

# Beacon Location Service

## A Location Service for Point-to-Point Routing in Wireless Sensor Networks

Jorge Ortiz, Chris R. Baker, Daekyeong Moon, Rodrigo Fonseca, Ion Stoica  
Computer Science Division, University of California, Berkeley  
Berkeley, CA, USA  
{jortiz, crbaker, dkmoon, rfonseca, istoica}@eecs.berkeley.edu

### ABSTRACT

In this paper we present Beacon Location Service (BLS): a location service for beacon-based routing algorithms like *Beacon Vector Routing* (BVR) [8] and S4 [19]. The role of a location service is to map node names to topologically meaningful addresses that can be used for routing. We evaluate an implementation of BLS that works on top of BVR. BLS resolves the destination node's name to BVR coordinates and then uses BVR to route the source message to the destination node.

Together, BVR with BLS offer a complete solution for scalable point-to-point routing in sensor networks. We describe our design and its implementation and evaluate BLS in simulation and on a sensor network testbed. Our results show it can sustain a high look-up and routing success rate (*greater than 90%* on most network sizes we tested) over various send rates and network sizes. We also compare our results to two existing point-to-point routing schemes for sensor networks, and show that BLS+BVR performs comparably to state of the art point-to-point routing protocols. Finally, we demonstrate a simplified distributed hash table application implemented over BLS.

**Categories and Subject Descriptors:** C.2.1 [Network Architecture and Design]: Wireless Communication

**General Terms:** Design, Performance.

**Keywords:** sensor networks, location service, point-to-point.

## 1. INTRODUCTION

While one-to-many and many-to-one routing techniques provide much of the routing needs for wireless sensor networks, point-to-point routing is still important for many applications. One example is data-centric storage [23], where data is stored in specific locations in the network based on its name<sup>1</sup>. Another use is for management and control of the network, when commands need to be sent to specific nodes. This type of routing is also used in the pursuer-evader game (PEG) described in [6], where the nodes of the wireless sensor network build a tracking tree with the root as the evader.

<sup>1</sup>In the original proposal a geographic hash table is suggested but, if the network can route directly to names, this primitive can be used instead.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IPSN'07, April 25-27, 2007, Cambridge, Massachusetts, USA.  
Copyright 2007 ACM 978-1-59593-638-7/07/0004 ...\$5.00.

The pursuer may use a point-to-point routing scheme to traverse the tracking tree when searching for the evader. Finally, networking stacks for sensor networks like ZigBee [2] and 6LowPAN [16] will require robust point-to-point routing algorithms.

Point-to-point routing can be implemented in many ways, with specific tradeoffs in scalability. One particular class of routing algorithms that scale well, because nodes only need to know the addresses of their neighbors, are coordinate-based schemes, either geography-based, real coordinates (*e.g.* [13, 15]), or topology-based, virtual coordinates (*e.g.* [22, 20, 8]). However, a source node needs to obtain the coordinates of a destination before sending a message to it. Moreover, virtual coordinates may change over time as connectivity and node placement changes temporally.

*Beacon Vector Routing* (BVR)[8] is an existing coordinate-based point-to-point routing scheme for wireless sensor networks that has guaranteed delivery, is robust with respect to a dynamic network, and is simple and efficient. In BVR, nodes have topology-derived virtual coordinates, defined as a vector of distances to a subset of the nodes called beacons. Each beacon is the root of a minimum spanning tree and is chosen randomly across the network. Routing is accomplished by using greedy forwarding. The next hop is determined by a distance function over the current and destination node's coordinates. On occasion, the forwarding mechanism reaches a local minimum for the distance function. At that point BVR resorts to routing the message to the beacon nearest the destination. This is known as fall-back mode. If the message reaches the beacon, BVR resorts to scoped flooding to guarantee delivery.

BVR sets up a virtual coordinate system similar to Landmark Routing [26]. In Landmark Routing, routers in the network are assigned a coordinate vector as an address and routing is done over the virtual coordinate space with greedy forwarding. The coordinate vector consists of a set of router IDs that range from 0 to  $H$  hops from the destination router. When a source sends data to a destination, it finds the router ID in the destination coordinate vector that is closest to the destination and sends the packet to that router; that router forwards the packet to the next router in the vector until the destination is reached. The last router ID in the coordinate vector is the router ID of the destination (since it is 0 hops from itself). Routers that are in every nodes' routing table are considered global landmarks. In BVR, beacons are global landmarks and BVR coordinates consist of  $B$  components where the  $B$ th component is the hopcount from the node to the  $B$ th beacon for some  $n \leq B$  and  $B$  is the number of beacons in the network.

Like Landmark Routing, BVR cannot route without a location service. In this paper, we address this problem directly with the Beacon Location Service (BLS) – a location service which maps application-specific identifiers (node names) to virtual coordinates (node addresses). BLS assumes a flat namespace for identifying

nodes in the network as well as a topologically meaningful address space that identifies a node's location. It does not assume that the address space, or the name space, can be decomposed hierarchically. BLS is different from previous schemes for two main reasons: first, it is adapted to the constraints of a sensor network where memory capacity, processing power, and communication bandwidth are factors limiting the design and implementation that are not present or as severe in other systems such as the Internet or mobile ad-hoc networks. Second, because of the particularities of BVR, BLS takes advantage of the fact that all nodes know how to reach all beacons. Lastly, because BVR coordinates are derived from network topology, they may change over time and there is an added level of complexity in maintaining appropriate state with minimal overhead as opposed to other coordinate-based schemes (*i.e.* those that use static geographic coordinates).

Given the assumptions that guided its design, BLS not only solves the address resolution problem in BVR, but can fulfill this need in other point-to-point routing schemes. The recently proposed Small State and Small Stretch Routing Protocol [19] can also make use of BLS as nodes are identified with unique ID's, node addresses are topologically configured, and there is also a set of globally known beacons.

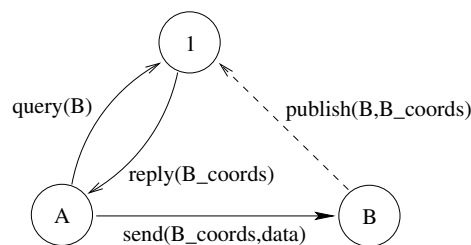
The remainder of this paper is organized as follows: related research efforts are discussed in §2. Following this, we discuss the design (§3) and implementation (§4) of BLS. We present our evaluation in §5, and §6 concludes the paper.

