

Pervasive Embedded Networks for Supporting Multi-Robot Activities

Keith J. O'Hara and Tucker R. Balch

The BORG Lab
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0250
{kjohara, tucker}@cc.gatech.edu

Abstract

We are interested in the application of a low-cost, pervasively distributed network to support cooperative multi-robot tasks. We consider a heterogeneous system composed of small, embedded, immobile, possibly sensor-less, communication nodes and larger mobile robots equipped with sensors and manipulators. The embedded network serves as a pervasive communication and computation fabric, while the mobile robots provide sensing and actuation. The embedded network supports effective cooperative foraging by coordinating coverage patterns and by providing nearly optimal path planning without the network nodes or the robots having global knowledge or localization capabilities. In addition, the network can provide paths for the robots in dynamic environments where paths are created and destroyed dynamically.

Introduction

We are interested in the application of low-cost, pervasively distributed network nodes to support cooperative multi-robot tasks. In this work we consider a heterogeneous system composed of small, embedded, immobile sensor-less communication nodes and larger mobile robots equipped with sensors and manipulators. The embedded network serves as a pervasive communication and computation fabric, while the mobile robots provide sensing and actuation. We refer to the embedded nodes as forming a *sensor-less* network to distinguish the approach from those where the network nodes also have sensors. In our work the embedded nodes provide only modest computation and communication for the team.

As noted above, we depart from the usual approach where the embedded nodes are equipped with sensors. There are a number of arguments in favor of sensor-less embedded networks. First, the cost and power requirements for simpler embedded nodes is lower, thus enabling us to distribute more of them. Second, it is likely that even for a network with sensor nodes, certain activities for which the nodes do not have sensors will need to be conducted. For instance, we may want to utilize a network that has already been deployed (e.g. wildfire monitoring), to support a new applica-

tion (e.g. search and rescue). Unanticipated applications can be addressed on deployed networks by equipping the mobile robots with appropriate sensors. In this way application-specific hardware is concentrated on the smallest portion of the team – the mobile platforms.

The embedded network creates navigation networks for guiding mobile robots in various tasks such as coverage, recruitment, and path planning. We demonstrate all three of these distributed skills in a multi-robot foraging task. We show that a pervasive network of embedded nodes can enable a team of mobile robots to accomplish complex tasks effectively. We analyze the sensitivity of the approach to different team sizes, environmental complexities, and deployment configurations.

The algorithm essentially works as a distributed variant of the popular wave-front path planning algorithm, or a breadth-first search from the goal, propagating paths from the goal location. It also shares some qualities with randomized path planning algorithms such as probabilistic roadmaps (PRM) [(Kavraki *et al.* 1996)]. It can be seen as a physical manifestation of these types of path planning techniques. The embedded nodes make up the vertices of the path planning graph, and the network connections between them are the edges of the graph. Mobile robots can then use reactive navigation to traverse the graph by visiting the vertices (i.e. the embedded nodes) to the goal. In order to respond to changes in the environment this graph has to be maintained as edges are added and removed. For power-efficiency reasons we prefer algorithms that minimize communication between the embedded nodes. In particular, we want to concentrate the effort along paths robots are taking, thus saving power without any loss in performance.

Related Work

Path planning [(Latombe 1991)] has been one of the most active and fundamental research area in mobile robotics. Techniques such as D* [(Stentz 1994)], D*-lite [(Koenig & Likhachev 2002)], and ERRT[(Bruce & Veloso 2002)] are able to plan paths incrementally in dynamic environments. Konolige presents a path planning algorithm based on navigation functions to generate a gradient field which can be used to optimally guide a robot from any point in space to the goal. [(Konolige 2000)] This technique is similar to the common wave-front path planning algorithm, so it shares

some characteristics with our approach as well. All these algorithms, as do most path planning algorithms, assume the path is being planned by one robot using a map, either given a priori, or built incrementally. Our technique does not rely on maps or a single robot to create the plan, but uses the network to plan paths in parallel in dynamic environments for multiple mobile robots.

Parunak et al developed a technique for coordinating multiple unmanned air vehicles (UAVs) using synthetic pheromones. [(Parunak, Brueckner, & Sauter 2002; Parunak, Purcell, & O'Connell 2002)] Inspired by pheromone communication in insects, they create potential fields for guiding the UAVs around threats to goal locations in a distributed manner. The technique they developed used uniformly placed (tiled as hexagons) "place" agents to store the pheromone and evaporate it over time, and "walker" agents to spread and react to the pheromone. The walker agents consisted of the UAV agents which physically move over the place agents and "ghost" agents which walk over the place agents virtually. We do not assume the embedded nodes are arranged uniformly in any structure. We also rely on a distributed dynamic programming solution to the path planning problem, rather than an approach based on the dynamics of insect pheromones.

