network routing into motion control results in a complex optimization problem requiring joint optimization of motion and routing. The resulting systems that solve this problem are either centralized, requiring global coordination but able to obtain globally-optimal solutions, or distributed, needing only local information but susceptible to local minima. The system developed in [16] is an example of a globally-optimal centralized system that is able to control a team of robots in a complex environment using global coordination, while maintaining a minimum end-to-end rate between a robot and an access point. In contrast, the system developed in [9] is an example of a distributed system that can also maintain a minimum end-to-end rate between a robot and an access point, but local minima limit it to simple environments.

In this paper, we propose a hybrid system that combines the benefits of both a distributed and a globally-optimal centralized system while avoiding their deficiencies. To achieve this we construct a multi-layer feedback system composed of two distinct layers, an outer and inner layer, each with specific responsibilities. The outer layer is responsible for the infrequent global coordination and the inner layer is responsible for the motion control and network routing at the robot level. The behavior of this system is evaluated in simulations and experiments. We begin the paper by defining the problem that is under consideration in this paper - Section II. Then we detail the construction and implementation of the hybrid system - Section III. We then describe the experimental platform and operating environments - Section IV. Next we present the results of two simulations, the first highlights the limitations of the distributed system in [9] that are not present in the hybrid system and the second demonstrates the scalability of the hybrid system to teams with over 20 robots - Section V. Shifting to empirical validation, we offer results from two experiments comparing the performance of the hybrid system to the the centralized system in [16] - Section VI. The first experiment demonstrates the superior performance of the hybrid system to the centralized system and the second demonstrates the ability of the hybrid system to mitigate unexpected occurrences that cause the centralized system to fail. Finally, we present an application of the fundamental algorithms presented here to a long-duration monitoring task - Section VII.

II. Problem Formulation

In this paper we consider a team of N mobile robots operating in a known environment. The position of robot i in the environment is $x_i(t) \in \mathbb{R}^3$ and the collection of all the robot positions, called a formation, is $\mathbf{x}(t) \in \mathbb{R}^{3N}$. The team is deployed at time t_0 in formation $\mathbf{x}(t_0)$ and given a task that it has to accomplish by time t_f. We assume that the task is given in the form of a scalar convex function, $\Gamma(\mathbf{x}) : \mathbb{R}^{N\times 3} \to \mathbb{R}$ whose minimum $\mathbf{x}^*$ is the desired final formation. If the team's trajectory satisfy $\mathbf{x}(t_f) = \mathbf{x}^*$ we say that the task has been successfully completed. To model the kinematics of a single robot, we begin with a single input control system, $\dot{x}_i(t) = f(x_i(t), u_i(t))$, with input $u_i(t)$. We only consider robots with simple dynamics that are assumed fully controllable so that we can write $\dot{x}_i(t) = u_i(t)$.

Our goal here is to find control inputs $\dot{\mathbf{x}}(t)$ for the team as a whole. These control inputs are required to complete the task successfully, while avoiding environmental obstacles and collisions with each other. Define then $\mathcal{O}$ as the set of environmental obstacles and the configuration free space $\mathcal{F}$ as the set of formations $\mathbf{x} \in \mathbb{R}^{3N}$ for which robots don't collide with each other, namely, formations such that $\|x_i(t) - x_j(t)\|^2 > \delta$, and for which they remain outside of the obstacle space, namely, $x_i(t) \notin \mathcal{O}$. Using this definition of free space and the integral team trajectory that follows from full controllability, we can write trajectory planning as the optimization problem,

$$\min_{\dot{\mathbf{x}}(t)} \Gamma\left(\mathbf{x}\left(t_f\right)\right) \tag{1}$$
$$\text{s. t. } \mathbf{x}(t) = \mathbf{x}(t_0) + \int_0^t \dot{\mathbf{x}}(s) ds, \quad \mathbf{x}(t) \in \mathcal{F},\ t \in [t_0, t_f].$$

In (1), we find a trajectory whose final formation $\mathbf{x}(t_f)$ minimizes the task function $\Gamma(\mathbf{x})$ while evolving according to the control law $\mathbf{x}(t) = \mathbf{x}(t_0) + \int_0^t \dot{\mathbf{x}}(s)ds$ and staying in configuration free space $\mathcal{F}$. This problem formulation is well understood and a variety of solution methodologies exist that provide a guarantee that $\mathbf{x}(t_f) = \mathbf{x}^*$ if this is possible. In this paper we modify (1) by adding communication constraints as we explain in the following section.

A. Communication Links and Networking

We model communication using the normalized point to point rate $R(x_i, x_j) : \mathbb{R}^6 \to [0, 1]$ that measures the information rate from robot i at position x_i to robot j at position x_j. Using these point to point rate functions we construct the rate matrix, $\mathbf{R}(\mathbf{x}) \in \mathbb{R}^{N\times N}$, with entries $R_{ij}(\mathbf{x}) = R(x_i, x_j)$. The information flow over the network is specified by a set of routing variables $\alpha_{ij} \in [0, 1]$, that denote the fraction of time that robot i transmits data to robot j. As with the communication rates, we group the variables in the matrix $\boldsymbol{\alpha} \in \mathbb{R}^{N\times N}$, with entries α_{ij}. Further observe that since they represent time fractions we must have that $\sum_{j=1}^N \alpha_{ij} \leq 1$ for all i. With these definitions it follows that the product $\alpha_{ij} R(x_i, x_j)$ is the rate at which data is transmitted over the communication link from robot i to robot j. We can then compute the communication rate margin for robot i as the difference between the rate at which information is sent to other robots and the rate at which it is received,

$$a_i(\boldsymbol{\alpha}, \mathbf{x}) = \sum_{j=1}^N \alpha_{ij} R(x_i, x_j) - \sum_{j=1,\, i\notin\mathcal{D}}^N \alpha_{ji} R(x_j, x_i). \tag{2}$$

In the second summation in (2) the set $\mathcal{D}$ represents the information destinations. They are excluded because destinations $i \in \mathcal{D}$ don't send information out. In order to prevent unbounded growth of the communication queues and ensure network stability we must have $a_i(\boldsymbol{\alpha}, \mathbf{x}) \geq 0$ for all i. With this restriction the value of $a_i(\boldsymbol{\alpha}, \mathbf{x})$ can now be reinterpreted as the rate at which robot i can add data to the network.

The formulation in (2) is for a static formation and a single information flow. In a robotic team we have dynamic formations and multiple information flows. To allow for dynamic formations we add a dependence on time to (2),

$$a_i(\boldsymbol{\alpha}(t), \mathbf{x}(t)) = \sum_{j=1}^{N} \alpha_{ij}(t) R(x_i(t), x_j(t)) - \sum_{j=1,\, i \notin \mathcal{D}}^{N} \alpha_{ji}(t) R(x_j(t), x_i(t)). \quad (3)$$

This allows the routing solutions to adjust as the team's formation changes. For the multiple information flows, we assume that the communication constraints are given as a collection of K quality of service (QoS) requirements, where each requirement consists of a set of destinations, $\mathcal{D}_k \subseteq \{1, \ldots, N\}$, and a minimum data rate, $a_{i,m}^k \in [0 \ldots 1]$ for all terminals i. When all of the QoS requirement are satisfied, we say that network integrity is achieved. To accommodate the multiple QoS requirements, we extend $\boldsymbol{\alpha}$ from $\mathbb{R}^{N \times N}$ to $\mathbb{R}^{N \times N \times K}$. This results in a per requirement version of (3),

$$a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) = \sum_{j=1}^{N} \alpha_{ij}^k(t) R(x_i(t), x_j(t)) - \sum_{i=1,\, i \notin \mathcal{D}_k}^{N} \alpha_{ji}^k(t) R(x_j(t), x_i(t)). \quad (4)$$

As with the single flow formulation, α_{ij}^k represents a time fraction, thus it must be $\sum_{k=1}^{K} \sum_{i=1}^{N} \alpha_{ij}^k = \sum_{j,k} \alpha_{ij}^k \leq 1$ for all i. Using (4) we can write a set of constraints that when satisfied ensure that for the given trajectory, the series of routing solution preserve network integrity. These constraints can be written as,

$$a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) \geq a_{i,m}^k, \qquad \sum_{j,k} \alpha_{ij}^k(t) \leq 1, \quad (5)$$

and are required to hold for all terminals i, flows k, and times $t \in [t_0, t_f]$. Adding the constraints in (5) to the problem statement in (1) results in the full mobility and communication optimization problem,

$$\begin{aligned} &\min_{\dot{\mathbf{x}}(t)} \Gamma\left(\mathbf{x}\left(t_f\right)\right) && (6) \\ &\text{s. t. } \mathbf{x}(t) = \mathbf{x}(t_0) + \int_0^t \dot{\mathbf{x}}(s) ds, \quad \mathbf{x}(t) \in \mathcal{F}, \\ &\qquad a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) \geq a_{i,m}^k, \quad \sum_{j,k} \alpha_{ij}^k(t) \leq 1. \end{aligned}$$

where the constraints are assumed to hold for all terminals i, flows k, and times $t \in [t_0, t_f]$.

The goal of this paper is to find a trajectory that is optimal in the sense of solving problem (6). Methods to plan coordinated trajectories at a central location that approximate a solution to (6) exist [16]. The drawback of this centralized strategy is the cost of aggregating environmental information and disseminating plans which, e.g., makes it difficult to react to changing conditions. Distributed methods to find local minima of (6) also exist [9]. Their drawback is the limited progress that such a controller can make in a complex environment. This paper develops a hybrid methodology that utilizes a centralized controller to feed intermediate points to a distributed controller that is responsible for the determination and execution of motion and communication variables as we explain in the following section.

