RoamHBA: Maintaining Group Connectivity In Sensor Networks

Qing Fang[*] Jie Liu[†] Leonidas Guibas[‡] Feng Zhao[§]

ABSTRACT

This paper presents a new group communication scheme, *roamingcast*, for collaborative information processing in wireless sensor networks. Roamingcast enables efficient communication among a subset of mobile terminals in a collaboration group. Unicast and multicast communication can be considered as special cases of roamingcast in which the subset contains one and all group members, respectively. We propose a *Roaming Hub Based Architecture* (RoamHBA, pronounced as 'rumba') as one solution to support roamingcast. We present the distributed construction and dynamic update of a multicast tree, referred as the *roaming hub*. This roaming hub has the property that an average pair of terminals communicate using the hub with only constant degradation in path length compared to the best possible path. We have developed network layer protocols implementing this mechanism and evaluated their performance in comparison with roaming restricted flooding. We simulated our design using NS-2.

Categories and Subject Descriptors

H.1 [**Information Systems**]: [models and principles]; H.4 [**Information Systems**]: [information systems application]

General Terms

Algorithms, Design

Keywords

Sensor networks, Network architecture, Network protocols, Applications of sensor networks

[*]Department of Electrical Engineering, Stanford University, Stanford, CA 94305. Email: jqfang@stanford.edu
[†]Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, U.S.A. E-mail: jliu@parc.com
[‡]Department of Computer Science, Stanford University, Stanford, CA 94305. Email: guibas@cs.stanford.edu.
[§]Palo Alto Research Center. E-mail:zhao@parc.com

1. INTRODUCTION

The collaboration group is a useful abstraction to support collaborative information processing in sensor networks [1]. Maintaining connectivity among a group of roaming agents arises in applications such as collaborative exploration, pursuer-evader games [2], and identity management in multi-target tracking [3]. Maintaining connectivity among a group of mobile agents requires multicast routing support at the network layer.

In this paper, we study an efficient mechanism that supports symmetric multicast routing in a fixed, densely deployed wireless sensor network. In this scenario, a subnetwork is used to connect a group of processes that reside in sensor nodes. While physical node locations are fixed, the processes may migrate from one node to another following physical events being tracked. Therefore, end points of the subnetwork may hop, from time to time, to other nodes in their neighborhood. Thus mobility is implemented by adding or deleting new nodes at end points of the subnetwork. This differs from common mobile ad hoc networks where all physical nodes are mobile and differs from Internet routing, where network processes are mostly fixed. Although it shares common goals with routing in both mobile ad hoc networks and the Internet, routing in wireless sensor networks has its own unique qualities and deserves to be studied. For example, relatively stable sensor neighborhoods form an important characteristic to be exploited in algorithm design.

2. RELATED WORK

There have been significant amount of research on multicast routing in mobile ad hoc networks [4, 5, 6] in recent years. The majority of prior works in ad hoc network multicast have concentrated on building source specific forwarding trees, mesh based multicast forwarding paths or shared multicast trees in mobile ad hoc environments. We are interested in routing algorithms and techniques for building an efficient, shared, many-to-many communication network.

The optimal many-to-many multicast routing structure we consider is a tree in the network connectivity graph that satisfies the graph theoretic properties of a Steiner tree. Given a graph $G = (V, E)$ and a set of terminals K, the Minimum Steiner Tree (MStT) is the subtree of G connecting the K terminals that has the least total weight. Vertices u in V but not in K are also permitted in the subtree. Computing MStTs in general is known to be NP-complete [7]. On-line algorithms proposed for multicast routing in the Internet have all focused on polynomial-time approximations [8, 9, 10]. However, all the MStT polynomial-time approximation algorithms require $O(n)$ global routing tables at each node, where n is the number of nodes in the network. Although recent work [11] has shown that $O(\log n)$ routing table entries at each node can give a bounded approximation to pairwise distances among n nodes, maintaining this information in a distributed manner is still an unsolved problem.

Backbone-based routing in multihop ad hoc wireless networks has been proposed in recent years [12, 13, 14, 15, 16]. All backbone nodes form a dominating set within the network connectivity graph. Prior works have used the backbone mostly for disseminating control information. The dominating set approach may not be well-suited for applications with event locality in a large scale sensor networks. For example, if the sensed activities are confined in a small region within the network, a backbone consisting of nodes that form a dominating set of the whole network is unnecessary. Even within this small region, not all nodes are actually participating in the group communication. Therefore, there is no need to ensure these nodes can be reached within one hop from the backbone, which is what the dominating set method provides. The dominating set approach makes sense in ad hoc networking environment since the assumption there is that all nodes may potentially participate in network communication activities. In sensor networks, networking events are typically caused by events in the physical world. Thus they tend to be spatially localized and temporally correlated. Usually only a small subset of nodes out of a vast network participate in supporting a given task. This subset of nodes may change gradually, as it continuously tracks the physical phenomenon of interest. Therefore, from computational point of view, a dominating set solution is overly expensive. Power efficiency is a critical factor that has to be taken into consideration in designing algorithms for sensor network applications. Unnecessary involvement of sensor nodes for communication purpose should be avoided as energy consumption for inter node communication is high.