Like Parunak, Payton et al present an approach for large scale multi-robot control referred to as "Pheromone Robotics" inspired by biology. [(Payton *et al.* 2001)] They use a system based on virtual pheromones, by which a team of mobile robots use short-range communication to accomplish cooperative sensing and navigation. In Payton's work "virtual pheromones" are communicated over an ad hoc network to neighboring robots. In contrast, in our approach information is not distributed by the mobile robots, but rather by the relatively static, embedded nodes scattered throughout the environment.

Both Batalin et al [(Batalin & Sukhatme 2003b; 2003a)] and Li et al [(Li, DeRosa, & Rus 2003)] have developed similar approaches using heterogeneous teams composed of mobile nodes and an embedded network. The network of embedded nodes, creates a "Navigation field" [(Batalin & Sukhatme 2003b)], which mobile nodes can use to find the their way around. They differ in how they compute this navigation field. Batalin et al use Distributed Value Iteration [(Batalin & Sukhatme 2003b)]. In their approach, the embedded nodes use estimated transition probabilities between nodes to compute the best direction to suggest to a mobile robot for moving between a start and goal node. These transition probabilities are established during deployment and both the robots and sensor nodes have synchronized direction sensors (e.g. digital compass). Our approach does not require the nodes to store transition probabilities, instead we rely on the communication network to establish the navigation paths. Also, in our approach the mobile robots only need a local sense of direction in order to move toward the correct embedded node. Neither the robots or the embedded nodes need any shared sense of direction. Batalin et al (Batalin & Sukhatme 2003a) also use communication nodes as "markers" in aiding mobile robots in the exploration problem. The embedded nodes offer a suggested un-

explored direction for the mobile robots to follow. Unlike our approach, their embedded nodes do not communicate with each other, but only to mobile robots.

Li et al are able to generate an artificial potential field for navigation based on the obstacles and goals sensed by the network. [(Li, DeRosa, & Rus 2003)] This potential field is guaranteed to deliver the mobile node to the goal location via an danger-free (obstacle-free) path. The field is created by the embedded nodes propagating goal-ness or danger to neighboring nodes. In our approach the embedded nodes do not have sensors, this capability is provided by the mobile nodes, and thus can not sense obstacles directly. We assume the obstacles are sensed indirectly by the resulting communication topology. The later three of these approaches, as well as ours, use distributed dynamic programming [(Bertsekas 1982)] to create the navigation field.

The sensor networks community has also worked on various related problems. For instance, the Berkeley Motes project (Culler *et al.* 2001) is concerned with many aspects of power efficient sensor networks. The key idea of making the networking protocols data-driven and application-aware is the premise behind "Directed Diffusion". (Intanagonwatt, Govindan, & Estrin 2000) Rather than relying on an application transparent, point-to-point style communication, the idea of directed diffusion is to define anonymous gradients of interest for relaying information from information sources to their respective sinks.

Problem Statement and Assumptions

Our system is composed of mobile robots with sensors and actuators supported by an embedded immobile network of nodes without environmental sensors. We assume the embedded network nodes have the following capabilities:

- **Limited computation and memory**, on the order of a PIC microprocessor with 2K ROM and 256 bytes of RAM operating at 4 MHz.
- **Short range communication** with adjacent nodes up to 5 meters distant.
- **Communication is blocked by navigation obstacles.**

We assume the robots and embedded nodes communicate using a short-range medium that is occluded by the same objects that occlude navigation (e.g. walls). Line of sight between nodes implies open space for navigation.

We have implemented a hardware platform to these specifications, the GNAT (see Fig. 1(a)). The GNAT is a low-power, omni-directional, infrared device costing about 30 dollars to build. The GNAT uses a Microchip 16F87 PIC for computation, and its very modest human interface consists of a button and two human-visible light LEDs. For gnat-to-gnat communication, it uses four remote-control class infrared receivers and emitters. GNATs have the unique capability of controlling via software the current supplied to the human-visible and infrared LEDs. This permits us to control the communication range of the GNATs dynamically.

The mobile robots in our system are somewhat more capable. We assume they support:

- Communication with embedded nodes;

- Relative bearing estimation to nearby embedded nodes;
- Local attractor and obstacle sensing; (e.g. food objects);
- Attractor object grasping.

