2005-4

Roadmap Query Protocols for Sensor Network

Authors: Sangeeta Bhattacharya, Nuzhet Atay, Gazihan Alankus,

Corresponding Author: sangbhat@cse.wustl.edu

Abstract: Autonomous mobile entity navigation through dynamic and unknown environments is an essential part of many mission critical applications like search and rescue and fire fighting. The dynamism of the environment necessitates the mobile entity to constantly maintain a high degree of awareness of the changing environment. This criteria makes it difficult to achieve good navigation performance by using just on-board sensors and existing navigation methods and motivates the use of wireless sensor networks (WSNs) to aid navigation. In this paper, we present a novel approach that integrates a roadmap based navigation algorithm with a novel network query protocol called Roadmap Query (RQ). RQ enables collection of frequent, up-to-date information about the surrounding environment, thus allowing the mobile entity to make good navigation decisions. Simulation results under realistic fire scenarios show that RQ outperforms existing approaches with respect to both navigation performance and communication cost in highly dynamic environments. To validate our protocol further, we present a mobile agent based implementation of RQ along with preliminary experimental results, on Mica2 motes.

Notes:
Autonomous mobile entity navigation through dynamic and unknown environments is an essential part of many

Type of Report: Other

RQ and present empirical results in section V. Related work is discussed in section VI, followed by conclusion, in section VII.

II. Problem Formulation

The fundamental problem that we address in this paper is to find a *safe path* for a mobile entity through a sensor field from a start point p_s to a goal point p_g. We define a safe path as a path that is clear of *static* and *dynamic obstacles*. In addition to path safety, a shorter path length as well as lower path traversal time is desirable.

The problem is simple and uninteresting in the absence of static or dynamic obstacles, where a straight line from p_s to p_g would serve as the desired path. The problem gets interesting, in the presence of static obstacles and has been extensively studied in the field of motion planning [5], [6], [7], [8]. However, the problem becomes really difficult with the involvement of dynamic obstacles (e.g., a spreading fire). In such scenarios, continuous awareness of the surroundings needs to be maintained, in order to navigate the region without colliding with the dynamic obstacles. Previous solutions that deal with just static obstacles are incapable of meeting this requirement and hence cannot be applied to such scenarios. WSNs, on the other hand, provide us with the opportunity to meet such real time requirements. The use of WSNs, however, imposes some new requirements on the query protocols for collecting real-time information from WSNs. Firstly, since WSNs have limited resources, it is necessary to minimize the network resource usage as much as possible. Secondly, it is desirable to increase the network responsiveness, or in other words, to decrease the network query delay, such that a high degree of awareness of the surroundings can be maintained continuously. Both of these requirements translate into the need for highly efficient query protocols with low communication cost.

In this paper, we consider the particular scenario where the dynamic obstacle is a fire. The temperature of the region traversed by the mobile entity in such scenarios, is therefore a function of time and is affected by the location and movement of fire. In this case, the problem can be restated as that of finding a safe path for a mobile entity, from start to goal, without the mobile entity getting burnt. This specific problem has important applications such as assisting firemen or residents to evacuate an area on fire. Here, a safe path is defined as one where the maximum temperature along the path is below a certain temperature threshold Δ_T. Cooler paths are thus considered safer than hotter paths. The requirements for a shorter path length, lower path traversal time and low communication cost remain the same.

We present our solutions to this problem in the following section. Even though we consider the specific scenario where the dynamic obstacle is a fire, our solutions can be generalized to other types of dynamic environments where safety is defined by changing sensory values (e.g., lakes polluted by hazardous fluid, chemical spills).

III. Navigation Approach

Our approach to the navigation problem consists of two components; a navigation algorithm and a query strategy. Both components work together to safely guide the mobile entity to the goal. The navigation algorithm computes the path that should be taken by the mobile entity to reach the goal, based on knowledge of the changing environment. This knowledge is provided by the query strategy, which is responsible for collecting the required information from the WSN and delivering it to the mobile entity. These two components are discussed next, with the navigation algorithm being discussed first, followed by a discussion of possible query strategies.

A. Navigation Algorithm

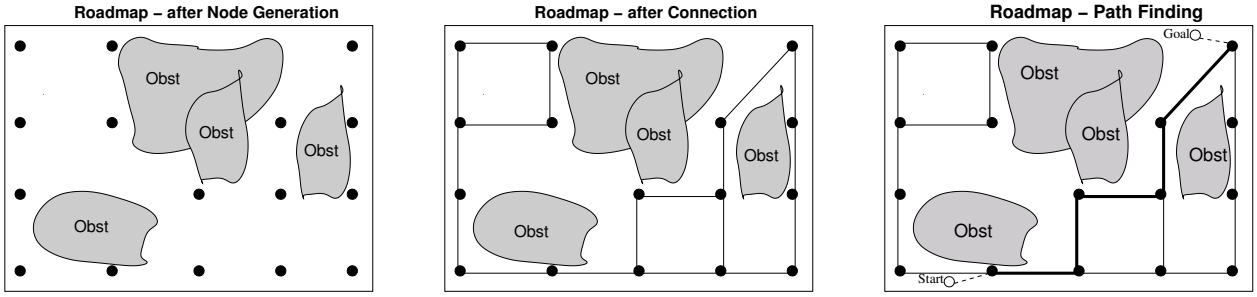
The goal of the navigation algorithm is to find a feasible path for a given mobile entity that takes the mobile entity from a given start to a given goal location. Although several solutions have been suggested [5], nowadays, nearly all practical systems use roadmap methods [3]. Roadmaps are analogous to the highways of the real world. They encode representative feasible paths in the environment. The aim is to search for obstacles in the limited region of the environment. Instead of searching every point in space, only the points that are on the possible paths of the mobile entity are searched.

A typical roadmap algorithm has a roadmap construction stage followed by a path finding stage. In the roadmap construction stage, the roadmap nodes are selected using a sampling technique (see Figure 1(a), Node Generation). These nodes are controlled for feasibility, i.e., it is ascertained that the mobile entity will not be in collision if it is placed at these locations. Feasible nodes are then connected to form a graph (see Figure 1(b), Connection). The edges of the graph, represent paths from one node to another node. The resulting edges are again controlled for feasibility. The graph that is obtained, after the edges that result in collision have been discarded, is the roadmap. Once the roadmap is obtained, a feasible path from start to goal is found by finding a sub-path from the starting location to the roadmap, a sub-path from the roadmap to the goal and a sub-path in-between, within the roadmap. An algorithm for Roadmap Methods can be summarized as below:

RMs: Roadmap Methods
I. Preprocessing: Roadmap Construction
  1. Node Generation (find collision-free locations)
  2. Connection (connect nodes to form roadmap)
     (repeat as desired)
II. Path Finding
  1. Connect start/goal to roadmap
  2. Find path in roadmap between connection nodes

Our navigation algorithm extends the traditional roadmap algorithm by incorporating information about the danger level on the path, which is collected through a WSN.

(i) Node Generation: Our node generation technique is inspired by the Probabilistic Roadmap Method (PRM) [6]. While PRM uses probabilistic sampling to obtain roadmap nodes, we deterministically place the roadmap nodes on a uniform grid (Figure 1(a)). We choose a grid for simplicity and for sampling the area uniformly. Another benefit of using a grid is that roadmap information can be easily included in a query message without increasing the message size too much (explained in section III-B.3). While grid sampling may fail in high-dimensional problems, it is sufficient for navigation on two dimensional surfaces. Note, that there is no direct correlation between the wireless sensor locations



(a) (b) (c)

Fig. 1. A roadmap. (a) After node generation, (b) after the connection phase, and (c) using it to find a path.

and the roadmap nodes. That is, while the roadmap nodes are uniformly placed virtual points, the sensors can be non-uniformly distributed in the region.