## 2. RELATED WORK

There are several research efforts focused on point-to-point routing in ad-hoc wireless sensor networks. Each proposal can be categorized into a specific type of scheme: shortest path based (*e.g.* AODV [5], DSDV [21]), geographic coordinate-based (*e.g.* GPSR [13], GOAFR+ [15]), virtual coordinate-based (*e.g.* No-Geo [22], GEM [20], BVR [8]), hierarchical (*e.g.* Landmark [26]), and DHT-Based (*e.g.* VRR [4]). Of these, all but the shortest path and the DHT schemes need a location service to route to names.

One way to determine a destination node's location is through a service that relies on flooding of information. There are many works that take this approach (*e.g.* DREAM [3], and LAR [14]). In general, flooding is not scalable for large distributed systems; several studies have shown a large variance in broadcast reliability in low-power networks [24, 27, 28]. We believe that a location service which relies on flooding has too high a cost for the limited resources of sensor networks. Therefore, we do not pursue this class of routing/location service protocol further in this section. However, we will revisit this in our evaluation when we compare our location service with the ZigBee [2] standard routing protocol, AODV. The remainder of this section will focus on work related to location services not reliant on flooding. We cite three representative examples: VRR, GLS, and Landmark.

The recently proposed Virtual Ring Routing (VRR) [4] scheme borrows from overlay routing algorithms in Distributed Hash Tables (DHT). VRR does not require a location service. All node identifiers are mapped to a logical ring of identifiers. When a sender wants to send to a destination node, the packets are routed using these identifiers over the logical ring. To maintain the ring, each node keeps paths to its two logical successors and two logical predecessors in the ring. Packets make greedy progress throughout the network by minimizing the difference between the next hop's ID and the destination node's ID until the packet reaches the destination. VRR is quite effective in providing scalability in point-to-point routing for sensor networks. Although different from our location service it is comparable in flexibility, workloads, overhead,



**Figure 1: Rendezvous scheme for the Beacon Location Service. Node A is sending to node B; each rendezvous at beacon 1.**

and routing success rate. In the evaluation section we address these comparisons directly through experimentation. We refer the reader to [8] for more details on point-to-point routing related work, and now focus on work related to location services.

The Grid Location Service (GLS) [18] is a scalable location service for geographic ad-hoc routing. It is designed primarily for mobile cell phones with GPS. GLS takes a decentralized approach by distributing the location information about the network through local decision making. All nodes act as location servers. Each node periodically publishes its location-to-ID mapping to a set of nodes in the network. It chooses these nodes based on a predetermined grid hierarchy. Nodes wishing to send to a destination use the same hierarchy to find the closest location server. The location server forwards the packet to the destination; upon receiving the packet, the destination node can reply with its coordinates.

There are several key differences between GLS and our approach. First, the coordinate system created in BVR is not amenable to a grid hierarchy<sup>2</sup>. Second, GLS has a different model for its nodes. The nodes are assumed to be mobile, GPS-enabled, computation and memory capable, and not power constrained. Third, location servers in BVR are the beacons themselves, not arbitrary nodes. Finally, in our location service, beacons do not forward queries but instead return the location of the requested node.

The Landmark Routing [26] system suggests a similar location service to GLS. The location service maps node IDs to coordinates. These coordinates, or landmark address, is a set of node identifiers of designated landmarks (in order of decreasing area covered by each landmark) closest to the node. Each node publishes its landmark address information to a location server by hashing its own ID and using the result as the landmark address for the location server. If no location server exists at that landmark address, the node with the closest landmark address acts as the location server. Similar to GLS, a node wishing to send a packet uses the same process (when the mapping was published) to find the location server. We adopt the same method of querying the location server and sending directly, however we add additional optimizations to combat look-up latency and node failure.

## 3. DESIGN

The design of BLS is driven by a set of basic assumptions. First, there exists a flat namespace that uniquely identifies individual nodes in the network and there is a topologically meaningful address space that identifies the *location* of a each node in the network. We do not assume that the address space can be hierarchically decomposed. Given these assumptions, our approach is to use a rendezvous scheme like the one shown in Figure 1. Nodes publish

<sup>2</sup>BVR does not lend itself to straightforward decomposition of space because BVR coordinates are not Cartesian. GLS depends on spatial decomposition.

their address to a well known location. When sending a message to another node, one looks up the current address at the rendezvous point, and then sends a message to the destination.

The simplest way to implement such a location service would be to use a single node as the *location server* responsible for maintaining coordinates. All nodes know the coordinates of this server and retrieve the mapping from this server directly. The main issues with this setup are limited memory and routing failure. Motes are heavily memory-constrained and, as the network size increases, the amount of state to maintain simply outgrows the amount of memory available on a mote. Furthermore, even if one node could hold all the state information, queries to the node from rest of the nodes in the network can quickly lead to congested paths that cause increased packet loss [7, 9] and reduced look-up/routing success.

Therefore it is desirable to distribute this database. We leverage the fact that in BVR there is a globally-known set of nodes that are used as anchors for routing in the network, referred to as *beacons*. All nodes in BVR have direct routes for reaching all beacons. These beacons are natural places for the rendezvous points.

In BLS, all nodes periodically publish their coordinates to a beacon (Coordinate Publishing §3.2), deterministically chosen using the node's ID (Beacon Resolution §3.1). When node A wants to reach node B, it uses the same algorithm to choose a beacon, based on B, and sends a look-up message to that beacon (Coordinate Lookup §3.3). The beacon replies to A, which can then send a message to B. Our system uses consistent hashing to choose beacons.

### 3.1 Beacon Resolution

Beacon resolution is important for the location service in two ways: for nodes to *query* the destination node's coordinates given a node identifier, and to *publish* their coordinates to beacons.

To avoid undesirable effects when adding new beacons we use a simple implementation of consistent hashing [12] for beacon resolution. This algorithm uses a simple hash function to map the node identifier as a key into a hash value. Each of the beacon identifiers are also hashed and then sorted in a logical ring similar to Chord [25]. The node identifier hash value is used to find the appropriate beacon responsible for that node: the first beacon identifier whose hash value is greater than or equal to the node identifier's hash value. If all hash values are less than the node identifier's hash value, then the mapping wraps around to the beacon with the lowest hash value. Simply stated, the first beacon hash value in the clockwise direction from the hash value of the node is the beacon responsible for storing that node's coordinates.