Connectivity among roaming agents has been studied in the literature. Luo *el al.* proposed a two-tier data dissemination mechanism in large-scale wireless sensor networks [17]. Their work addresses connectivity issues between a moving source and sinks in a sensor network. Our work differs from this in two aspects: first, our communication end-points are symmetric as opposed to statically designated as sources or sinks; second, their work focuses on solving data dissemination problems, where providing information to sinks that can be at any node in the network is of major concern. Our work focuses on supporting applications that coordinate groups of sensors in the vicinity of multiple external physical events of interest, such as identity differentiation in tracking, collaborative exploration, etc. In such applications, although group memberships change overtime, there are limitations on the range of space where a new member can possibly appear. For example, for identity management in multi-target tracking, while new group members may join in at any time, they will not appear at arbitrary nodes in the network because identity differentiation is only needed when a new target is near another that has already been tracked. Because of this, there is no need to build a system-wide infrastructure for connectivity purposes. Instead, we choose to build a routing tree that as small as possible to support communication needs within the group. Recent work by Blum *el al.* also studied connection services for sensor networks [18]. Their work presented a middleware architecture for coordination services in sensor networks that facilitates interaction between groups of sensors. Our work differs from theirs in that our work addresses routing algorithms at the network layer, while their work is at the transport layer, assuming support from underlying routing infrastructure. Similar to their work, we define and maintain abstract entities to facilitate communication between moving events in the network.

The basic idea for backbone-based routing is to overlay a virtual infrastructure on an ad hoc network to help disseminate control information. From the protocol standpoint, existing ad hoc routing protocols can be classified into three categories: proactive, reactive and the combination of the two. Proactive routing protocols have each host maintain global topology information so that a route can be provided immediately when requested. Protocols in this category suffer from lower scalability and high protocol overhead. Reactive routing protocols produce routes *on-demand*. Each host computes a route for a specific destination only when necessary. Topology changes that do not influence active routes do not trigger route maintenance functions, thus reducing communication overhead. The third category maintains partial topology information in some hosts. However, all these protocols rely on some form of flooding. Proactive protocols rely on flooding for the dissemination of topology update packets. Reactive protocols rely on flooding for route discovery. Flooding suffers from the notorious *broadcast storm problem* [19] that result in excessive redundancy, contention, and collision. These cause high protocol overhead and interference to ongoing traffic. For backbone-based routing, routing protocols are operated over the backbone infrastructure, replacing flooding mechanisms with broadcast over the backbone.

3. ROAMINGCAST

Let us consider the example of multi-target tracking in sensor networks. In order to facilitate in-network processing, a leader-based scheme assigns, for each target in the sensor field, a sensor node as the home for an agent tracking that target [20]. As the targets move, the corresponding agents hop from node to node, following the estimates of the target locations. However, when multiple targets come to close proximity, their signal signatures may mix, which causes the identities of tracks to mix. Since it may be impossible to resolve immediately the target identities, we have to tolerate the identity confusion until they can be resolved later, either because the individual signal to noise ratio becomes high enough, or because other sensing modalities are available. When the confused targets move away from each other and additional information becomes available, a virtual communication route needs to be maintained between them to share the new information in a timely manner. After some time, the identities of multiple targets may become mixed and all the relevant agents need to stay in communication and share new sensing information as it becomes available.

We use the term *roamingcast* for the communication patterns necessary to support such collaborating agents. Formally, given a group of mobile agents G in a sensor network, roamingcast is a group communication scheme that enables communication among any subset S of group members G. Roamingcast is different from multicast, as in multicast $S = G$. Furthermore, since S is not known *a priori* and can change from one request to another, a multicast tree cannot provide routes without incurring excessive overhead in order to build different trees for each data distribution. Within the multicast framework, additional information is needed to differentiate each member in the group. This means that some type of location lookup service or route registration is necessary and calls for stateful implementations.

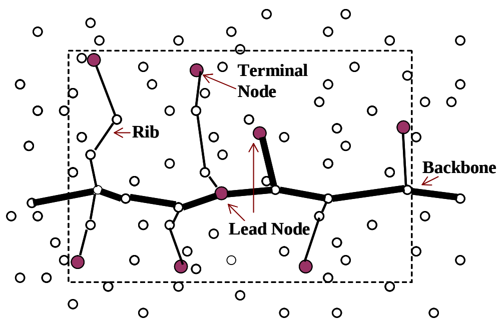

Figure 1. A roaming hub consists of two backbones. The backbone is shown as the darker horizontal and vertical lines. Eight terminal nodes (shaded) are connected to a horizontal active backbone.

We assume that sensors are densely deployed in large scale in an ad hoc fashion. Our work focuses on distributed construction and dynamic updates of a many-to-many communication tree, the roaming hub. An average pair of terminals communicate using the roaming hub with only constant degradation in path length as compared to the best possible path. In addition, we want the structure to have reasonable maintenance overhead. Figure 1 shows an example of a roaming hub structure in a sensor network.

We begin with introduction of the *Roaming Hub Based Architecture (RoamHBA)* outlining the major architectural and functional components in our design. We propose a network layer solution that provides routing services for roamingcast. Network end-to-end control and reliability is handled at the transport layer and is beyond the scope of this paper.

4. ROAMING HUB BASED ARCHITECTURE (ROAMHBA)

Geographic routing [21, 22, 23] has emerged as an effective routing technique in sensor networks. For geographic routing to work, a location service is needed to provide destination locations. In roamingcast, destination locations change from time to time without acknowledging the senders. This leads to our introduction of RoamHBA. We first introduce definitions of some of the architectural and functional entities that will be used in RoamHBA.

Definition 4.1. *An* agent *is a process that acts on behalf of some application entity, a process that performs some functional duty in support of another agent, or a process that realizes protocol semantics.*

A *mobile agent*, is an agent that tracks the moving physical phenomenon of interest that the sensor network is monitoring. It processes and stores application states related to the sensing task at hand. For our work, we want to take out application specific attributes and only keep the ones that are relevant to network layer functionalities, the mobility aspect. The node that a mobile agent resides in is the *terminal node*.

A junction node, is the node at which a terminal node connects to the backbone, possibly through a multi-hop path. A *junction agent*, is an agent representing a mobile agent on the backbone. A junction agent always resides in a junction node, and the two have a 1-to-1 mapping.