(ii) Connection: Once the roadmap nodes are generated, the nodes that are neighbors of each other in the grid are connected using a simple local planner [9] such as the common straight-line local planner (Figure 1(b)). A straight-line local planner returns success if the mobile entity can reach from one location to another location when it moves in a straight line. As a result of this process, we obtain a graph, (roadmap), such that the mobile entity can traverse the edges of the graph, without colliding with any obstacles. In a typical roadmap method, the edges are weighted based on their length. These weights are then used to find the shortest path on the roadmap. Since we are interested in finding the safest path, we use an edge weight function that balances safety with efficiency. The edge weight function used, is a weighted function of the normalized maximum edge temperature and the normalized edge length. The weight of an edge e is thus

$$W_e = \begin{cases} \alpha(\frac{\delta_e}{\Delta_M}) + (1-\alpha)(\frac{l_e}{L}) & \delta_e < \Delta_T \\ \infty & \delta_e \geq \Delta_T \end{cases} \quad (1)$$

where δ_e is the maximum temperature on e, l_e is the length of e, Δ_M is the maximum possible temperature, L is the maximum edge length among all roadmap edges E and $\alpha \leq 1$ is the weight given to the temperature field.

δ_e in equation 1 is calculated as below.

$$\delta_e = max(\delta_s), s \in S \quad (2)$$

where S is the set of sensors that cover edge e and that have responded to the query. A sensor is said to cover an edge if the edge or part of the edge lies within its sensing circle. On the other hand, an edge is said to be covered if certain points on the edge are covered. The points on the edge that need to be covered is dependent on the query strategy and will be discussed later. If an edge e is not covered by the sensors that respond to the query, then W_e is pessimistically set to ∞ so as to avoid traversing that edge.

(ii) Path Finding: In this step, the designated start and goal locations are connected to the neighboring roadmap nodes(see Figure 1(c), Path Finding). Once again, a local planner is used for these connections. At this point, a path from start to goal can be found, that consists of a sequence of sub-paths, first connecting the mobile entity's current location to the roadmap, then following the roadmap edges, and finally, connecting the roadmap to the goal location. Since, there may be more than one path to the goal, the mobile entity should select the most efficient (safe) path, i.e., a path that has the lowest weight among all paths. This path can be found by applying Dijkstra's shortest path algorithm [10] on the roadmap.

In the particular scenario considered in this paper, the edge weights of the roadmap are updated, as new sensor readings are obtained, thus modifying the path in order to avoid the spreading fire. Note, that by using a roadmap algorithm, we gain an advantage over other motion planning techniques in terms of efficiency in the computation while maximizing our objective (i.e., staying away from the danger).

B. Query Strategies

In this section, we discuss four different query strategies called Global Query (GQ), Local Query (LQ), Roadmap Query (RQ), and Robust Roadmap Query (RRQ). Both GQ and LQ serve as baselines for this work, while RQ and RRQ are two new protocols that are optimized for our roadmap-based navigation strategy.

GQ is an intuitive global query strategy in which the mobile entity queries the entire network once and computes the entire path from start to goal based on the obtained information. Unlike GQ, LQ is a spatiotemporal query strategy suggested in [2], where the mobile entity progressively computes sub-paths to the goal based on data collected from surrounding areas, as it traverses the roadmap. However, LQ queries *all* nodes in a query area and hence, is not optimized for the navigation strategy. We present RQ and RRQ, in order to address this drawback of LQ. Unlike LQ, RQ requires only a few nodes that cover roadmap edges in a query area to respond per query and hence has very low communication cost. RRQ is very similar to RQ, but has the added capability of handling node failures.

All query strategies use data aggregation, in order to reduce communication cost. Data aggregation, in all the query strategies, is achieved, using a query tree formed during the query process. Data is aggregated at each level of the tree, by merging the children's information received by a parent node with the information of the parent node, into a single query reply. Our current implementation simply packages multiple replies into a single packet. More sophisticated aggregation mechanisms such as compression and topographic mapping [11] can also be incorporated into our protocols. The query response of each node is timed, in order to facilitate data aggregation. This timing, is based on the time period T_w that the mobile entity waits, in order to obtain the query results. T_w represents the query delay and is tuned in each protocol, to obtain a high percentage of query results.

The different query strategies are discussed in detail in the following sub-sections.

1) Global query (GQ): In this approach, the mobile entity broadcasts a query message, which is flooded through-out the entire network. On receiving the query message, the nodes respond with their location and temperature. The information from the nodes is then aggregated and delivered to the mobile entity which then computes the edge weights of all roadmap edges E in accordance with equation 1. The mobile entity then computes a safe path from start p_s to goal p_g and follows that path. If the mobile entity fails to obtain a safe path from the temperature information obtained from the wireless sensor network, the mobile entity issues subsequent queries until it obtains a safe path. Though this approach is intuitive, it is quite clear that this approach is not a suitable solution for two reasons; (1) it un-necessarily queries all the sensors and hence suffers from a high communication cost; (2) it does not deal with the dynamism of the environment.

2) Local Query (LQ): In this approach, the mobile entity finds its way to the goal by computing sub-paths to the goal, using temperature information obtained from local queries, as it moves towards the goal. The mobile entity issues a local query at the end of each sub-path. The query area of the local query is centered at the mobile entity and thus, moves with the mobile entity. All and only the nodes in the query area respond to the query with their location and temperature information. As mentioned earlier, the mobile entity waits for a period T_w to receive the reponses to the query. After the wait period, the mobile entity uses the information obtained to compute the weight of the roadmap edges that fall within the query area. The edge weights are computed according to equation 1. Using the edge weights, the mobile entity then computes the best sub-path P_i from its current location, towards the goal. If the mobile entity is unable to obtain a safe sub-path, it re-issues the query, while if it does obtain a safe sub-path, it moves along the chosen sub-path. On reaching the end of the sub-path, the mobile entity once again issues a local query and the process described above is once again repeated. The whole process is repeated till the mobile entity reaches the goal or gets burnt. The path followed by the mobile entity is thus $P = \{P_1, ..P_k\}$ where P_i is a sub-path obtained from query i.

In our implementation of LQ, we assume a circular query area of radius R_q. We also assume that the mobile entity carries an on-board sensor through which it remains aware of the temperature at its current location. While traversing a sub-path, if the mobile entity finds that the temperature at its current location is above the threshold Δ_T, it stops moving and re-issues a local query.

LQ, unlike GQ, was designed to deal with the dynamism of the environment. However, LQ still queries all the nodes in a query area, thus resulting in high communication cost and long query delay due to network contention. As shown in our simulations (section IV), the high communication cost can severely affect the performance of navigation and endanger the mobile entity in a dynamic environment.

3) Roadmap Query (RQ): We developed RQ to deal with the shortcomings of LQ. RQ was designed to minimize the communication workload on the network, by optimizing the query strategy in accordance with the chosen roadmap-based navigation strategy. Since the navigation strategy only requires the maximum temperature along the roadmap edges, the query message, in RQ, is forwarded only along the roadmap edges lying within the query area. This forwarding pattern reduces the number of nodes that forward the query message and is achieved by requiring that the queried nodes have knowledge about the global roadmap and that they maintain 1-hop neighborhood information. Since we use a grid as the roadmap, the first requirement is easily met by including the location of the bottom left corner of the grid and the grid square size in the query message. Each queried node uses this information, to construct the grid. The second requirement requires all nodes in the network to maintain neighborhood information which may introduce some overhead. But this is acceptable, since the same neighborhood information is also required by other common services such as routing and power management [12], [13], [14]. This information is maintained, by having each node broadcast periodic beacons, also referred to as hello messages [15]. The period at which the hello message is broadcasted, is called the *hello period*. On receiving a hello message, the receiving node records the sending node as its neighbor.

RQ reduces the communication cost by not only reducing the number of nodes that forward a query message but also by reducing the number of nodes that respond to a query. RQ requires all nodes that forward the query message, to respond to the query. Nodes that hear the query message, but do not forward the message, respond to the query only if they satisfy certain conditions that are discussed later. Thus, in RQ, nodes that rebroadcast the query message, form a backbone of nodes along the roadmap edges that fall within the query area. Non-backbone nodes that respond to the query form leaves that are attached to the backbone nodes, thus resulting in a tree structure with the mobile entity as the root. The formed tree is used to aggregate the sensor results and deliver them to the mobile entity.