Our implementation of consistent hashing uses the beacon identifier (which is a different name-space from the node identifier) as the key. The advantage of hashing on the beacon identifier, rather than node identifier, is that upon the birth of a new beacon (in the case of a previous beacon failure), the beacon identifier does not change. We implemented a bully algorithm for electing a new beacon if one goes down. The new beacon simply takes the old beacon's place with respect to the beacon identifier. All of the nodes that mapped to the old beacon would map to the new beacon. Furthermore, this implementation has the advantage of minimal movement of mappings if new beacons are added dynamically (in addition to original beacons)<sup>3</sup>.

### 3.2 Coordinate Publishing

Nodes must publish their coordinates to beacons. Publishing, and sending updates, is triggered by three events: a timer expires, a beacon is added to the network, or the local coordinates change.

<sup>3</sup>Under a constant number of beacons, simple modulo and this implementation of consistent hashing have the same effect.

The first is motivated by the fact that the beacons keep the ID-to-coordinate mappings as *soft-state*. Furthermore, mappings may also be evicted from the beacon's cache. The expiration of the timer (with random jitter) ensures the local coordinates remain fresh with the appropriate beacon and do not synchronize with other updates. Updates are also sent when a beacon is added to the network a node may have to republish its coordinates to a new beacon. Finally, when a node's coordinates change, it must publish to the beacon to ensure correct routing. Since BVR is robust enough to handle slightly stale coordinates we can reduce the coordinate publishing rate to minimize BLS-related traffic.

The beacon processes updates of node mappings and stores these values in a small table. If the ID-to-coordinate mapping already exists, the coordinates are refreshed; otherwise an entry is replaced in the cache.

### 3.3 Coordinate Look-up

Coordinate look-up is the part of the location service by which a node desiring to send a message to a destination receives the destination node's coordinates given the destination node's identifier.

Beacons act as location service repositories. The basic look-up algorithm consists of the node sending a query to the appropriate beacon via the network. Since beacons are global landmarks of the network all nodes can automatically route to any beacon. The appropriate beacon is determined using beacon resolution described in §3.1. Once a beacon receives a query packet, it searches through its local cache and returns coordinates of a matching entry. If there is no matching entry, an error is returned to the sending node. Instead of the location service querying the beacon, the following simple techniques can be employed locally as an optimization to the basic look-up algorithm:

1. The *Neighbor Table* contains nodes which are considered neighbors by BVR's link estimation. By looking up the destination in the neighbor table, the sending node may short-circuit more expensive retrieval from a local cache (*i.e.* if implemented in FLASH memory) or the network (via beacons).
2. The *Local Cache* stores mappings between node coordinates and identifiers which have previously been queried from the beacons.

### 3.4 Eavesdropping

The purpose of using distributed look-up servers is to distribute traffic and processing overhead across the network. The more beacons we assign in a given network, the better the load will spread across multiple paths. However, increasing the number of beacons also increases the packet size, since BVR coordinates have the same number of components as beacons. Therefore we must be careful to balance the network load across more routes and maintain a reasonable packet size. Because this is not easy to configure and it is unclear how to optimize, we are not convinced that adding beacons is the correct solution.

With beacons as look-up repositories, we introduce a congested paths problem, whereby the route to each beacon becomes congested with look-up queries. This is a problem often encountered on the Internet, when a server with limited capabilities has an object of interest to many people and machines along the path to the server become congested. We address this problem with BLS eavesdropping: an intermediate node learns (caches locally) coordinate information from reply and publish packets when forwarding location service traffic. If a node has the answer to a query directed to a beacon it may send a reply on behalf of a beacon rather than forwarding the query along the path.

**Table 1: Parameter values tuned in simulation.**

Parameter	Value	Notes
Update Frequency	100sec	95% coordinate registration within 2 updates
Initial Query Timeout	5sec	Exponential Increase
Query Resends	1	More resends minimally effect performance
Server Reply Buffer Entries	4	Beacons may handle concurrent queries
Server Cache Size	#notes/#beacons	

The opportunistic caching takes advantage of reply and publishing traffic that traverses the nodes along the path to a beacon. This may be best for workloads where the number of destinations is larger than the local cache so that nodes are often performing look-ups to the beacons. Furthermore, if a node fails to receive a reply for a query, upon resend it may set a flag such that intermediate nodes do not intervene, and the reply may only originate from the beacon. The trade-off for eavesdropping is reduced query-reply latency and more evenly distributed location service traffic, versus “staleness” of coordinates and increased memory footprint.

BVR examined its routing success rate under varying node dynamics. In the evaluation nodes were randomly killed and the impact on overall routing success was measured. They discovered that even after more than 80% of the nodes die, the overall routing success rate is still greater than 95%. Similar results were obtained when beacons were randomly killed. This implies that BVR coordinates offer good “hints” as to the location of a node, so routing can still succeed even if components of the coordinate vector are incorrect (and suggests that coordinate “staleness” is handled quite well by BVR); BLS eavesdropping should show similar success under these conditions. Furthermore, our beacon election bully algorithm prevents routes from changing too much, so that BVR can still route packets with a high success rate.

## 4. IMPLEMENTATION

Our prototype uses TinyOS [1, 11] and approximately 3500 lines of NesC [10] code. BLS and BVR combined occupy approximately 3700 bytes of RAM. The local and beacon caches are small fully-associative cache containing node identifier-to-coordinate mappings. Once the cache is full, entries are replaced using a LRU-clock replacement policy. When a node performs a look-up and is waiting for a reply from the beacon, the message is queued to a send buffer in the location service, each with its own query timer. The query timer is responsible for checking if a previous query has completed and signaling the interface with the appropriate result for the layer above the location service (*e.g.* an application). If the query is still busy, it is assumed the query was lost and a new query is sent. Only one additional re-send is made before the interface signals failure. Each time a resend occurs, the time-out interval is increased exponentially.

## 5. EVALUATION

Our goal is to evaluate BLS in a simulation environment to verify its functionality, establish the necessary parameters for operation, and motivate potential improvements. We also use simulations to evaluate the scalability of our design. Then we deployed BLS/BVR on a sensornet testbed to further validate those results. On the testbed we perform two experiments: 1) to measure the location service performance given the parameters established in simulation, and 2) to compare the performance of BLS/BVR to two point-to-point routing schemes – *tinyAODV* and *VRR*. Finally, we

demonstrate an application running on our system and show some performance results.

