2005-41

Roadmap Query for Sensor Network Assisted Navigation in Dynamic Environments

Authors: Bhattacharya, Sangeeta; Atay, Nuzhet; Alankus, Gazihan; Lu, Chenyang; Bayazit, O. Burchan; Roman, Gruia-Catalin

November 23, 2005

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.

Roadmap Query for Sensor Network Assisted Navigation in Dynamic Environments

Sangeeta Bhattacharya, Nuzhet Atay, Gazihan Alankus,
Chenyang Lu, O. Burchan Bayazit and Gruia-Catalin Roman
Department of Computer Science and Engineering
Washington University in St. Louis
Campus Box 1045, One Brookings Drive
St. Louis, MO 63130-4899, USA
{sangbhat,atay,gazihan,lu,bayazit,roman}@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.

1 Introduction

Mobile entity navigation is a crucial part of many mission critical applications like fire fighting and search and rescue operations in disaster areas. These scenarios usually involve dynamic environments that make navigation dependent on up-to-date knowledge of the changing environment. Moreover, information about a large region around the mobile entity is required in order to achieve good navigation performance. For example, in the case of a robot navigating a region on fire, the robot would need real-time temperature information about the surrounding areas in order to navigate the region without getting burnt. Also, due to the highly dynamic and unpredictable nature of fire, frequent temperature information of the surrounding areas would be needed for continuous awareness of the neighboring environment. On-board sensors have

a limited sensing range and hence cannot provide sufficient information required to make good navigation decisions. Wireless sensor networks (WSNs), on the other hand, present new opportunities to obtain frequent, up-to-date information about a large expanse of the surrounding area. Information obtained from the WSN can be used by the mobile entity to make good navigation decisions, with reduced risks. Moreover, WSNs are easily deployable and are also economically feasible. Once deployed, a WSN can serve several mobile entities and can also be employed to co-ordinate the movement of multiple mobile entities.

The use of WSNs for navigation in dynamic environments presents important new challenges. Since frequent sensor data updates are required to maintain continuous awareness in dynamic environments, the data collection process can induce a heavy communication workload on the WSN, which usually has limited bandwidth and energy. The resulting network contention and congestion may cause excessive communication delay and loss of sensor data, which may significantly affect the safety and navigation performance of the mobile entity. Therefore, it is important to design efficient query protocols that can collect updated sensor data needed for safe navigation at minimum communication cost.

In this paper, we present a novel roadmap-based approach for navigation in dynamic environments. Our approach consists of two components; a roadmap-based navigation algorithm for the mobile entity and a distributed query protocol called *Roadmap Query (RQ)* for the WSN. The navigation algorithm searches for a safe path to the goal, by exploring a roadmap in the region based on updated sensor data collected from the WSN using the roadmap query. The roadmap query protocol achieves communication cost savings by querying nodes only in the vicinity of the mobile entity and by using a sampling strategy that queries only a few selected nodes lying along roadmap edges in the query area. The sampling strategy eliminates communication cost resulting from the collection of unnecessary and redundant data, while still enabling RQ to provide sufficient data needed for successful navigation in dynamic environments.

The main contributions of this paper are as follows. (1) We present the Roadmap Query protocol that is optimized for navigation in dynamic environments. (2) We present a mobile agent based implementation of RQ on a Mica2 mote testbed. (3) We integrated the NIST Fire Dynamics Simulator (FDS) [1] with NS-2 to obtain a simulation environment that allowed us to evaluate our protocols under realistic fire scenarios. Simulations in this environment brought to light several non-intuitive cases that arise due to the highly dynamic nature of fire, enabling us to further improve our query strategy. (4) Our simulation results show that RQ outperforms several existing protocols in terms of navigation performance as well as communication efficiency and tolerance to node failures.

This paper is organized as follows. We present related work in Section 2 followed by a formal definition of the problem, in Section 3. This is followed by a detailed presention of the navigation algorithm and the RQ query strategy, in Section 4 and Section 5, respectively. We then present our agent based implementation of

RQ, together with experimental results in Section 6. This is followed by simulation results in Section 7 and conclusion in Section 8.

2 Related Work

Several methods [2, 3, 4, 5] for robot navigation have been proposed in the past. These methods either assume a priori knowledge of the environment or use on-board sensors to avoid obstacles. A priori knowledge of the environment does not help in navigation when the environment changes continuously. On the other hand, on-board sensors have limited sensing range and hence cannot provide information about a sufficiently large region, which is required to make good navigation decisions in dynamic environments. Recent work in this area suggests integrating sensor networks with mobile entities to enable navigation in the presence of dynamic obstacles. The proposed methods fall into two distinct categories.

The first category uses some form of global flooding initiated by the goal, by the dynamic obstacle or by the mobile entity itself. This approach is unsuitable for dynamic environments since the need to constantly maintain a high degree of awareness of the changing environment (e.g., a spreading fire) would cause frequent flooding of the network. Thus, this approach may suffer from high communication cost and network contention, which would lead to poor navigation performance. It also wastes network energy, thereby decreasing network lifetime. Protocols suggested in [6] and [7] fall into this category. Both protocols construct global navigational fields to guide the robot to the goal. In [6], the goal generates an attractive potiential field that pulls the robot towards the goal, while an obstacle (static or dynamic) generates a repulsive potential field that pushes the robot away from the obstacle. We will henceforth refer to this method as the Dartmouth Algorithm (DA). Unlike DA, the method in [7] uses value iteration to compute the magnitude of directional vectors that guide the robot to the goal. The approach proposed in [7] is used in [8] and [9] for robot coverage and exploration of space and for multi-robot task allocation. The approach presented in [10] also falls into this category. It addresses navigation of mobile sensor nodes. The protocol assumes a sensor network consisting of both mobile and static sensor nodes. Mobile sensor nodes are guided to event locations to provide more coverage of the area. Thus, in this approach, the goal is an event location. The goal initially floods the network to locate a suitable mobile sensor node. Mobile sensor nodes respond to the flood by sending a response to the goal. The protocol then creates a navigation field around the path taken by the response, to draw the mobile node to the goal. Since the navigational field is only around a path that does not change until the goal changes, this approach cannot efficiently handle dynamic obstacles. Unlike the above approaches, the approach used in [11] assumes that a path already exists in the network, and uses controlled flooding to guide the robot to the start of the path, after which the robot follows the path. This approach is not applicable to dynamic environments where an initially safe path may quickly

become unsafe due to changing conditions.