III. Hybrid System

The goal of the hybrid system is to drive an arbitrary number of mobile robots through a complex environment while maintaining a minimum QoS in order to complete a given task. To accomplish this, we propose an architecture that consists of a two stage feedback system shown in Figure 1. This architecture is composed of an outer centralized planning feedback loop and an inner distributed control feedback loop. The process is initiated by the user providing a global task function $\Gamma(\mathbf{x})$ to the outer loop. This begins the planning process which generates a set of dense *candidate* trajectories for the system that we denote as $\tilde{\mathbf{x}}(t) = \{\tilde{x}_i(t)\}_{i=1}^N$. Given the model that has been provided as input to the outer loop, these trajectories give an approximate robust solution to a stochastic formulation of (6) – see Section III-A. The candidate trajectory $\tilde{\mathbf{x}}(t)$ is never executed but rather fed to a waypoint generator that converts the dense trajectories into a series of waypoints for each robot,

$$\mathcal{X}_i = \left\{\tilde{x}_i(\tau_w)\right\}_{w=1}^W. \quad (7)$$

The waypoints in (7) are sampled at the same set of times $\{\tau_w\}_{w=1}^W$ for all robots and represent a decomposition of (6) into subproblems that can be solved by the distributed control inner loop.

The waypoints in (7) serve as sequential inputs to the distributed control loop. In contrast to the centralized loop which only operates only when a new task is given, the distributed loop operates continuously on each robot. The distributed controller accepts a target location $x_{i,0}$ as input and attempts to drive the robot to that location while avoiding physical obstacles and preserving network integrity. The process that is used to implement this driving is distributed in that it relies on communication between adjacent robots only – see Section III-B. When robot i receives a new set of waypoints $\mathcal{X}_i$ from the global planner its waypoint curator is responsible for updating the target location $x_{i,0}$. This is done by setting $x_{i,0}$ to the first waypoint in the series, i.e., by making $x_{i,0} = \tilde{x}_i(\tau_1)$. Then, when the distance to the target location falls bellow a given tolerance $\omega > 0$, namely, when $\|x_i(t) - x_{i,0}\| \leq \omega$, the waypoint is declared reached and $x_{i,0}$ is updated to the next waypoint in the series. The curator advances though successive waypoints until the final waypoint is reached at which time the task is declared accomplished for robot i.

Notice that the candidate trajectory $\tilde{\mathbf{x}}(t)$ generated by the centralized planner is optimal for the model that is available. However, given the possibility for model mismatch, the trajectory is not necessarily optimal during actual deployment. The

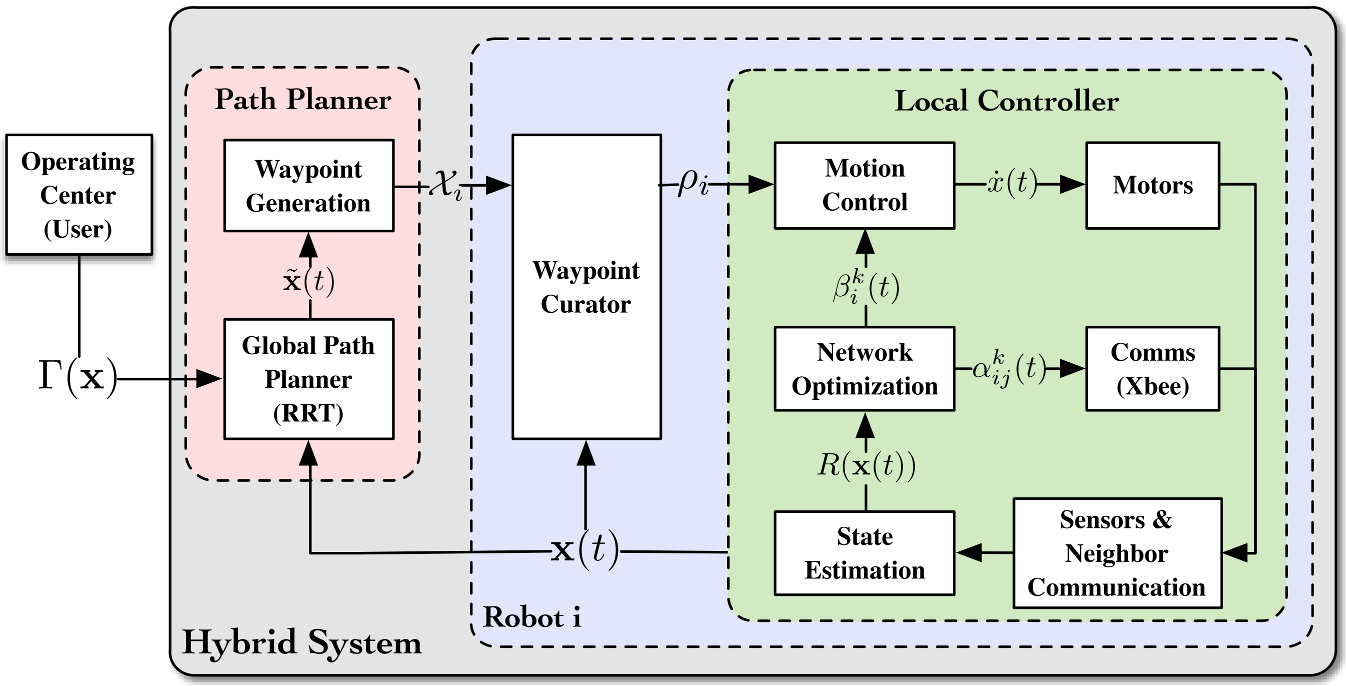


Fig. 1: Hybrid architecture diagram. The red indicates the outer, centralized, loop of the system while the green indicates the inner, local controller, loop.

distributed controller corrects for this mismatch because, due to the small communication overhead of its implementation, it can adapt to the conditions observed during execution. Thus, the hybrid system proposed here resolves the lack of adaptability of the centralized planner while avoiding the local minima that can limit the progress of the distributed control loop. We describe the centralized and distributed loops in the following sections.

A. *Centralized Path Planning*

The purpose of global path planning is to find a trajectory that solves (6). The challenge in finding this plan is that we want to construct long term trajectories that visit points in space for which the channel reliabilities that appear in (6) can't be measured. Solving this problem is therefore not possible because the constraints can't be evaluated. We overcome this problem with the introduction of a probabilistic formulation [16]. Reinterpret then R_{ij} as a Gaussian random variable with mean $\bar{R}_{ij}$ and variance $\tilde{R}_{ij}$. With this modeling assumption, the flow rates in (4) become random variables as well and we can reformulate the satisfaction of (5) in a probabilistic manner. To do so, introduce a tolerance ϵ and replace the deterministic constraint in (5) by the probabilistic constraint that requires the target rates $a_{i,m}^k$ to be achieved with probability at least $1-\epsilon$,

$$\mathrm{P}\left[a_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t)) \geq a_{i,m}^k\right] > 1-\epsilon, \tag{8}$$

Observe that the flow $a_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t))$ has a normal distribution because the rates R_{ij} are assumed to be Gaussian and $a_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t))$ is a linear function of R_{ij} for given $\mathbf{x}$. As it follows from (4), the mean $\bar{a}_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t)) := \mathbb{E}\left[a_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t))\right]$ of this Gaussian variable can be written as

$$\bar{a}_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t)) = \sum_{j=1}^{N} \alpha_{ij}^k(t)\bar{R}(x_i(t),x_j(t)) - \sum_{j=1, i\notin\mathcal{D}_k}^{N} \alpha_{ji}^k(t)\bar{R}(x_j(t),x_i(t)), \tag{9}$$

and the corresponding variance $\tilde{a}_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t)) := \mathrm{var}\left[a_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t))\right]$ is given by the expression

$$\tilde{a}_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t)) = \sum_{j=1}^{N} (\alpha_{ij}^k(t))^2\tilde{R}(x_i(t),x_j(t)) + \sum_{j=1, i\notin\mathcal{D}_k}^{N} (\alpha_{ji}^k(t))^2\tilde{R}(x_j(t),x_i(t)). \tag{10}$$

Using the mean and variances in (9) and (10) and letting $\Phi^{-1}(\epsilon)$ stand for the inverse Gaussian complementary cumulative distribution function, we can write the probability in (8) as

$$\frac{\bar{a}_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t)) - a_{i,m}^k}{\sqrt{\tilde{a}_i^k(\boldsymbol{\alpha}(t),\mathbf{x}(t))}} \geq \Phi^{-1}(\epsilon). \tag{11}$$

The constraint in (11), being dependent on the probabilistic model variables $\bar{R}_{ij}$ and $\tilde{R}_{ij}$, can be evaluated by the global path planner. We therefore modify (6) to write the optimization problem

$$\min_{\dot{\mathbf{x}}(t),\boldsymbol{\alpha}(t)} \Gamma\left(\mathbf{x}\left(t_f\right)\right) \tag{12}$$
$$\text{s. t.} \quad \mathbf{x}(t) = \mathbf{x}(t_0) + \int_0^t \dot{\mathbf{x}}(s)ds, \quad \mathbf{x}(t) \in \mathcal{F},$$
$$\bar{a}_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) \geq a_{i,m}^k + \Phi^{-1}(\epsilon)\sqrt{\tilde{a}_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t))},$$
$$\sum_{j,k} \alpha_{ij}^k(t) \leq 1,$$

where, as in (6), the constraints hold for all terminals i, flows k, and times $t \in [t_0, t_f]$.