### 5.1 Metrics

To evaluate our location service, we use the following performance metrics:

- **Location Service Traffic** BLS introduces additional network traffic due to publishing, queries, and replies. With this metric we evaluate the amount of overhead in the number of messages sent and received (including forwarding) by all nodes in the network. Our goal is to minimize this metric and distribute it evenly across all the nodes.
- **Total Hopcount Per Route Query** For our experiments, this measure is actually evaluated in the worst case. Because we do not use local caching, all send requests require a coordinate look-up before routing the data packet to the intended destination. The use of local caching and eavesdropping is to minimize the average hopcount per route query to an amortized cost similar to a normal BVR route query.
- **Location Service Success Rate** This is the number of replies divided by the number of queries, not counting resent queries, and duplicate or late replies. This metric illustrates the ability of BLS to supply the required coordinates (implies success in publish attempt, query, and reply) without routing to the destination.
- **Total Routing Success Rate** Finally, this metric shows, from the application’s perspective, the success rate when routing to a destination; it includes the success rate of BLS (publish, query, reply) and BVR (routing with given coordinates to the destination).

### 5.2 Methodology

All experiments for BLS are with look-ups in the local cache and neighbor table turned off; this represents the worst case for BLS because every request to send to a destination results in a look-up query. The cache at each beacon is sized such that all nodes which publish to that beacon will fit in the cache. This setup allows us to focus on the performance of BLS without the added complexity in dealing with caching policy or cache sizing. BLS performance can be arbitrarily improved by increasing the cache size, at the cost of memory footprint and look-up latency. For each experiment a warm-up period of several minutes allows the link estimators, BVR coordinate system, and node publishing to stabilize.

Our simulation results are generated using a TinyOS, bit-level radio simulator known as TOSSIM [17]. We use five network sizes (25, 50, 100, 200, and 400 nodes) with the following properties: constant number of randomly placed beacons, each network is routable by plain BVR with at least a 90% success rate, and each network has approximately the same node density. We use the lossy radio model to approximate testbed conditions.

Our testbed experiments consisted of 35 Berkeley `mica2dot` (Chipcon CC1000 433MHz radio, 4KB RAM, 128KB ROM) wireless sensor motes. The motes are spread across the northern half of Soda Hall at U.C. Berkeley with an approximate network diameter between 3 and 4 hops. All motes are connected via an Ethernet back-channel enabling remote programming, debugging, and logging. The send rate is the aggregate rate at which new queries are introduced into the network as a whole, measured in queries per second or  $q/s$ . Successful queries (which reach the beacon and reply with a positive result) result in a dummy data packet sent to the destination.

### 5.3 BLS with BVR: Results in TOSSIM

The experiments in this section are designed to measure BLS performance and overhead with respect to our evaluation metrics in the TOSSIM simulator. Due to space limitation we do not present the results for beacon election and several performance-tuning experiments we ran in simulation.

#### 5.3.1 Parameter Tuning

We tuned the system parameters in simulation until we achieved adequate performance to run in a testbed environment. Table 1 shows a short summary of the tuned parameter values in TOSSIM. These values were set to balance the performance with the memory footprint as we were limited to only 4KB of RAM on the `mica2dot` motes.

Memory constraints have a direct impact on the efficacy of both the routing layer’s performance as well as the performance of BLS. Increasing the number of beacons increases the amount of state each node must keep and increases the memory footprint. Furthermore, the performance of BVR directly affects the performance of BLS, as it sits directly above BVR.

#### 5.3.2 Simulation Performance

We ran BLS in simulation over various send rates and network sizes to determine the performance range of BLS. Specifically we were concerned in the third bullet in our list of evaluation metrics: *location service success rate*. Figure 2(a) was the first set of results collected from TOSSIM. For these experiments we turned off both eavesdropping and local cache/neighbor table look-ups. Therefore all send requests initiated a look-up query/reply in BLS in addition to a normal BVR data packet to the destination. For these experiments we attempted to improve the performance by adding more beacons for larger networks to spread out the message load more evenly across the network. However the results remained poor; the added BLS message load was too much for the network to handle. We ran the experiment for network sizes ranging from 25 to 200 nodes.

Figure 2(b) was collected with the same experimental setup but with the *eavesdropping feature turn on*. The results were dramatically different and shows the efficacy of eavesdropping in improving overall BLS performance. Without needing to increase the number of beacons larger than 8, we were able to route BLS query/reply packets with greater than 85% success rate for a 400 node network.

#### 5.3.3 Effects of Eavesdropping

We performed two experiments in simulation to demonstrate the effects of eavesdropping to reduce the BLS message overhead. One experiment used 35 nodes (similar to our testbed size), and the other experiment used 100 nodes. Both experiments were set up with the same parameters and sent a total number of messages proportional to the network size. Our hypothesis was that eavesdropping would reduce the total number of BLS messages, where BLS

**Table 2: Statistics showing effects of eavesdropping on BLS message overhead in simulation. This table shows the minimum, maximum, and mean number of messages sent per node in the network. When eavesdropping is enabled, the 35 node network sees 16.4% reduction in mean message overhead per node while the 100 node network sees a reduction of 26.8%.**

Experiment	Min	Max	Mean
35 nodes w/o eavesdropping	34	305	168.3
35 nodes w/ eavesdropping	33	295	140.7
100 nodes w/o eavesdropping	37	1171	347.2
100 nodes w/ eavesdropping	43	792	254.1

messages consist of the query and reply messages sent and received for each node. Figure 3(b) and Figure 3(c) show the CDF for the number of BLS messages sent by a percentage of nodes in the network for the 35 and 100 node networks. For each experiment, one can see that eavesdropping reduces the number of BLS messages. Table 2 gives the statistics showing the effects of eavesdropping on BLS message overhead in simulation. On average, the reduction of BLS messages was 26.8% for the 100 node network versus only a 16.4% reduction on the 35 node network. This demonstrates that while eavesdropping reduces the BLS message overhead per node, larger networks reap more dramatic improvements with eavesdropping than smaller networks.

