# Practical Connectivity-based Routing in Wireless Sensor Networks using Dimension Reduction

Shao Tao
School of Computing
National University of Singapore
Email: shaot@comp.nus.edu.sg

A. L. Ananda
School of Computing
National University of Singapore
Email: ananda@comp.nus.edu.sg

Mun Choon Chan
School of Computing
National University of Singapore
Email: chanmc@comp.nus.edu.sg

***Abstract*—Connectivity-based routing protocols provide an attractive option for point to point communication in wireless networks due to its potential for low routing overhead. However, when the entire hopcount vector is used to address each node, the communication and storage overhead in the packets are often so high that it is not feasible to implement existing connectivity-based routing protocols infeasible on resource-constrained sensor networks. In this paper, we apply the technique of dimension reduction, in particular Principle Component Analysis(PCA), to the hopcount vectors. Compared to the original hopcount vector, the embedding coordinates preserve the network geometry with much lower overhead, making their use much more practical on current sensor platform. Simulation results show that the coordinates computed by PCA can achieve higher packet delivery ratio, lower path stretch and shorter flooding range in local minimum cases. We have also implemented the PCA algorithm on MICAz motes and conducted experiments in a testbed containing 48 nodes deployed on two floors of an office building. With the use of $9$ landmark nodes and only 3 dominant components, the PCA coordinates can achieve $95\%$ of the delivery ratio obtained using full hopcount vector and maintain an low path stretch of $1.12$.**

## I. Introduction

Wireless sensor networks provide a flexible platform to support a variety of applications such as ecological monitoring[1], intrusion detection and security surveillance. The traditional on-demand ad-hoc routing protocols require a path discovery procedure for inter-node communication, which may lead to intensive bandwidth consumption in a resource-constrained sensor network. The geographic routing protocols achieve scalability by using node location for packet forwarding. However, accurate position information is hard to obtain and violation of the unit disk assumption in real deployments may result in persistent routing failures. In connectivity-based routing, a group of nodes are designated as the *landmarks* $L_i (i \in [1, k])$, propagating beacon messages to the network. Each node measures the distance to the $k$ landmarks to create a hopcount vector $H = [h_1, h_2, \ldots, h_k]$, where $h_i$ is the hopcount to landmark $L_i$. Routing is performed by treating the hopcount vector $H$ as the $k$-dimensional coordinates which can be accessed through a location service[2].

Routing performance is directly affected by the number of landmarks used during packet forwarding, creating a trade-off between control overhead and routing efficiency. Existing connectivity-based routing protocols require a large number of landmarks to attain good performance. For example, in BVR[3], 10$\sim$90 nodes are selected as beacons, out of which 10 beacons closest to the destination are selected for routing. Such large number of landmarks and routing components is very expensive for resource constraint sensor nodes, making such approach difficult to implement in practice.

In this work, we exploit the observation that as sensor networks are physically deployed in a 2D or 3D space, the intrinsic dimensionality of the topology is usually much lower than the number of landmarks. Therefore, dimension reduction technique[4] can be applied to extract the major axises of the connectivity graph and project each node on to a lower -dimension Euclidean space.

Our approach uses of Principal Component Analysis. Each node constructs a hopcount matrix, describing the pair-wise distance of the landmarks. The embedding algorithm apply *singular value decomposition* to the matrix to extract the most significant dimensions. The process outputs a transformation matrix that can capture the largest variance in the network topology. The inter-node distance is well preserved through the first few components in the coordinates and the embedding coordinates are more resilient to degenerate landmark set. Experiment results show that by compressing the hopcount vectors into three dimensional coordinates, the nodes can maintain $95\%$ of the packet delivery ratio (relative to using full hopcount vectors) with a path stretch of only $1.12$.

The rest of the paper is organized as follows. The related work on dimension reduction and connectivity-based routing protocols is given in Section II. The details of the embedding procedure are explained in Section III. The simulation results on large scale performance comparison are presented in Section IV. The testbed settings and experimental results are detailed in Section V. The conclusion and future works are given in Section. VI.

## II. Related Work

Connectivity-based routing protocols perform packet forwarding by minimizing the distance computed from the hopcount vectors between the nodes and the landmarks. Logical Coordinate Routing(LCR)[5] and Hop-ID Routing(HIR)[6] select the next hop as the node that can reduce the Euclidean distance to the destination according to their *logical coordinates* or *hop-ids* in the form of singleton Lipschitz

embedding[7]. The Beacon Vector Routing(BVR)[3] applies a similar technique relying on the weighted Manhattan distance metric.

