

# Distributed Path Planning for Robots in Dynamic Environments Using a Pervasive Embedded Network

Keith J. O'Hara  
The BORG Lab  
College of Computing  
Georgia Institute of Technology  
kjohara@cc.gatech.edu

Tucker R. Balch  
The BORG Lab  
College of Computing  
Georgia Institute of Technology  
tucker@cc.gatech.edu

## Abstract

*We investigate the application of a low-cost, pervasively distributed network to plan paths for mobile robots in environments with dynamic obstacles. We consider a heterogeneous system composed of small, embedded, immobile, possibly sensor-less, communication nodes and larger mobile robots equipped with sensors and manipulators. We develop and analyze techniques for distributed path planning when the environment is dynamic and paths are destroyed and created. We demonstrate that the embedded network provides nearly optimal path planning without the network nodes or the robots having global knowledge or localization capabilities. A power-aware technique is presented that is able to respond to changes in the environment quickly, but concentrates network messages along the path the robot is currently using. Through simulation we analyze the communication cost and performance of the techniques.*

## 1. Introduction

We investigate the application of a low-cost, pervasively distributed network to plan paths for mobile robots in environments with dynamic obstacles. 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 network is responsible for planning paths for the mobile robots even though paths are being created and destroyed dynamically.

Path planning is one of the most fundamental and well-studied problem in mobile robotics. Techniques such as  $D^*$ [1] and  $D^*$ -lite[2] plan paths in dynamic environments where paths are created and destroyed. Inspired by these

techniques, we have developed a technique for distributed path planning in environments where obstacles appear and disappear. Traditional path planning algorithms for dynamic environments, such as  $D^*$ [1], typically require a single mobile robot to build a map, update the map as the environment changes, and then finally plan over the map. Instead, we use an embedded network distributed throughout the environment to approximate the path-planning space and use the network to compute the path in a distributed fashion.

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. 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 the path the robot is taking, thus saving power without any loss in performance.

## 2. Related Work

Path planning [3] has been one of the most active and fundamental research area in mobile robotics. Techniques such as  $D^*$  [1] and  $D^*$ -lite [2] 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. [4] 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 incre-

mentally. 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. [5, 6, 7] 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. [8] 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 [9, 10] and Li et al [11] have developed similar approaches, but use heterogeneous teams composed of mobile nodes and an embedded network. The system of distributed agents, in this case a network of embedded nodes, creates a “Navigation field” [9], which mobile nodes can use to find their way around. They differ in how they compute this navigation field. Batalin et al use Distributed Value Iteration. [9] 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. Our approach does not require the nodes to store transition probabilities, instead we rely on the communication network to establish the navigation paths.

Li et al are able to generate an artificial potential field for navigation based on the obstacles and goals sensed by the network. [11] 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 [12] to create the navigation field.

In previous work [13] we demonstrated that the embedded network supports effective cooperative forging by coordinating coverage patterns and by providing nearly optimal path planning without the network nodes or the robots having global knowledge or localization capabilities. Quantitative results illustrated the sensitivity of the approach to different network sizes, environmental complexities, and deployment configurations. In particular, we investigated how performance is impacted by the density and precision of network node placement. 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 complete network, the approach does not work since the network can not guide the robots effectively. We did not investigate the algorithm’s performance in dynamic environments or any kind of power-aware operations.

### 3. 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 (which we have implemented, see Figure 8) have the following capabilities:

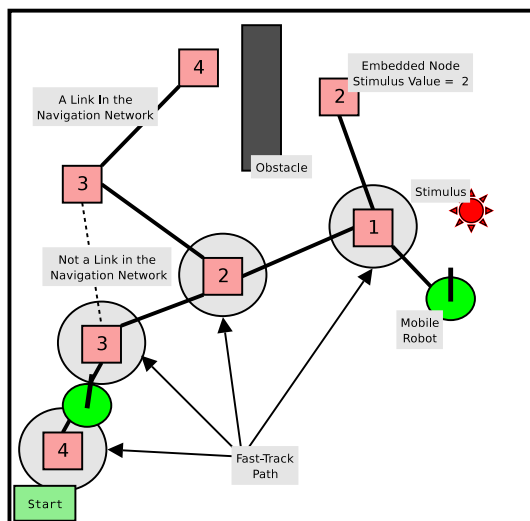
- **Limited computation and memory**, on the order of a PIC microprocessor with 2K ROM and 256 bytes of RAM operating at 4 MHz.
- **Short range communication** with adjacent nodes up to 5 meters distant.
- **Communication is blocked by navigation obstacles.**

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.