The second category of protocols do not use global flooding but instead use a local query strategy to achieve navigation. Our earlier work, presented in [12], which we will henceforth call Local Query (LQ), falls into this category. The suggested approach builds a roadmap of the area, which is used by the mobile entity to navigate the region. The path from the start to the goal is built incrementally as the mobile entity traverses the region, by querying all nodes in the vicinity of the mobile entity. This approach avoids global flooding by restricting the mobile entity to a roadmap and by making local decisions regarding the next set of roadmap edges to be traversed, in order to safely reach the goal. However, since only information along the roadmap edges is needed to make navigation decisions, this method wastes energy and bandwidth by collecting information from all nodes in the query area. In contrast, the roadmap query suggested in this paper addresses this drawback by performing only selective sampling to collect only necessary information. The amount of necessary information is dependent on the condition of the environment and changes with the change in environment. Our sampling strategy eliminates collection of both redundant as well as unnecessary data and adapts to the changing environment.

LQ integrates the roadmap navigation strategy with a local query strategy similar to DCTC [13] and MobiQuery [14]. Other query strategies exist in literature but are not suitable for navigation. Query services like TinyDB [15] and Directed Diffusion [16] use a flooding based approach initiated by the querying entity and hence are more suitable for data collection by stationary entities. TTDD [17] is a proactive protocol, where a source initiates the setup of global forwarding paths to itself, in the form of a grid. These forwarding paths are used by the mobile entities to obtain information from the sources. Since a global grid is formed per source, TTDD would initiate the formation of a large number of global grids in a dynamic environment, thus incurring high communication cost. This makes TTDD unsuitable for navigation in dynamic environments. Another protocol that supports mobile entities is SEAD [18]. SEAD builds and maintains a data dissemination tree rooted at a source, with the mobile entities forming the leaves of the tree. Thus, SEAD is more suitable for applications where multiple mobile entities query a static query area.

3 Problem Formulation

The navigation 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 *dynamic obstacles*. Dynamic obstacles can be either mobile entities like a car or a spreading substance like fire, chemical spill etc. Thus, a dynamic obstacle is any obstacle whose location or shape changes with time. Good navigation performance is defined in terms of path safety, short path length and low path traversal time.

In this paper, we consider fire as the representative example for a dynamic obstacle. Thus, the temperature of the region traversed by the mobile entity is 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. The mobile entity is assumed to get burnt if it is at a location that has temperature higher than a threshold Δ_{burn}. A safe path is defined as one where the maximum temperature along the path taken by the mobile entity remains below the threshold Δ_T, while the mobile entity is on the path. Cooler paths are thus considered safer than hotter paths.

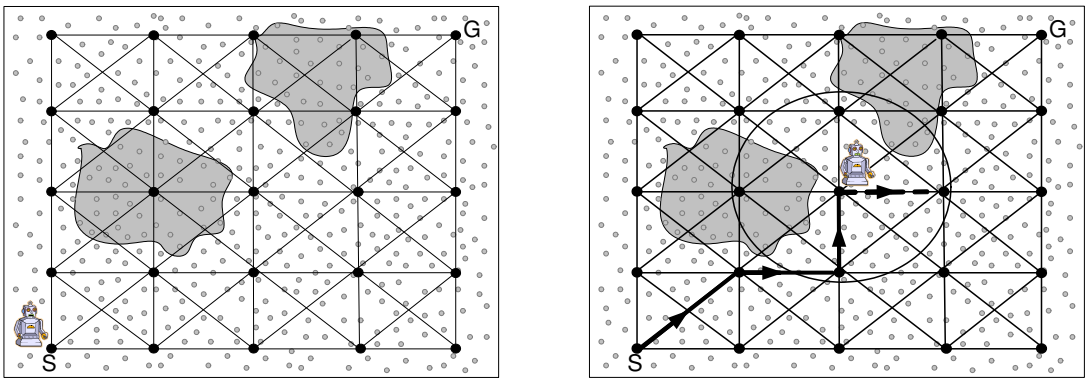
We make the following assumptions in the paper: (i) The WSN nodes form an ad hoc network. (ii) Nodes are location aware. (iii) The mobile entity communicates with the WSN through an on-board gateway device (e.g. PDA, stargate) (iv) Nodes have a limited sensing range R_S. R_S is chosen such that if the temperature sensed by a node is below the threshold Δ_T, then the temperature at any point within the sensing range is below the threshold Δ_{burn} at which the mobile entity gets burnt. This allows us to assume that edges with nodes having temperature above Δ_T are unsafe. The sensing range is thus dependent on the tunable parameter Δ_T. A lower Δ_T results in a longer sensing range.

Even though we consider the specific scenario where the dynamic obstacle is a fire, our solution 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).

4 Navigation Algorithm

Our navigation algorithm adapts the roadmap navigation method [3] to make it more suitable for dynamic environments and for integration with WSNs. The roadmap navigation method builds a roadmap of the region, that is used by the mobile entity to reach the goal. A roadmap is a weighted graph that comprises of a set of possible paths in the environment. The weight of a path represents the path safety and length. When the mobile entity searches for a safe path to the goal, it only considers the paths on the roadmap instead of all possible paths in the entire region. Thus, the roadmap based approach has low computational complexity. Furthermore, it is particularly suitable for sensor network assisted navigation, since it reduces the amount of sensor data that must be collected from the WSN by requiring information only along the roadmap edges.

The navigation algorithm consists of first constructing a roadmap and then incrementally finding safe sub-paths leading to the goal. Sub-paths are selected based on edge weights that are repeatedly updated using temperature information obtained from the WSN. More specifically, after constructing the roadmap, the mobile entity issues a roadmap query to obtain the maximum temperature along the roadmap edges within its query area. The query area is a circle with a certain query radius R_q, centered at the current



(a) Roadmap Generation (b) Sub-path Selection

Figure 1: Working of the Navigation Algorithm. The figures show the grid roadmap. Roadmap nodes are colored black and sensor nodes are colored gray. The start is denoted by the letter S and the goal is denoted by the letter G. The gray shaded regions denote fire. Figure (a) shows the roadmap constructed initially. Figure (b) shows the sub-path selection stage in which the robot chooses the sub-path that falls inside the query area (shown by the circle). The solid lines show the path taken by the robot to reach its current location. The dotted line with an arrow shows the selected sub-path and direction of navigation.

location of the mobile entity $p_e(t)$. The mobile entity waits for a time T_w to receive the query result which consists of a list of maximum temperatures along the roadmap edges within the query area. At the end of the wait period T_w, the mobile entity calculates the edge weights based on the temperature information obtained and finds a safe sub-path to the goal. 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. This entire process is repeated every time the mobile entity reaches the end of a sub-path, until the mobile entity safely reaches the goal.