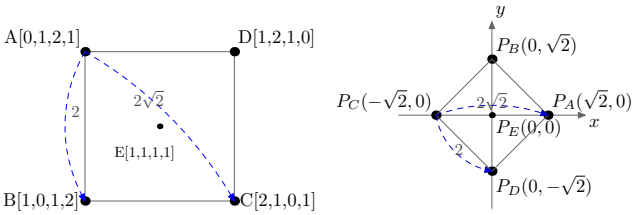
Landmark placement is crucial for network distance estimation[8] and connectivity-based routing. Clustered landmark nodes with low path diversity will generate highly correlated hopcount distance and become less effective to differentiate nodes at different locations. Zhang *et al.*[9] proposed a hierarchical landmark selection approach that groups the candidates into clusters. Landmarks are then chosen from both nearby and distant clusters, in order to improve the granularity of the embedding and capture the global network connectivity. Srinivasan *et al.*[10] conducted a performance comparison of different landmark selection methods based on randomization, clustering, hierarchical structure and min/max inter-landmark distance. The result shows that the heuristics based landmark selection algorithms provide nearly identical performance as random selection in general, while determining the critical landmark number required for a satisfactory embedding is a non-trivial task.

Using graph embedding technique to compute virtual coordinate has been applied for node localization in wireless networks. The NoGeo[11] protocol selects the perimeter nodes from the network boundary and allows each node to compute and refine its coordinates through iterative updates with its neighbors. Both Vivaldi[12] and GSpring[13] treat the network as a spring system, where each link has a normalized length. By applying the contraction and expansion rules according to the node distance, the network system converges to a stabilized state, where the node location can be determined independently. Although the iterative methods can achieve good results for network distance estimation, they suffer from long convergence time, require the network deployment to have some special structure and may be vulnerable to local minimum conditions.

Another popular application of network embedding algorithm to compute virtual coordinates is distance and/or delay estimation over the Internet. In [14], the distance between two nodes in the Internet is estimated by iteratively minimizing a potential energy function. Mao *et al.*[15] applies matrix factorization to obtain an incoming and an outgoing vector for each node, such that the network distance between two nodes is the dot product of the two vectors.

PCA is a popular technique used in data analysis[16] and image processing[17]. In the networking area, ICS[18] and Virtual Landmarks[19] are two algorithms that perform *Principal Component Analysis*(PCA)[20] to embed each node into a $k$-dimensional Euclidean space. The resulting coordinates are used to predict network latency.

Instead of predicting the network latency (and distance) over the Internet, the purpose of this work is to reduce routing overhead while achieving satisfactory routing performance in resource constraint sensor network. In addition, as the network latency may violate the triangle inequality property[21] assumed by the Euclidean embedding approaches, the network dimensionality often varies among different datasets. For connectivity-based routing, when all nodes are placed in a 2D or 3D Euclidean space, the geodesic distance measured in hopcount generally conform to the triangle inequality and symmetry properties. (This property always holds for unit disk graph.) Our approach is practical in sensor network and we are able to implement a distributed version of the PCA algorithm on MICAz motes and have conducted experiments on a medium scale testbed for performance evaluation.




(a) Connectivity Graph (b) Embedded Topology in $\mathbb{R}^2$

Fig. 1. An Embedding Example with 4 Landmarks: $A$, $B$, $C$ and $D$

## III. Protocol Design

In this section, we will present the details of the dimension reduction procedure with dimensionality analysis and highlight some implementation issues.

### A. Embedding with Dimension Reduction

In a network with $k$ landmarks, each node measures the minimum hopcount distance $d_i$ to each landmark. The hopcount vector $(d_1, d_2, \ldots, d_k)$ is utilized to address the nodes. In order to apply PCA, the vector distances between all pairs of landmark nodes are broadcasted to all nodes. Once a matrix containing all these vector distance information is obtained, each node independently performs the PCA-based dimension reduction process. The result provides nearly isometric embedding coordinates and ensures the transformed node distance in the embedded graph will approximate the original value. The new virtual coordinate of each node can be computed using the transformation matrix and the original hopcount vector.