A *rib*, is the path consisting of a set of nodes (rib nodes) between a mobile agent and its junction node. It is established by a mobile agent connecting to the backbone via geographic forwarding.

Roaming Hub is the rendezvous entity that all mobile agents communicate with. The *backbone* together with the ribs connecting to it forms the roaming hub. Figure 2 (i) illustrates the components of RoamHBA. Figure 2 (ii) illustrates the logical and physical relations between agents and nodes.

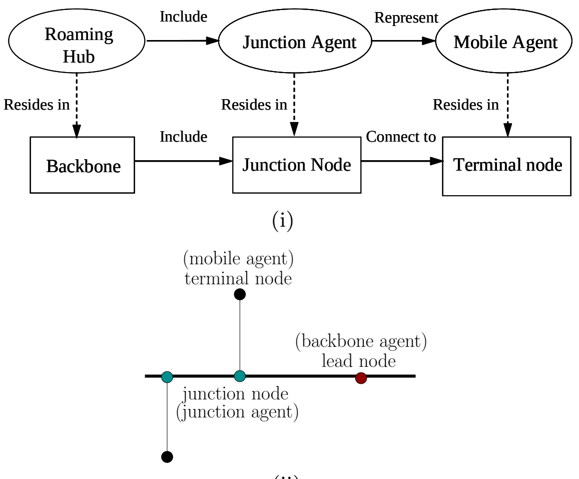

Figure 2. (i) Roaming Hub Base Architecture (RoamHBA); (ii) illustration of the relations between logical agents and physical nodes that agents reside in. The line in bold is the backbone. Logical entities residing in the physical nodes are shown in parentheses next to their corresponding physical nodes. Lead node and backbone agent will be defined in section 5.2.

On the one hand, the Roaming Hub performs dynamic mapping of mobile agents to nodes. It provides group member location service by mapping routes to the agents, and by storing location information on the backbone. In case of unicast, a mobile agent can obtain location information of the other mobile agents from the hub, hence point-to-point geographic routing is possible. On the other hand, the roaming hub serves as the multicast routing tree and actually participates in routing and data dissemination.

5. ROAMING HUB FORMATION

We assume wireless sensor nodes are deployed in the plane, each with a constant communication range. Each node can get its location information either by GPS or other location services [24, 25, 26, 27]. Node density of the network is high enough so that greedy forwarding can always make progress. In case this assumption does not hold due to node power outage or adverse channel conditions, hole bypassing techniques [28] can be used to build 'detours' to overcome the local minimum problem.

5.1 Greedy forwarding in horizontal/vertical direction

Let R be the transmission radius. To greedily construct a horizontal/vertial path passing node $L(x_0, y_0)$, node L picks two of its neighbors that are closest to the points (x_0+R, y_0) and (x_0-R, y_0), say, node $A(x_1, y_1)$ and $B(x_2, y_2)$. Node A then picks one of its neighbors that is closest to point (x_1+R, y_0), while node B picks one of its neighbors that is closest to point (x_2-R, y_0), and so on. The value of y_0 is recorded in the packet to be overlaid by the nodes to regulate the choice of next hop on the backbone branch, so that the path will not drift up or down in the y direction. Exchanging x-coordinates with y-coordinates, we can construct a vertical path accordingly.

5.2 The backbone

Definition 5.1. *A bounding box, is a virtual rectangle defined by dual pairs of x coordinates (x_{min}, x_{max}) and y coordinates (y_{min}, y_{max}), such that* x_{min}=min$(x_1, x_2, ...x_k)$, x_{max}=max$(x_1, x_2, ...x_k)$, y_{min}=min$(y_1, y_2, ...y_k)$, y_{max} = max$(y_1, y_2, ...y_k)$. *k is the number of mobile agents. The bounding box state, includes these dual pairs and the agent ID associated with each of them.*

Intuitively, the bounding box is the smallest rectangle that 'covers' all the mobile terminals in the group. The bounding box state is used to decide on the location and span of the backbone, as well as to decide which backbone (the horizontal or the vertical) mobile agents should connect to. The bounding box state is propagated along the backbone. Every backbone node stores a copy.

Backbones can have different shapes. The principles we have used in deciding the shape of backbone are: a) maximal sharing of common paths among the mobile agents, b) easy maintenance, and c) easy inter-agent connections through greedy forwarding. We chose to use a backbone that is cross-shaped consisting of a horizontal part and a vertical part. We refer to these as *the horizontal backbone* and *the vertical backbone*. The two backbones are connected, each consisting of a set of sensor nodes (backbone nodes) connected via a multi-hop path.

The active backbone The backbone oriented in the direction of the longer side of the bounding box is the active backbone. To provide hysteresis and avoid instabilities, switching the backbone only occurs after the current active backbone is oriented in the direction of the shorter side of the bounding box and the ratio of the longer side to the shorter side is larger than $c(c > 1)$. In Figure 1, the horizontal backbone is active.

Backbone location Both the horizontal and vertical backbones are located such that there are approximately equal numbers of mobile agents on both sides of the backbone.

Property 5.1. *Given a set of n nodes v_i, $i = 1, 2, \cdots, n$, all connecting to a horizontal line l via vertical lines, the optimal y location for line l is the median of the y-coordinates of all v_i such that the total vertical line length is shortest. The claim also holds if we exchange horizontal with vertical. Proof of this property is in appendix 13.1.*

From this point on we confine our discussion to the horizontal backbone. The vertical case is completely analogous.

Each backbone is associated with a *backbone agent*. A backbone agent resides in the *lead node*, which is also a backbone node. The lead node is chosen during the previous backbone migration so that the lead node is a terminal node with its y-coordinate chosen to be close to the median of all the terminal nodes' y-coordinates. A backbone agent is responsible for establishing the backbone at the beginning of a backbone migration. After the migration, the backbone agent decides when a migration is needed. Before a new migration takes place, the backbone agent chooses the next lead node and migrates to that lead node, repeating the above process. Section 6.3 describes this in detail.