Thus, the roadmap navigation algorithm has three stages (i) roadmap generation, (ii) roadmap edge weight updation based on environmental information obtained from the sensor network, and (iii) sub-path selection. The roadmap generation stage occurs just once at the start of the algorithm, while the roadmap edge weight updation and sub-path selection stages occur repeatedly till the mobile entity safely reaches the goal. Details of these stages are presented next.

(i) Roadmap generation: Traditional Probabilistic Roadmap Methods (PRM) [3] randomly choose points in space to construct a roadmap. However, we choose to have a grid as the roadmap as shown in Figure 1(a). A benefit of using a grid is that roadmap information can be easily included in a query message without significantly increasing the message size (explained in Section 5.1). The grid points are called *roadmap nodes* and the edges are called *roadmap edges*. Note that there is no direct correlation between roadmap nodes and sensor nodes. The roadmap nodes are virtual points that are placed in space, without considering sensor locations.

(ii) Roadmap edge weight updation: The roadmap edge weights are used to find a short, safe sub-path on the roadmap that leads to the goal, as the mobile entity traverses the roadmap. In order to balance path safety and path length, we use an edge weight function that is a weighted function of the normalized

maximum edge temperature and the normalized edge length. The maximum edge temperature is provided by the RQ protocol that queries the WSN. 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 based on recent query results, 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. The tunable parameter α determines the tradeoff between safety and efficiency (in terms of path length).

δ_e in equation 1 is obtained from the query result and is approximated as below.

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

where S is the set of sensors that cover edge e and δ_s is the temperature at a sensor $s \in S$. 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 are determined by the query protocol 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.

(iii) Sub-path selection: A sub-path consisting of edges lying within the query area is selected based on the updated edge weights. We choose the sub-path that is a part of the path with the least weight from the start to the goal, at that instant. The path with the least weight is found using the Dijkstra's shortest path algorithm [19] on the roadmap. This stage is illustrated in Figure 1(b).

5 Design of Roadmap Query

In this section, we present the novel RQ query protocol used to collect temperature information from the WSN. RQ collects updated temperature information from sensors covering the roadmap edges in a query area with radius R_q. It is issued by the navigation algorithm, every time the mobile entity reaches the end of a sub-path, until the mobile entity reaches the goal. In addition, to improve safety, it is also issued when the temperature at the mobile entity location rises above the threshold Δ_T. The temperature at the mobile entity location is obtained using an on-board sensor.

In the following sub-sections, we first present the basic RQ protocol and then present an extension to the protocol, required to handle node failures. Following that, we present an analysis that shows that the RQ protocol successfully collects sensor readings from nodes along all roadmap edges lying in the query area,

under certain conditions.

5.1 Basic Roadmap Query Protocol

RQ minimizes the communication workload on the network by reducing the number of nodes involved in the query process. This is achieved by optimizing the query protocol in accordance with the roadmap-based navigation algorithm. Since the navigation algorithm requires the maximum temperature only along the roadmap edges, the query message, is forwarded only along the roadmap edges lying within the query area. Moreover, due to the high density of sensor nodes, the query message is not forwarded by all nodes along an edge, but only by some selected nodes. These selected nodes form a backbone of nodes along the roadmap edges that fall within the query area. RQ requires all backbone nodes to respond to the query. Non-backbone nodes that hear the query message respond to the query only if their sensor reading is above Δ_T. The backbone and non-backbone nodes form a tree structure with the mobile entity as the root. The formed tree is used to aggregate the query results and deliver them to the mobile entity. Thus, RQ reduces communication cost not only by reducing the number of nodes that forward the query message but also by reducing the number of nodes that respond to a query, within a query area.

In order to achieve the communication cost reduction, RQ requires the queried nodes to have knowledge of the roadmap and to maintain 2-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 calculate the grid points and edges. The second requirement requires all nodes in the network to maintain 2-hop neighborhood information which may introduce some overhead. Neighborhood information is maintained, by having each node periodically broadcast beacons, also referred to as *hello messages* [20]. Hello messages contain the ID of the sending node and the IDs of its 1-hop neighbors. 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 and also stores the neighborhood information of the sending node. Each entry in the neighborhood table is associated with a timestamp that corresponds to the time the most recent hello message was received from that neighbor. The timestamp field is used to detect failed neighbors. We have described a simple neighborhood management technique but more sophisticated techniques [21] can also be used. Note that, similar neighborhood information is also required by other common services such as routing and power management [22, 23, 24].

RQ uses two 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*. The forwarding rule identifies

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 timeout at the right time.

When Query reply is received

1. *If* result not yet sent *then* store *else* discard.

When *SendTimer* times out

1. Send aggregated query result to parent.

Figure 2: Roadmap Query (RQ) Algorithm.

backbone nodes while the reply rule identifies non-backbone nodes.

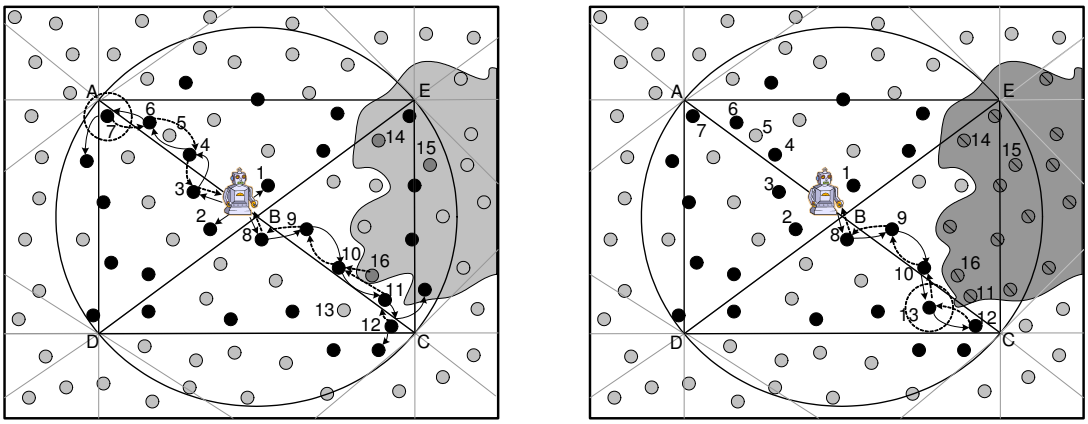
Forwarding Rule: By the forwarding rule, if a node receives a query message that is being propagated along edge $e = \overrightarrow{p_{e1}p_{e2}}$ where p_{e1} and p_{e2} are the endpoints of the edge, and the arrow denotes the direction of query message propagation, then, the node rebroadcasts the 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 [1]. Note that, by this method, only a few nodes along the edge, called backbone nodes, rebroadcast the query message. Thus, some nodes do not rebroadcast the query message, thinking that another node that is closer to the endpoint will rebroadcast the message. These nodes listen for a certain time interval to see if the query message is rebroadcasted. If the query message is not rebroadcasted within this time, these nodes rebroadcast the message. This method takes care of situations where the node selected to rebroadcast the query message does not receive the query message due to collision or other factors.