These robot capabilities are sufficient for cooperative foraging in the presence of an embedded network.

Significantly, there are a few assumptions we do not make. In particular, **we do not assume localization or mapping capabilities** on the part of the robots or the embedded nodes. No mobile robots or embedded nodes are expected to perform localization or mapping. Furthermore **we do not assume the environment is static**. Obstacles to navigation can appear and disappear. We expect the network to automatically adapt to dynamic conditions.

In this work do not address the deployment of the embedded nodes. We assume they have already been placed in the environment, but their positions are unknown and the uniformity of their placement can vary. In fact, one objective of this work is to assess the impact on performance with respect to different network sizes, environment complexities, and deployment configurations.

Given the system of robots and network nodes described above, we would like to solve a multi-robot foraging problem. Foraging is a well-studied, canonical multi-robot task (Balch & Arkin 1994; Goldberg & Mataric 2002). In this task a robot team is initialized at a “homebase” location, from which they should begin to explore the environment in search of attractor (food) objects. Once a cache of attractor objects is discovered, this information should be disseminated to the other robots, along with a means for them to navigate to the cache. Finally, all of the attractor objects should be collected by the robots and delivered to homebase. We have decomposed the overall problem into the following sub-problems:

- Cooperative coverage: enable a team of mobile robots to completely explore the area covered by embedded nodes. For efficiency, robots should avoid traveling through areas already explored by other robots.
- Recruitment: alert the team to new, critical information. In the context of a foraging task, the discovery of a food cache is an example recruitment situation.
- Path planning: Without requiring localization capabilities, provide an efficient route for each robot, located anywhere in the environment, to a goal location.

Approach

The embedded network creates navigation networks for guiding, or routing, mobile robots in various tasks such as coverage, recruitment, and path planning. We use navigation networks to accomplish three different steps in the task: 1) directing the robots to visit uncovered areas, 2) directing the robots to a discovered goal, and 3) directing them home. Navigation networks for each of these tasks are present in the sensor-less network simultaneously. A mobile robot can then follow whichever navigation network corresponding to it’s current sub-task goal. A navigation network is illustrated

in Fig. 1(b). We are able to use navigation networks to complete the multi-robot foraging task in complex environments without mapping or localization.

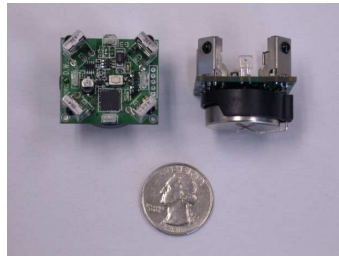
A critical issue in large multi-robot systems concerns how information is routed and presented throughout the network. “Virtual pheromones” (Payton *et al.* 2001) are an example application specific protocol for sharing information between robots. It provides general way to propagate and decay information. We follow Payton’s virtual pheromone technique and assume the communication paths are similar to the navigation paths, and use this to propagate navigation information. To create a navigation network for a particular goal we use a distributed dynamic programming approach; specifically, a variation of the distributed Bellman-Ford algorithm. The Bellman-Ford equation (Bellman 1957) for finding the shortest path from i to j is:

$$D(i, j) = \min_{k \in \text{neighbors}} d(i, k) + D(k, j)$$

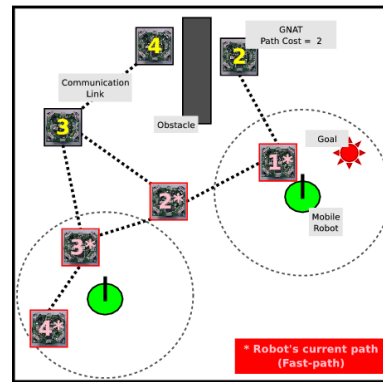
Where $D(i, j)$ is the path cost from i to j , and $d(i, k)$ is the distance between i and k . It can be used to find the shortest path to a destination from all nodes. The distributed version of Bellman-Ford was created for network routing protocols (Ford & Fulkerson 1962). In the distributed network routing version, neighbors share their path costs and the distance between nodes is usually measured in hops. We use distributed Bellman-Ford to effectively create a tree of shortest paths from every node to the goal – this tree is the navigation network. The sensor-less network can be thought of as “routing” the mobile robots to their destination. However, note that the embedded nodes do not know the position of their neighbors, so they are not directing the robot in any direction. The embedded nodes can be thought of as accomplishing a type of passive routing, because, although they provide path cost, the mobile robots make the decision where to go next.