Backbone span The horizontal/vertical backbone spans the full length of the bounding box in the horizontal/vertical direction plus a parameter x_c/y_c. Determination of x_c and y_c will be discussed in section 6.1.

5.3 Mobile agents connecting to the backbone

When a packet is forwarded towards/away from the backbone, it is referred to as travelling upstream/downstream.

Mobile terminals connect to the active backbone in the vertical or horizontal direction depending on which backbone is active (vertical for horizontal backbones and vice versa), which in turn depends on the aspect ratio of the bounding box. A path to the backbone is constructed by an agent sending a path finding packet upstream towards the backbone. The packet is forwarded at each hop using geographic greedy forwarding described in section 5.1. Such a packet carries agent state including its ID and location. When the packet reaches the backbone, a junction agent is created at the junction node and agent state is stored there. A question that arises is if it is possible for two paths to cross each other without the crossing being detected.

Property 5.2. *When an expanding path crosses an existing path, there will always be at least one node on the existing path that is within the radio range of a node on the expanding path. Hence, crossing event can always be detected. Proof is in appendix 13.2.*

Multicast path length

Property 5.3. *The sum of all pairwise terminal distances using the roaming hub is at most:* $2\sqrt{2}\times$*(the sum of all pairwise unicast distances). In other words, an average pair of terminals communicating using the roaming hub has only constant degradation in path length as compared to the best possible path.Proof for this property is in appendix 13.3.*

6. ROAMING HUB UPDATES

In the previous section we discuss the formation of the roaming hub structure. Ultimately, due to agent mobility, routes provided by the roaming hub degenerate over time. Therefore, it is necessary to update the roaming hub so that property 5.3 still holds. The way to perform such updates in a distributed fashion is one of the major contributions of this paper. We organize our description of dynamic updates by logic network entities that are responsible for these updates.

6.1 Mobile agent

Maintaining agent connectivity to the backbone Figure 1 shows a snapshot of the multicast path tree when each mobile agent had just connected to the backbone. When a mobile agent moves after some time, its path to the backbone can be extended by adding the new terminal node to its path. These are strictly incremental changes occurring locally. It is possible that a mobile agent crosses its own path as it moves about. Our protocols handles this by removing loops from an existing route. Figure 3 illustrates this idea.

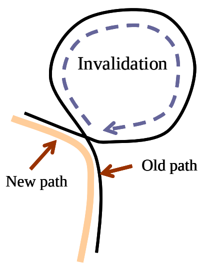

Figure 3. An example of erasing route loops due to agent mobility. When a route crossing is detected, an invalidation packet is sent out to invalidate the downstream route starting from where the detection is made until the invalidation packet returns to the same node .

Route updates using distance based rule When an agent's new location has significant displacement away from its prior location where the last direct connection to the backbone was made, another direct connection to the backbone is needed. There are different ways to measure the 'significance' of displacements. Referring to Figure 4, let node s be the node that the agent resides when the previous connection to the backbone was made. Let node t be the node that the agent currently resides. Let r be the distance from node s to node t. Let d be the distance of node s from the backbone. We used the ratio r/d to measure the relative extent of agent displacement and referred to it as the *free ratio*. When this ratio reaches a constant $\rho(\rho < 1)$, the mobile agent should initiate a new connection to the backbone. We call this rule the *distance-based rule*. The region inside the circle of radius $\rho \times d$ is referred as the *free region*.

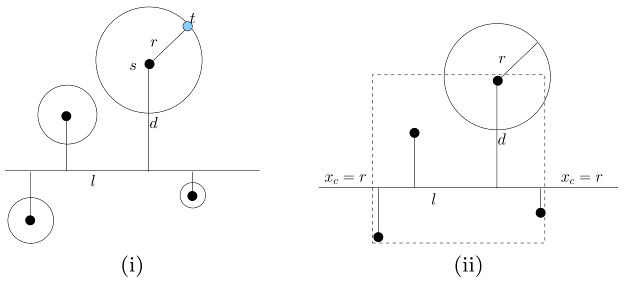

Figure 4. Line l represents the backbone. black nodes represent terminal nodes. (i) illustrates free regions as circles; (ii) illustration of the relation between the bounding box and the backbone span .

With this scheme, for given mobility, the farther away an agent is from the backbone, the less frequently it updates its path to the backbone. There are three reasons for this. First, to maintain property 5.3, the ideal location for the horizontal/vertial backbone is at the median y or x location. When agents are far away from a backbone, we can care less about if they have crossed the backbone or not. Second, the farther away an agent is, the higher cost it will incur to reconnect to the backbone. Third, the farther away the agent, the more detour it can tolerate before its path to the the backbone degrades significantly.

Determining and adjusting the backbone span We focus on the horizontal backbone. In section 5.2, we stated that the length of a horizontal backbone is approximately the length of the bounding box in x direction plus an additional length x_c. The extra length of x_c is necessary because the backbone must be long enough so that when a mobile agent makes a new connection to the backbone via geographic forwarding, the connecting path will be guaranteed to cross the backbone. This is illustrated in Figure 4 (ii). As every backbone node stores a copy of bounding box state, the largest y offset from the backbone y-coordinate among all the mobile agents, d in Figure 4, can be computed locally. If $x_c = d \times \rho$, a direct connection from any of the mobile agents to the backbone will be guaranteed to cross the backbone.

When a new bounding box update is propagated on the backbone, a backbone node decides if it should remain on the backbone based on its calculations of the backbone end points according to the rule introduced above. If the decision is 'no', this node first, continues propagating bounding box update to its next neighbor on the backbone if it has one; second, clears its state as a backbone node. If the decision is 'yes', it updates its local copy of the bounding box state and passes on the update packet to its next neighbor on the backbone. If this backbone node is at the end of the backbone, it is responsible for extending the backbone until it extends beyond the end point of the backbone based on the calculation. Extending backbone is accomplished by geographic forwarding discussed in section 5.1.