Reply Rule: The reply rule states that a node should send a query reply, if (i) it is a backbone node, or, (ii) its temperature is above Δ_T and it covers a roadmap edge that falls within the current query area. The first condition draws query results from the minimum number of nodes that entirely cover all the roadmap edges within the query area. The second condition adapts the number of nodes responding to the query, according to the danger level. This condition enforces the safety of the path by drawing query results from nodes that lie along the roadmap edges within the query area and that sense a dynamic obstacle (e.g. fire).

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 [2], 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

[1] A node knows if a neighbor can hear the same query message by checking its neighborhood table that contains 2-hop neighborhood information.

[2] Node i sets node j as its parent only if the link between both nodes is symmetric. This information is obtained from the neighborhood table.



(a) (b)

Figure 3: Working of RQ protocol. The figures show the backbone (colored black) and non backbone nodes (colored dark gray) selected by the RQ protocol in response to a query issued by a robot positioned at B. The roadmap edges within the query area (solid black circle) are colored black. Figure (a) shows the query message (solid arrow) and query reply (dotted arrow) propagation along edges BA and BC. The shaded region represents a region with temperature above the threshold. Figure (b) shows a possible case where RQ fails. The shaded region in this figure represents very high temperature at which all nodes in the region have failed. Failed nodes are crossed out. In this case, even though the fire has spread across edge BC, the mobile entity considers the edge safe since there are still enough active nodes with temperatures below the threshold, along edge BC, to cover the entire edge.

results at a time that facilitates data aggregation. 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 timeout at that time. When the timer times out, node i sends an aggregated query result, deduced from its information and the information obtained from its children, to its parent. If node i and its children are along the same edge, the MAX function is applied to the sensor readings. Otherwise, the results are just merged into one message.

The query reply time 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 within the query area. Thus, the timer is set to timeout 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. The RQ algorithm is shown in Figure 2 and is illustrated in Figure 3(a).

Figure 3(a) illustrates different aspects of the RQ protocol. (i) It shows the backbone (colored black) and non-backbone (colored dark gray) nodes selected by the RQ protocol in response to a query issued by a robot positioned at B. The resulting query area is shown by a solid circle centered at B. The roadmap edges lying within the query area are colored black. (ii) The figure illustrates the query message (solid arrow) propagation along edges BA and BC. The query message is propagated from node 2 to node 3, to node 4 and then to node 6 and finally to node 7 along edge BA. The query message propagation along edge BA stops

at node 7 since it covers endpoint A. Node 7 covers endpoint A since A lies within node 7's sensing range (shown by dotted circle centered at 7). Node 7 then propagates the message along the adjoining roadmap edges in the query area. Note that the query message is forwarded by node 4 and then by node 6 and not by node 5. This is because of the forwarding rule. When node 5 hears the query message from node 4, it sees that it has a neighbor, node 6, that is also node 4's neighbor (hence, it must have also heard the query message) and that is closer to the endpoint A. Thus, by the forwarding rule it does not rebroadcast the query message. (iii) The figure illustrates the outcome of the reply rule when a portion of the query area (shaded region) has temperature above the threshold. By the reply rule, nodes 14, 15 and 16 in this area must reply to the query, since their temperatures are above the threshold and they cover a roadmap edge lying within the query area. These nodes are thus, non-backbone nodes. (iv) The query reply (dotted arrow) propagation along edges BA and BC is also shown. Note how a non-backbone node, node 16, becomes a leaf node, under parent node 10.

5.2 Extension to Handle Node Failures

Robustness to node failures is especially important in dynamic environments since nodes can be destroyed by harsh environments such as fire. The basic RQ protocol cannot handle certain situations arising due to node failures, as shown in Figure 3(b). In the figure, the shaded region depicts a spreading fire, that causes the nodes in the region to fail. Due to node failures, edges BE and EC are not covered by working nodes. Hence, the robot does not receive sufficient information about these edges and considers them unsafe. However, edge BC is completely covered since even though the fire burns node 11, node 13 takes its place in forwarding the query message, thus giving the robot the false impression that the edge is safe. If the robot were to choose to traverse edge BC, it would collide with the dynamic obstacle, which in this case, is fire and get burnt. This scenario shows the importance of fault-tolerance in dynamic environments. Therefore, we extend RQ to avoid such situations.

In order to make RQ fault-tolerant we include node failure information in the query results. The mobile entity, uses node failure information to avoid paths with failed nodes, assuming that node failures are due to destruction by fire. To obtain node failure information, nodes responding to a query are required to not only send their sensor reading but also send a list of failed neighbors that cover roadmap edges in the current query area. Also, the reply rule is modified slightly such that nodes now send a query reply if (i) they are backbone nodes, (ii) their temperatures are above a threshold and they cover a roadmap edge lying within the query area or (iii) they have failed neighbors that cover roadmap edges lying within the query area.

It can be seen that with these modifications RQ is successful in situations like the one depicted in Figure 3(b). This is because, by the modified reply rule, the robot is informed about the failed nodes 11 and

16 by either node 10, node 12 or node 13. Since node failure is assumed to be due to destruction by fire, the robot infers that the edge BC is not safe and does not traverse that edge. Thus, the modifications make RQ robust to situations where the fire destroys only some nodes along an edge leaving enough working nodes with temperatures below Δ_T to cover the edge, which give the mobile entity the false impression that the edge is safe. Cases where the fire destroys enough nodes such that an edge is no longer covered is already handled by the basic protocol.

The modified RQ protocol depends on node failure information which is easily obtainable. 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, since a lower value of n will give rise to more false positives while a higher value of n will result in delayed awareness of danger, thus leading to poor navigation performance. In our simulations, we set $n = 2$. 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 this node knows to be alive. Any such node is removed from the list before it is passed up the tree.

5.3 Analysis

In this section, we show that RQ successfully gathers the information required by the navigation algorithm within a query area, under the following two conditions.

The first condition is 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 at all. A sensing covered network is desirable, as it increases the mobile entity's awareness of the surroundings thus improving its navigation path.

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, i.e., $R_C \geq 2R_S$. The double range property guarantees network connectivity in a sensing covered network [25] and hence is a desirable property for such networks. Since the sensing range depends on the temperature threshold Δ_T, we can achieve the double range property by selecting an appropriate Δ_T.