### 5.4 BLS with BVR: Results in Testbed

The results with eavesdropping were so compelling that we decided the run all of our experiments on the testbed with eavesdropping enabled. Therefore, unless otherwise stated, eavesdropping is enabled for all of the testbed results presented in this section.

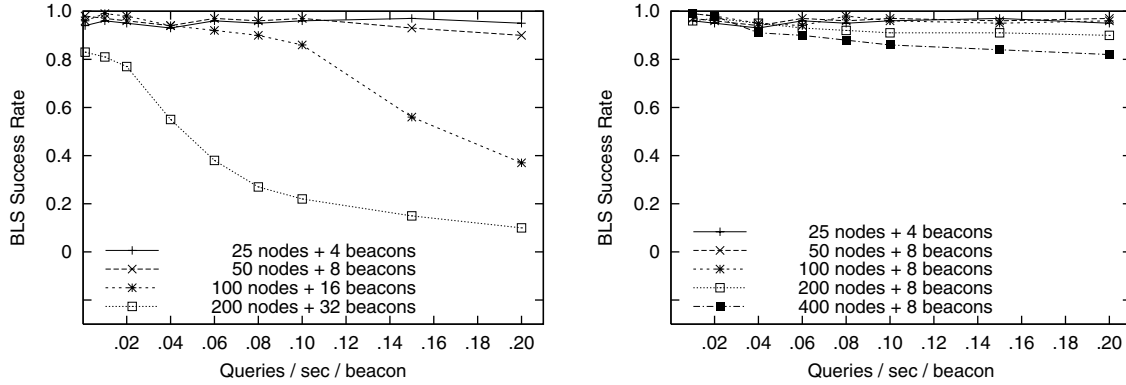
#### 5.4.1 BLS Success Rate

For these experiments, three independent trials were run each with the same send rate. The results of each trial for a particular send rate are averaged. Source and destination motes are chosen randomly. Figure 4 shows the results from these experiments. As shown in Figure 4, the location service can maintain a high success rate (greater than 85%—both routing and total) over a range of send rates ( $0.1 - 2q/s$ ). The total routing success rate will always be less-than or equal to the location service success rate because it includes the routing of data packets (which may fail).

BLS success rate and total routing success rate decrease as the send rate increases, due to increased load in the network. The load in the network is observed as “stuck” or “contentious” routes. Contention is caused by the send or forward interfaces of a node being busy when a packet needs to be sent. Stuck routes are those which cannot make progress; either the BVR coordinate system has broken down, or the particular node cannot route to any of its neighbors. In both cases, the packet is dropped. Also, due to the increase in network congestion, the latency (not shown) between sending a query and receiving a reply increases linearly with the send rate. These problems are part of the much larger question of resource sharing (e.g. congestion avoidance or control) in sensornets.

#### 5.4.2 Effects of Eavesdropping

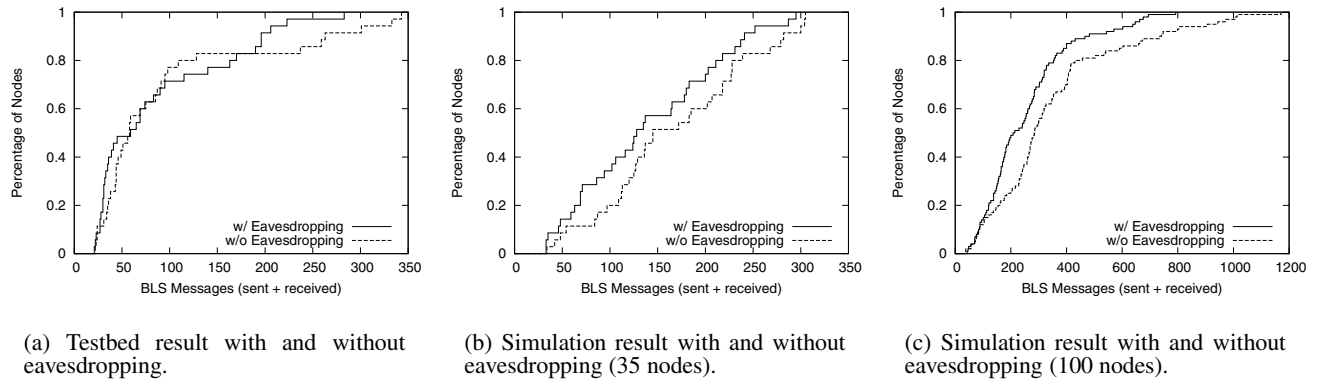
We wanted to show the effects of eavesdropping in the testbed as well, however, we only have a relatively small network to work with. We set up the same experiment as before and measured the BLS messages overhead. Figure 3(a) shows the CDF plot of the



(a) Results with eavesdropping disabled.

(b) Results with eavesdropping enabled.

Figure 2: BLS success rate results obtained in simulation.



(a) Testbed result with and without eavesdropping.

(b) Simulation result with and without eavesdropping (35 nodes).

(c) Simulation result with and without eavesdropping (100 nodes).

Figure 3: Effect of eavesdropping on BLS message overhead (queries and replies sent and received) in simulation and on the testbed

number of BLS messages sent by a percentage of nodes. This data backs up the findings given in §5.3.3. Table 3 summarizes the statistics of the eavesdropping effects on this small testbed network. Again, there is a slight improvement with eavesdropping: 11% reduction in the average number of BLS messages per node. We believe that if we were to run this same experiment on a larger testbed, we would expect to see similar results to those shown in Figure 3(c).

### 5.4.3 BLS Hopcount

We set out to evaluate hopcount distributions for BLS when routing to a destination node. This consists of queries and replies, to and from the beacon, as well as the data route to the destination. We wanted to determine the hopcount distribution for each type of BLS message. Furthermore, we wanted to understand how the overall hopcount changes as the send rate varies. Again, we ran these experiments with local-caching turned off so that each route request requires a coordinate look-up query to a beacon<sup>4</sup>. Also, to justify hopcount distribution, note that the network diameter is between 3 and 4 hops.

<sup>4</sup>Therefore, in the best case, the minimum total hopcount to a destination is 3 (*i.e.* one query, one reply, and one data route).