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

- Communication with embedded nodes;
- Relative bearing estimation to nearby embedded nodes;
- Local obstacle and attractor sensing;

Significantly, there are a few assumptions we do not make. In particular, **we do not assume localization or**



**Figure 1. An illustration showing how the network guides a mobile robot to a goal location.**

**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.

#### 4. The Distributed Path Planning Algorithm

The embedded network creates a navigation network for guiding, or routing, mobile robots in various tasks such as coverage, recruitment, and path planning. We use the network in this work strictly for path planning. In general, several navigation networks can be present in the network simultaneously. A mobile robot can then follow the navigation network corresponding to its current goal. An illustration of a simple navigation network is illustrated in Figure 1. We are able to use navigation networks to complete distributed path-planning in dynamic environments without mapping or localization.

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 [14] 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 sources. The distributed version of Bellman-Ford was created for network routing protocols [15]. 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 embedded 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 a path cost, the mobile robots make the decision where to go next.

As the mobile robots discover attractors, they broadcast this information to neighboring embedded nodes and navigation trees are created with roots near the attractor. As the information is propagated throughout the network by the embedded nodes, the hop-count, or path-cost, increases. Any mobile robot can then approach the network, access the relevant navigation network, and descend to the root of the tree, eventually reaching the original goal.

By using a short-range communication medium that is occluded by obstacles to navigation the communication paths carve out free-space. As also pointed out by Payton [8] and Li [11], this results in a kind of distributed physical path-planning. Since other robot algorithms, in addition to path-planning, rely on dynamic programming solutions, we can create overlay networks for them similarly using the distributed Bellman-Ford algorithm.

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.  $\infty$ ) heartbeat timeout. Two screenshots of a simulation are shown in Figure 2.

## 5. Experiments

We implemented this system in the TeamBots multi-robot simulation environment. The control systems were encoded in the Clay behavioral architecture [16] and used motor schemas [17] for the local navigational tasks of going toward the embedded nodes and avoiding obstacles. In all the experiments 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.

To quantify the performance of the technique in a sample dynamic environment we conducted the following experiment. 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.

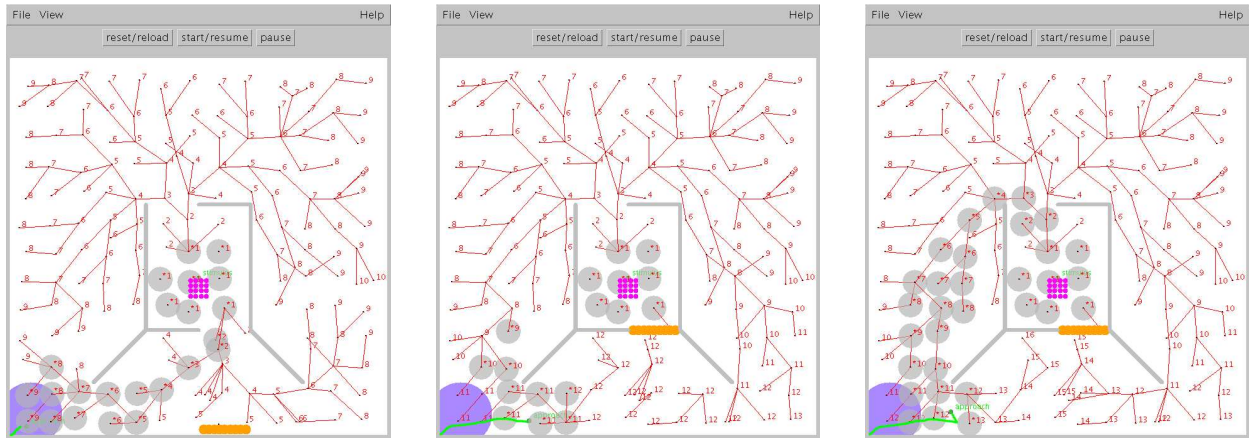
To understand the impact of the heartbeat interval we ran the experiment with increasing heartbeat intervals (0-25 timesteps). Each configuration was run 10 times with different random seeds, the graphs show means and error-bars indicate one standard deviation from the mean. In the baseline approach, 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  $\infty$ . Both techniques are able to repair the plan and provide a new path for the robot. In Figure 4 the time to reach the goal as a function of the heartbeat interval is presented. As one would expect, both techniques suffer the 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 Figures 5 and 6 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.