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's 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 formally prove the following property 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 edge e.*

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 edge e.

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

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 guaranteed to be received by every node covering a roadmap edge lying completely within the query area.

This property of RQ is very useful in environments that do not cause node failures (e.g., chemical spill). Environments like fire that cause node failures violate the sensing coverage condition, in which case RQ can only provide best effort service. We note that it is extremely difficult to provide any guarantees in the presence of node failures. However, as shown in our simulations, RQ can still provide sufficiently good performance under a number of realistic scenarios.

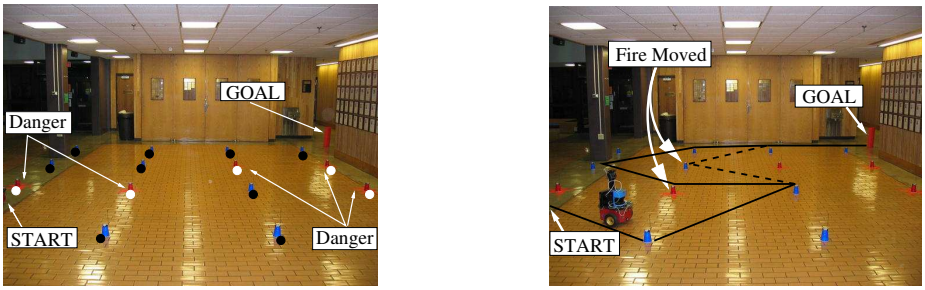
6 Implementation

We implemented the basic RQ protocol on Agilla [26, 27], a mobile agent middleware for the TinyOS [28] and Mica2 [29] mote platform. A mobile agent based implementation enables RQ to be used in a pre-deployed WSN without requiring the RQ protocol to be pre-installed on the WSN. A WSN running some other application can be quickly re-utilized to run the RQ protocol by just injecting mobile agents containing the protocol into the network. The capability to flexibly reprogram a WSN for a different application is particularly important to WSNs that have limited storage and long operational lifetime [26, 27, 30]. For example, a WSN deployed in a building for temperature monitoring can be quickly re-programmed to run the RQ protocol in case of a fire emergency. The RQ protocol can then be used to guide people safely out of the building. An agent based implementation has the additional benefit of ease of programming [26].

An Agilla application consists of one or more mobile agents that coordinate with each other, to achieve application-specific behavior. An agent is programmed using a high-level language supported by Agilla. Agilla provides primitives for an agent to move and clone itself from sensor node to sensor node while carrying its code and state, effectively reprogramming the network. New mobile agents can be injected onto a sensor node, 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 node. Multiple agents can communicate and coordinate through local or remote access to tuple spaces. Prior experiences with Agilla have demonstrated that it can provide efficient and reliable services needed by highly dynamic applications such as fire tracking [27]. More details about the Agilla middleware can be found at *http://mobilab.wustl.edu/projects/agilla/index.html*.

6.1 RQ using Agents

In the agent based implementation of RQ, the mobile entity injects an *explorer agent* into the network that collects the edge weights and delivers them to the mobile entity. Once injected into the network, the explorer agent clones itself on nodes lying along the roadmap edges according to the forwarding rule. The reply rule is applied to determine the agents that need to respond to the query. The agent migration sets up a tree structure along the roadmap edges within the query area, which is used to collect the query result. Per node query results are aggregated such that a list of per-edge-maximum-temperatures is forwarded along the tree branches to the mobile entity through remote tuple space operations. The mobile entity processes the query result and takes appropriate action as explained before.



(a) (b)

Figure 4: (a) The experimental environment (8×8 meters). (b) Spreading fire. When the fire spreads, the robot avoids initial path (dotted line) and follows a safer path (solid line).

6.2 Experiments

We used a Pioneer-3 DX robot by ActiveMedia [31], as the mobile entity in our experiments. The Mica2 mote network was arranged in a 4x4 grid as shown in figure 4(a). 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 4(a)), 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 4(b). As is seen, the robot successfully avoids dangerous places by staying close to motes with normal temperature.

Dynamic fire. The dynamic fire scenario 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 4(b). The solid line shows the path followed by the robot, while the dotted line represents the initial path.

These experiments demonstrate that a robot can use our agent based implementation of RQ, to successfully find a safe path in the presence of dynamic obstacles. We further evaluate the performance of RQ and compare it with existing approaches through simulations under realistic fire scenarios.

7 Simulation Results

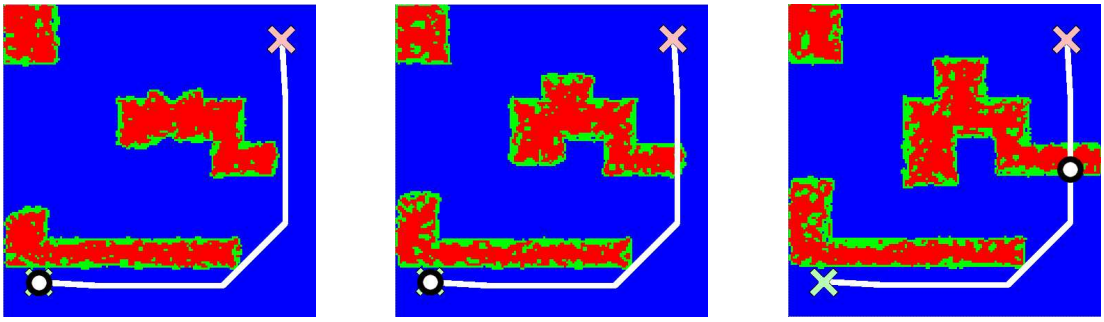
In this section, we present the results obtained from simulations in NS-2. We evaluate and compare RQ with existing protocols, using 9 different realistic fire scenarios, obtained using the NIST Fire Dynamics Simulator (FDS) [1]. In all the scenarios, the fire starts in different locations scattered over the region and then spreads over the region with time. 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.

We evaluate the RQ protocol both with and without the extension for handling node failures to observe the difference in performance caused by the extension. We refer to the basic RQ protocol as B-RQ and the RQ protocol with the extension as Robust RQ (R-RQ), in this section. We also compare our protocol to other approaches like LQ [12] and DA [6] which were discussed earlier in the related works section (Section 2).

As mentioned earlier, LQ is a spatiotemporal query that queries all nodes in the vicinity of the mobile entity. The mobile entity uses the information delivered by LQ to compute sub-paths to the goal, as it traverses the region. This method also uses the roadmap based navigation method to compute the paths and hence requires information only along roadmap edges. Thus, this algorithm wastes energy by collecting unnecessary information and therefore suffers from high communication cost.