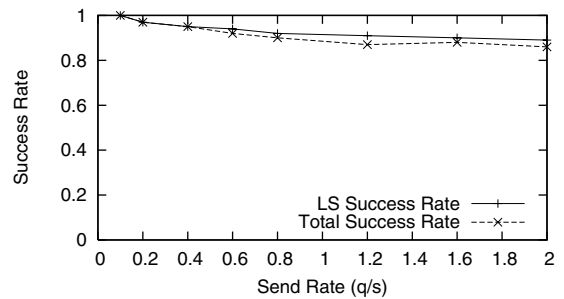
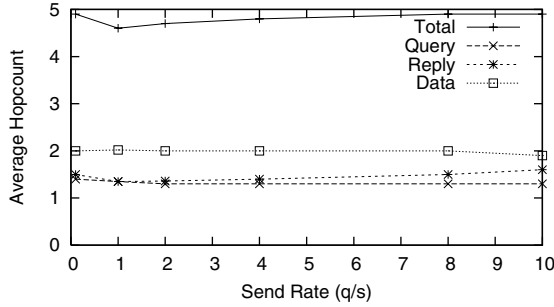
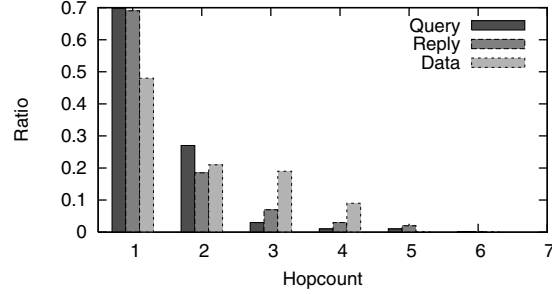


Figure 4: BLS query/reply success rate as well as the overall routing success rate from the perspective of the application (Total Success Rate). Results were obtained in testbed environment.



(a) Average hopcount distribution per send rate.



(b) Representative hopcount distribution run (10q/s). The hopcount ranges from 1–6 hops.

**Figure 5: BLS Hopcount distributions for different message types. These results were obtained in a *testbed* environment.**

Figure 5(a) shows the hopcount distribution over various send rates. Notice that the average hopcount for all three message types changes very little. BVR is fairly consistent in the routes it chooses to get to and from beacons. The overall average hopcount stays relatively consistent around 4.8 hops per route, indicating that on average an attempt to route to a destination takes 1.8 additional hops more than the absolute minimum of 3. Figure 5(b) shows the hopcount distribution for various BLS traffic for a specific, but representative, run on the testbed. The hop count values for each type (e.g. Query) are normalized to the total number of messages of that type. This figure shows approximately 70% of the Queries and Replies require only 1 hop.

## 5.5 BLS/BVR vs. tinyAODV

The goal of this experiment is to compare the performance of BVR (with BLS) to a sensornet implementation of Ad-hoc On-demand Distance Vector routing (tinyAODV). The code for tinyAODV has previously been tested in simulation; we deployed and debugged it on the testbed for this comparison. Despite that fact that tinyAODV uses flooding we included it in the evaluation because it is the Zig-Bee standard routing protocol.

### 5.5.1 Experimental Setup for Comparison

Our experiments for comparison between the two protocols consisted of routing packets between sources and destinations in such a way so as to highlight the actual routing performance and not the effects of caching policies in either protocol. Since the destination cache size in the AODV code was 7, we chose a destination set of size 7 and tested the routing success from a random source to a destination node. Clearly the cache size can be easily adjusted to fit the number of destinations you wish to set up routes to, but what-

**Table 3: Statistics showing effects of eavesdropping on BLS message overhead in the *testbed*. For this 35 mote deployment, we see an 11% reduction in average message overhead per node when eavesdropping is enabled.**

Experiment	Min	Max	Mean
w/o eavesdropping	32	343	96.9
w/ eavesdropping	21	283	86.2

ever value you choose is mostly inconsequential, as demonstrated below.

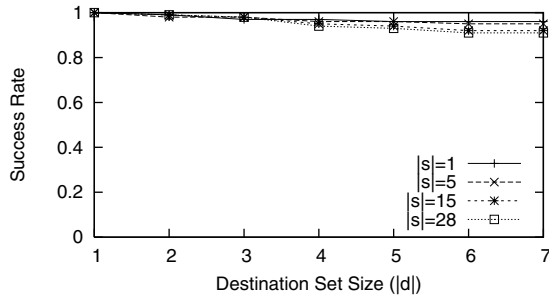
The components of the experiment consists of a destination set, a sub-destination set, a source set, and a sub-source set. The destination set  $D$  is a static set of nodes chosen randomly from the testbed. It is the set of all possible destinations where packets may be routed to. There are 7 nodes in the destination set (the same size as the cache in tinyAODV). The sub-destination set  $d$  contains randomly chosen subset of destinations from the destination set  $D$ ; the sub-destination set varies in size from 1, 2, . . . 7. The remaining 28 nodes are in the source set  $S$ . These nodes all generate traffic to be routed to a node in the destination set. The sub-source set  $s$  contains a randomly chosen subset of sources from the source set; the sub-source set varies in size ( $|s| = \{1, 5, 15, 28\}$ ).

For this experiment we attempted to route 20 messages from each source in the sub-source set to each destination in the sub-destination set at the send rate of  $1q/s$ . For tinyAODV, we did not count the first send request to each destination as this request was used to establish a route (by flooding) to the destination address in the send request packet. For this experiment, local caching in the location service *is* enabled with the cache size set to 7 entries. The success rate is determined by the total number of send requests made divided by the total number of unique packets received at the destination.

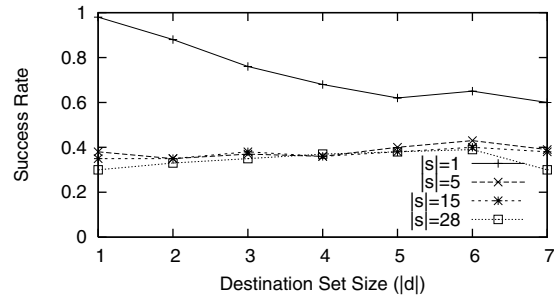
### 5.5.2 Results

Figure 6 shows the results of these experiments. The sub-destination set size is varied along the x-axis while the y-axis shows the total routing success rate. There are separate curves for each sub-source set size. From Figure 6(a) one can see that BVR with BLS is minimally affected by the size of either source or destination subsets, enabling it to route with greater than 90% success rate for all cases. Figure 6(b) shows the results for the same experiment using tinyAODV. Here we see that for a single source, the performance is the best over a range of destination subsets, however, when additional sources begin sending to the destination subset, the performance drops significantly.