As the mobile robots discover stimuli, they broadcast this information to neighboring embedded nodes and navigation trees are created with roots near the stimuli. The embedded nodes then propagate this information to their neighbors, and so on. Any mobile robot can then approach the sensor-less network, access the relevant navigation network, and descend to the root of the tree, eventually reaching the original goal.

Using a static sensor-less network provides several advantages over fully mobile networks and static sensor networks. The first benefit is cost. By having the bulk of our system be composed of cheap communication and computation nodes, we lower the cost and power requirements of the entire team. In addition, the embedded network can be used generally, for instance, when the desired sensor for the application is not known beforehand. Another related advantage is that the bulk of the application-specific hardware is concentrated in the smallest portion of our team – the mobile platforms – allowing the network to be used in a general manner. Another advantage is that the majority of the system is relatively static and connected. We say relatively because the topology can indeed change, but we assume for the most part, the network will be fully connected and fairly static.



(a) The GNAT



(b) Navigation Network

Figure 1: (a) The GNAT is a low-power, omni-directional, infrared device. (b) An illustration of a navigation network.

When using all mobile nodes, as done by Payton (Payton *et al.* 2001), we must assure the network stays connected.

One of the key requirements for using navigation networks in dynamic environments is the ability of the network to respond to changes in the environment. The nodes must keep track of their neighbors and act appropriately when changes occur. One approach would be for every node to constantly monitor the status of its neighbors. Although this approach would result in speedy reconfiguration, it is also wasteful since it doesn't focus the communication on any part of the system. Instead, we would prefer the nodes along the path of the robot to monitor their neighbors often, but have the other nodes not along the path update less frequently (or not at all). Once the path has been initialized, certain nodes are designated to be on the *fast-path*, meaning they should monitor the status of their neighbors more frequently.

Two routines run in parallel on the embedded nodes - the routine responsible for receiving incoming messages and another for sending messages to neighbors. When a message is to be sent, the node will propagate a path-cost value of 1 plus the minimum path-cost value of all its alive neighbors.

When an embedded node, A , receives a message it updates a data-structure that keeps track of the status of its neighbors. If this neighbor is a mobile robot then the node is on the fast-path. Also, if a node's neighbor, B , is on the fast-path and B is A 's child (i.e. B 's hop-count is 1 more than A 's hop-count) then A is also on the fast-path. The fast-path indicates the path the robot is currently taking. The value of the node's heartbeat message depends on whether or not it is on the fast-path. Again, nodes on the fast-path send heartbeat messages frequently to monitor their neighbors.

A node will broadcast a message if any of the following four conditions are met. One, if a mobile robot is near, two, its heartbeat timeout has expired, three, its path-cost to the goal has changed, or four, a heartbeat request by its child node has been sent. The last condition allows for children to make sure the connection to their parent still alive. One

key idea is that the fast-path changes, and propagates, when paths are destroyed or created. Because of this, we can have nodes not on the fast-path have a very high (e.g. ∞) heartbeat timeout. Two screenshots of a simulation are shown in Figure 2.

The goal navigation network allows a robot from anywhere in the environment to find a path to a goal that was sensed by another mobile robot. By using a short-range communication medium that is occluded by obstacles to navigation the communication paths carve out free-space. It is obvious that if we filled the space with embedded nodes arranged in a grid, and gave the nodes range sensors, we would see a picture very similar to many grid-based planning approaches. In addition, although the communication channels may approximate the free-space, having real robots also reinforce these paths is desirable. However, we currently do not do this. The path planning space is approximated by the communication network, but how many nodes are required for this approximation to hold, and how uniformly do they have to be arranged? We address these questions in the experiments below. The home navigation network is an instance of a goal navigation network, with the homebase being the goal. This creates a tree rooted at the homebase, assuring the robots can return home.

Next, we consider a mechanism for building a coverage navigation network. It is a straightforward application of the goal navigation network, where each unvisited node is a goal. If a node has been visited, it uses the default scheme of propagating one of its neighbor's values. The mobile nodes are then offered paths to reach the closest unvisited nodes. This approach assures all nodes will be visited.