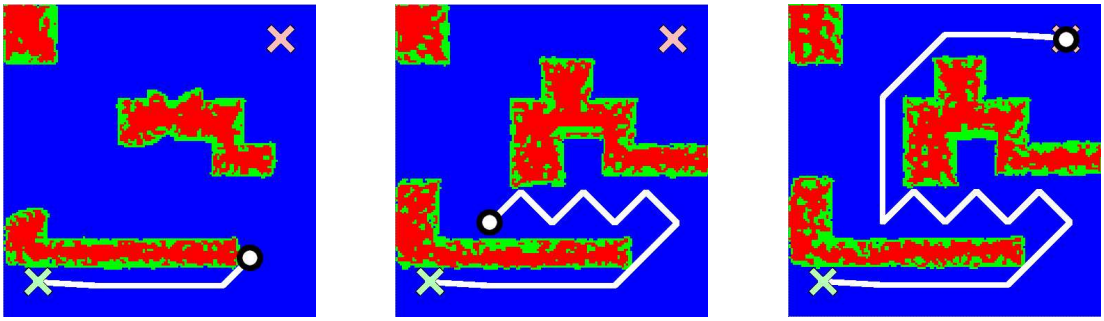
DA, on the other hand, employs a global data dissemination approach, where a positive potential field is set up by the goal and a negative potential field is set up by the obstacle. The potential fields are set up throughout the network and the resultant gradient of the combined potential fields is used to direct 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 back 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.

In addition to LQ and DA, we also compare our protocol to an intuitive approach called Global Query (GQ). In this approach, the mobile entity broadcasts a query message, which is flooded throughout 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 in accordance with Equation 1 to obtain a complete path from the start to the goal. Since this method employs global data collection, it has a high communication cost. In



(a) (b) (c)

Figure 5: 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 blue region represents the safe region and has temperature below 60^oC. The red and green regions represent fire with temperature above 150^oC and above 60^oC, respectively.



(a) (b) (c)

Figure 6: 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 blue region represents the safe region and has temperature below 60^oC. The red and green regions represent fire with temperature above 150^oC and above 60^oC, respectively.

addition, it also suffers from a long query delay, which significantly reduces a mobile entity's awareness of the region. This leads to situations where the mobile entity gets burnt while traversing a path that changes from being safe to being unsafe, due to lack of awareness of the changing environment. This situation was observed in a simulation run and is shown in Figure 5. The white line depicts the safe path that is initially computed by the mobile entity. As the mobile entity traverses the path, a part of the path is engulfed by fire. The mobile entity is unaware of this and continues on the path and gets burnt. In this particular such situation, the mobile entity actually detects the rising temperature via its on-board sensor and issues a new query. However, due to the significant query delay, the fire spreads to the location of the mobile enitity before it finds a safe path and burns the mobile entity. The outcome of using the RQ protocol for the same scenario is shown in Figure 6. Since the RQ protocol uses local queries with low query delay (due to low communication cost) it computes successive safe sub-paths that successfully lead it to the goal, around the regions on fire.

Since LQ and RQ query only nodes within a local query area, we also evaluate the performance of just these two 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 roadmap grid size.

Each simulation was run with 900 nodes uniformly distributed in a $450m \times 450m$ area. 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 $90m \times 90m$. The communication range and bandwidth of the nodes were set to 45m and 40kbps, respectively. The sensing range of the nodes was obtained using the maximum temperature gradient δT at the border of a fire. Thus $R_S = \frac{\Delta_{burn} - \Delta_T}{\delta T}$. δT was found to be $4.5^oC/m$ from the simulation scenarios. Δ_T and Δ_{burn} were set to 60^oC and 150^oC, respectively, assuming a robot as the mobile entity. These settings result in $R_S = 20m$. Note that $R_C \geq 2R_S$, satisfying the double range property.

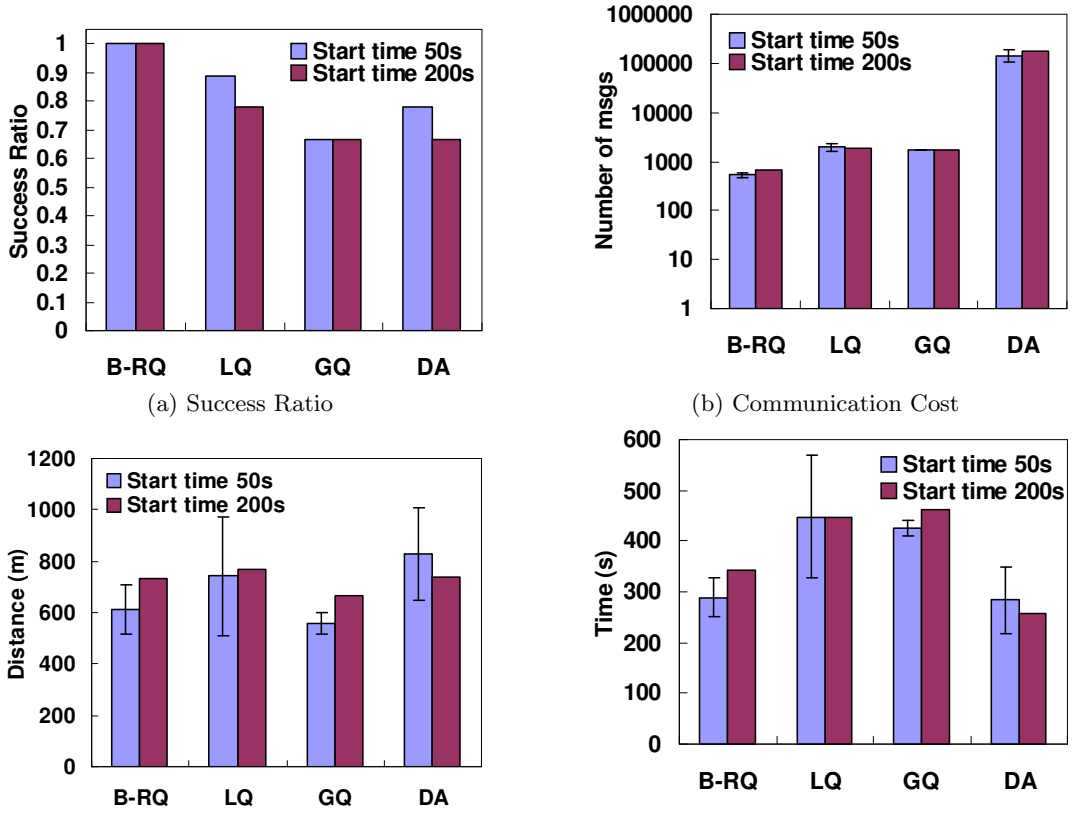
As mentioned in Section 4, an edge is considered covered if certain points on the edge are covered. The selection of the points differs with the query protocol. Since LQ and GQ query all nodes within a query area, any number of points (≥ 2) on an edge can be considered in these algorithms. Note that the query area in LQ is a circle (of certain radius) centered at the mobile entity location while the query area in GQ is the entire region. In our simulations of LQ and GQ, we considered coverage of five equidistant points on an edge to indicate edge coverage. However, since fewer nodes respond to queries in RQ, we considered coverage of only the endpoints of an edge to indicate edge coverage in RQ.