As an illustration, assuming four landmark nodes $A$, $B$, $C$ and $D$ are connected consecutively as shown in Fig. 1(a), the hopcount vectors of the 4 landmarks are $[0,1,2,1]$, $[1,0,1,2]$, $[2,1,0,1]$ and $[1,2,1,0]$. These vectors form a pair-wise landmark distance matrix $M$ as given in Equation 1. We create a matrix $M'$ by normalizing each row of $M$ with a zeroed mean, such that $M'_{ij} = M_{ij} - (\sum_{x=1}^{k} M_{ix})/k$.

$$M = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 1 & 2 & 1 & 0 \end{pmatrix}, \; M' = \begin{pmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix} \quad (1)$$

$$M' = U \cdot S \cdot V^T, \quad \text{where}$$

$$U = \begin{pmatrix} -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}, S = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$V^T = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \end{pmatrix} \quad (2)$$

By applying *Singular Value Decomposition*(*SVD*) on the normalized distance matrix $M'$, $[U, S, V^T] = svd(M')$, the result matrixes are computed by Equation 2. The diagonal matrix $S$ contains the significance values $\sigma_1, \sigma_2, \ldots, \sigma_n$ for all components in a decreasing order. Matrix $U$ is the transformation matrix for the PCA embedding. The PCA embedding coordinates $P_A$ for landmark $A$ can be computed as in Equation 3, where $A'$ is vector $A$ normalized with a zeroed mean. The PCA embedding coordinates for $B$, $C$ and $D$ are $P_B = [0, \sqrt{2}, 0, 0]$, $P_C = [-\sqrt{2}, 0, 0, 0]$ and $P_D = [0, -\sqrt{2}, 0, 0]$. By taking the first two components in the result vector as the embedding coordinates $(x, y)$ in $\mathbb{R}^2$ space, the four landmark nodes can be plotted on a plane, which resembles their original structure as shown in Fig. 1(b). It is clear that the inter-node distance remains unchanged in this isometric embedding, $dist(A, B) = dist(P_A, P_B)$. For a normal node $E$ with a hop count vector of $E = [1, 1, 1, 1]$, its embedding coordinates can be computed from the transformation matrix $U$ as $P_E = E' \cdot U = [0, 0, 0, 0]$. The $\mathbb{R}^2$ coordinates of E is $(0, 0)$, which is consistent with the relative position of $E$.

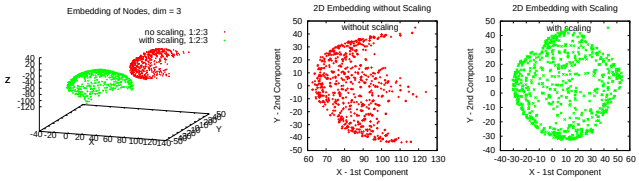
$$P_A = A' \cdot U = [-1, 0, 1, 0] \cdot \begin{pmatrix} -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{pmatrix}$$

$$= [\sqrt{2}, 0, 0, 0] \quad (3)$$

The zeroed-mean normalization is a necessary step, without which the first component in the output will represent the curvature of the data samples[22]. Incidentally, this normalization step is not performed in ICS. Fig. 2 provides an embedding example of 800 nodes. One way to evaluate the PCA generated virtual coordinates is by looking at the projection on the x-y plane which indicates that only the coordinates computed with scaling can correctly reflect the relative node position. As shown in Fig. 2(b), the projection without normalization fails to recover the original 2D topology.

### B. Critical Dimension

A key issue in the dimension reduction procedure is to determine the critical dimensionality required. The rule of thumb[22] is to draw a scree plot of the principal components and select the *90 percentile* – the first $k$ components that contribute to $90\%$ of the total variance. The variance contribution of the $i_{th}$ component is computed as $\sigma_i / \sum_{j=1}^{n} \sigma_j$, where $\sigma_i$ is a diagonal entry in matrix $S$. In the example at Fig. 1(b), the contributions of the first two components are both $0.5$.




(a) 3D Embedding (b) 2D: no scaling (c) 2D: with scaling

Fig. 2. An Embedding Example with and without Scaling

We use simulation to examine the critical dimensionality required. Nodes are deployed in a 2D and 3D area with a side length of $400m$ and the communication range is $30m$. The number of nodes and landmarks are varied in different scenarios. The distribution of principal components is displayed in the scree plot at Fig. 3. In Fig. 3(a), when all nodes are placed on a plane, each of first two components in the coordinates contributes to $23\% \sim 25\%$ of the total variance, while each non-intrinsic component contributes less than $7\%$. In Fig. 3(b), when the intrinsic dimensionality becomes three, each of the first three components contributes to $12\% \sim 15\%$ of the variance, while each of the rest contributes less than $5\%$.