Although the coverage solution generated is not optimal in the sense of shortest circuit to visit all the nodes (i.e. the traveling salesman problem) it does assure all nodes are visited. Depending on the configuration of the embedded nodes, this can assure systematic coverage of the terrain, even though neither the robots or the embedded nodes have any localization capabilities or a map. In addition, both algo-

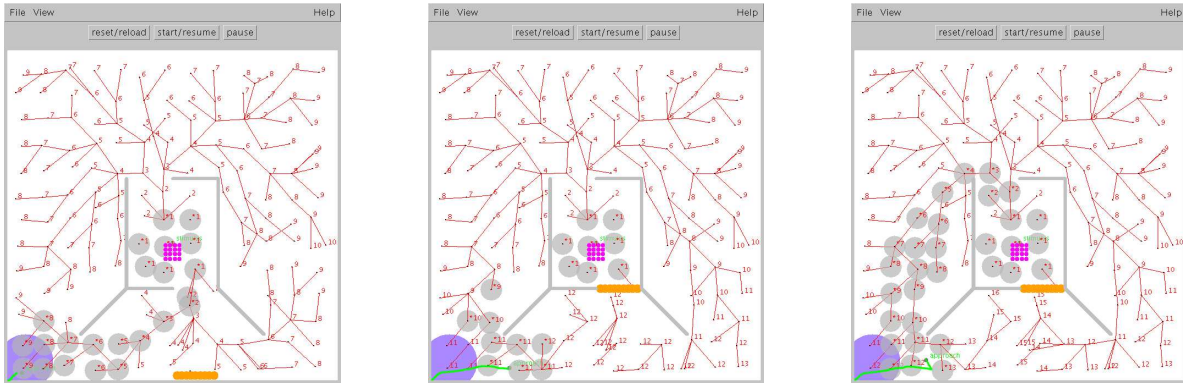


Figure 2: A sequence of screenshots of a simulation using the distributed path planning algorithm. The first shows the initial plan, the second is a snapshot after the door has closed and while the nodes are repairing the plan, and the third is the final plan. The numbers indicate the nodes' distances to the goal, the lines between nodes indicate the parent-child relationship. The gray circles around the nodes indicate the path the robot is taking and thus determine which nodes are monitoring their neighbors more frequently (i.e. on the fast path). The goal location is represented by the pink circles in the center, the mobile robots by the green circles, their trails by the green lines, and obstacles by the gray and yellow lines.

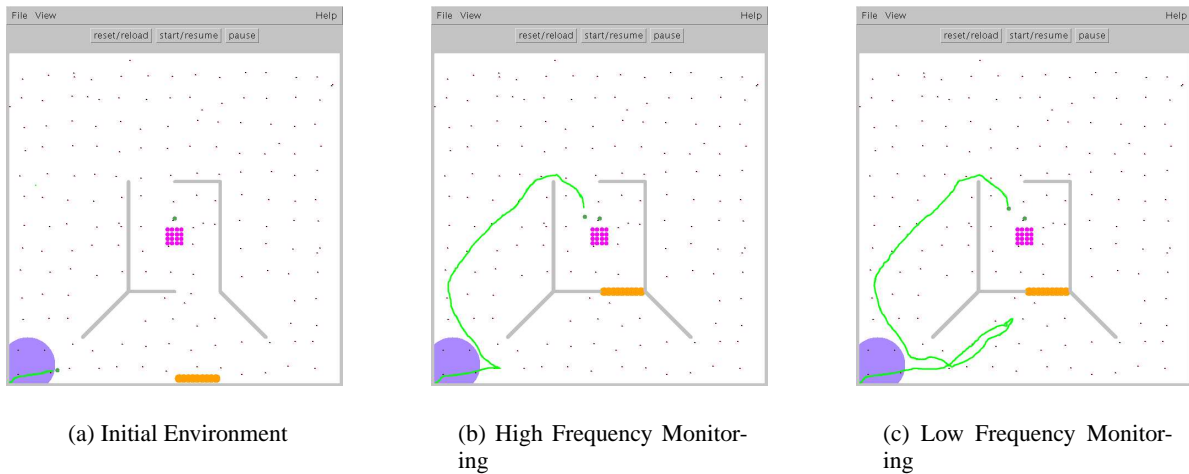


Figure 3: Distributed path planning in a dynamic environment using two different frequencies for neighbor monitoring. In the first screenshot we see the environment in which the initial plan was formed. In the last two screenshots we see the path in which the original plan chosen has been blocked. In the first case, the network is able to sense this quickly, repair the plan, and offer a new path to the robot. In the second case, the network, running at a slower speed, takes longer to sense the broken path. As a result, the robot takes longer to reach the goal. The goal location is represented by the pink circles in the center, the mobile robots by the green circles, their trails by the green lines, and obstacles by the gray and yellow lines.