The underlying reason for the bad performance of tinyAODV is the use of flooding to establish routes between sources and destinations. When we examined the cache hit rate in the local tinyAODV cache, we noticed the plot closely resembled the overall success rate. We also noticed that the success rate was above 90% when a route was in the local cache. This suggests that the routing success rate is driven by the efficacy of route establishment. Since routes

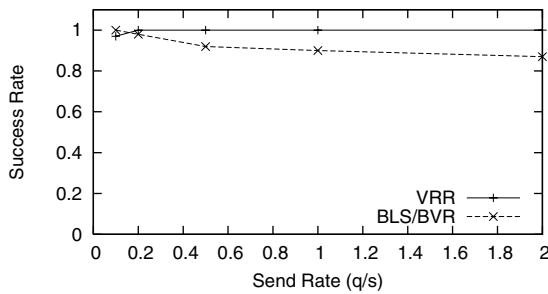


(a) Total routing success rates for BVR (with BLS).

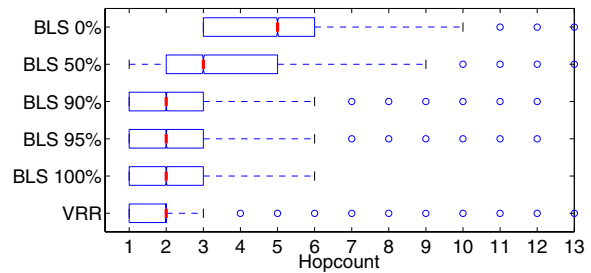


(b) Total routing success rate for tinyAODV.

**Figure 6: BVR with BLS and tinyAODV total routing success comparison while varying size of source set  $s$  and destination set  $d$ . Results were obtained in *testbed* environment.**



(a) Total routing success rates for BLS/BVR in comparison with VRR.



(b) Boxplots comparing the hopcount distribution of VRR with BLS/BVR.

**Figure 7: Two comparisons of BLS/BVR and VRR. Figure *a* shows the overall routing success over various send rates while figure *b* shows the hopcount distribution of VRR and of BLS/BVR with different local cache hit rates. Both results were obtained in the *tested* environment.**

are established with network floods, as you increase the number of sources trying to establish routes, you decrease the likelihood that a route will be successfully established. Unfortunately we could not include those graph due to space constraints.

Overall these experiments demonstrate that BVR with BLS allows for more flexibility in the behavior of applications requiring point-to-point routing. Using tinyAODV, applications are forced to send from one source to a small destination set; however, with BVR the source and destination sets can be small or large without fear of a performance penalty.

## 5.6 BLS/BVR vs. VRR

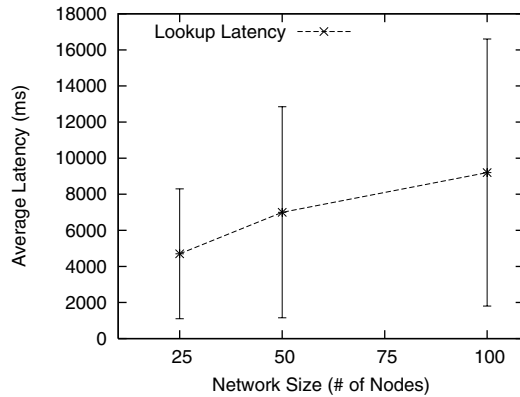
Virtual Ring Routing [4] is a point-to-point routing scheme that uses a distributed hash function to map individual node ID's to a logical ring of values. If a source node wishes to send a packet to a specified destination, the source node simply computes the hash value of the destination node and send the packet to the next logical successor in the ring that is closest to the computed value.

At startup each node computes its hash value and sends a join request to its nearest neighbors. After several minutes (convergence time is dependent on network size), the ring converges and every node in the network is becomes routable. The location service is

implicit in this design as the distributed hash function actually performs the mapping to the destination node when the hash value it computed.

For these experiments on the testbed we simply chose a random source and random destination over various send rates. If Figure 7(a) we plot the routing success rate of VRR and BLS/BVR as we varied the aggregate sending rate in the network. VRR outperformed our system with respect to overall routing success rate, but we were encouraged that they were indeed comparable.

Figure 7(b) compares the distribution of hopcounts between VRR and BLS/BVR. The data comprises about 35,000 packets for VRR and about 14,000 packets for BLS/BVR. The graph represents the distributions as *boxplots*. For each distribution the box extends from the 25th to the 75th percentiles. The thick mark shows the median, and the vertical lines show the points that are up to 1.5 times the inter-quartile distance away from the median. The circles represent outliers. Based on a testbed run of BLS/BVR with the local cache disabled (BLS 0%), we simulated different hit rates for the local cache, up to 100%. A BLS/BVR hopcount adds the hops taken for the lookup, the response, and the actual route to the destination, unless there is a hit in the local cache: in this case only the route to the destination is counted. VRR performed quite well, with the



**Figure 8: Demonstration of working distributed hash table on top of BLS/BVR, running in simulation. The results show the average latency and standard deviation for 25, 50, and 100 node simulations.**

median and 75th percentile at 2 hops. Some routes, however, were rather longer. We note that if the traffic pattern is such that the local cache hit rate is high, BLS/BVR’s performance is very close to that of VRR, even with the occasional lookups. The testbed in this experiment consists of 35 nodes, and the diameter is around 3 or 4 hops. We note that the worst case for both protocols was 13 hops, but the worst case for BVR alone (equivalent to a local cache hit rate of 100%), was much lower.

Since these are preliminary results there are still questions that need to be addressed in order to fully understand this comparison. For future work we intend to run BLS/BVR and VRR over larger networks in simulation and a testbed environment. VRR also keeps more state about the network than BLS/BVR. In fact, our system only maintains local information, such as nearest neighbors, BLS local cache mappings to coordinates, and other local state. Running a state comparison would also be beneficial for comparing the two systems. Finally, BLS is highly dependent on BVR for good performance. The performance would greatly be improved with a more efficient beacon-based routing layer.