The period T_w, during which the mobile entity waits for the query results, was determined experimentally for LQ, GQ and RQ. In each of these algorithms, T_w was set to a value that permitted at least 90% query results to be received by the mobile entity, per query. As expected, the value of T_w was found to increase with the number of nodes involved in a query. T_w was set to 250s in GQ. The values of T_w used for LQ and RQ are given in Table 1. T_w represents the minimum query latency of a query protocol. The other tunable parameter h_{max}, which represents the maximum hop count and is used in RQ 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

Table 1: Query latency T_w of LQ and RQ.

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 safely 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 scenarios where the mobile entity successfully reaches the goal. (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 results. Note that the latter depends on the performance of the query



(a) Success Ratio

(b) Communication Cost

(c) Path Length

(d) Path Traversal Time

Figure 7: Performance Comparison in the absence of node failures.

protocol. (4) *Communication Cost*, defined as the average number of messages sent per scenario in which the mobile entity successfully reaches the goal.

In the following subsections we first present the results obtained from simulations without node failures and then present the results obtained from simulations with node failures. All performance results, except success ratio, are averaged over scenarios where the mobile entity successfully reaches the goal, in all protocols being compared. In addition to the mean we also present 90% confidence intervals for all cases except for the case where the robot start time is 200s and there are no node failures. For this particular case, there are only two scenarios where the mobile entity safely reaches the goal, in all protocols being compared. Confidence intervals for this case would be statistically insignificant and hence have been omitted.

7.1 Performance in the Absence of Node Failures

7.1.1 Performance Comparison

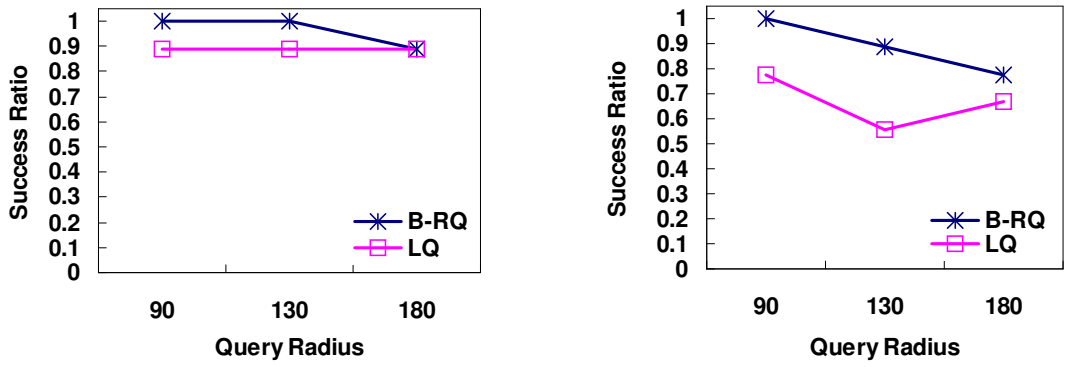
In this sub-section, we compare the navigation performance and communication cost of GQ, LQ, B-RQ and DA. The query radius of B-RQ and LQ is set to 90m in these simulations. Since R-RQ is designed specifically to handle node failures, we do not include it in this section.

Success Ratio: Figure 7(a) shows the success ratio of the different protocols at start times 50s and 200s. As seen in the figure, B-RQ performs better than all the other algorithms and enables the mobile entity to safely reach the goal in all tested scenarios. LQ does not perform as well, because unlike B-RQ, it queries all the nodes in a query area and hence incurs a long query delay. The long query delay slows down the mobile entity's progress towards the goal. As a result, the mobile entity sometimes gets caught up amidst the spreading fire with no safe path leading to the goal, or gets burnt while waiting for the query results.

The success ratio of GQ is the lowest. This is because, GQ does not update the selected path to the goal based on the changing environmental state. At a start time of 50s, the mobile entity usually gets burnt due to lack of awareness of the fire encroaching the chosen path. On the other hand, 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. The success ratio of DA is lower than that of B-RQ and LQ, because of high data loss due to contention caused by the high communication cost of DA (as shown in Figure 7(d)).

Communication Cost: A comparison of the communication cost incurred by B-RQ, LQ, GQ and DA is presented in Figure 7(b). 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 in the network to maintain the potential fields, resulting in periodic flooding of the entire network. In comparison, B-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. In our simulations, B-RQ achieves 73% lower communication cost than LQ and 70% lower communication cost than GQ for a start time of 50s. For a start time of 200s, the savings are 63% and 62% over LQ and GQ, respectively. The large difference in communication cost between B-RQ, LQ and GQ, is not clearly visible in the figure, due to the logarithmic scale. This low communication cost is one of the main advantages of B-RQ, which enhances its navigation performance. The efficiency of B-RQ can also potentially increase network lifetime.

Path Length: Figure 7(c) compares the *path length* obtained by the different algorithms. We see from the figure, that GQ obtains the shortest path length, on average. This is expected, since GQ obtains a global view of the surroundings by querying all the nodes in the network and hence, can choose the best path available. DA however fails to obtain a short path length, even though it maintains a network wide potential field. This is again because of data loss due to its high communication cost. The path achieved by B-RQ is shorter than that of LQ, and is slightly higher than that of GQ. Since B-RQ and LQ perform local queries, the resultant path length is expected to be worse than that obtained from an algorithm with a global view. However, we note that the differences in path lengths are statistically insignificant, based on



(a) Start time 50s

(b) Start time 200s

Figure 8: Effect of query radius on success ratio in the absence of node failures.