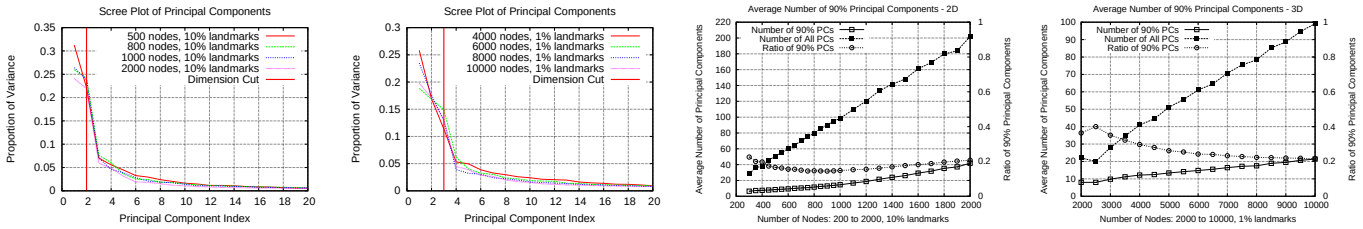
The 90-percentile components are illustrated in Fig. 3(c) and Fig. 3(d). When $10\%$ and $1\%$ of nodes are randomly selected as the landmarks for the 2D and 3D networks, $90\%$ of the total variance comes from the first $20\%$ of the components. Given that the actual network topology has a limited degree of freedom, the number of dominant components should remain relatively stable.

Based on the above results, it would seem that the critical dimension needed is fairly large and the potential for dimension reduction is limited. However, as our application is routing (rather than say data analysis or image processing), the need to include 90% of the total variance may be unnecessarily high. In fact, if the deployment is in a 3D space, as little as 4 principal components may suffice in some deployments.

In practice, we can resort to empirical measurements to assist in the dimension selection. As we will shown later in Section IV-D, the critical dimensionality for connectivity-based routing in 2D and 3D networks is only between 5 to 7 for uniformly random node placement, which is much smaller than the 20 components required for $90\%$ variance coverage in a network with 1000 nodes.

### C. Routing with Fallback

Assuming a source node $S$ with PCA coordinates $[s_1, s_2, \ldots, s_k]$ has a packet for destination $T$ at $[t_1, t_2, \ldots, t_k]$. Node $S$ will search among all its $m$ neighbors to find the next hop that can minimize the distance to $T$. During the greedy




(a) 2D: Scree Plot (b) 3D: Scree Plot (c) 2D: 90-Percentile (d) 3D: 90-Percentile

Fig. 3. Distribution of Principal Components in 2D and 3D Networks

**Algorithm 1** Routing Algorithm with Fall-back

Let $p$ be the packet from src $S$ to dst $T$
$p$ arrives at node $P$ with $m$ neighbors $N_i (i \in [1, m])$
Dst $T$'s coordinates $p.t = (t_1, t_2, \ldots, t_k)$ in $k$-dim space
Distance vector to $T$ is $p.dv = [d_1, d_2, \ldots, d_k]$
// update the minimum distance to $T$
**for** $i \leftarrow 1, k$ **do**
  **if** $dist_i(P, T) < p.dv[i]$ **then**
    $p.dv[i] = dist_i(P, T)$ ▷ update $d_i$ to avoid loop
  **end if**
**end for**
// search for the next hop
**for** $i \leftarrow k, 1$ **do** ▷ for each dimension
  $nexthop = null$
  **for** $j \leftarrow 1, m$ **do** ▷ for each neighbor
    **if** $dist_i(N_j, T) < p.dv[i]$ **then**
      $p.dv[i] = dist_i(N_j, T)$, $nexthop = N_j$
    **end if**
  **end for**
  **if** $nexthop \neq null$ **then**
    return $nexthop$ ▷ return the greedy neighbor
  **end if**
**end for**
flood the packet $p$ for $dist_k(P, T)$ hops. ▷ local minimum reached

forwarding step, upon reaching a local minimum node, a fall-back mechanism is activated as depicted in Algorithm. 1. For each destination $T$ in a $k$-dimensional Euclidean space, the packet $p$ contains a distance vector $[d_1, d_2, \ldots, d_k]$ to node $T$, where $d_i (i \in [1, k])$ represents the minimum Euclidean distance encountered from any visited node to $T$, computed from the first $i$ components of their embedding coordinates. If an intermediate node $P$ has no such a neighbor that can bring the packet closer to $T$ by $d_k$ in the $k$-dimensional space, $P$ will search in the space of dimension ($k$-1) by examining distance $d_{k-1}$. The fall-back procedure continues until the next hop is found or the current node $P$ is a local minimum for all $d_i$. If the fallback mechanism fails, the scoped flooding will be used as the last resort.