The formulation in (12) is termed robust routing in [16] where the constraint in (11) is shown to define a second order cone as long as $\epsilon < 0.5$ – which is not restrictive since we want ϵ to be small. Therefore, the determination of routing variables $\boldsymbol{\alpha}$ that satisfy this constraint can be written as a second order cone program if the formation $\mathbf{x}(t)$ is given. This implies that determining routing variables for a given formation can be done in polynomial time by using convex programing techniques [18]. In particular, checking if routing variables that satisfy the constraint in (11) exist is tractable, which in turn implies that finding formations that are feasible for the problem in (12) is also tractable. This is exploited in the solution of (12) with a Rapidly Exploring Random Tree (RRT) [19] as we explain in Section III-A1.

Do notice that acquiring an accurate probabilistic model of reliabilities is itself challenging. The values of $\bar{R}_{ij}$ and $\tilde{R}_{ij}$ are dependent on shadowing and fading effects that can vary substantially in different propagation environments. The problem formulation in (12) circumvents this problem with the use of the robust routing constraint in (8). If the available propagation model is rough, this is captured in large values for the variances $\tilde{R}_{ij}$, which in turn result make it difficult to find formations that satisfy (8). This leads to conservative plans that can later be refined by the distributed controller which, different from the global planner, can rely on online modification of the propagation model.

1) Rapidly exploring random tree: The robust routing constraints in (12) modify the configuration free space $\mathcal{F}$. On top of physical obstacles and collision avoidance, we also need to remove formations for which satisfying (11) is not possible – which, as we argued before, can be done in polynomial time. We explore the resulting free space with a RRT. The RRT algorithm is initialized by first setting the current valid formation as the root of a tree. Then the following process is repeated until a formation that satisfies the task objective is added to the tree, $\Gamma(\mathbf{x}) = \Gamma(\mathbf{x}^*)$. A random point from the configuration space, corresponding to a formation, is drawn. If the formation does not satisfy (11) then is it discarded and another point is sampled. When a formation that satisfies (11) is found, the nearest node in the tree is found. For this configuration space a simple Euclidean distance is used to determine the nearest node. Now using the nearest node as a starting point the system attempts to reach the sampled point under the motion dynamics of the platform. The system either reaches the sampled point at which time the point is added to the tree with the a branch from the nearest node to it, or some obstacle in the environment prevents a simple path. If the system is prevented from reaching the destination the formation that corresponds to the halting point is checked against (11). If the halting point is feasible it is added to the tree with a branch from the nearest node, otherwise the sample is discarded and the search process is repeated. When a formation that satisfies the task objective is added to the tree, the process is terminated.

The path through the tree starting at the current formation to the goal formation is then extracted. Since a node can only be added to the tree if the flow constraints are satisfied it is guaranteed that for every node in the final path the flow constraints are satisfied. This path corresponds to a feasible trajectory for each robot from its current location to a final location.

B. Distributed Controller

The purpose of the distributed controller is to manage the mobility and network routing of an individual robot using the waypoints generated by the centralized controller [cf. (7)]. This dual mandate requires that we solve both the motion control and the network routing. To accomplish this, we run concurrently a continuous-time motion-gradient control and a discrete-time dynamic computation of optimal communication variables [9].

For the motion-control portion of the distributed controller we employ a navigation function that is capable of driving the robot to a goal location $x_{i,0}$ while avoiding obstacles [20], [21]. However, obstacles here are not physical but determined by the need to guarantee network integrity. Assume then that routing variables $\boldsymbol{\alpha}(t)$ are given – their adaptive computation is explained in Section III-B1 – and recall that network integrity is defined as the satisfaction of the QoS requirements in (5). If we further introduce a strictly positive tolerance $e > 0$ we can thus define the obstacle function for robot i associated with the kth constraint as

$$\beta_i^k(\mathbf{x}(t)) \triangleq \sum_{j=1}^{N} \alpha_{ij}^k(t) R_{ij}(\mathbf{x}(t)) - \sum_{j=1}^{N} \alpha_{ji}^k(t) R_{ji}(\mathbf{x}(t)) - a_{i,m}^k + e. \tag{13}$$

The function $\beta_i^k(\mathbf{x}(t))$ is positive when the k^{th} QoS requirement for robot i is satisfied within the tolerance e for the current formation $\mathbf{x}(t)$, and negative otherwise. This allows a gradient controller to treat the zero points of $\beta_i^k(\mathbf{x}(t))$ as the border of a virtual obstacle that, if crossed, would result in a violation of the integrity of the kth flow. Observe that, different from the centralized controller, this QoS constraint can be accurately evaluated at the current location because the propagation model can be adapted to observations. Also notice that the tolerance e simply implies a reduction of the minimum acceptable rate from $a_{i,m}^k$ to $a_{i,m}^k - e$. They are kept separate to emphasize that the distributed controller requires some leeway to increase its range of motion for a given set of communication variables.

The obstacle define by the function $\beta_i^k(\mathbf{x}(t))$ in (13) is associated with robot i and flow k. For robot i all the QoS obstacle functions can be combined into the single network integrity obstacle function,

$$\beta_i(\mathbf{x}(t)) = \min_{k=1,\dots,K} \ \beta_i^k(\mathbf{x}(t)). \tag{14}$$

Integrity of all flows at robot i is guaranteed within the tolerance e if the joint obstacle function is $\beta_i(\mathbf{x}(t)) > 0$. To create an attraction to $x_{i,0}$ we use the goal potential function $\rho_i\left(\mathbf{x}(t)\right) = \|x_i(t) - x_{i,0}\|^2$. Using this definition of $\rho_i(\mathbf{x}(t))$ and the obstacle function in (14) we can define the navigation function,

$$\phi_i\left(\mathbf{x}(t)\right) = \frac{\rho_i\left(\mathbf{x}(t)\right)}{\left(\rho_i\left(\mathbf{x}(t)\right)^\kappa + \beta_i\left(\mathbf{x}(t)\right)^2\right)^{1/\kappa}}, \tag{15}$$

where the order parameter satisfies $\kappa > 2$ and has to be chosen sufficiently large. This navigation function has the desirable properties of being $\phi_i\left(\mathbf{x}(t)\right) \in [0, 1]$ always, satisfying $\phi_i\left(\mathbf{x}(t)\right) = 0$ when $\mathbf{x}(t) = x_{i,0}$, and being such that $\phi_i\left(\mathbf{x}(t)\right) \to 1$ when a QoS requirement is about to be violated. Taking advantage of these properties we can drive robot x_i towards $x_{i,0}$ while guaranteeing network integrity with the gradient descent controller

$$\dot{x}_i(t) = -\nabla_{x_i}\phi_i\left(\mathbf{x}(t)\right). \tag{16}$$

The value of κ is used to control the regions that are affected by the obstacles, the larger κ is the more localized the effects are to the obstacles. As shown in [20], [21], the controller in (16) is able to reach $x_{i,0}$ while avoiding obstacles that are not intersecting and spherical. The obstacle defined by (14) is not spherical and it may be that (16) stops at a local optimum. This is not a concern because the sampling of waypoints is done fine enough to preclude this possibility. Notice that in the complex environments considered here it is also necessary to avoid physical obstacles. This is standard problem that we can solve, e.g., with a modification of (14) to include the distance to these physical obstacles.

Assuming that feasible routing variables $\boldsymbol{\alpha}(t)$ satisfying $a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) \geq a_{i,m}^k$ are available for all configurations for which these variables exist, the controller in (16) coupled with proper generation of waypoints would drive the team to a configuration that solves (6). What is left, therefore, is the design of a distributed mechanism to find these feasible routing variables. We do so in the following section.

1) Adaptation of routing variables: The motion control of the robot is predicated on the virtual obstacles created in (13) which are computed directly from the routing solution $\boldsymbol{\alpha}(t)$. When starting at a waypoint and moving to the next, we have available the routing solution $\boldsymbol{\alpha}(t)$ that has been computed by the centralized controller. This solution can be used for initialization, but an accurate description of the obstacle space necessitates the routing solutions $\alpha(t)$ used in the controller in (16) to adapt as the robots move. In order to adapt these variables we adopt a modified version of (6) in which only the network integrity constraints are used.

Specifically, extract the network integrity constraint $a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) \geq a_{i,m}^k$ from (6) and rewrite it as $a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) = a_i^k \geq a_{i,m}^k$. The idea here is to adapt the routing variables so that the rates a_i^k are as large as possible – but not smaller than the minimum requirement $a_{i,m}^k$. To do so introduce weights $w_i^k > 0$ and $w_{ij}^k > 0$ and define the weighted proportional fair utility $U_i^k(a_i^k) = w_i^k \log(a_i^k)$ as well as the weighted quadratic penalty terms $V_{ij}^k(\alpha_{ij}^k) = -w_{ij}^k(\alpha_{ij}^k)^2$ that we incorporate into the optimization problem

$$\begin{aligned}\boldsymbol{\alpha}(t) = &\operatorname*{argmax}_{a_i^k, \alpha_{ij}^k} \ \sum_{k=i}^{K}\sum_{i=1}^{N}\left[U_i^k(a_i^k) + \sum_{j=1}^{N} V_{ij}^k(\alpha_{ij}^k)\right] \\ &\text{s. t.} \qquad a_i^k(\boldsymbol{\alpha}, \mathbf{x}(t)) = a_i^k \geq a_{i,m}^k, \quad \sum_{j,k}\alpha_{ij}^k(t) \leq 1.\end{aligned} \tag{17}$$

Some remarks are in order. To guarantee that a solution to (6) is found we need to find, for any given spatial configuration $\mathbf{x}(t)$, a set of routing variables that satisfy $a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) = a_i^k \geq a_{i,m}^k$ for all robots i and flows k. However, there are, in general, many variables that satisfy these constraints. The formulation in (17) resolves this indeterminacy by selecting the variables $\boldsymbol{\alpha}(t)$ that maximize the objective $\sum_{k=i}^{K}\sum_{i=1}^{N} U_i^k(a_i^k) + \sum_{j=1}^{N} V_{ij}^k(\alpha_{ij}^k)$. Since these variables are optimal in (17) they are feasible in particular, but the presence of the fair utility term $U_i^k(a_i^k) = w_i^k \log(a_i^k)$ also makes the difference between the achieved rate $a_i^k(\boldsymbol{\alpha}(t), \mathbf{x}(t)) = a_i^k$ and the minimum rate $a_{i,m}^k$ large. Assuming that rates $R_{ij}(\mathbf{x})$ change slowly in space, this allows more freedom of movement for fixed routing variables and, consequently, less frequent recomputation of the solution of (17). The quadratic penalty terms $V_{ij}^k(\alpha_{ij}^k) = -w_{ij}^k(\alpha_{ij}^k)^2$ hedges the solution against errors in the estimation of the rates $R_{ij}(\mathbf{x})$ because they ensure that a link is not overly utilized when similar links are available.