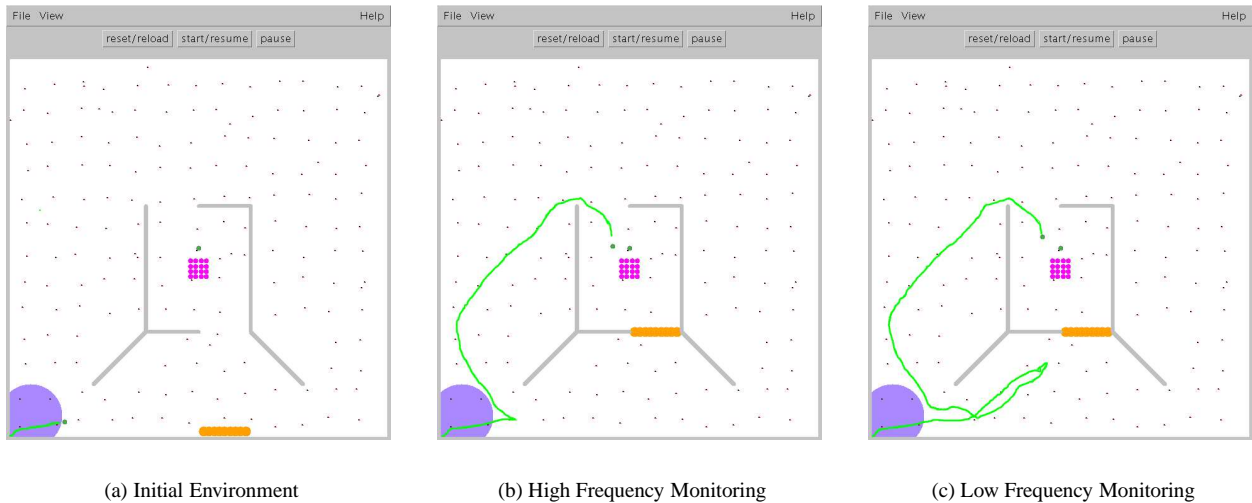
Finally, a summary graph is shown in Figure 7 where the average number of messages sent per timestep is plotted against the time to reach the goal for both techniques. We see that we don't need to sacrifice performance for more efficient communication using the fast-path technique since the communication is efficient for all the rates, including constant monitoring.

## 6. Discussion

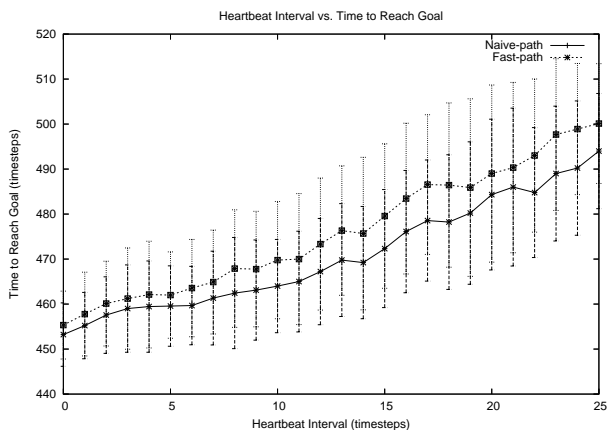
We investigated the application of a low-cost, pervasively distributed network to plan paths for mobile robots in environments with dynamic obstacles. We developed and analyzed two different techniques for distributed path planning when the environment is dynamic and paths are destroyed and created. Both techniques are able to repair the plan when the environment changes and provide paths for