RQ uses two simple rules to determine which nodes should forward the query message or respond to the query. We call the rule that determines if a node should forward the query message, as the *forwarding rule* and the rule that determines if a node should respond to a query, as the *reply rule*. By the forwarding rule, if a node receives a query message that is being propagated along edge $e = \overrightarrow{p_{e1}p_{e2}}$ where p_{e1} and p_{e2} are the endpoints of the edge, and the arrow denotes the direction of query message propagation, then, the node rebroadcasts the message only if it covers edge e and is the closest to p_{e2} among its neighbors that can also hear the same query message. By this method, only a few nodes along the edge rebroadcast the query message. The reply rule states that a node should send a query reply, if it has rebroadcasted the query message in accordance with the forwarding rule, or, if its temperature is above Δ_T and it covers a roadmap edge that falls within the current query area. The forwarding rule and the reply rule are enumerated in figure 2.

Given these two rules, RQ works as follows. On receiving a query message, a node i that lies within the query area, sets the sending node j as its parent, and sets its hop count h_i to $h_j + 1$ where h_j is the hop count of the sending node and is contained in the query message. The hop count is used to send the query results at a time that facilitates data aggregation.

Forwarding Rule: for rebroadcasting Query message
If received Query msg from the robot *then*
  $p_r \leftarrow$ robot location on the roadmap
  $N \leftarrow$ neighbors that can hear the robot
  If closest to p_r among N then rebroadcast msg
else
  $E_q \leftarrow$ edges that I cover in current query area
  $N \leftarrow$ neighbors that can hear the query msg
  If src of query msg is along any edge $e \in E_q$ *then*
    $p_e \leftarrow$ endpoint of e in the direction of query msg propagation
    If closest to p_e among N *then* rebroadcast msg

Reply Rule: for responding to a Query message
If node rebroadcasted query msg *then*
  send Query reply
else
  If node covers some edge in current query area *and* temperature at node is above Δ_T *then*
    send Query reply

Fig. 2. Roadmap Query (RQ) Forwarding Rule, for query message re-broadcast and Reply Rule, for sending query response.

Node i then applies the forwarding rule to determine if it should rebroadcast the message. If the node is required to rebroadcast the message, it rebroadcasts the message and then applies the reply rule to determine if it should respond to the query. If it needs to respond to the query, it calculates the time t_r at which the result needs to be sent and sets a timer to fire at that time. As mentioned earlier, t_r is calculated such that it facilitates data aggregation and is set equal to $t_0 + \frac{h_{max}-h_i}{h_{max}} \times T_w$, where t_0 is the time at which the mobile entity sends the query request and h_{max} is a tunable parameter that indicates a maximum possible hop count. Thus, the timer is set to fire after time $T_r = t_r - t_c$ where t_c is the time at which the node receives the query message. A node waits for time interval T_r to receive query replies from its children. When the timer fires, node i sends its query result, which includes its information as well as information obtained from any children that it may have. The RQ algorithm is shown in figure 3.

The mobile entity actions in RQ remain the same as that in LQ. That is, the mobile entity waits for a time T_w to receive the query results. At the end of the wait period T_w, the mobile entity finds the best sub-path to the goal using the temperature information obtained. It is important to note that by reducing the number of nodes involved in a query, RQ can achieve a significantly lower query delay T_w than LQ. If the mobile entity finds a safe sub-path, it starts moving along the sub-path. If no such sub-path is found, the mobile entity re-issues the query. Here also, the mobile entity is assumed to have an on-board temperature sensor through which the mobile entity remains aware of its surrounding temperature. If its surrounding temperature rises above the threshold, the mobile entity stops moving and issues a query, to find a safer sub-path.

Analysis: RQ reduces communication cost by imposing the forwarding rule shown in figure 2. In RQ, the query message of a particular query is propagated only along the roadmap edges that fall within the query area. In this section, we derive two conditions under which we can guarantee that the query

When Query message is received
1) Accept if in current query area.
2) Set sending node as parent.
3) Set h_i to hop count in msg plus 1.
4) Apply the forwarding rule to see if msg should be re-broadcasted. If yes, re-broadcast msg.
5) Apply the reply rule to see if query result should be sent. If yes, calculate time to send result and set timer *SendTimer* to fire at the right time.

When Query reply is received
1) *If* result not yet sent *then* store *else* discard.

When *SendTimer* fires
1) Send aggregated query result to parent.

Fig. 3. Roadmap Query (RQ) Algorithm.

message is received by all nodes that cover any roadmap edge lying within the query area.

The fist condition is that of a sensing covered network. A sensing covered network is one, where every point in the region is covered by at least one sensor. Without this network property, it is impossible to guarantee that a roadmap edge is covered by any sensor. A sensing covered network is highly desirable, as it enables the mobile entity to obtain more information about the surroundings and hence make better navigation decisions.

The second condition is the double range property, by which, the communication range R_C of a node is at least twice the sensing range R_S of the node. That is, $R_C \geq 2R_S$. The double range property guarantees network connectivity in a sensing covered network [16] and hence is a desirable property for such networks.

Query message propagation, in RQ, starts at a node s, that receives the query message from the mobile entity, and is closest to the mobile entity location. From node s, the query message is forwarded along edges covered by s and then along edges that are connected to them, and so on. Message propagation from one edge to another occurs at nodes that cover the intersection point of two or more edges. Note, that only roadmap edges that *completely* lie within the query area, are considered per query. Since these edges lie completely within the query area, they form a connected subgraph. Given the above, we can prove the following properties of RQ.

Lemma 1: Given a sensing covered network, with $R_C \geq 2R_S$, a query message at a node i that covers any roadmap edge e lying completely within the query area, is guaranteed to be propagated along the edge.

Proof: Let edge $e = \overrightarrow{p_{e1}p_{e2}}$ where p_{e1} and p_{e2} are the endpoints of the edge, and the arrow denotes the direction of query message propagation. If node i covers endpoint p_{e2} then the message has already traversed edge e and cannot be propagated further along edge e. Hence, let us assume that node i does not cover endpoint p_{e2}. Node i must thus, forward the message to a neighbor that covers edge e and that is closer to endpoint p_{e2} than node i is. Since the network is sensing covered, there are a set of nodes $S \neq \phi$ that cover the edge and are closer to endpoint p_{e2} than i is. Also, there is a node $j \in S$ who's sensing circle either intersects or is tangential to the sensing circle of node i. The maximum distance between node i and any such node j is therefore $2R_S$. Thus, if $R_C \geq 2R_S$, node i is bound to have a neighbor j in the direction of query

message propagation and can forward the message to node j. This is applicable to all nodes that cover edge e and hence, the query message at node i is guaranteed to be propagated along the edge. ■

Theorem 1: Given a sensing covered network with $R_C \geq 2R_S$, RQ can deliver a query message from the mobile entity to every node covering a roadmap edge lying completely within the query area.

Proof: In a sensing covered network, a query message from the mobile entity is received by a node s that covers the mobile entity's location, which corresponds to a grid point p_o. Node s covers roadmap edges E_s that have p_o as an endpoint and that lie completely within the query area. From lemma 1 we can guarantee that the query message propagates along all edges $e \in E_s$. On reaching the endpoint of the edges in E_s, the query message gets forwarded to the connected edges by the nodes covering the endpoints connecting the edges. Since the roadmap edges lying completely within the query area are connected and since the query message is guaranteed to propagate along an edge once it is received by a node that covers it (lemma 1), the query message is quaranteed to be delivered to every node covering a roadmap edge lying completely within the query area. ■

We note, that sensing coverage and the double range property are sufficient but not necessary conditions for guaranteeing query message delivery to all nodes covering roadmap edges lying within the query area. RQ can still provide best effort services when these conditions are violated.

4) Robust Roadmap Query (RRQ): RRQ extends the RQ protocol to make it more robust to node failures due to destruction by fire. To ensure optimal navigation performance even in the presence of node failures, RRQ requires each node that responds to a query, to not only send its sensor reading but also inform the mobile entity about failed neighbors that cover roadmap edges in the current query area. The new reply rule is shown in figure 4. The mobile entity uses the node failure information to avoid paths with failed nodes, assuming that the failed nodes have been burnt. This assumption is required, because the navigation strategy uses an optimistic approach, in that it considers an edge safe, if it has heard from enough nodes that cover the edge and have temperatures lower than the threshold. Given a situation where the fire burns nodes on one side of the edge, but does not affect nodes on the other side of the edge, without node failure information, the mobile entity would assume that the edge is safe and may traverse the edge, only to get burnt by the spreading fire. This situation as well as other special cases can be avoided by having the nodes send node failure information.