The problem formulation in (17) answers the question of which routing variables to plug in the definition of the obstacle function in (13) but, as formulated, (17) requires global coordination to compute the optimal routing solution. A distributed method to solve (17) follows from the observation that, for a given spatial configuration $\mathbf{x}(t)$, the problem is convex and can therefore be equivalently solved in the dual domain with a gradient descent method. Introduce then a non-negative dual variables $\lambda_i^k(t_n)$ associated with each of the $a_i^k(\boldsymbol{\alpha}, \mathbf{x}) = a_i^k$ constraints in (17), where t_n is used to track the current iteration. These variables can be grouped into a matrix, $\boldsymbol{\lambda}(t_n) \in \mathbb{R}^{N\times K}$. Using the dual variables and the constraints we can write the Lagrangian,

$$\begin{aligned}\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{x}) = \sum_{k=i}^{K}\sum_{i=1}^{N}\Bigg[&U_i^k(a_i^k) + \sum_{j=1}^{N} V_{ij}^k(\alpha_{ij}^k) \\ &+ \lambda_i^k\left(\sum_{j=1}^{N}\alpha_{ij}R(x_i, x_j) - \sum_{j=1,\, i\notin\mathcal{D}}^{N}\alpha_{ji}R(x_j, x_i) - a_i^k\right)\Bigg].\end{aligned} \tag{18}$$

We can rearrange the terms in (18) into a sum of local

Lagrangians, $\mathcal{L}(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{x}) = \sum_{i=1}^{N} \mathcal{L}_i(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{x})$, where

$$\mathcal{L}_i(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{x}) = \sum_{k=1}^{K} U_i^k(a_i^k) - \lambda_i^k a_i^k + \sum_{j=1}^{N} \left[V_{ij}^k(\alpha_{ij}^k) + \alpha_{ij}^k R_{ij}(\lambda_i^k - \lambda_j^k) \right]. \quad (19)$$

Notice that $\mathcal{L}_i(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{x})$ only depends on robot i's information, a_i^k, λ_i^k, and α_{ij}^k, as well as only the λ_j^k's for which $R_{ij} > 0$. This indicates that in order to compute the value of $\mathcal{L}_i(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{x})$ robot i is only required to collect the λ_j^k of its immediate neighbors. This can be achieved by a simple exchange of λ_i^k between all neighboring pairs. Upon receipt of its neighbors' variables $\lambda_j^k(t_n)$ robot i is able to compute its optimal rates and its part of the routing solution, at time t_n, by solving,

$$a_i^k(t_n), \{\alpha_{ij}^k(t_n)\}_{j=1}^{N} = \operatorname{argmax} \mathcal{L}_i(\boldsymbol{\lambda}(t_n), \boldsymbol{\alpha}(t_n), \mathbf{x}(t_n)).$$
$$\text{s.t.} \qquad a_i^k \geq a_{i,m}^k, \ \sum_{j,k} \alpha_{ij}^k(t) \leq 1. \quad (20)$$

After the optimal rates and routes are determined for time t_n the next step is to update the value of λ_i^k. To maintain the non-negative requirement for λ_i^k, we use a non-negative projection $\mathbb{P}[y]$, which returns y is $y \geq 0$ and 0 if $y < 0$. Using this projection we update $\lambda_i^k(t_n)$ by following $\nabla_{\lambda_i^k} \mathcal{L}_i(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{x})$, using the values of $a_i^k(t_n)$ and $\alpha_{ij}^k(t_n)$ found in (20),

$$\lambda_i^k(t_{n+1}) = \mathbb{P}\left[\lambda_i^k(t_n) - \epsilon \Big(\sum_{j=1}^{N} \alpha_{ij}^k(t_n) R_{ij} - \sum_{j=1}^{N} \alpha_{ji}^k(t_n) R_{ji} - a_i^k(t_n) \Big) \right], \quad (21)$$

These updated values are then shared with all the robots within communication range so they can be used in the next iteration of (20). This process is repeated and converges to the optimal routing solution when the formation is static. If the formation is changing the resulting solutions will be near optimal, and the deviation from optimality is dependant on the frequency of the iterations and the allowable velocity of the robots.

IV. Experimental Configuration

A. Robotic Platform

For this paper, we use at team of *Scarab*s [22], a custom built robot designed at the University of Pennsylvania, as our robotic platform. The newest version of the *Scarab* includes onboard computing, a Hokuyo UTM-30LX scanning laser range finder with a 30 meter range, and two Robo Claw 5 amp Motor Controllers. The motors are used to drive two of the three wheels, while the third is a passive omni-directional wheel. Since the *Scarab* is a small differential-drive platform, it is straight-forward to apply feedback linearization in order to obtain appropriate control inputs given the kinematic control laws presented in this paper. The on-board computer contains an Intel i5 3.8 GHz processor, 4 GB of RAM, and a 60 GB SSD hard drive with a full installation of Ubuntu 12.04 LTS. An image of a standard *Scarab* can be seen in Fig. 2.

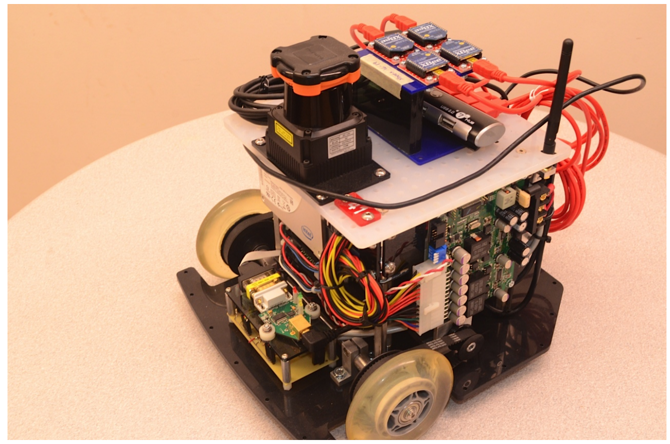


Fig. 2: The newest generation of the *Scarab*s. The XBees are mount on top of the platform behind the Hokuyo.

For wireless communication between *Scarab*s we use the Digi International XBee transceivers. These modules allow the user to control frequency and power. The XBee radios are capable of transmission on 16 evenly spaced channels in the 2.4 GHz spectrum. The XBee radio also allows for 5 discrete power levels, ranging from -10 dBm to 0 dBm. The XBee transmits data via a fixed packet size of 100 bytes, with a preamble the result is an effective payload size of 90 bytes for each transmission.

As shown in Fig. 2, each *Scarab* in these experiments contains 4 XBees. Each Xbee is configured to transmit at the minimum power of -10 dBm to force reliance on the other robots on the team while keeping the size of experiments manageble. Additionally, each XBee is responsible for communication on a different frequency. The frequencies chosen are evenly spaced to allow for maximum signal isolation between radios. This allows for the communication between one pair *Scarab*s to not interfere with communication between another pair of *Scarab*s, which is important as our routing solution does not consider interference.

The hybrid system, as well as all of the software used on the robots, is implemented in the Robotic Operating System (ROS), specifically Hydro. This allows for similar operation in both simulations and experiments. For localization, the AMCL library in ROS is used which leverages an existing map of the environment, the odometery from the robots motors, as well as the laser readings from the Hokuyo to provide an accurate estimate of the robot in the environment.

B. Environments

In this paper, two distinct environments are used. The first is the Levine building, Fig. 3a, and the second is the Towne building, Fig. 3c, both at the University of Pennsylvania. These two environments were chosen due to their different RF characteristics. These differences are derived from the construction date and materials used in the two buildings. The Levine building was built in 1996 and consists of mostly metal framing and drywall, while the Towne building was built in early 1900's and consists of mostly brick and concrete. These two environments allow for a better test suite for the hybrid system. An image of the *Scarab*s operating in the Levine environment can be seen in Fig. 3b. Due to large differences in the RF environments and to demonstrate the flexibility of the hybrid system to mismatched channel models, we are using a function that is a polynomial fitting of experimental curves found in the literature [23] for the local controller channel model.

V. Simulations

In this section, we highlight the benefits of our hybrid approach over a distributed system, while retaining the benefits of such a system. In the first set of simulations, a team of 4 mobile robots and 1 access point are given the task of moving one specific robot to a goal location in a complex environment. Two goal locations are used and it is shown that purely distributed operation fails while the hybrid system can successfully reach the goal. In the second simulation a large team is tasked with supporting one robot moving through a complex environment. This simulation demonstrates the ability of the hybrid system to scale with the number of robots in the team. While the physical communication layer is not simulated in these scenarios, the systems are operating as they would during a deployment; rates are estimated, dual variables are exchanged, routes are computed and motion is constrained based on the underlying network obstacles. A set of experiments with the full system, including the physical communication layer, are presented in Section VI.