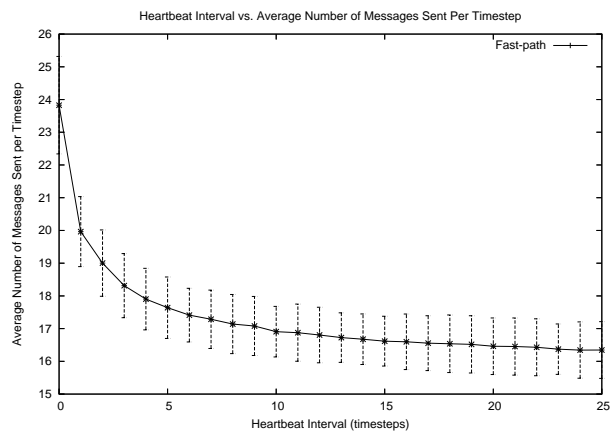
**Figure 2.** A sequence of screenshots of a simulation using the distributed path planning algorithm. The first shows the initial plan, the second is a snapshot after the door has closed and while the nodes are repairing the plan, and the third is the final plan. The numbers indicate the nodes' distances 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.



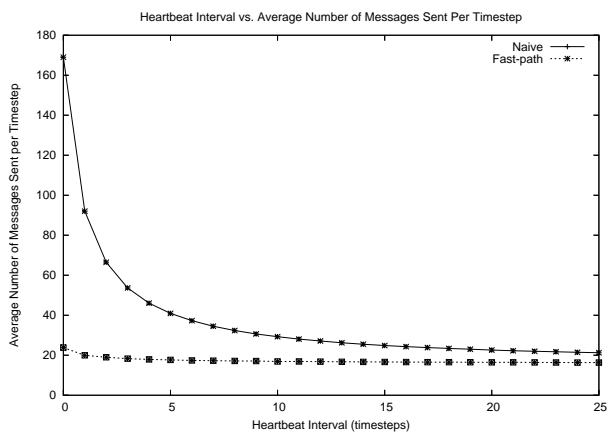
**Figure 3.** Distributed path planning in a dynamic environment using two different frequencies for neighbor monitoring. In the first screenshot we see the environment in which the initial plan was formed. In the last two screenshots we see the path in which the original plan chosen has been blocked. 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.



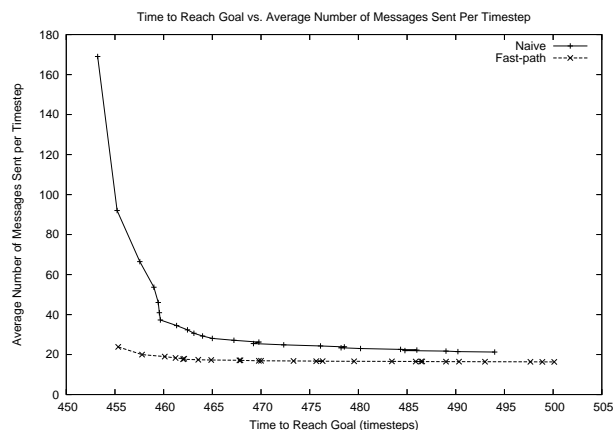
**Figure 4.** The time to reach the goal as a function of the heartbeat interval. The bottom curve represents the baseline technique where all the nodes send heartbeats at the same interval. The top curve represents the fast-path technique. Both techniques suffer the same performance degradation as the heartbeat interval is increased.