Since each node maintains a neighborhood table and receives periodic hello messages from its neighbors, a node knows if a neighbor has failed, if it hasn't heard from the neighbor in n hello periods. The choice of n has to be made carefully, as a lower value of n will give rise to more false positives and a higher value of n will result in delayed awareness of danger, thus leading to poor navigation performance. In our algorithm, we choose $n = 2$. To reduce the chances of false positives we make use of the aggregation phase to *verify* the failed nodes list contained in the query reply. Thus, when a node receives a query reply from a child, it checks if the child's node failure list contains any node that the parent knows to be alive. Any such node is removed from the list before it is passed up the tree.

Reply Rule: for responding to a Query message
If node rebroadcasted query msg *then*
  send Query reply*
else
  If node covers some edge in current query area *and*
      temperature at node is above Δ_T *then*
    send Query reply*
else
  If node has failed neighbors that cover roadmap
      edges in the current query area *then*
    send Query reply*

* Here, Query reply contains sensor reading and list of failed neighbors.

Fig. 4. Robust Roadmap Query (RRQ) Reply Rule, for sending query response.

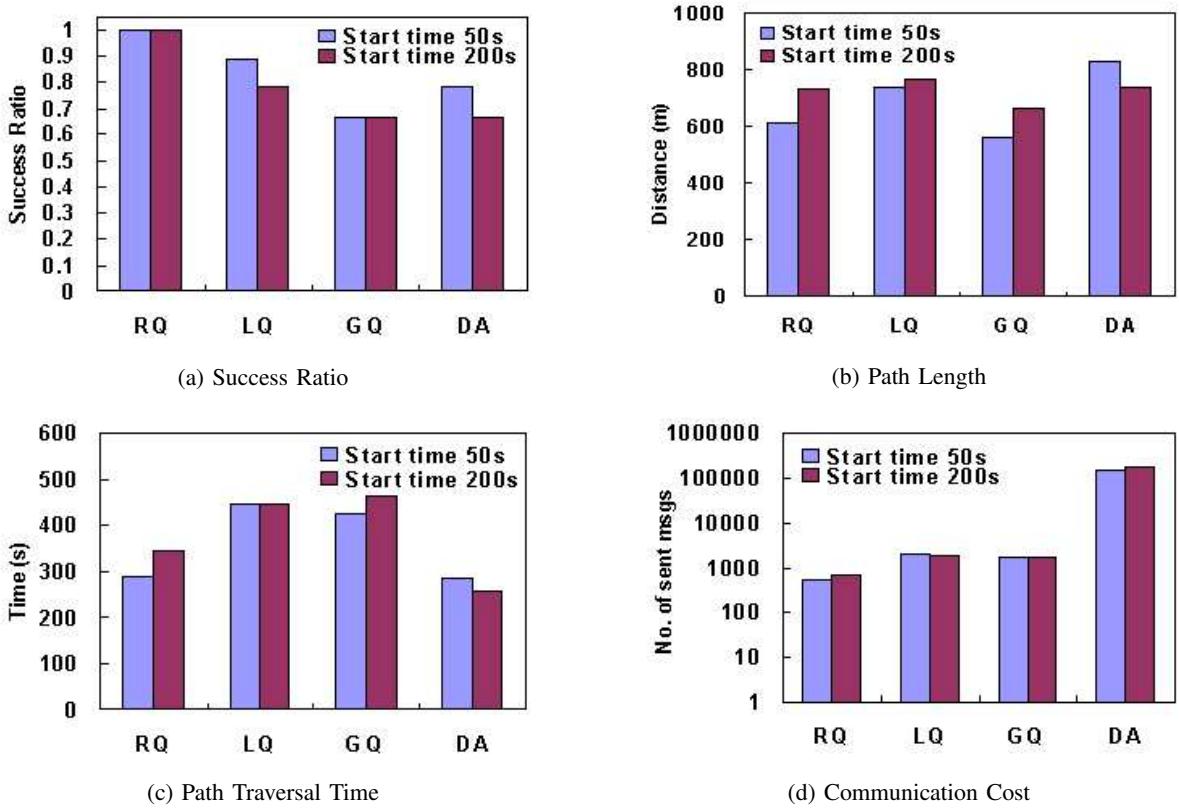
IV. Simulation Results

In this section, we present the results obtained from simulations in NS-2. We simulated Roadmap Query (RQ), Robust Roadmap Query (RRQ), Global Query (GQ), Local Query (LQ), and the algorithm presented in [1], which we will henceforth refer to as the Dartmouth Algorithm (DA). The Dartmouth Algorithm uses a potential field method, whereby, sensors located at or near obstacles (i.e., those with high temperatures) disseminate a negative potential field to the entire network, while the goal disseminates a positive potential field to the entire network. The resultant gradient of the combined potential fields directs the mobile entity to the goal. As every node maintains the potential field counters, the mobile entity just queries local nodes to learn of the resultant gradient. A major draw of this approach is that any change in obstacle state (for example, when the fire spreads to a node and its temperature rises) will trigger the flooding of the entire network to update the potential field counter of every node. Therefore, while DA may work well in a relatively static environment, it may introduce extremely high communication cost in a dynamic environment.

We evaluate and compare the protocols, using 9 different realistic fire dynamics scenarios, obtained using the NIST Fire Dynamics Simulator (FDS) [4]. In all the scenarios, the fire starts in different locations scattered over the region and then spreads over the region with time. Since this behavior presents two different environments, (1) which is very dynamic and occurs in the beginning, when the fire is still spreading and most of the region is still not on fire, and (2) which is less dynamic and occurs when a large area of the region is already under fire, we test the performance of the algorithms in both environments, by starting the mobile entity at two different times of 50s and 200s, after the fire starts spreading.

Since LQ, RQ and RRQ are spatiotemporal queries, querying only nodes within a local query area, we also evaluate the performance of these algorithms under different query radiuses of 90m, 130m and 180m. Query radiuses lower than 90m could not be used, due to the limit imposed by the selected grid size.

Each simulation was run with 900 nodes uniformly distributed in a 450m×450m area. The communication range, sensing range and bandwidth of the nodes was set to 45m, 20m and 40kbps, respectively. The node sensing range was



(a) Success Ratio

(b) Path Length

(c) Path Traversal Time

(d) Communication Cost

Fig. 5. Performance Comparison without node failure.

carefully chosen such that the temperature sensed by a node gives a good indication of the maximum temperature within the sensing circle. The sensing range value was based on the temperature gradient δT at the border of a fire, the temperature threshold Δ_T used by the mobile entity to avoid danger, and the temperature Δ_{burn} at which the mobile entity gets burnt. Mathematically, $R_S = \frac{\Delta_{burn} - \Delta_T}{\delta T}$. In the simulations, δT was obtained from the simulation scenarios, while Δ_T and $Delta_{burn}$ were set to 60^oC and 150^oC, respectfully, assuming a robot as the mobile entity. R_C was set to satisfy the property $R_C \geq 2R_S$.

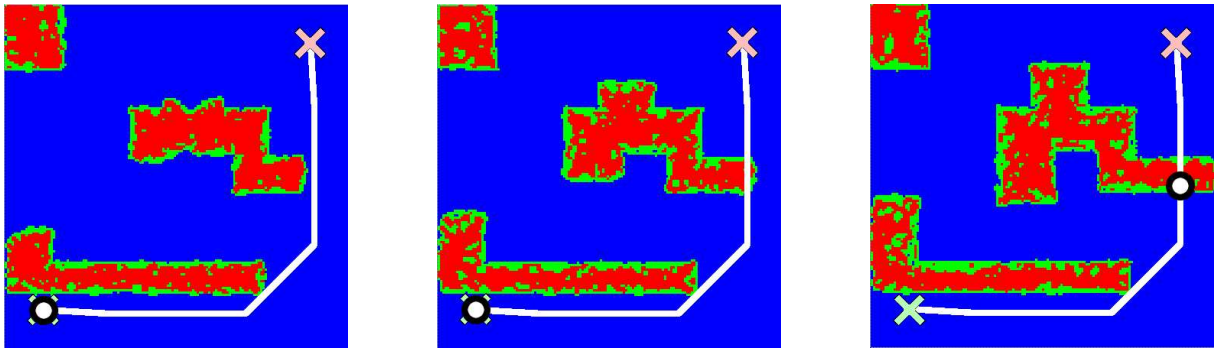
The mobile entity's velocity was set to $3m/s$ and a 5×5 grid was used as the roadmap, with each block in the grid, measuring 90mx90m. The period T_w, during which the mobile entity waits for the query results, was determined experimentally for LQ, GQ, RQ and RRQ. In each of these algorithms, T_w was set to a value that permitted a high percentage of query results to be received by the mobile entity. As expected, the value of T_w was found to increase with the number of nodes responding per query. T_w was set to 250s in GQ. The values of T_w used for LQ, RQ and RRQ are given in table I. T_w represents the query delay achievable using a query protocol. The other tunable parameter h_{max}, which represents the maximum hop count and is used in RQ and RRQ to determine the time at which a node should send the query response, was also determined experimentally and was set to 6, 10 and 12 for query radius 90m, 130m and 180m, respectively.