A. Local Minima

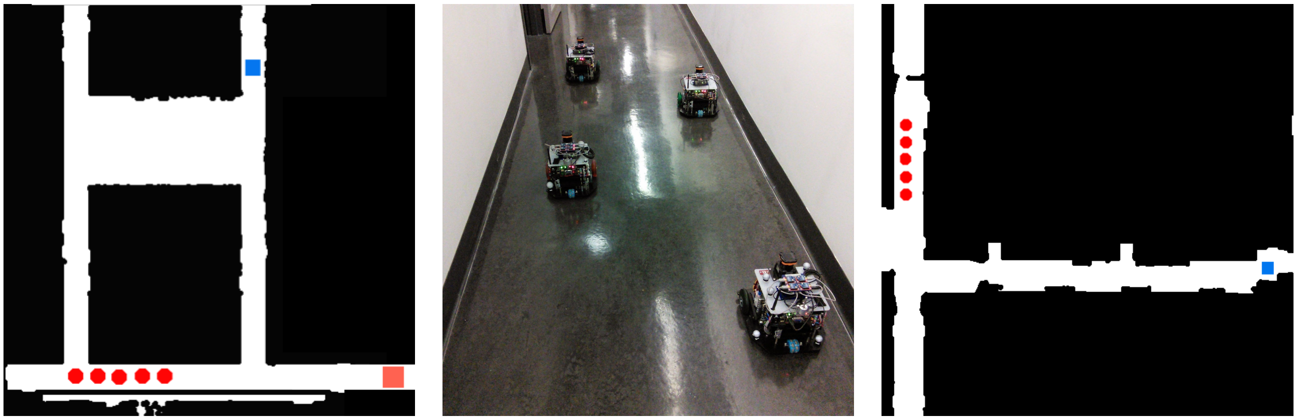
In this set of simulations, we demonstrate the limitations of a purely local controller. Using only the controller described in Section III-B, the team of 4 robots and an access point are given the task of driving *Scarab*40 to a specific goal location. For all three simulations in this section, the goal is 19 meters away from the access point, only the location of the goal is changed. The first location given was straight along the lower hallway in the Levine map, Fig. 3a, as indicated by the red square. The second was around the lower right corner in the same building, which is indicated by the blue square in the Fig. 3a. The resulting trajectories for all three simulations are plotted in Figs. 4a-4c. In Fig. 4a, it can be seen that the robots successfully assemble into a formation that allows the sensing robot to reach the goal, as indicated by the final position of *Scarab*40 being inside the red square. In contrast, Fig. 4b shows that the local controller alone is not capable of driving the team into a valid formation when the goal is around the corner. This is due to the local minima that is created by the attractive force of the goal being cancelled out by the repulsive force from the wall and the attractive force from network preservation. The final simulation in this section shows the performance of the hybrid system when given the same task of turning the corner. As seen in Fig. 4c the team is able to successfully turn the corner and assemble into a formation that allows the sensing robot to reach the goal, as indicated by *Scarab*40 reaching the blue square. This is achieved because each robot is given a series of 3 goals locations that change the location of the local minima and thus allow the team to reach a valid final formation.

B. Large Scale Deployment

Another scenario that we explore in simulation focused on the ability of the hybrid system to operate in a complex environment when the team size is large, $N = 25$. The environment used in this scenario is one floor of the Levine and Towne buildings, shown in Fig. 5a, which has over $850\ \mathrm{m}^2$ of floor space. The access point, $i = 25$, and the sensing robot, $i = 24$, are indicated by the thick red and green axes, while the remaining 23 support robots, $i = \{1, \ldots, 23\}$ are indicated by red arrows. The team begins in a formation $\mathbf{x}((t_0)$ located in the upper left corner of the Levine building. It is tasked with supporting a single QoS requirement with $a^1_{24,m} = 0.3$, $a^1_{j,m} = 0.0$ for all $j \neq 24$, and $\mathcal{D}_1 = 25$, while robot 24 moves to the goal location, x_g, in the upper right corner of the Towne building. In this environment the shortest path from $x_{24}(t_0)$ to x_g is over 200 m. Upon receipt of x_g the global planner determines trajectories for each robot, which are then passed to the waypoint generator and converted into waypoints for the local controllers.

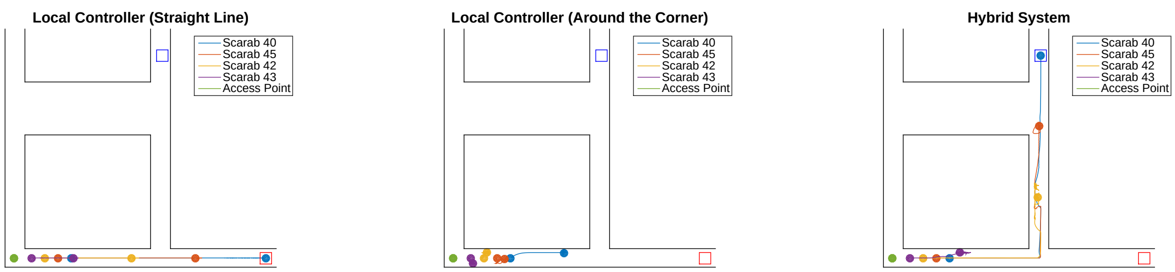
Remark. *Due to the size of the environment and the number of robots, the global planner restricted samples for the RRT to points that were within a meter of robot 24's shortest path. While this restriction limits the set of possible final formations, it allows the system to find feasible trajectories more quickly, as long as there are sufficient number of robots on the team.*

With the waypoints from the global planner, the local controllers begin executing their trajectories. Snapshots of the team's formation in the environment at 0, 300, 600, and 900 seconds are shown in Figs. 5a, - 5d. As shown in the figures, the team is able to successfully deploy into a formation that allows robot 24 to successfully traverse the environment and reach x_g. In this deployment every robot is critical to the data path from robot 24 to the access point due to the complexity of the environment. Note, since each robot is critical to the network, each robot has sufficient back haul to support robot 24's data back to the access point. Therefore, given the problem formulation in (17) any location, $\hat{x}_{24}$, where $R(\hat{x}_{24}, x_j) \geq a^1_{24}$, is a location at which robot 24 can collect data. With this understanding we see that robot 24 is able to retrace its path back to the access point and network integrity will be maintained for the duration of its travel. Since this environment is more complex than the environment in Section V-A, it can be safely assumed that even with knowledge of their final location, the local controllers would would not be able to successfully reach those locations, due to local minima. This highlights that not only is it necessary to find a final formation that supports x_g but intermediate waypoints are needed to ensure proper avoidance of local minima. This simulation was run on a 2.7 Gigahertz Intel i7 laptop with 16 Gigabytes of RAM to demonstrate the lightweight nature of the local controller. After the waypoints were determined, all 25 local controllers ran in parallel in real time.



(a) Partial map of Levine. (b) Image of the *Scarab*s in Levine. (c) Partial map of Towne 3rd floor.

Fig. 3: Environments used in simulations and experiments



(a) Waypoint is straight ahead, no obstacles. Local controller is able to achieve the goal. (b) Waypoint is around a corner. Local controller fails to achieve the goal. (c) Waypoint is around a corner. Hybrid system is able to achieve the goal.

Fig. 4: Simulation results for local and hybrid systems. For all tests the goal location is 19 meters away.