6.2 Junction agents

When a mobile agent's new connection reaches the backbone, a new junction agent is created at the new junction node. The new junction agent does the following: First, it sends out a packet along the backbone to invalidate the old junction agent that belongs to the same mobile agent. Before the old junction agent expires, it sends out an invalidation packet to invalidate the old rib associated with itself. Figure 5 illustrates the invalidation process. Second, the new junction agent decides if the existing bounding box state needs an update.

Updating the bounding box To minimize network traffic, we want to localize control information processing as much as possible. This is the main reason for summarizing agent locations using bounding box state and replicating the state along the backbone. Whenever an agent announces a new location by making a new connection to the backbone, the junction node needs to decide if the bounding box state needs an update. The criterion is whether or not the existing bounding box is still the smallest rectangle that covers all the terminal nodes. If 'yes', no update is needed. Otherwise, the junction node computes the new bounding box state and broadcasts it over the backbone. There is no need

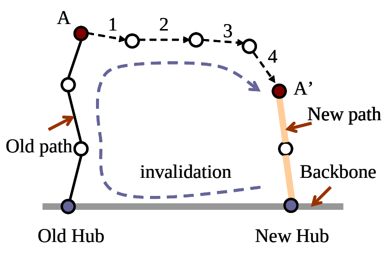

Figure 5. The agent was at node A when the previous connection to the backbone was made. After some time, the agent hops to node A'. The agent updates its registration on backbone by making a new connection to the backbone. The new junction node informs the old junction node. The old junction node invalidates the old path.

to enforce global ordering in these distributed updates because all conflicts can be resolved locally by attaching each update with agent ID and update sequence number for that agent.

6.3 Backbone agent

A backbone agent resides in the lead node of the backbone. Ideally, we would have equal numbers of mobile agents on each side of a backbone. However, frequent backbone migrations, although yielding smaller total path length, incur significant control packet overhead and disruption to ongoing data traffic. In our simulation, the rule for backbone migration is that when the number of agents on one side is a fraction, γ, of the total number of agents, the backbone will be migrated. The choice of the value of γ reflects the tradeoff point between path length and roaming hub update overhead.

After the initial construction of the backbone, the backbone agent is responsible for monitoring the number of mobile agents on each side and deciding when a backbone migration is needed. Whenever a mobile agent crosses the backbone, the crossing event can be detected by backbone nodes. Backbone nodes report such events to the backbone agent so that the backbone agent can update its count of agents on each side. When the backbone agent decides that a migration is necessary, it polls all the junction agents to get their mobile agents' locations. It identifies the new lead node and creates a new backbone agent at the lead node. The new backbone agent initiates a new backbone construction, and notifies all backbone nodes on the old backbone of the migration. A *death timer* is set at each backbone node on the old backbone. All junction nodes request their terminal nodes to connect to the new backbone. All nodes on the old backbone will time out after the death timers expire. When junction nodes time out, they also invalidate their old ribs of their terminal nodes.

7. ROUTING OVER THE ROAMING HUB

Roamingcast routing over the hub is very simple. All mobile agents send packets upstream towards the backbone. Packets are broadcast on the backbone. Whenever a packet encounters the junction node of one of its destined mobile agents, the packet is copied and forwarded downstream following that rib route to the mobile agent.

For unicast, the sender can request the current location of the destined mobile agent by querying the junction agent of that agent on the backbone, and then use geographic forwarding to reach the destination.

Due to the broadcast nature of the wireless medium, the route mapped out by a rib may not be the best route to distribute data packet. Figure 6 shows one example.

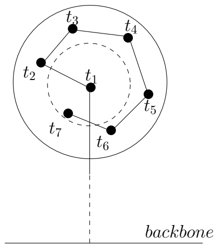

Figure 6. Data packets use shortcuts whenever possible. The dotted circle shows node's radio range. The solid circle shows the free region. Node t_1 can directly reach node t_7. Hence, a data packet can be delivered directly between them instead of following the route.

8. PERFORMANCE CONSIDERATIONS

Backbone-based approaches for ad hoc network routing fall, in general, within the category of proactive routing. The introduction of a virtual bounding box enables us to make backbone construction and updates reactive to mobile agents' geometric configurations. When a mobile agent is involved in some data exchange, the data packet can piggyback the agent's location information to its junction agent. When no data exchange takes place for a mobile agent, the agent only makes its location known to its junction agent when it establishes a new route to the backbone via geographic forwarding following the distance based rule. At the junction node, bounding box parameters may be changed based on the new location reported to the junction agent. If such changes take place, it will in turn affect backbone location and/or backbone length. Roaming hub updates are necessary only because we want to provide good routes for mobile agents, but at a price of taxing on network resources for such route updates. The tradeoff between good routes and low update cost depends on application requirements. For example, if an application requires low latency communications, good routes have to be maintained even with high maintenance overhead. We will discuss issues relating to energy efficiency.

8.1 Amortizing roaming hub updates over data communications

Short routes for packet deliveries reduce total internode communication cost. The more frequently data communications occur, the more advantageous good routes will be. If data exchange happens frequently enough, the maintenance cost may well be justified by savings on routing cost and route length. For a mobile agent not communicating frequently, its location changes may not be of much concern to the rest of the group. Hence data exchange frequency needs to be factored into the decision process along with distance based rule in determining when the mobile agent should reestablish a new route to the backbone. This leads to route updates using a *communication based rule*. The communication based rule is the following: an agent does not initiate an update on its path to the backbone without upstream/downstream traffic from/to the agent for some time interval τ. The interval τ is a constant that is twice as large as the average data communication interval.