	90m	130m	180m
LQ	20	40	60
RQ	10	20	40
RRQ	10	20	40

TABLE I
T_w VALUES USED FOR LQ, RQ AND RRQ.

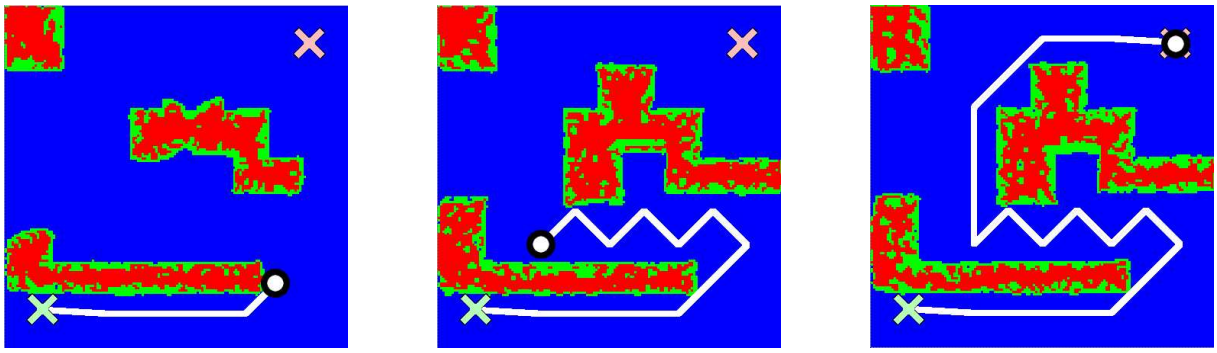
As mentioned earlier (in section III-A), an edge is considered covered if certain points on the edge are covered. The selection of the points differs with the query strategy. Since LQ and GQ query all nodes within a query area, any number of points (≥ 2) on an edge can be considered in these algorithms. In our simulations of LQ and GQ, we consider five equidistant points on an edge. However, since much fewer nodes are queried in RQ and RRQ, we consider coverage of only the endpoints of an edge to indicate edge coverage in these algorithms.

We use the following metrics to evaluate the performance of the different algorithms. (1) *Success Ratio*, defined as the ratio of the number of scenarios in which the mobile entity reaches the goal to the total number of scenarios. This is the most important metric for the application. (2) *Path Length*, defined as the average length of the path taken by the mobile entity, from start to goal over different scenarios. (3) *Path Traversal Time*, defined as the average time taken by the mobile entity to reach the goal, over scenarios where the mobile entity successfully reaches the goal. The path traversal time includes the time that the mobile entity spends on moving and the time it remains stationary while waiting for the query



(a) (b) (c)

Fig. 6. Global Query Snapshots. The circle depicts the mobile entity; the cross at the bottom left corner marks the starting point; and the cross at the top right corner marks the goal. The red region has temperature above 150^oC, the green region has temperature above 60^oC while the blue region has temperature below 60^oC.



(a) (b) (c)

Fig. 7. Roadmap Query Snapshots. The circle depicts the mobile entity; the cross at the bottom left corner marks the starting point; and the cross at the top right corner marks the goal. The red region has temperature above 150^oC, the green region has temperature above 60^oC while the blue region has temperature below 60^oC.

results. Note that the latter depends on the performance of the query protocol. (4) *Communication Cost*, defined as the average number of messages sent per scenario in which the mobile entity reaches the goal.

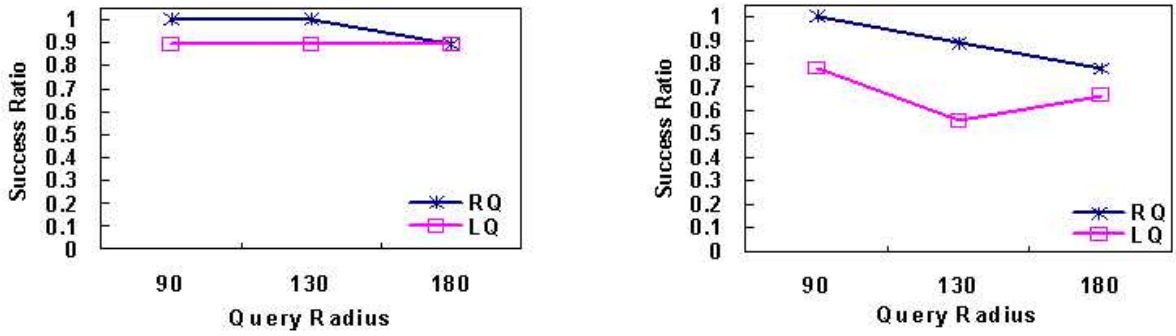
In the following subsections we first present the results obtained from simulations without node failure and then present the results obtained from simulations with node failure. In the simulations with node failure, nodes fail due to destruction by fire at a temperature of 150^oC.

A. Without Node Failure

1) Performance Comparison: In this sub-section, we compare the navigation performance and communication cost of GQ, LQ, RQ and DA. The query radius of RQ and LQ is set to 90m in these simulations. Since RRQ is designed specifically to handle node failures, we do not include it in this section.

a) Success Ratio: Figure 5(a) shows the success ratio of the different protocols at start times 50s and 200s. As seen in the figure, RQ performs better than all the other algorithms, and achieves a success ratio of 1 for both start times. Although LQ is similar to RQ, it does not perform as well, because unlike RQ, it queries a large number of sensors and hence incurs a long query delay. The long query delay slows down the mobile entity's progress towards the goal, and sometimes causes the mobile entity to be caught up amidst the spreading fire with no safe path leading to the goal. Sometimes, the mobile entity also gets burnt while waiting for the query results. In comparison, the success ratio of GQ is quite low. This is because, in GQ, the selected path to the goal is not updated, based on the changing environmental state, causing the mobile entity to get burnt in dynamic environments, where the fire encroaches the chosen path. However, this explanation applies only at start time equal to 50s. At a start time of 200s, the mobile entity usually fails to obtain a safe path, because, by the time the mobile entity obtains the query results, which can take as long as 250s, most of the region is already engulfed by fire, disconnecting the start from the goal. Figure 6 shows snapshots of a particular scenario, where the mobile entity gets burnt on using GQ while figure 7 shows snapshots of the same scenario, where the mobile entity reaches the goal on using RQ. The success ratio of DA is also not as good as that of RQ or LQ. The low success ratio is attributed to high data loss due to contention caused by the high communication cost of DA.

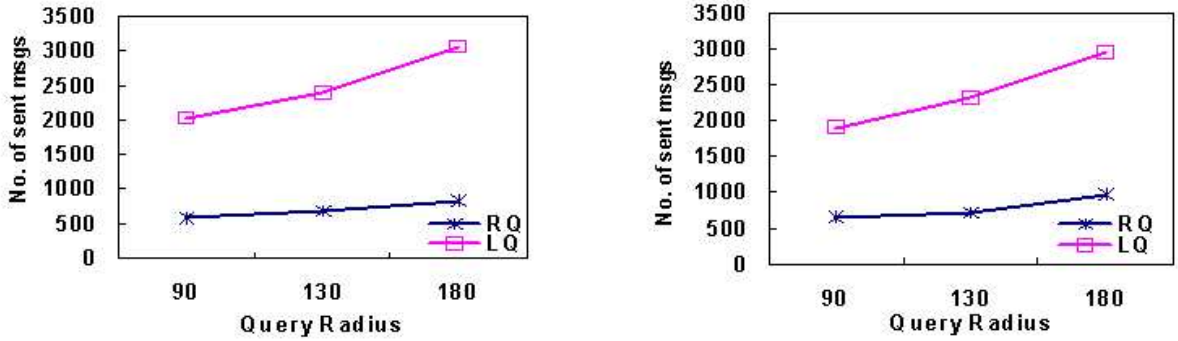
b) Path Length: Figure 5(b) compares the *path length* obtained by the different algorithms. We see from the figure, that GQ obtains the shortest path length. This is expected, since GQ obtains a global view of the surroundings by querying all the nodes in the network and can hence, choose the best path available. DA however fails to obtain a short



(a) Start time 50s

(b) Start time 200s

Fig. 8. Effect of query radius on success ratio without node failure.