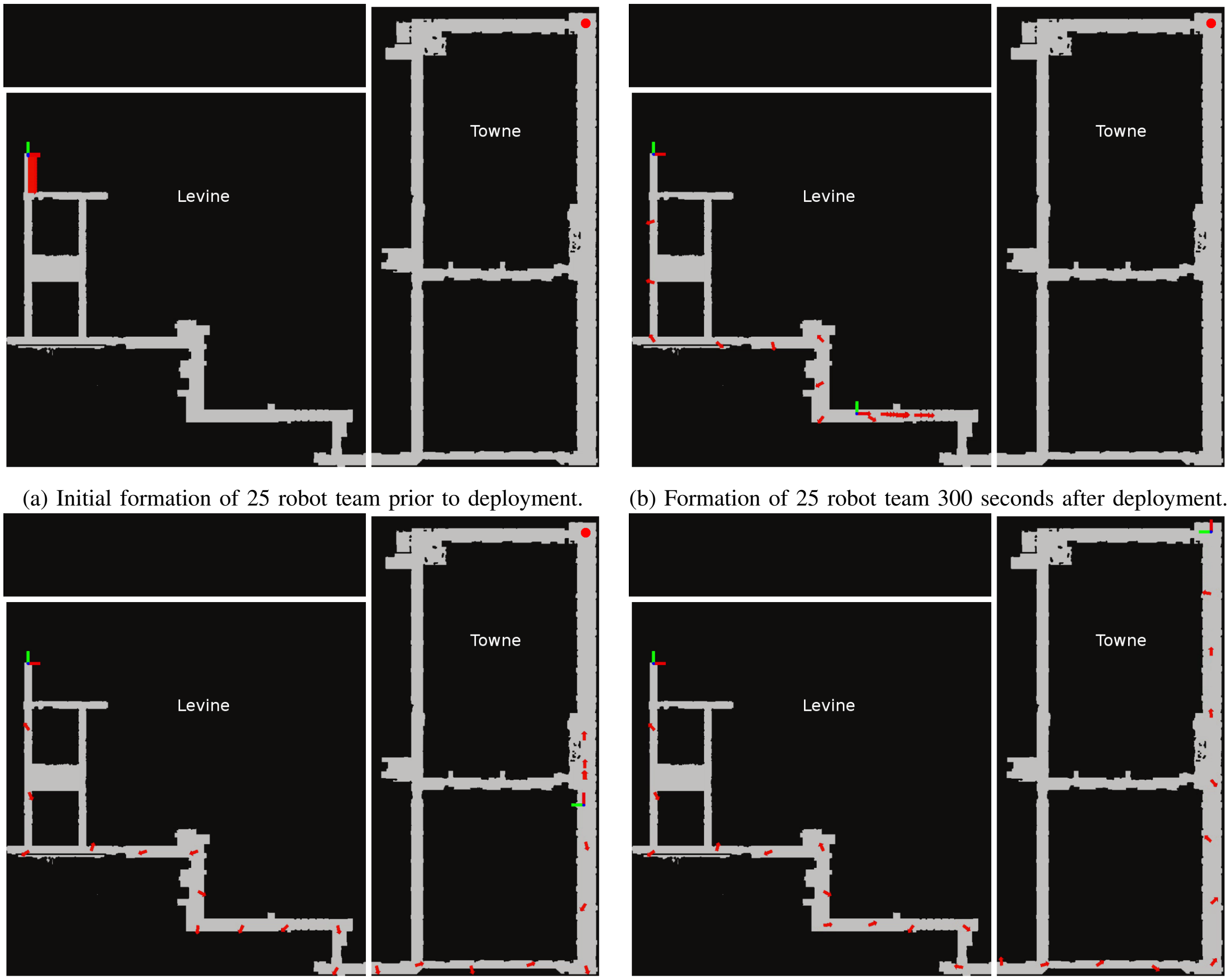
VI. Experimental Evaluation

Our work is motivated by the uncertainty and difficulty in modeling real-world wireless communication. Since our primary objective is to maintain a reliable wireless communication network, it is important that we evaluate the system under the realistic RF conditions described in Section IV. As noted previously, for these experiments we used the *Scarab* platform, [22], with XBee transceivers.

The first set of experiments we ran, Section VI-A, compared the hybrid system to the full system developed by Fink et al in [16]. This system consists of two centralized parts, the path planner and the motion controller. The path planner is the same as the one used in our hybrid system. The motion controller executes the plans determined by the path planner in a synchronous closed loop manner. This means that each robot is given a location to drive to and then wait till given the next location. The next location is not published until after all of the robots have reached their goal. This is implemented in order to preserve the guarantee that at each intermediate formation network integrity is preserved. While this approach does provide more control over the evolution of the formation and underlying wireless network, it is rigid and susceptible to breakage. An example of a scenario that would cause such a breakage is shown in Section VI-B. In that set of experiments one of the support robots incurs a temporary motor failure inbetween two formations and the results causes the centralized system to lose network integrity, while the hybrid system preserves network integrity.

A. System Comparison

In the initial set of experiments, we compare the successful packet transmission of our hybrid system to the centralized system developed by Fink et al. There were three sets of experiments run for this section, each set consisted of ten runs, with only one data flow, $a^1_{1,m} = 0.5$. The first two sets provide the comparison between the hybrid and centralized systems in the Levine building, while the third highlights the performance of the hybrid system in a different environment, the Towne building. For the first two sets, the centralized planner was used to find the trajectories that allowed the team to complete the goal, which was reach the blue square from the initial formation shown in Fig. 3a. With these trajectories the waypoint generator was used to reduce the number of waypoints to three as shown in Fig. 6a. These sets of waypoints were then used by both the centralized motion controller and the local controllers, to remove any bias incurred by different input waypoints. The results of the ten runs are plotted in Fig. 7a, where the solid line represents the average over all the runs and the dotted



(a) Initial formation of 25 robot team prior to deployment.

(b) Formation of 25 robot team 300 seconds after deployment.

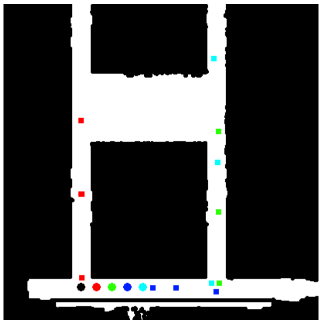
(c) Formation of 25 robot team 600 seconds after deployment.

(d) Formation of 25 robot team 900 seconds after deployment.

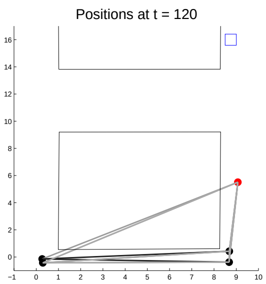
Fig. 5: Evolution of a 25 robot team that is supporting one robot, indicated by red and green axis, from the initial starting formation in the upper left corner to the goal location in the upper right corner, indicated by the red circle.

envelope shows the one σ bounds. There are a few items to note, first there is a portion of the data in which the average success rate for the centralized system falls below 0.5, this is mostly due to a mismatch between the channel model and the actual environment. The second item to notice is how well the hybrid system performs. Even the one σ bound stays above the required data rate. This is mostly due to the robots locally optimizing their trajectory and not moving in straight lines. Another item to note is the spread on the one σ bounds. Since the centralized system is including variance of the channel the spread is much less than the hybrid system which is only using expected value of the channel. Also since the hybrid system allows for deviations to locally optimize, the trajectories taken by the robots is not always the same compared to the tightly controlled trajectories executed by the centralized system. The final item to note is the divergence of the results for the two systems at 12 meters. While the hybrid system continues to exceed the required data rates the centralized system drops off dramatically to marginally meeting the requirements. The reason for this is at 12 meters the sensing robot turns the corner and must rely on the support robots to relay data back to the access point. Since the centralized system is attempting to increase robustness and maximize performance it must balance link diversity with throughput. This conservative approach is useful when planning but it does not leverage the current state of the environment and team formation. In contrast the local controller in the hybrid system is constantly optimizing for performance based on the environment and team's formation. Therefore it can achieve a higher level of performance when compared to the centralized systems due to better utilization of current information. An example of this is seen in Fig. 6 where the location and routing probabilities are plotted for one run of the experiment. For both systems, two snapshots in time are taken, $t = 120$ and at the completion of the task. In the first time instance, the formations are not identical. This is due to the local deviations performed by the hybrid system, but the final formations match.

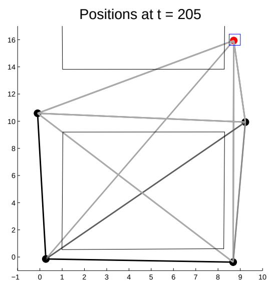
In the third set of experiments for this section the same task, drive around a corner, was completed but in the Towne building shown in Fig. 3c. Again the hybrid system was given



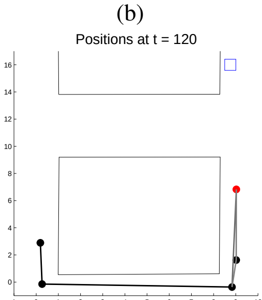
(a) The waypoints used in Section VI-A.



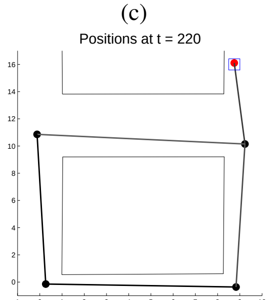
(b)



(c)

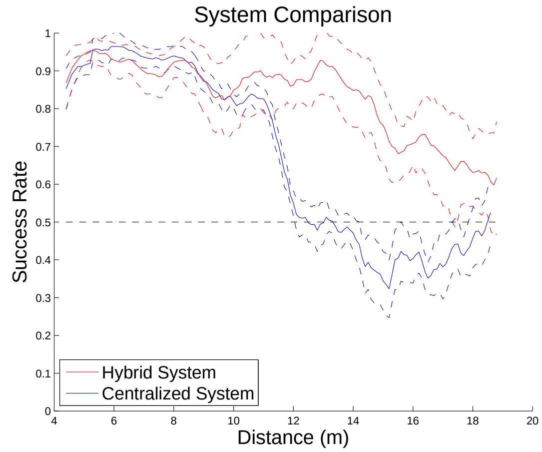


(d)

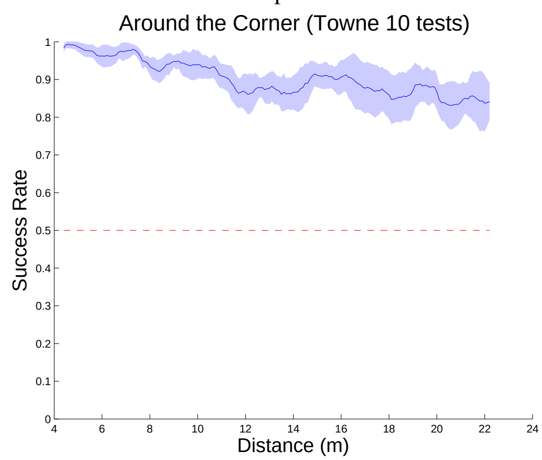


(e)

Fig. 6: These figures show a series of formations and the resulting routing probabilities experienced during the experiments in Section VI-A. Figures (b)-(c) correspond to the centralized system experiments and figures (d)-(e) correspond to the hybrid system experiments. The darkness of the lines connecting the robots indicate the routing probabilities used for that link.



(a) Experimental results for centralized and hybrid systems in the Levine building. The solid line is the average performance and the dashed colored lines are +/- 1 σ bounds. The black dashed line is the minimum input data rate for the lead robot.