8.2 Mobility pattern

Mobility patterns have a big impact on the performance of roaming hub based routing. The amount of maintenance overhead is largely determined by agents' moving speed as well as mobility patterns. The roaming hub based routing is well equipped to handle a scenario where mobile agents' movement has some locality, such as a random walk with or without a drift. The maintenance overhead of roaming hub comes mainly from backbone and rib migrations. The roaming hub provides a mobile agent with a 'mapping service' about the other mobile agents. The higher the resolution of the map, the more costly it is. We introduced a distance based rule and a communication based rule to 'obscure' the view presented to the junction agent. We further obscure the view presented to the other junction agents by only presenting them the virtual bounding box. The hierarchical filtering of information is intended to encapsulate and localize structural updates as much as possible. With some level of locality in agents' movement pattern, this hierarchical filtering mechanism becomes very effective.

9. SIMULATIONS

We assume all sensor nodes have the same computation, communication and storage capabilities. Each node maintains a neighbor list storing coordinates of its neighbors.
Soft states:

1. Rib nodes maintain a dynamic array of ports. A port is created for one agent that has its rib through the node. One port is associated with one agent's connection to the backbone at the node. At each port, upstream and downstream neighbors are recorded. These are used for forwarding purpose. There are no routing information kept in the node. Introduction of port is necessary because multiple agents may cross each other's path therefore different network 'flows' need to be maintained at a rib node. Ports are also needed for invalidating old paths. A port takes 3 integers of memory.
2. Backbone nodes store bounding box parameters (4 doubles).
3. Mobile agents store backbone locations (2 doubles).
4. Junction agents store their mobile agent's location (2 doubles).
5. Backbone agents store numbers of agents on each side (2 integers)

Agents as logical entities move from node to node. It is possible that one node has multiple agents at the same time.

9.1 Simulation setup

RoamHBA is implemented at the network layer using NS-2. Geographic forwarding as described in section 5.1 is used in constructing the roaming hub. In our simulations, each node learns its one-hop neighborhood and store this information locally. A control packet header includes the direction the packet travels. Based on this information, a node n, can decide the next hop for this packet, and append this information to the packet header. All neighbors receive this packet but only the designated neighbor processes and possibly overlays the packet based on its local state and information encoded in the packet.

802.11b within NS-2 was used for MAC support. We modified the channel grid keeper to speed up the simulations so that networks with more than 300 nodes can be simulated in reasonable amount of time.

9.2 Evaluation

The topology of the scenario used is as follows: 1800 sensors placed on a perturbed grid in a 250m by 250m 2-D space. Each sensor node has transmission radius of 15 meters. There are seven mobile agents scattered across an area of about one sixth of the 2-D space. We use the total number of transmissions to approximate the total energy consumption.

9.2.1 Tradeoffs between the data packet rate and the frequency of roaming hub updates

We are first interested in knowing, for a fixed data rate, how the free ratio ρ (defined in section 6.1) affects the total data packet transmissions and the control packet transmissions. Figure 7 shows that the more frequent the route updates (the smaller the ρ) are, the less transmissions are necessary to deliver data to destinations, but at a price of more control packets. There is an optimal range for ρ that minimizes the total number of transmissions.

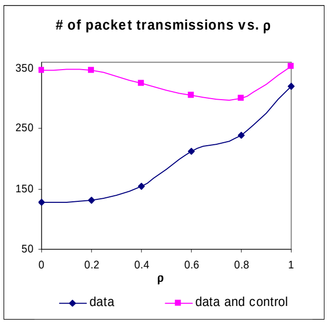

Figure 7. Number of packet transmissions vs. ρ

We next study, for a fixed data rate, how the criteria (γ value defined in section 6.3) for migrating backbone affect total data packet transmissions and control packet transmissions. Figure 8 shows that frequent backbone location adjustments help shortening total data path but drastically increase control overhead. Allowing some extent of 'imperfections' in the backbone location can reduce overall energy consumptions. For the simulated scenario, $\gamma = 2/7$ is optimal.

Figure 9 replicates the total number of packet transmissions curve from Figure 8. In addition, we vary the data packet rate and plot the numbers of total packet transmissions side by side in this figure. As data rate increases, the benefits of maintaining good routes dominate backbone migration overhead. It is evident from the figure that the curves are 'rolling' slowly counterclockwise. This means that more responsive backbone adjustments benefit high data rate network in terms of overall energy efficiency.

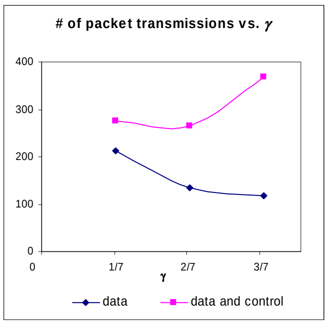

Figure 8. Number of packet transmissions vs. γ

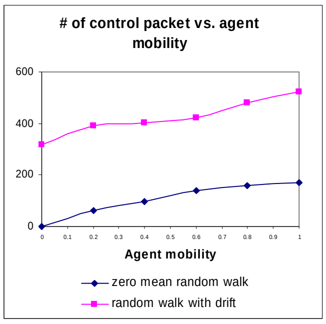

Figure 9. Number of control packet vs. agent mobility

9.2.2 Comparison with roaming restricted flooding

roaming restricted flooding can be another possible scheme to support group communications in sensor networks. It is used to be compared with our scheme. The basic idea is to let all mobile agents share the bounding box state through flooding. All agents also communicate with one another by flooding inside the bounding box. When updates on the bounding box are necessary, a mobile agent notifies other agents by flooding the restricted region. Therefore, roaming restricted flooding is essentially a restricted flooding with the restricted region adapting to agents locations and mobility.