(a) Start time 50s

(b) Start time 200s

Fig. 9. Effect of query radius on communication cost without node failure.

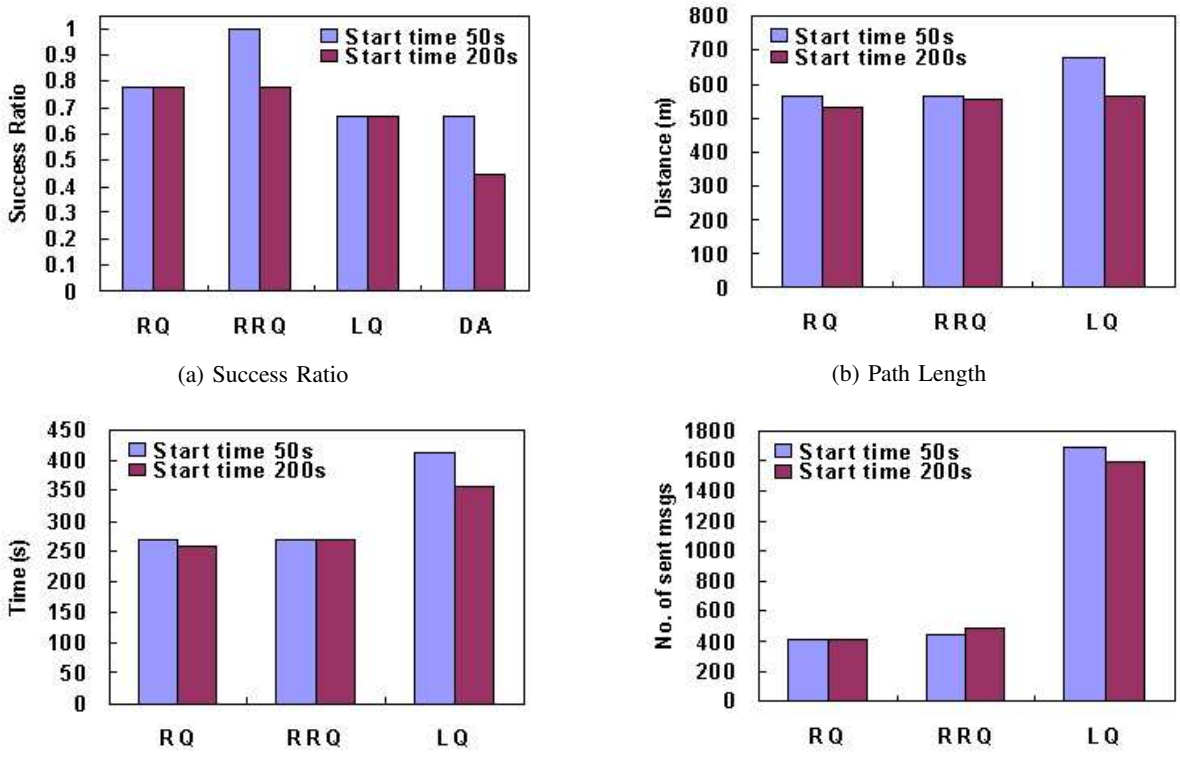
path length, even though it is a global algorithm. This is again because of data loss due to its high communication cost. The path achieved by RQ is shorter than that of LQ, and is slightly higher than that of GQ. Since RQ and LQ are based on local queries, the resultant path length is expected to be worse than that obtained from using a global algorithm.

c) Path Traversal Time: Figure 5(c) shows the path traversal time obtained by the different protocols. From the figure, we see that DA achieves the least path traversal time. This is because, DA has very low query delay compared to the other algorithms, as it requires the mobile entity to query only nearby nodes. The path traversal time of RQ is also quite low and is comparable to that of DA. This is because, the query delay in RQ is also quite low, since only a few sensors in a query area are required to participate per query. LQ and GQ, on the other hand, attain high path traversal times, since they query a large number of nodes and hence have long query delays.

d) Communication Cost: A comparison of the communication cost incurred by RQ, LQ, GQ and DA is presented in figure 5(d). From the figure, we see that DA has an extremely high communication cost. The high communication cost of DA is expected, since it requires all nodes of the network to maintain the potential fields, resulting in periodic flooding of the entire network. In comparison, RQ has the least communication cost, since it queries only a few nodes that lie along the roadmap edges in a query area, per query. This is one of the main advantages of RQ, which enhances it's performance as well as increases network lifetime.

e) Discussion: We observe the following, in the above comparisons. GQ achieves the shortest path length, due to its global view, but does so at the cost of high communication cost and high path taversal time. However, it also has a low success ratio compared to the other algorithms. The cause of the low success ratio (explained earlier) emphasizes the need for continuous awareness in dynamic environments. LQ addresses this requirement and enables the mobile entity to stay aware of the changing environment, but still performs poorly due to high communication cost. DA, on the other hand, requires the involvement of all the nodes in the network, thus incurring a high communication cost, which in turn lowers its navigation performance and also affects network lifetime. Overall, RQ achieves the highest success ratio and lowest communication cost, as a result of its efficient forwarding and query reply rules. These results demonstrate the importance of optimizing the query protocols, in order to navigate successfully in dynamic environments.

2) Effect of Query Radius: In this sub-section, we compare the effect of query radius on LQ and RQ. The query radius represents the mobile entity's region of awareness. A larger query radius implies a larger region of awareness, which potentially implies better navigation performance. However, higher awareness comes at the price of higher communication cost and higher query delay, which have negative impacts



(a) Success Ratio

(b) Path Length

(c) Path Traversal Time

(d) Communication Cost

Fig. 10. Performance Comparison with node failure.

on the navigation performance. Hence, the best navigation performance can be expected at a small, but yet, sufficiently large query radius. This reasoning is supported by our simulation results and is visible in figure 8 which shows the effect of query radius on the success ratio of the algorithms. The success ratio of the algorithms do not improve with increase in query radius, due to the increase in query delay with increasing query radius. A longer query delay implies that the mobile entity takes longer to navigate the region, and since, the region gets more difficult to navigate with time, due to the spreading fire, the slower the mobile entity, the lower the success rate. Longer query delay also increases the danger of the mobile entity getting burnt while waiting for query results. The path length and path traversal time (not shown here) also degrade marginally with increasing query radius. These results demonstrate that the impact of query delay dominates the potential benefit of a large query area in our settings. The optimal query radius thus depends on the network bandwidth and the grid size.

Figure 9 shows the effect of query radius on communication cost. As expected, the communication cost of both the algorithms increases with increase in query radius. LQ has a steeper increase in communication cost than RQ since it queries all the sensors in the query area unlike RQ, which only queries a few sensors along the roadmap edges. From the above results, we observe that a query radius of 90m achieves the best results for the chosen simulation settings.

B. With Node Failure

Since it is important to design robust protocols that can tolerate node failures caused by harsh environments, we now compare the performance of the algorithms in the presence of node failures. As mentioned earlier, nodes are assumed to fail at temperature 150^oC.

1) Performance Comparison: In this sub-section, we compare the navigation performance and communication cost of LQ, DA, RQ and RRQ. GQ is not considered in these simulations, due to its poor performance in the earlier simulations. The query radius of LQ, RQ and RRQ is set to 90m in these simulations.

a) Success Ratio: Figure 10(a) compares the success ratio of the algorithms. From the figure, we see that LQ and DA perform worse than RQ, just like in the previous simulations that did not consider node failure (figure 5(a)). We also note that the success ratio of RQ is lower in figure 10(a) than in figure 5(a). This is because, in these simulations, the mobile entity does not receive temperature information from failed nodes located in regions having temperature above 150^oC. This is a problem when some nodes covering a roadmap edge are burnt but there are other working nodes that cover the edge and have temperatures below the threshold Δ_T. In such cases, the mobile entity assumes the edge to be safe and traverses it, only to be burnt by the fire.