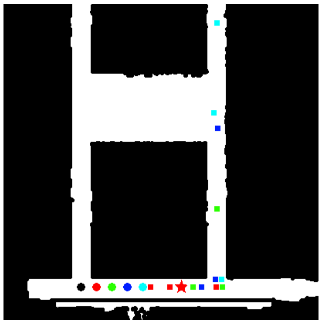
(b) Experimental results for the hybrid system in the Towne building. The solid line is the average performance and the shaded region is the +/- 1 σ bounds. The red dashed line is the minimum input data rate for the lead robot.

Fig. 7: Experimental results for the Levine and Towne buildings.

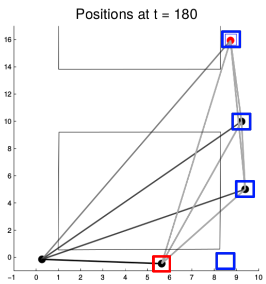
the blue square as a goal location for the sensing robot, and the initial formation is indicated by the red circles. Again ten experiments were run with $a^1_{1,m} = 0.5$, and the results are plotted in Fig. 7b. The system performs remarkably well, with the one σ bounds well above the desired results. This is most likely due to the Towne building having wider hallways compared to Levine and therefore the amount of multi-path interference being reduced when the robots are in the center of the hallway. Also, the same model parameters were used as in the Levine building, thus the superior performance could indicate that the channel model is conservative with respect to the RF environment in Towne when compared to Levine.

B. Dynamic Response

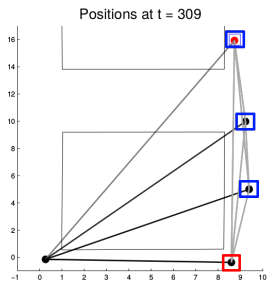
In this section of tests, we highlight a major benefits of using a local controller, as opposed a centralized waypoint system, namely dynamic response to unexpected events. In these experiments, as with those in the previous section, the goal was to drive around the corner in the Levine building to a goal location, but during deployment one of the robots has a temporary restriction to its motion. A temporary restriction in motion could be caused by events such as an actual failure of the physical motor or an obstacle or person blocking the path of the robot. Similar to the previous section, feasible trajectories were found and passed to the waypoint generator. The resulting waypoints are shown in Fig. 8a, as in the previous section each robot has three waypoints. This set of experiments were run just as the previous section was but when *Scarab*43 reaches the red star in Fig. 8a its motor is



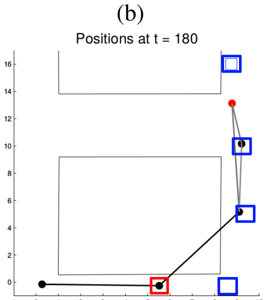
(a) The waypoints used in Section VI-B.



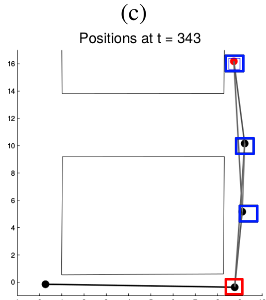
(b)



(c)

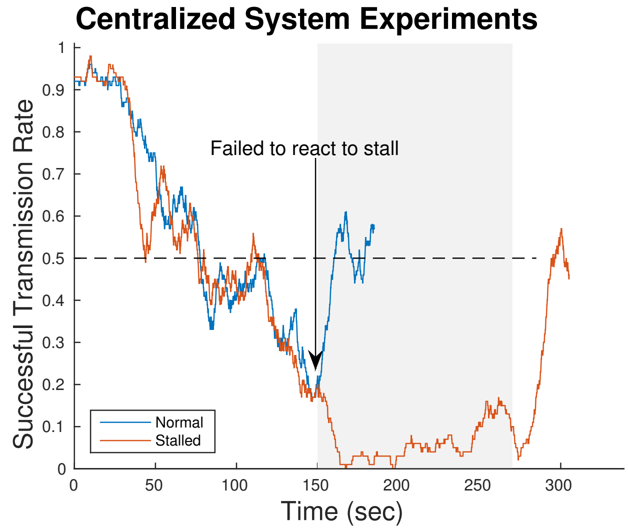


(d)

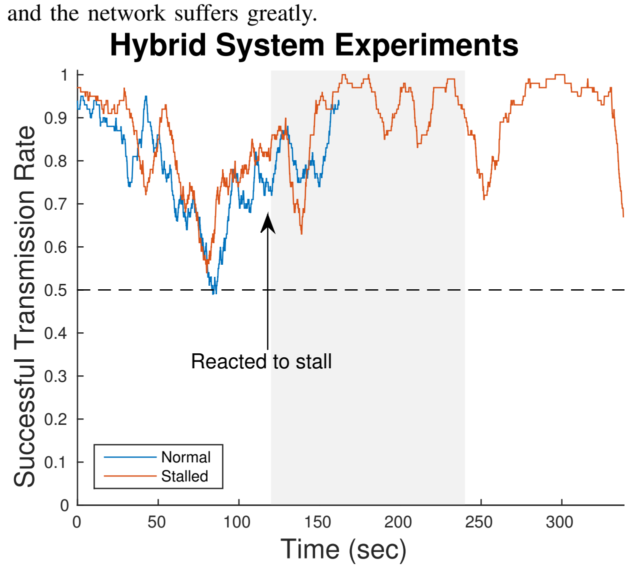


(e)

Fig. 8: These figures show a series of formations and the resulting routing probabilities experienced during the experiments in Section VI-B. Figures (b)-(c) correspond to the centralized system experiments and figures (d)-(e) correspond to the hybrid system experiments. Figures (b) and (d) show a snapshot of the formation when *Scarab*43 has stalled. The darkness of the lines connecting the robots indicate the routing probabilities used for that link.



(a) The centralized system fails to adjust to the motor failure and the network suffers greatly.

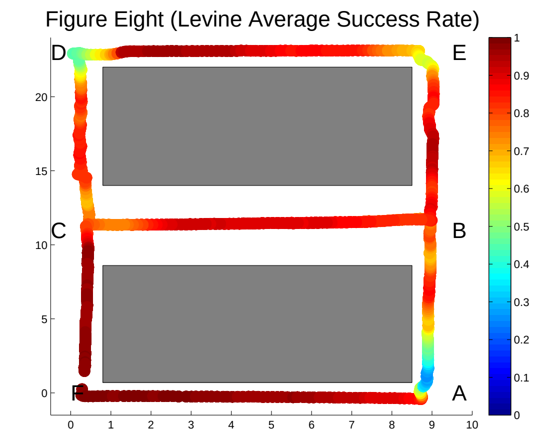


(b) The hybrid system is able to adjust the motion of the robots to overcome the motor failure.

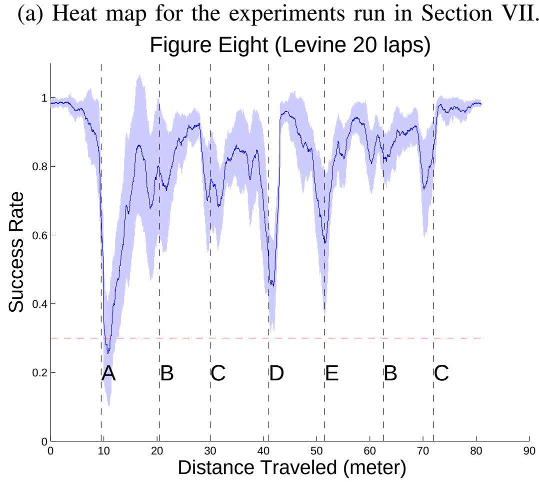
Fig. 9: Experimental results highlighting the hybrid systems ability to dynamically adjust to motor failures. In both figures two separate experiments are plotted. The blue line is from an experiment under normal conditions and the red line is from an experiment where there is a motor failure. The shaded region indicates the time the motor failed for the stalled experiment.

disabled for 120 seconds, to simulate a temporary restriction in motion. In Figs. 8b and 8d we plot the team's formation for the centralized and hybrid systems during the stall period, and in Figs. 8c and 8e we plot the formations at the completion of the experiment. In these plots the sensing robot is a red circle, the support robots are black circles, the final team formation is shown as blue squares, and *Scarab*43 is highlighted by a red square. Notice that since *Scarab*43 stalls after the second set of waypoints in Fig. 8a the centralized system attempts to reach the final formation. This is seen in Fig. 8b by all the robot except *Scarab*43 reaching their goal location. After *Scarab*43 recovers from the stall it moves to it's final location and the team is in the correct final formation. This does not occur when the hybrid system is used due to the team dynamically reacting to the stall and preventing the sensing robot from advancing farther. As shown in Fig. 8d by the red circle not reaching its blue square.

To analyze the network performance of these tests we ran two more experiments where *Scarab*43 does not stall using the same configuration as the prior tests. The results of the two experiments for the centralized and hybrid systems are plotted in Figs. 9a and 9b. In these plots the red and blue lines are the data rate of system with and without the stall, which is indicated by the shaded region. It can be seen that prior to the stall the two lines are in agreement for both systems as is expected since there has not been an unexpected event yet.



(a) Heat map for the experiments run in Section VII.



(b) The average and 1 σ bounds for the the experiments run in Section VII.

Fig. 10

When the stall occurs we see that the two lines in Fig. 9a diverge, while the they do not in Fig. 9b. The divergence in Fig. 9a is due to the formation deviating greatly from the one that was verified by the centralized planner. After the stall is recovered from we see that the network performance returns to the desired value. In contrast in Fig. 9b we see that the network performance never suffers from the robots being out of position. This is because when the stall occurs the other members of the team react accordingly, specifically the sensing robot halting its motion. These experiments show how the hybrid system is more robust to dynamic changes in the environment and other obstacles that may arise during the execution of a task when compared to the more brittle waypoint synchronization of the centralized approach.