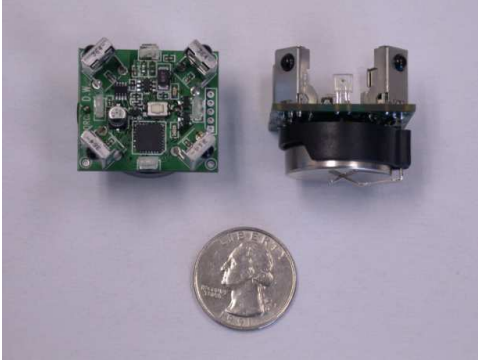
**Figure 6.** The average number of messages sent by the network per timestep as a function of the heartbeat interval. A zoomed in version of Figure 5.



**Figure 5.** 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.



**Figure 7.** A graph summarizing the experiments. The average number of messages sent by the network per timestep is plotted against the time to reach the goal for both techniques. The top curve is the baseline approach and the bottom curve is the fast-path approach.



**Figure 8. The GNAT is a low-power, omnidirectional, infrared device under development.**

a mobile robot to reach a goal. The first technique is able to respond to changes in the environment very quickly but does this at high communication cost. The second approach is able to respond to changes in the environment at the same speed, but with far fewer messages because it concentrates the messages along the path the robot currently resides.

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.

We have implemented a hardware platform to evaluate this approach on real mobile robots, the GNAT, see Figure 8. The GNAT is a low-power, omnidirectional, infrared device costing about 20 dollars to build. In addition, we are investigating ways the network can be used for other types of tasks rather than purely the navigational variety. For instance, it can be used for complex multi-robot coordination, task allocation, or distributed learning. Also, more extensive investigations into power-aware operations are planned.

## References

- [1] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1994.
- [2] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," in *Proceedings of the International Conference on Robotics and Automation*, 2002.
- [3] J. Latombe, *Robot motion planning*. Boston: Kluwer Academic Publishers, 1991.
- [4] K. Konolige, "A gradient method for realtime robot control," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2000.
- [5] J. Sauter, H. Van Dyke Parunak, S. Brueckner, and R. Matthews, "Tuning synthetic pheromones with evolutionary computing," in *Evolutionary Computation and Multi-Agent Systems (ECOMAS)*, R. E. Smith, C. Bonacina, C. Hoile, and P. Marrow, Eds., San Francisco, California, USA, 7 2001, pp. 321–324.
- [6] H. V. D. Parunak, S. Brueckner, and J. Sauter, "Synthetic pheromone mechanisms for coordination of unmanned vehicles," in *Proceedings of First International Conference on Autonomous Agents and Multi-Agent Systems*, 2002, pp. 449–450.
- [7] H. V. D. Parunak, M. Purcell, and R. O'Connell, "Pheromones for autonomous coordination of swarming uavs," in *Proceedings of First AIAA Unmanned Aerospace Vehicles, Systems, Technologies, and Operations Conference*, 2002.
- [8] D. Payton, M. Daily, R. Estowski, M. Howard, and C. Lee, "Pheromone Robotics," *Autonomous Robots*, vol. 11, pp. 319–324, 2001.
- [9] M. Batalin and G.S.Sukhatme, "Sensor network-based multi-robot task allocation," *Proceedings of International Conference on Intelligent Robots and Systems (IROS 2003)*, October 2003.
- [10] M. Batalin and G. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems Journal, Special Issue on Wireless Sensor Networks*, 2003.
- [11] Q. Li, M. DeRosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, 2003.
- [12] D. P. Bertsekas, "Distributed dynamic programming," *IEEE Transactions on Automatic Control*, vol. 27, no. 3, pp. 610–616, 1982.
- [13] K. O'Hara and T. Balch, "Pervasive sensor-less networks for cooperative multi-robot tasks," in *Submitted for publication*, 2004.
- [14] R. E. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.
- [15] L. Ford and D. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton University Press, 1962.
- [16] T. Balch, "Clay: Integrating motor schemas and reinforcement learning," Tech. Rep. GIT-CC-97-11.
- [17] R. Arkin, "Motor schema based mobile robot navigation," *International Journal of Robotics Research*, vol. 8, no. 4, pp. 92–112, 1989.