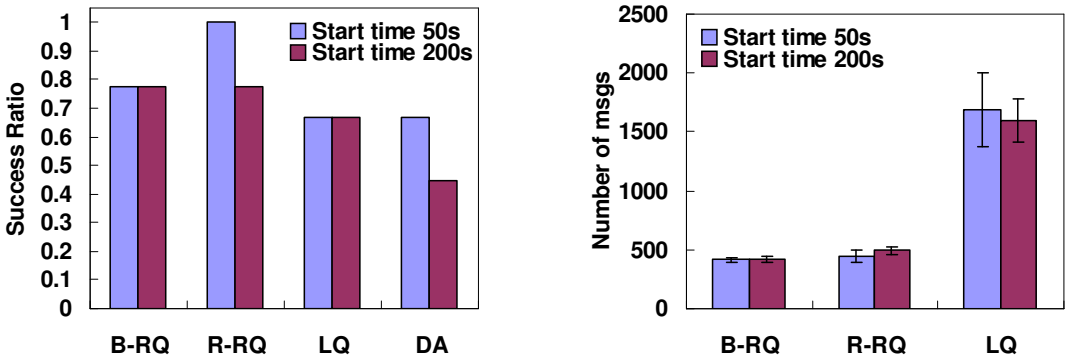
the confidence intervals.

Path Traversal Time: Figure 7(d) shows the path traversal time obtained by the different protocols. From the figure, we see that DA achieves the least path traversal time, on average. 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 B-RQ is comparable to that of DA. This is because, B-RQ also has a low query delay, since it queries only a few nodes per query. LQ and GQ, on the other hand, have higher path traversal times, since they query a large number of nodes and hence have long query delays.

In summary, 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. More importantly, it 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, floods the entire network to update the potential field, thus incurring a high communication cost, which in turn lowers its navigation performance. Overall, B-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 protocol in accordance with the navigation algorithm, in order to navigate successfully in dynamic environments.

7.1.2 Effect of Query Radius

In this sub-section, we compare the effect of query radius on LQ and B-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



(a) Success Ratio (b) Communication Cost

Figure 9: Performance Comparison in the presence of node failures.

communication cost and longer query delay, which have negative impacts 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 the increase in query radius, due to the parallel increase in query delay. A longer query delay implies that the mobile entity takes longer to navigate the region. Since the region gets more difficult to navigate with time, due to the spreading fire, the longer the query delay, the lower the success rate. Longer query delay also increases the danger of the mobile entity getting burnt while waiting for query results. 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.

7.2 Performance in the Presence of Node Failures

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. Nodes are assumed to fail at a temperature of 150^oC.

7.2.1 Performance Comparison

In this sub-section, we compare the navigation performance and communication cost of LQ, DA, B-RQ and R-RQ. GQ is not considered in these simulations, due to its poor performance in the earlier simulations. The query radiuses of LQ, B-RQ and R-RQ were set to 90m in these simulations, since it was found to be the optimal query radius from simulation results presented in Section 7.1.2.

Success Ratio: Figure 9(a) compares the success ratio of the algorithms. From the figure, we see that LQ and DA perform worse than B-RQ, just like in the previous simulations that did not consider node

failures (Figure 7(a)). We also note that the success ratios of B-RQ, LQ and DA are comparatively lower in Figure 9(a) than in Figure 7(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 only some nodes covering a roadmap edge fail, leaving 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 (as shown in Figure 3(b)).

R-RQ effectively improves the success ratio of B-RQ when the mobile entity starts at 50s, at which time many new nodes start failing, due to the spreading fire. This is because R-RQ informs the mobile entity about failed nodes, thus warning the mobile entity about danger areas. On the other hand, R-RQ does not outperform B-RQ when the mobile entity starts at 200s, since by that time, the environment is relatively stable. In contrast, the performance of LQ and DA are affected significantly by node failures. The success ratio of LQ falls by 14% (for start time of 50s) and 24% (for start time of 200s), while the success ratio of DA falls by 14% (for start time of 50s) and 34% (for start time of 200s). This demonstrates that R-RQ is particularly important in dynamic environments.

Communication Cost: Figure 9(b) shows the communication cost incurred by B-RQ, R-RQ and LQ. The commmunication cost of DA is not shown as it is significantly higher than the other protocols. As expected, the communication cost of LQ is much higher than that of B-RQ and R-RQ since it queries all the nodes in the query area. The communication cost of R-RQ is only slightly more than that of B-RQ since it requires nodes to respond if they have failed neighbors even if their temperatures are below the threshold. More specifically, R-RQ achieves 73% savings in communication cost at a start time of 50s and 69% savings in communication cost at a start time of 200s, over LQ.

In summary, we note that R-RQ succeeds in more cases than B-RQ does, without significantly increasing the communication cost. This shows that the extension incorporated in RQ to handle node failures, is indeed effective. With the extension, RQ achieves upto 49% improvement in success ratio over LQ and upto 77% improvement in success ratio over DA (as shown in Figure 9(a)).

8 Conclusion

In summary, the contributions of this paper are four-fold. First, we propose a new sensor network assisted approach for mobile entity navigation in dynamic environments. A key novelty of our approach is the integration of roadmap-based navigation techniques with efficient query protocols for wireless sensor networks. Second, we present a Roadmap Query protocol (RQ) that is specially optimized for collecting spatiotempo-

ral data needed for motion planning. Third, our simulation results demonstrate that RQ can significantly improve the success ratio of navigation while introducing minimum communication cost under realistic fire scenarios and node failures. RQ is shown to achieve upto 77% improvement in success rate over the most commonly used navigation algorithms, with significantly large savings in communication cost. In addition, it achieves upto 49% improvement in success ratio over our earlier protocol with upto 73% savings in communication cost. 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 RQ on Mica2 motes and a robot.

References

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

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

[3] 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.

[4] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *WAFR'98.*

[5] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning," in *ICRA'00.*

[6] Q. Li, M. D. Rosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *MobiCom'03.*

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

[8] 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.

[9] ——, "Sensor network-based multi-robot task allocation," in *IROS'03.*

[10] A. Verma, H. Sawant, and J. Tan, "Selection and navigation of mobile sensor nodes using a sensor network," in *PerCom'05.*

[11] P. Corke, R. Peterson, and D. Rus, "Coordinating aerial robots and sensor networks for localization and navigation," in *DARS'04*.

[12] G. Alankus, N. Atay, C. Lu, and B. Bayazit, "Spatiotemporal query strategies for navigation in dynamic sensor network environments," in *IROS'05*.

[13] 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.

[14] 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*.

[15] 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.

[16] 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.

[17] 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*.

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

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

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

[21] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multhop routing in sensor networks," in *Sensys'03*.

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

[23] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

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

[25] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration in wireless sensor networks," *TOSN*, vol. 1, no. 1, pp. 36–72, 2005.

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

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

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

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

[30] P. Levis and D. Culler, "Mate: A tiny virtual machine for sensor networks," in *ASPLOS X*.

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