VII. Application

The previous sections demonstrated, through simulations and experiments, that the hybrid system is able to control the motion of the team so that the sensing robot is able to reach a specific location. In this section we demonstrate that by building upon this ability, we can extend the system to complete complex tasks with minimal user input. One such task is long duration monitoring or patrolling a series of hallways. For this task the sensing robot is not moving to a specific location, but instead the requirement is to visit multiple sensing locations, all the time maintaining the desired QoS.

We begin by decomposing the task of patrolling a hallway into a series of operations. First, the system determines a path for the patrol robot that visits all the sensing locations and returns to its current location to create a loop. This loop allows for repeated execution of the generated path without compromising the QoS. Next, the global planner uses this path to determine a goal formation for the support robots that maintains the QoS for the majority, if not entirety, of the patrolling robot's motion. This goal formation, including the first sensing location, is then used as the desired formation for the RRT in global planner. With this desired formation the system operates just as it does in the single location scenario. After finding the trajectories and disseminating the waypoints, the local controllers drive the robot to their goal locations. Upon reaching their goals, the robots are able to adjust their location to optimize the communication network in response to the rest of the team. This allows the team to react to locations along the patrolling robot's path that are not supported by the goal formation, but are still feasible for patrolling.

For this experiment we use a team of 3 robots supporting a patrol robot as it moves through a figure eight hallway. The location of this experiment is the Levine building and the desired QoS is set to $a^1_{4,m} = 0.3$. The team of 4 robots and an access point begin in the lower left corner near location F in Fig. 10a with sensing location (A, B, C, D, E, B, C). The global planner uses this order of sensing locations to determine an optimal formation for the support robots. The resulting formation covers the entire path by placing the support robots at locations B, C, and D. With the path covered, every location along the patrol robot's path will have sufficient network connectivity to support the required QoS. Thus, the local controllers are not required to deviate from the formation. In this experiment the robot executes the figure eight path a total of 20 times. The resulting data rates for each lap are overlaid in Fig. 10. In Fig. 10a we plot the average data rate, signified by the color, at each location along the path. In Fig. 10b, we plot the average and one σ bounds as a function of distance traveled. The vertical dotted line indicate the waypoints. As with the previous experiments, even the one σ bound is above the required rate, $a^1_{4,m} = 0.3$, for the majority of the experiment. Note that other than right after location A, the system maintains the required QoS. This drop off is consistent across laps, as evidenced by the σ bounds not spreading out. We attribute this result to the delay in the convergence of the routing algorithm to the new optimal solution. This is due to the dramatic change in the solution from a direct path to the access point to a multi-hop path through two support robots.

VIII. Discussion

In this paper a hybrid architecture, composed of a centralized planner and local controller, that provides motion control and network routing in order to complete a task for a multi-robot team in known environments was proposed. The centralized planner is used to successfully generate trajectories that allow the team to move through the environment while avoiding local minima. While the local controller is used to determine the network routing, as well as execute the trajectories generated by the centralized planner. Deviations from the trajectories are allowed if they are determined to enhance the performance of the network. This system is distributed on the execution side in the sense that each robot is controlling both their network routing and motion, based solely on its own and its neighbor's information.

It is worth noting that even though this hybrid system relies on a centralized path planner, the amount of time spent in the panning phase can be made to be a small fraction of the execution time. This can be achieved by reducing the size of the configuration space, which can be achieved in a few ways. One way is to break the task up into smaller steps. This can be achieved by finding a path from the current location to the goal location for the sensing robot and creating a series of sub-goals along that path. After the first sub-goal is reached then the next sub-goal is provided and so on till the ultimate goal is reached. Another way to achieve a smaller configuration space is to allow the robots that are purely in a support role be unlabeled. This can be achieved by modifying the RRT process, such that after a candidate formation is determined to maintain the network we do not require a specific association between a location in the formation and a support robot. Allowing any support robot to go to any of the locations, so long as every location is covered, effectively reduces the size of the configuration space. Additionally, for the experiments in this paper the rates were measured over a connectionless protocol, which represent the minimum level of achievable performance. Including a confirmation based protocol, such as Multi-Confirmation Transmission Protocol (MCTP) from [24], can greatly increase the successful transmission rate with minimal overhead.

Our ongoing work and future plans focus on extending the hybrid system to the third dimension. Currently the system operates on the assumption of ground robots operating on a single floor of a building. In future experiments we plan to augment the team with flying platforms, one of which will be the sensing robot. This extension will greatly increase the value of the system by freeing the team from the ground plane. This will allow for operation in more complex environments while providing a new vantage point for the sensing robot. Another area of interest in the ability of the team to operate in unknown environments. This requires that a map of the environment be constructed online and then disseminated to the team, while still preserving the network. This is area is under active of research [25], [26] but the inclusion of the network constraint complicates the motion of the mapping robots greatly since trajectories must be followed that prevent loss of network integrity. With the ability of the team to operate in unknown environments in 3-D the hybrid system will provide a robust and reliable platform on which even more complex tasks can be completed.

References

[1] M. Zavlanos and G. Pappas, "Potential fields for maintaining connectivity of mobile networks," *Robotics, IEEE Transactions on*, vol. 23, pp. 812–816, Aug 2007.

[2] Y. Kim and M. Mesbahi, "On maximizing the second smallest eigenvalue of a state-dependent graph laplacian," *Automatic Control, IEEE Transactions on*, vol. 51, pp. 116–120, Jan 2006.

[3] E. Stump, A. Jadbabaie, and V. Kumar, "Connectivity management in mobile robot teams," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1525–1530, May 2008.

[4] M. De Gennaro and A. Jadbabaie, "Decentralized control of connectivity for multi-agent systems," in *Decision and Control, 2006 45th IEEE Conference on*, pp. 3628–3633, Dec 2006.

[5] M. Schuresko and J. Cortés, "Distributed motion constraints for algebraic connectivity of robotic networks," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 1-2, pp. 99–126, 2009.

[6] M. Ji and M. Egerstedt, "Distributed coordination control of multiagent systems while preserving connectedness," *Robotics, IEEE Transactions on*, vol. 23, pp. 693–703, Aug 2007.

[7] M. Zavlanos and G. Pappas, "Distributed connectivity control of mobile networks," *Robotics, IEEE Transactions on*, vol. 24, pp. 1416–1428, Dec 2008.

[8] D. Spanos and R. Murray, "Motion planning with wireless network constraints," in *American Control Conference, 2005. Proceedings of the 2005*, pp. 87–92, June 2005.

[9] M. Zavlanos, A. Ribeiro, and G. Pappas, "Network integrity in mobile robotic networks," *Automatic Control, IEEE Transactions on*, vol. 58, pp. 3–18, Jan 2013.

[10] G. Notarstefano, K. Savla, F. Bullo, and A. Jadbabaie, "Maintaining limited-range connectivity among second-order agents," in *American Control Conference, 2006*, pp. 6 pp.–, June 2006.

[11] H. Lundgren, E. Nordstrom, and C. Tschudin, "The gray zone problem in ieee 802.11b based ad hoc networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 6, pp. 104–105, June 2002.

[12] O. Tekdas, W. Yang, and V. Isler, "Robotic routers: Algorithms and implementation," *The International Journal of Robotics Research*, 2009.

[13] Y. Mostofi, "Communication-aware motion planning in fading environments," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 3169–3174, May 2008.

[14] Y. Mostofi, A. Gonzalez-Ruiz, A. Gaffarkhah, and D. Li, "Characterization and modeling of wireless channels for networked robotic and control systems - a comprehensive overview," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 4849–4854, Oct 2009.

[15] Y. Yan and Y. Mostofi, "Robotic router formation in realistic communication environments," *Robotics, IEEE Transactions on*, vol. 28, pp. 810–827, Aug 2012.

[16] J. Fink, A. Ribeiro, and V. Kumar, "Robust control of mobility and communications in autonomous robot teams," *Access, IEEE*, vol. 1, pp. 290–309, 2013.

[17] J. Fink, *Communication for teams of networked robots*. PhD thesis, University of Pennsylvania, 2011.

[18] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra and its Applications*, vol. 284, no. 1–3, pp. 193 – 228, 1998.

[19] J. Kuffner and S. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, pp. 995–1001 vol.2, 2000.

[20] E. Rimon and D. Koditschek, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412 – 442, 1990.

[21] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *Robotics and Automation, IEEE Transactions on*, vol. 8, pp. 501–518, Oct 1992.

[22] N. Michael, J. Fink, and V. Kumar, "Experimental testbed for large multirobot teams," *Robotics Automation Magazine, IEEE*, vol. 15, no. 1, pp. 53–61, 2008.

[23] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 121–132, Aug. 2004.

[24] J. Stephan, J. Fink, B. Charrow, A. Ribeiro, and V. Kumar, "Robust routing and multi-confirmation transmission protocol for connectivity management of mobile robotic teams," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 3753–3760, Sept 2014.

[25] M. Di Marco, A. Garulli, A. Giannitrapani, and A. Vicino, "Simultaneous localization and map building for a team of cooperating robots: a set membership approach," *Robotics and Automation, IEEE Transactions on*, vol. 19, pp. 238–249, Apr 2003.

[26] M. Pfingsthorn, B. Slamet, A. Visser, and N. Vlassis, "Uva rescue team 2006; robocup rescue-simulation league," in *Proc. CD of the 10th RoboCup International Symposium*, 2006.