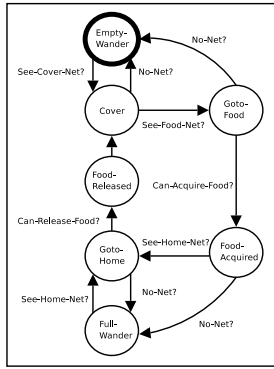


Figure 4: The behavior state diagram of the mobile robots for the foraging task. The robots start in the empty-wanderer state.

rithms can be used in dynamic coverage scenarios by changing their state to unvisited after a certain amount of time has passed. The algorithm works well for both single and multi-robot exploration.

Experiments

We combined the three navigation networks (food source, homebase and coverage) to complete a multi-robot foraging task. We implemented this system in the TeamBots multi-robot simulation environment. The control systems were encoded in the Clay behavioral architecture (Balch 1997). The state diagram for the mobile robots which shows when robots switch from one behavioral mode to another is illustrated in Figure 4. In all the experiments we used 8 mobile robots with grippers, and 16 attractors in a group to represent a food source. All the robots had limited sensing and communication ranges of 4 meters that were occluded by obstacles. We tested the technique in three different $36 \times 36 m^2$ environments of increasing complexity. The three environments are shown in Figs. 5(a), 5(b), and 5(c).

Two key factors impacting how the system performs are the number of embedded nodes and how they are placed. As mentioned previously, when we have a large number of embedded nodes deployed uniformly we effectively have a real grid-world and the navigation networks are accomplishing distributed path-planning. Much work has dealt with trying to optimally deploy a sensor network for these tasks. In this research, however, we assume that the network is approximately uniformly distributed, but with random placement error. Placement error in a real system could be due to error in deployment or changes over time. Since our approach does not depend on the embedded nodes being localized, it is robust to changes in placement.

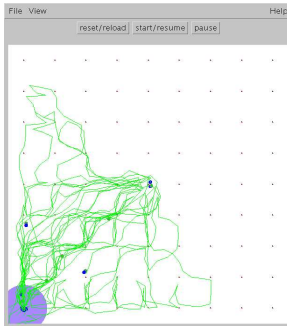
We ran experiments with 81, 121, 169, 225, 289, and 361 embedded nodes. Additionally, we varied the error in placement using the following technique. First, we placed the nodes uniformly across the space, then added error to each node’s position by some random amount, the average distance from original position was varied: from 0, .5, and 2, to 10 meters. In the case of 10m average error, placement is

essentially uniform random. Each experimental configuration was run 10 times. The graphs show mean performance, with errorbars denoting standard deviations.

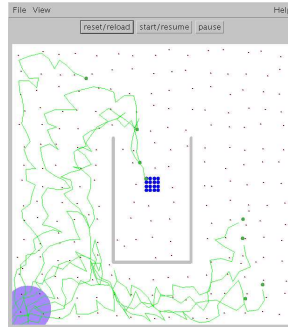
We present the results of the complete foraging task. Space limitations preclude presenting the individual analyses of the coverage and delivery sub-tasks. The results show the average time, in timesteps, to deliver each of the 16 attractors. A delivery time of 7200 timesteps (simulation time-out) were used for undelivered attractors. The results of the first obstacle free environment are shown in Fig. 5(d). The results of the second and third more complex environments are shown in Figs. 5(e) and 5(f). We see that as we increase the error in placement and the complexity of the environment, more nodes are needed to maintain the same level of performance. This is due to the fact that with a small number of nodes and large amount of error, the navigation network is disconnected and isn’t able to guide the robots in the foraging task. Instead, they must rely on a random walk to cover the space and purely local reactive navigation.

To quantify the performance of the technique in a sample dynamic environment we conducted the following experiment. We used 2 mobile robots and a group of 16 attractors to represent the goal location. The first robot was placed near the attractors in the center of the environment at the start of the experiment. This robot’s purpose was to insert the goal information into the network. The second robot was placed in the corner of the environment away from the goal and was responsible for navigating to the goal location. We used 180 embedded nodes. We placed the embedded nodes uniformly across the space, then added error to each node’s position by some random amount, the average distance from original position was .5 meters. The robots and embedded nodes had limited sensing and communication ranges of 4 meters that were occluded by obstacles. The environment was $36 \times 36 m^2$ and included non-convex obstacles. We initially let the network plan a path for the robot, but once the plan is established we shut a door along the path of the robot. The network has to sense this change and repair the plan accordingly. If every node in the network is constantly monitoring its neighbors, then the network will very quickly reconfigure. In contrast, if the network is slow to sense change, the robot will follow the old path until the network fixes the plan. The speed at which the network can respond to changes in the environment is dependent upon the rate at which the network nodes monitor their neighbors by sending heartbeat messages. Examples of this scenario with differing rates are shown in Figure 3.