RRQ effectively improves the success ratio of RQ when the mobile entity starts at 50s, at which time many new nodes start failing, due to the spreading fire. On the other hand, RRQ does not outperform RQ when the mobile entity starts at

200s, since by that time, the environment is relatively stable. This demonstrates that a robust query protocol is particularly important in dynamic environments.

RRQ performs better than RQ in some cases, because in RRQ, the mobile entity receives information about failed nodes and is thus warned about possible danger areas. Hence, the circumstances mentioned earlier in the previous paragraph, that cause the mobile entity to get burnt in RQ, do not occur in RRQ.

We note that, RRQ cannot deal with all types of failure situations. This includes situations where the neighbors of a failed node are not aware that the node has failed. Such situations occur if the neighbors know that the failed node exists and the time interval between the time that the node fails and the time that the neighbors respond to a query is less than 2 hello periods. It could also occur if the neighbors do not know that the failed node exists, which happens in situations where the failed node gets burnt right at the start of the simulation and hence, has not been able to inform its neighbors about its presence.

b) Path Length: Figure 10(b) compares the path length achieved by LQ, RQ and RRQ. Due to the poor performance of DA, we omit it in this as well as in future comparisons. As seen in the figure, the path length obtained by RQ and RRQ are similar and much shorter than that obtained by LQ.

c) Path Traversal Time: Figure 10(c) compares the path traversal time achieved by LQ, RQ and RRQ. The figure shows that both RQ and RRQ achieve similar path traversal time, while LQ achieves a much higher path traversal time. The longer path traversal time of LQ is due to its longer query delay.

d) Communication Cost: Figure 10(d) shows the communication cost incurred by RQ, RRQ and LQ. As expected, the communication cost of LQ is much higher than that of RQ and RRQ since it queries all the nodes in the query area. The communication cost of RRQ is only slightly more than that of RQ since it requires nodes to respond if they have failed neighbors even if their temperatures are below the threshold.

e) Discussion: From the above performance comparisons, we see that RQ and RRQ perform better than the other algorithms in both navigation performance and communication cost. We also note that RRQ succeeds in more cases than RQ does, without increasing the communication cost too much. Moreover, the path length and path traversal time achieved by RQ and RRQ are similar, thus making RRQ the best choice among the compared navigation algorithms.

V. Agent based Implementation of Roadmap Query

In addition to simulating both RQ and RRQ, we also implemented RQ on Agilla [17], [18], a mobile agent middleware for the TinyOS platform [19]. While our query protocols can also be implemented directly on an operating system such as TinyOS, an agent-based implementation has several important benefits. In addition to ease of programming, this approach allows the mobile entity to use a wireless sensor network even if it is not pre-programmed with the RQ protocol.

A. Agilla

Agilla is implemented on top of the TinyOS and motes platform. An Agilla application consists of one or more mobile agents that move or clone themselves throughout the network, coordinating with each other, to achieve application-specific behavior. An agent is a programmed, high-level language supported by Agilla. Agilla provides primitives for an agent to move and clone itself from sensor to sensor while carrying its code and state, effectively reprogramming the network. New mobile agents can be injected onto a sensor, thereby allowing new applications to be installed after the network has been deployed. To facilitate inter-agent coordination, Agilla maintains a local tuple space and neighbor list on each sensor. Multiple agents can communicate and coordinate through local or remote access to tuple spaces. Agilla also maintains a neighbor list on each sensor which contains the address of all one-hop neighbors. Prior experiences with Agilla has demonstrated that it can provide efficient and reliable services needed by highly dynamic applications such as fire tracking [18].

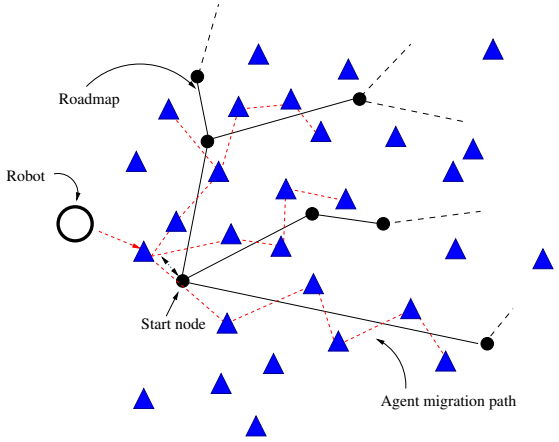


Fig. 11. Agent based temperature collection. An exploration agent migrates along the sensors (triangles) near the edges. Once it traverses the edge, it sends the temperature information back to the mobile entity

B. RQ Protocol using Agents

In our system, the agents are injected into the network by the mobile entity, in order to collect the temperature values from the sensors near the roadmap edges. Unlike the simulations, our agent implementation supports general roadmaps that can either be grids or arbitrary graphs. Figure 11 summarizes our implementation of the agent based temperature collection. In this system, the mobile entity injects an explorer agent into the nearest sensor in the WSN through a wireless interface. The explorer agent carries the roadmap edges that fall within the query area. Hence, the agent always knows the local topology of the roadmap. The first sensor that receives the agent becomes responsible for the temperature readings on the closest roadmap node. After injection, the agent starts exploring the roadmap edges. Exploration of an edge with an agent is done by migrating the agent, along the sensors lying alongside the edge. When an explorer agent reaches the end of an edge (i.e., a roadmap node), it creates a copy of itself for each outgoing edge. These copies, or clones, repeat the same process. For each outgoing edge, this cloning is done on the nearest neighboring sensor. If a clone fails to return back within a predefined period, the waiting agent can assign

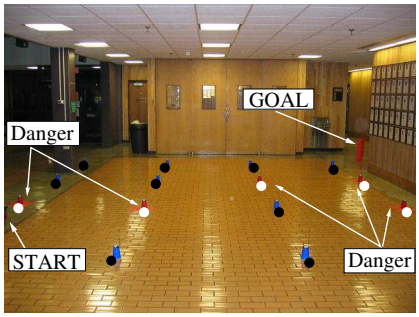


Fig. 12. The experimental environment (8×8 meters).

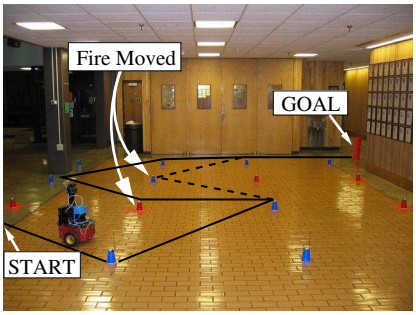


Fig. 13. Spreading fire. When the fire spreads, the robot avoids initial path (dotted line) and follows a safer path (solid line).

the maximum temperature to the corresponding edge. In our experiments, we have employed the former strategy.

The pseudo-code for each agent is presented below:

AGENT EXPLORATION
1. while (end of the edge is not reached)
2. migrate to the next sensor in the direction of the edge.
3. clone the agent to the nearest sensors in the direction of unvisited outgoing edges.
4. report back to the mobile entity with the temperature information.

C. Experiments

We performed a preliminary evaluation of our implementation on a physical testbed comprised of Mica2 motes and a robot. The movies of the experiments can be found at *http://www.cse.wustl.edu/∼bayazit/sn*.

In our experiments, we used a Pioneer-3 DX robot by ActiveMedia [20], as the mobile entity. The Mica2 mote [21] network was arranged in a 4x4 grid as shown in Figure 12. Each node was assigned an (x,y) coordinate based on its grid position. In the figure the node in the lower-left corner has the coordinate (1,1). The distance between each neighboring mote was set to 2 meters. Also, the robot controller carried a mote as a communication interface to the wireless sensor network.