Roaming restricted flooding is an appropriate scheme for comparison because it adapts its flooding area to be as small as enough to cover all the mobile terminals. This is important for the type of applications we consider, namely, collaborative group communication in a large network. Any scheme that does not adapt the scale of its communication subnetwork to mobile terminal locations will potentially perform poorly when mobile terminals do not spread out across the whole network.

We ignore the cost of maintaining the bounding box state across the mobile agents, assuming it is insignificant compared with data dissemination cost. Therefore, cost in restricted flooding only comes from data disseminations. If S is the area of the bounding box, flooding within this region has $O(S)$ cost for networks with constant node density. Cost for roaming hub based solutions comes from both data disseminations and roaming hub maintenance overhead. However, data dissemination cost is only $O(\sqrt{S})$. Obviously, the more frequently data exchange happens the more advantage the roaming hub based solution has.

We simulate the two schemes using the same topology scenario. Assume a single data source with constant packet rate. Take $\rho = 0.6, \gamma = 2/7$, and probability of each mobile agent to leave the current terminal node to be 0.5. Total number of packet transmissions are counted over 5 simulation runs, each with 1900 iterations. The average number of control packet transmissions per iteration for using roaming hub is 15.5. The average number of data packet transmissions is 25.3. For roaming restricted flooding, the average number of data packet transmissions is 594.2. Thus, if there is data exchange in $1/15$ of the time, roaming hub is a better solution than roaming restricted flooding. This conclusion is drawn based on exclusion of control cost for maintaining the bounding box for restricted flooding.

To summarize, RoamHBA is a more scalable solution than restricted flooding. When mobile agents span a large area, even restricted flooding is too expensive. Therefore, a more sophisticated routing service is necessary.

10. CONCLUSIONS

This paper studies a new way of networking, the Roamingcast, which arises from collaborative information processing in wireless sensor networks. We proposed RoamHBA as one of the solutions to support group connectivity. We demonstrated that for dense network where mobile agent's mobility has some locality, RoamHBA is both effective and practical.

11. ACKNOWLEDGEMENTS

The authors would like to acknowledge support by a grant from the DARPA Software for Distributed Robotics program under a subcontract from SRI, the DARPA Sensor Information Technology Program under contract F30602-00-C-0139, NSF grant CCR-0204486, ONR MURI grant N00014-02-1-0720, and a grant from the Stanford Networking Research Center.

12. REFERENCES

[1] J. Liu, M. Chu, J. Liu, J. E. Reich, and F. Zhao, "State-centric programming for sensor and actuator network systems," *IEEE Pervasive Computing Magazine*, 2003, to appear.

[2] Murat Demirbas, Anish Arora, and Mohamed Gouda, "A pursuer-evader game for sensor networks," in *Sixth Symposium on Self-Stabilizing Systems (SSS'03), San Francisco, CA, LNCS 2704*. June 2003, pp. 1–16, Springer.

[3] J.Shin, L.Guibas, and F.Zhao, "Distributed group management for track initiation and maintenance in target localization applications," in *IPSN*, 2003.

[4] J.G.Jetcheva and D.B.Johnson, "Adaptive demand-driven multicast routing in mult-hop wireless ad hoc networks," in *Proc. of the ACM Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc)*, October 2001.

[5] L.Ji and M.S.Corson, "Differential destination multicast-a manet multicast routing protocol for small groups," in *IEEE INFOCOM*, April 2001.

[6] R. Boivie, N.Feldman, Y.Imai, W.LIvens, D.Ooms, and O.Paridaens, "Explicit multicast (xcast) basic specification. internet draft (work i progress)," in *draft-ooms-xcast-basic-spec-04.txt, Internet Engineering Task Force*, January 2003.

[7] R. Hwang, D. Richards, and P. Winter, “The steiner tree problem,” *Annals of Discrete Mathematics*, 1992.

[8] Fred Bauer and Anujan Varma, “Aries: A rearrangeable inexpensive edge-based on-line steiner algorithm,” *IEEE Journal of Selected Areas in Communications*, July 1995.

[9] F.Bauer and A.Varma, “Distributed algorithms for multicast path setup in data networks,” in *IEEE GLOBECOM*, November 1995.

[10] J. Kadirire and G. Knight, “Comparison of dynamic multicast routing algorithms for wide-area packet-switched (atm) networks,” in *IEEE INFOCOM*, 1995.

[11] J.Gao and L.Zhang, “Well-separated pair decompositionfor the unit-disk graph metric and its applications,” in *35th annual ACM symposium on theory of computing*, 2003.

[12] A.Amis and R.Prakash, “Load-balancing clusters in wireless ad hoc networks,” in *Proc. 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, 2000.

[13] P.Sinha, R.Sivakumar, and V.Bharghavan, “Enhancing ad hoc routing with dynamic virtual infrastucutes,” in *IEEE INFOCOM*, 2001.

[14] J.Wu and H.Li, “On calculating connected dominating set for efficient routing in ad hoc wireless networks,” in *the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Comunications*, 1999.

[15] B.Das and V.Bharghavan, “Routing in ad hoc networks using minimum connected dominating sets,” in *ICC’97*, 1997.

[16] J.Gao, L.Guibas, J.Hershberger, L.Zhang, and A.Zhu, “Geometric spanner for routing in mobile networks,” in *MobiHoc*, 2001.

[17] Haiyun Luo, Fan Ye, Jerry Cheng, Songwu Lu, and Lixia Zhang, “TTDD: Two-tier Data Dissemination in Large-scale Wireless Sensor Networks,” *ACM/Kluwer Mobile Networks and Applications (MONET), Special Issue on ACM MOBICOM*, 2003.

[18] B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, S. Son, and J. Stankovic, “An entity maintenance and connection service for sensor networks,” in *The First International Conference on Mobile Systems, Applications, and Services (MOBISYS ‘03), California*, May 2003.