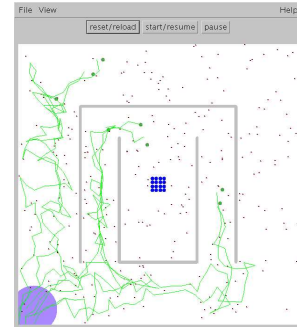
To understand the impact of the heartbeat interval we ran the experiment with increasing heartbeat intervals (0-25 timesteps). Each configuration was run 10 times with different random seeds, the graphs show means and error-bars indicate one standard deviation from the mean. In the baseline approach, all of the nodes use the same heartbeat interval. In the fast-track approach, only nodes on the fast-track use the heartbeat interval, the other nodes have their heartbeat value set to ∞ . Both techniques are able to repair the plan and provide a new path for the robot. In Figure 6(a) the time to reach the goal as a function of the heartbeat interval is presented. As one would expect, both techniques suffer the



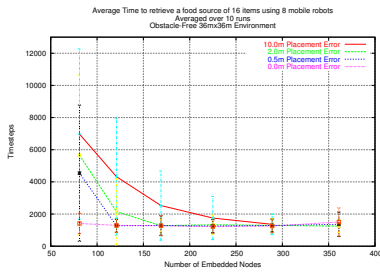
(a) Map 0



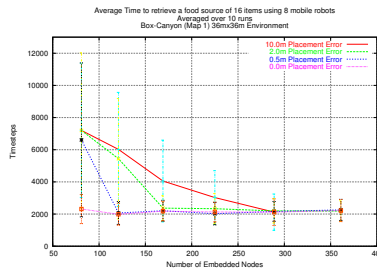
(b) Map 1



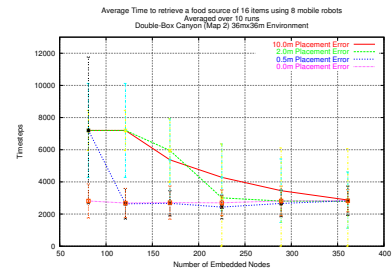
(c) Map 2



(d) Map 0

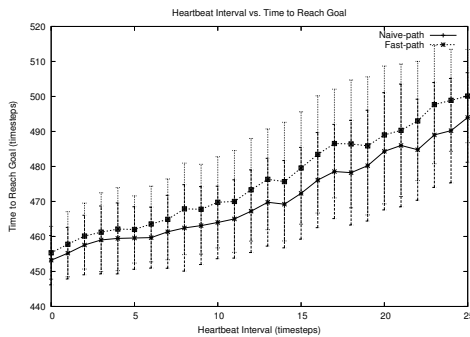


(e) Map 1



(f) Map 2

Figure 5: The average time to deliver each attractor as a function of the number of embedded nodes, the error in their placement, and the environment. The simulation environments. (a) Map 0 is the first obstacle-free environment. A sample configuration is shown with 81 embedded nodes distributed uniformly throughout the environment without any error in placement. (b) Map 1 is a more complicated environment with a box canyon. A sample configuration is shown with 225 embedded nodes distributed uniformly with .5 m of error in placement. (c) Map 2 the most complicated environment with two box canyons. A sample configuration is shown with 289 embedded nodes distributed uniformly with 10m of error in placement.



(a) (a)

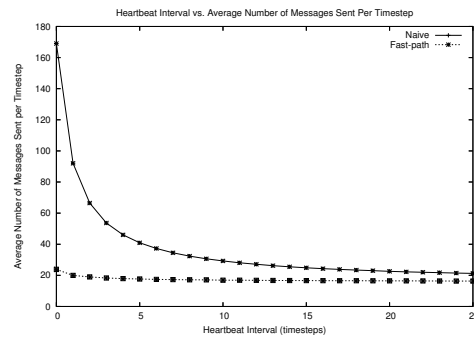


Figure 6: (a) The time to reach the goal as a function of the heartbeat interval. The top curve represents the baseline technique where all the nodes send heartbeats at the same interval. The bottom curve represents the fast-path technique. Both techniques suffer the same performance degradation as the heartbeat interval is increased. (b) The average number of messages sent by the network per timestep as a function of the heartbeat interval. The top curve represents the baseline technique where all the nodes have the same heartbeat interval. The bottom curve represents the fast-path technique. We see the fast-path technique uses far fewer messages at the higher frequencies.

same performance degradation as the heartbeat interval is increased. The fast-path method has slightly more overhead due to the time it takes for the fast-path to propagate.