The goal of the robot, in the experiments, was to move from (1,2) to (8,8) while avoiding the fire. Experiments were conducted with two types of fire: (a) static fire, and (b) dynamic fire. In the static fire experiments, the temperatures of the motes were fixed throughout the experiment. Fire was simulated by assigning predefined high temperature values (70^oC) to motes located at (1,3), (3,3), (7,3), (5,5), and (7,5) (motes with white dots in Figure 12), and 30^oC to the remaining motes. Dynamic fire was simulated by assigning the same predefined values as in the static fire, but the values were changed during the experiment. More specifically, the temperature of the mote located at (3,5) was increased while the temperature of the mote located at (3,3) was decreased, thus simulating a fire spreading northwards.

Static fire. The path found in this scenario is shown as the dotted line in Figure 13. As is seen, the robot successfully avoids dangerous places by staying close to motes with normal temperature.

Dynamic fire. This experiment shows the reaction of the robot when the fire changes location. In this case, the robot follows the same path as the static fire until it reaches (5,3). At this point, the fire at (3,3) moves to (3,5). The robot then successfully finds a new path to avoid the fire. This scenario is illustrated in Figure 13. The solid line shows the path followed by the robot, while the dotted line represents the initial path.

Through these experiments, we demonstrate that a robot can use our agent based implementation of RQ, to successfully find a safe path in a dangerous environment. A detailed empirical study of RQ and RRQ is part of our future work.

VI. Related Work

Several query services have been developed for wireless sensor networks, but most of them are unsuitable for navigation. These include Directed Diffusion [22], TinyDB [23], TTDD [24], SEAD [25], DCTC [26] and MobiQuery [27]. TinyDB and Directed Diffusion are designed for fixed query areas and hence are unsuitable for applications involving mobile entities with changing areas of interest. TTDD and SEAD are designed for mobile sinks, but with fixed areas of interest. DCTC and MobiQuery are however, more suited for navigation since they involve moving entities with moving query areas. These query services, if modified, can be used along with a navigation strategy, for navigation. But, since these query services query all nodes in the query area and are not optimized for navigation, they will potentially not have very good navigation performance. MobiQuery's capability of continuous query is however, very useful for navigation and can be integrated with RQ (and RRQ) so that the mobile entity does not have to stop and wait for the query results.

Many successful applications have been presented recently, that use wireless sensor networks for robot or mobile sensor navigation [1], [28], [29]. These algorithms utilize wireless sensor networks to compute a path for the mobile entity. Generally, they compute navigation fields that have global minimums at the mobile entity's destinations. These fields are computed using wavefront expansions. Such algorithms would give maximum clearance from the obstacles or dangers in static environments during the navigation. However, building the navigation field requires a flood of messages (all sensors must participate) increasing the communication overhead and power consumption. In a static environment, this cost could be negligible. However, when dynamically changing dangers are present, these algorithms require continuous flooding of the network whenever the danger changes. Our approach avoids these floods by utilizing only a small subset of nodes. We only gather the information from the nodes near the possible paths. We further reduce the number of active nodes by defining query ranges. Nodes lying outside the query range do not participate in the query and can go to sleep. Our approach is also more flexible. For example, in approaches

using wavefront expansion, if the WSN is initially deployed for one mobile entity but is later required to support multiple mobile entities with different goals, wavefronts of these goals would create problems unless the network is built to support several simultaneous wavefronts. Given the same situation, in our approach, different portions of the WSN can serve different mobile entities without message congestion. Finally, our agent based architecture gives us unprecedented flexibility that is lacking in other approaches. Using different agents, we can customize the same network for different tasks. For example, while some fire-extinguishing robots use agents on the wireless sensor network to direct them to the fire, other agents may direct other robots away from the fire.

There are also other applications of WSNs in robot navigation. In [30], a WSN discovers a path, to help an aerial robot reach it's goal. In [31], [32], [33] WSNs are used to direct robot(s), in order to explore the environment and replace broken sensors. Mobile entities are also used to increase connectivity of a WSN [34], [35], [36], [37].

VII. Conclusion

In summary, the primary contribution of this paper is four-fold. First, we propose a novel sensor network guided approach for mobile entity navigation in dynamic environments. A key novelty of our approach is the integration of roadmap-based motion planning techniques with efficient query protocols for wireless sensor networks. Second, we present two roadmap query protocols that are specially optimized for collecting spatiotemporal data needed for motion planning. Third, our simulations demonstrate that our roadmap query protocols can significantly improve the success ratio of navigation while introducing minimum communication cost under realistic fire scenarios and node failures. Our results highlight the importance of joint optimization of motion planning and sensor network query protocols for navigation in dynamic environments. Finally, we demonstrate the feasibility of our approach through a mobile-agent based implementation of roadmap query on MICA2 motes and a real robot.

References

[1] Q. Li, M. D. Rosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM Press, 2003, pp. 313–325.

[2] G. Alankus, N. Atay, C. Lu, and B. Bayazit, "Spatiotemporal query strategies for navigation in dynamic sensor network environments," in *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2005.

[3] N. Nilsson, "A mobile automaton: An application of artificial intelligence techqniques,," in *1st International Joint Conference on Artificial Intelligence*, 1969.

[4] K. M. et. al., *Fire dynamics simulator (version 4) technical reference guide*. National Institute of Standards and Technology, 2004.

[5] J. C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[6] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[7] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Robotics: The Algorithmic Perspective*. Natick, MA: A.K. Peters, 1998, pp. 155–168, proceedings of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[8] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000, pp. 995–1001.

[9] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," *IEEE Trans. Robot. Automat.*, vol. 16, no. 4, pp. 442–447, August 2000, preliminary version appeared in ICRA 1998, pp. 630–637.

[10] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, 6th ed. MIT Press and McGraw-Hill Book Company, 1992.

[11] J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek, "Beyond average: Towards sophisticated sensing with queries," in *IPSN'03*.

[12] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *MobiCom'00*.

[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *Mobile Computing and Networking*, 2001, pp. 85–96.

[14] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *MobiCom'01*.

[15] K. Whitehouse, C. Sharp, E. Brewer, and D. Culle, "Hood: a neighborhood abstraction for sensor networks," in *MobiSys'04*.

[16] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, 2003, pp. 28–39.

[17] C.-L. Fok, G.-C. Roman, and C. Lu, "Rapid development and flexible deployment of adaptive wireless sensor network applications," in *ICDCS'05*.

[18] ——, "Mobile agent middleware for sensor networks: An application case study," in *IPSN'05*.

[19] "TinyOS community forum," http://www.tinyos.net/.

[20] "Activmedia," http://www.activemedia.com.

[21] "Xbow," http://www.xbow.com.

[22] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 2–16, 2003.

[23] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS)*, vol. 30 (1), 2005.

[24] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A two-tier data dissemination model for large-scale wireless sensor networks," in *MobiCom'02*.

[25] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon, "Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks," in *SenSys'03*.

[26] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-Based Collaboration for Target Tracking in Sensor Networks," *IEEE Transactions on Wireless Communication*, vol. 3 (5), 2004.

[27] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya, "A spatiotemporal query service for mobile users in sensor networks," in *ICDCS'05*.

[28] M. A. Batalin, G. S. Sukhatme, and M. Hatting, "Mobile robot navigation using a sensor network," in *ICRA'04*.

[29] A. Verma, H. Sawant, and J. Tan, "Selection and navigation of mobile sensor nodes using a sensor network," in *IEEE International Conference on Pervasive Computing and Communications*, 2005, pp. 41–50.

[30] P. Corke, R. Peterson, and D. Rus, "Coordinating aerial robots and sensor networks for localization and navigation," in *Proceedings of the Seventh International Symposium on Distributed Autonomous Robotic Systems*, ser. Distributed Autonomous Robotic Systems 6. Springer-Verlag, June 2004. [Online]. Available: http://cmc.cs.dartmouth.edu/papers/corke:navigatesensors.pdf

[31] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 181–196, August 2004.

[32] ——, "Sensor network-based multi-robot task allocation," in *Proceedings of IEEE/RSJ International Conference On Intelligent Robots and Systems*, October 2003.

[33] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Deployment and connectivity repair of a sensor net with a flying robot," in *Proceedings of the Ninth International Symposium on Experimental Robotics*, June 2004.

[34] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM Press, June 2004, pp. 111–124.

[35] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, vol. 353.

[36] G. Wang, G. Cao, T. L. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *INFOCOM'05*.

[37] G. Wang, G. Cao, and T. L. Porta, "Movement-assisted sensor deployment," in *INFOCOM'04*.