### 5.7 Application: Distributed Hash Table

Finally, we implemented a “correctness” version of Chord [25] (*i.e.* only successor list, look-up, join, and stabilize; no finger table, etc.) which we call *tinyChord*. This implementation of Chord is not meant to be the most efficient or high performance. Instead, this experiment is meant simply to show a working application using the location service. As alluded to earlier, the location service is the interface to point-to-point routing using BVR. This experiment successfully demonstrates an application using the entire protocol stack. Our simple experiment consisted of three trials, each trial performed 100 DHT look-ups; only a single look-up in the DHT was active in the network at once. This resulted in a 99% DHT look-up success rate, and a look-up latency plotted in Figure 8 over various network sizes.

## 6. CONCLUSIONS

We presented the Beacon Location Service, BLS: a location service for Beacon Vector Routing (BVR), a point-to-point routing scheme for wireless sensor networks. BLS is an important part of BVR because it allows nodes to use unique identifiers for their destination, rather than keeping track of dynamic coordinate information. Our scheme uses the beacons as location servers which map

identifiers to coordinates. Source and destination nodes rendezvous at the beacon through queries and updates, respectively. Once the source node has the reply from the beacon it may route directly using BVR.

We tuned BLS in simulation and demonstrated its performance on large sensor networks. Furthermore, we have deployed BLS on a sensornet testbed to validate its performance. We compared BLS/BVR against two existing point-to-point routing solutions – tinyAODV and VRR on a sensornet testbed. Finally, we demonstrated a simplified version of Chord DHT using BLS.

From our experiments, we show that BLS/BVR performs well with random source and destinations over various send rates. Eavesdropping reduces the overall traffic in the network and improves the performance of BLS. When comparing BLS/BVR to tinyAODV, we find that it can sustain a very high routing success rate over various sizes of source and destination subsets, where tinyAODV cannot. We also found that despite running a non-optimized version of BLS we achieved comparable results to the more integrated VRR design. Overall we believe our contribution is in the design of a well-performing, simple, and robust location service design for a class of routing protocols that uses a set of anchors to create a virtual coordinate system over the network. With more efficient routing BLS would certainly continue to improve in performance.

There are several opportunities for future work with regard to the location service.

- Study the effects of caching, for both local node and beacons, under different organizations and policies, on update frequency, and location service success rate.
- Study the effects of mobility and churn on overall BLS/BVR performance as well as the effects of beacon placement on hopcount and locality. Moreover, we can study other optimizations to improve the performance of BLS/BVR under these conditions.

## 7. REFERENCES

- [1] TinyOS open-source website: <http://www.tinyos.net>, 2006.
- [2] ZigBee website: <http://www.zigbee.org/en/index.asp>, 2006.
- [3] S. Basagni, I. Chlamtac, V. R. Syrotiuik, and B. A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84, New York, NY, USA, 1998. ACM Press.
- [4] M. Caesar, M. Castro, E. Nightingale, G. O’Shea, and A. Rowstron. Virtual ring routing: Network routing inspired by dhts. In *Special Interest Group on Data Communications (SIGCOMM)*, pages 351–362. ACM, 2006.
- [5] S. Das, C. E. Perkins, and E. M. Royer. Ad hoc on demand distance vector (AODV) routing. Internet-Draft Version 4, IETF, October 1999.
- [6] M. Demirbas, A. Arora, and M. Gouda. A pursuer-evader game for sensor networks. In *SSS '03: Sixth Symposium on Self-Stabilizing Systems*, pages 1–16, 2003.
- [7] C. T. Ee and R. Bajcsy. Congestion control and fairness for many-to-one routing in sensor networks. In *Proceedings in the 2nd International Conference on Embedded Networked Sensor Systems*, pages 148–161. ACM, 2004.
- [8] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable

- point-to-point in wireless sensor networks. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI '05)*, pages 329–342. USENIX, 2005.
- [9] M. J. Freedman, E. Freudenthal, and D. Mazières. Democratizing content publication with coral. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI '04)*, pages 239–252. USENIX, 2004.
- [10] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*, pages 1–11, New York, NY, USA, 2003. ACM Press.
- [11] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, New York, NY, USA, 2000. ACM Press.
- [12] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663, May 1997.
- [13] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual ACM/IEEE international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, August 2000. ACM Press.
- [14] Y.-B. Ko and N. H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 66–75, New York, NY, USA, 1998. ACM Press.
- [15] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. In the proceeding of the 22nd symposium on the principles of distributed computing (podc). In *Geometric Ad-hoc Routing: Of Theory and Practice*, New York, NY, USA, 2003. ACM Press.
- [16] N. Kushalnagar, G. Montenegro, and C. Schumacher. 6LoWPAN: Overview, assumptions, problem statement and goals. IETF Internet-Draft, February 2007.
- [17] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: accurate and scalable simulation of entire tinyos applications. In *SensSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137, New York, NY, USA, 2003. ACM Press.
- [18] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *MobiCom '00: Proceedings of the 6th annual ACM/IEEE international conference on Mobile computing and networking*, pages 120–130, New York, NY, USA, August 2000. ACM Press.
- [19] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proceedings of the 4th Symposium on Networked Systems Design and Implementation (NSDI '07)*. USENIX, 2007.
- [20] J. Newsome and D. Song. In the proceedings of the first sensys (2003). In *GEM: Graph Embedding for Routing and Data-Centric Storage in Sensor Networks Without Geographic Information*, New York, NY, USA, 2003. ACM Press.
- [21] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, New York, NY, USA, 1994. ACM Press.
- [22] A. R. S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. In the proceedings of the first sensys (2003). In *Geographic Routing Without Location Information*, pages 76–88, New York, NY, USA, 2003. ACM Press.
- [23] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensor networks. *SIGCOMM Comput. Commun. Rev.*, 33(1):137–142, 2003.
- [24] F. Stann and J. Heidemann. Rmst: Reliable data transport in sensor networks. In *In the Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 102–112. IEEE, May 2003.
- [25] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.
- [26] P. F. Tsuchiya. The landmark hierarchy: a new hierarchy for routing in very large networks. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 35–42, New York, NY, USA, 1988. ACM Press.
- [27] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *In Proceeding for the First International Conference on Embedded Networked Systems (Sensys '01)*, pages 14–27, 2001.
- [28] J. Zhao and R. Govindan. Understanding packet delivery performance in dense wireless sensor networks. In *In Proceeding for the First International Conference on Embedded Networked Systems (Sensys '01)*, pages 1–13, 2001.