[19] s. Ni, Y. Tseng, Y.Chen, and J. Sheu, “The broadcast storm problem in a mobile ad hoc network,” in *MOBICOM*, August 1999.

[20] F. Zhao, J. Shin, and J. Reich, “Information-driven dynamic sensor collaboration,” *IEEE Signal Processing Magazine*, vol. 19, no. 2, pp. 61–72, 2002.

[21] B. Karp and H. Kung, “Gpsr:greedy perimeter stateless routing for wireless networks,” in *MobiCom*, 2000.

[22] Y. Yu, R. Govindan, and D. Estrin, “Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks,” Tech. Rep., UCLA/CSD-TR-01-0023, 2001.

[23] M.Mauve, J.Widmer, and H.Hartenstein, “A survey on position-based routing in mobile ad hoc networks,” *IEEE Network Magazine*, vol. 15, no. 6, pp. 30–39, November 2001.

[24] Jeffrey Hightower and Gaetano Borriello, “location systems for ubiquitous computing,” *IEEE Computer*, vol. 34, no. 8, pp. 57–667, August 2001.

[25] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava, “Dynamic fine-grained localization in ad-hoc networks of sensors,” in *Proc. MobiCom*, 2001, pp. 166–179.

[26] Andreas Savvides and Mani B. Strivastava, “Distributed fine-grained localization in ad-hoc networks,” *IEEE Transactions of Mobile Computing*, 2003.

[27] A.Ward, A.Jones, and A.Hopper, “A new location technique for the active office,” *IEEE Personnel Communications*, vol. 4, no. 5, pp. 42–47, October 1997.

[28] Q.Fang, J.Gao, and L.Guibas, “Locating and bypassing routing holes in sensor networks,” in *IEEE INFOCOM*, March 2004.

13. APPENDIX

13.1 Proof for property 5.1

Given a set of n nodes v_i, $i = 1, 2, ...n$. They all connect to a horizontal line l via vertical lines. See Figure 10 (i). We prove that the optimal y location for line l is the median of the y-coordinates of v_i so that the total length the vertical connecting lines is minimized.

Suppose that the i-th node v_i is at location (x_i, y_i). If y is the location of the horizontal line l, then the total length of vertical lines is:

$$\sum_{i=1}^{n} |y - y_i|$$

We can assume that $y_1 \le y_2 \le ... \le y_n$. if $y_k \le y \le y_{k+1}$, then the total length of vertical lines is

$$\sum_{i=1}^{k} (y - y_i) + \sum_{i=k+1}^{n} (y_i - y)$$

This is a piecewise-linear function. Taking derivative, we get:

$$k - (n - k) = 2k - n$$

This means that y is optimal when there are $n/2$ nodes below or on the line, and $n/2$ nodes above or on the line. Therefore, the total length is minimized by locating l at the *median* of the y-coordinates. The same argument holds for a vertical partition line.

13.2 Proof for property 5.2

We prove that when an expanding path crosses an existing path, there is at least one node on the existing path that is within the radio range of one of the node on the expanding path. Therefore, crossing event can always be detected.

We prove by contradiction. Referring to Figure 10 (ii), assume node u,v are on an existing path and edge uv is the edge to be crossed. Node x initiates a new path northbound using the greedy forwarding method as discussed in 5.1. It finds the next hop node y. Similarly node y picks the next hop to be node z. Assume neither node u nor v is within

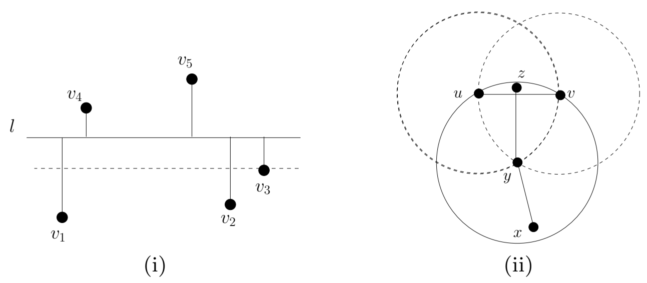

Figure 10. (i) The dotted line depicts the optimal location for horizontal line l; (ii) At least one of node u and node v is within distance 1 from at least one of node y and node z. All circles are unit disks.

radio range of node y, or node z. This means that both node y and z must lie outside of the two dotted circles. However, if edge yz are to cross edge uv, the only possible scenario is that edge $|yz| \geq \sqrt{3}$. This contradicts with the fact that z is within the radio range of node y. Therefore, the distance between them is at most 1. Hence, the path finding process always terminates.

13.3 Proof for property 5.3

We prove that the sum of all pairwise terminal distances using the roaming hub is at most: $2\sqrt{2}\times$(the sum of all pairwise unicast distances).

The $\sqrt{2}$ factor comes from replacing the Euclidean (L_2) distance with rectilinear (L_1) distance. Assume that the backbone is horizontal. The backbone paths and the optimal paths have horizontal portions of the same exact length, so the only difference comes from the use of the ribs in the backbone connections. For simplicity, let us assume that we have k terminals above the backbone with rib lengths a_i, $i = 1, 2, \cdots, k$, and k terminals below the backbone with rib lengths b_i, $i = 1, 2, \cdots, k$.

Then clearly, the vertical portion of the sum over all pairwise backbone paths is

$$(2k-1)(\sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i),$$

as every node connects to $2k-1$ others and each node's rib length is used this many times.

For terminal pairs that are across the backbone, the backbone and the true shortest path have identical contributions vertically. Since every terminal can be paired with at least k nodes on the other side, the true L_1 length sum over all pairs includes

$$k(\sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i).$$

Thus the total backbone connection path length over all pairs is at most twice the L_1 sum of distances between all pairs.