In Figure 6(b) the average number of messages sent by the network per timestep as a function of the heartbeat interval is shown. We see the fast-path technique uses far fewer messages when its heartbeat frequency is very high as compared to the baseline approach. This is because very few nodes are on the fast-path, so messages are not wasted on areas where there are no mobile robots. There is a lower-bound on the average number of messages per timestep because embedded nodes near mobile robots communicate every timestep in order to guide the mobile robots effectively.

We presented a technique for using a pervasive network of embedded sensor-less nodes to support multi-robot exploration and navigation in dynamic environments. We showed that with enough embedded nodes, the distributed physical path planning works even with very random, non-uniform, deployment of the embedded nodes in complex environments. In contrast, without enough nodes to form a connected network, the approach does not work since the network can not guide the robots.

The approach also has some notable limitations. For one, the system relies on the assumption that communication paths are similar to navigation paths. If this is not true then the embedded nodes will have to be more capable and sense obstacles directly. Also, small changes in the environment that do not impact the communication network can not be sensed, and therefore, can not be planned around. We believe this approach can be extended to include more capable embedded nodes, if the application demands it, but our goal was to show how much can be accomplished with very limited embedded nodes. Another variation would be to use real robot navigation experiences to reinforce the paths in the navigation networks. We are currently evaluating the technique on real robots.

References

- Balch, T., and Arkin, R. C. 1994. Communication in Reactive Multiagent Robotic Systems. *Autonomous Robots* 1(1):27–52.
- Balch, T. 1997. Clay: Integrating motor schemas and reinforcement learning. Technical Report GIT-CC-97-11, Georgia Institute of Technology.
- Batalin, M., and Sukhatme, G. 2003a. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*.
- Batalin, M., and Sukhatme, G. 2003b. Sensor network-based multi-robot task allocation. *Proceedings of International Conference on Intelligent Robots and Systems (IROS 2003)*.
- Bellman, R. E. 1957. *Dynamic Programming*. Princeton, NJ: Princeton University Press.
- Bertsekas, D. P. 1982. Distributed dynamic programming. *IEEE Transactions on Automatic Control* 27(3):610–616.
- Bruce, J., and Veloso, M. 2002. Real-time randomized path planning for robot navigation. In *IEEE Conference on Intelligent Robotics and Systems*.
- Culler, D. E.; Hill, J.; Buonadonna, P.; Szewczyk, R.; and Woo, A. 2001. A network-centric approach to embedded software for tiny devices. *EMSOFT 2001*.
- Ford, L., and Fulkerson, D. 1962. *Flows in Networks*. Princeton, NJ: Princeton University Press.
- Goldberg, D., and Mataric, M. 2002. *Robot Teams*. A K Peters Ltd. chapter Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks.
- Intanagonwiwat, C.; Govindan, R.; and Estrin, D. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, 56–67.
- Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4):566–580.
- Koenig, S., and Likhachev, M. 2002. Improved fast re-planning for robot navigation in unknown terrain. In *Proceedings of the International Conference on Robotics and Automation*.
- Konolige, K. 2000. A gradient method for realtime robot control. In *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*.
- Latombe, J. 1991. *Robot motion planning*. Boston: Kluwer Academic Publishers.
- Li, Q.; DeRosa, M.; and Rus, D. 2003. Distributed algorithms for guiding navigation across a sensor network. *The 2nd International Workshop on Information Processing in Sensor Networks*.
- Parunak, H. V. D.; Brueckner, S.; and Sauter, J. 2002. Synthetic pheromone mechanisms for coordination of unmanned vehicles. In *Proceedings of First International Conference on Autonomous Agents and Multi-Agent Systems*, 449–450.
- Parunak, H. V. D.; Purcell, M.; and O’Connell, R. 2002. Pheromones for autonomous coordination of swarming uavs. In *Proceedings of First AIAA Unmanned Aerospace Vehicles, Systems, Technologies, and Operations Conference*.
- Payton, D.; Daily, M.; Estowski, R.; Howard, M.; and Lee, C. 2001. Pheromone Robotics. *Autonomous Robots* 11:319–324.
- Stentz, A. 1994. Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*.