### D. Implementation Issues

*1) Data Structures:* As shown in Fig. 4, each node maintains three data structures: the hopcount vector *local_hv*, the list of neighbors *nbrlist* and the matrix containing the inter-landmark distance *hv_matrix*. *local_hv* stores the hopcount entries to all landmarks, where *parent_id* refers to the neighbor with the minimum hopcount to that landmark. *nbrlist* contains the neighbor records, in which the *last_heard* variable records the time when a neighbor's beacon is received. The PCA coordinates *loc* in a neighbor entry stores the 3D embedding coordinates and the neighbor's hopcount vector is stored in *hv*. *hv_matrix* keeps a record of the hopcount vectors for all landmark nodes in multiple rows, where each row contains the corresponding *landmark_id*, *parent_id* and its distance to other landmarks. The variable *parent_id* identifies the neighbor from which this landmark's hopcount vector is received.

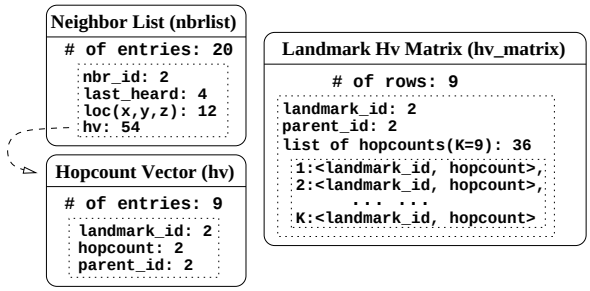



Fig. 4. Data structures in the nodes: *nbrlist*, *local_hv* and *hv_matrix*

*2) Memory Requirements:* The memory cost for the data structures in Fig. 4 is estimated in Table I. The local hopcount vector *local_hv* has maximally 9 entries in the format of [landmark_id, hopcount, parent_id], which consumes 54 bytes. Each neighbor entry uses 2 bytes for *id*, 4 bytes for *last_heard* timestamp and 12 bytes for PCA coordinates ($x$,$y$,$z$). The hopcount vector *hv* in the neighbor entry contains maximally 9 hopcount entries, each with a size of 6 bytes. The total number of bytes required for each neighbor is thus 72 bytes. Given a capacity of 20 neighbor entries, the *nbrlist* consumes 1440 bytes. The matrix of landmark hopcount vector *hv_matrix* has 9 rows, where each row contains a *landmark_id*, a *parent_id* and 9 hopcount entries of format [landmark_id, hopcount]. The size of *hv_matrix* is 360 bytes. The total memory cost for *nbrlist*, *local_hv* and *hv_matrix* is thus 1440+54+360 = 1854 bytes.

TABLE I
MEMORY COST(BYTES): 20 NEIGHBORS AND 9 LANDMARKS

| size of *id*, *hopcount* | *nbrlist* | *local_hv* | *hv_matrix* | total |
|---|---|---|---|---|
| 2 Bytes | 1440 | 54 | 360 | 1854 |
| 1 Byte | 880 | 27 | 180 | 1087 |

In a network with less than 255 nodes, it is adequate to use `uint8_t` type for node id and hopcount instead of

`uint16_t`, which will reduce the total memory cost from 1854 bytes to 1087 bytes and also reduce the control packet size during network initialization. This modification is applied in the experiments.

*3) Timer-based Operation:* The operation of the nodes are controlled by three timers: beacon timer, *nbrlist* refresh timer and the *landmark_hv* timer. The beacon timer allows each node to periodically broadcast a beacon packet containing its nodeid(*id*) and hopcount vector(*hv*). Once a beacon is received from a neighbor *nbr*, the node will either insert a *nbr* entry to *nbrlist* if it is from a new neighbor or update the *nbr.last_heard* timestamp for an existing neighbor. For each new landmark learned from that beacon, a new hopcount entry will be created in the local hopcount vector *local_hv*. For all entries in *local_hv* whose parent is *nbr*, if the corresponding landmark is not found in *nbr*'s beacon, those invalid entries will be erased. To avoid the count to infinity problem, a *clear_entry* message will be sent for each removed entry, such that the stale landmark records in the child nodes will be cleared accordingly. The details are presented in Algorithm 2.

**Algorithm 2** Actions triggered by beacon timer

```
1: each node broadcasts beacon every 10s. a beacon from nbr is received.
2: if nbr.id ∉ nbr_list then                          ▷ nbr is a new neighbor
3:     add nbr to nbr_list
4: else                                   ▷ update timestamp of an existing neighbor
5:     update nbr.last_heard to the current time.
6: end if
7: for each v ∈ nbr.hv do
8:     if v.landmark_id ∉ local_hv then              ▷ nbr has a new landmark
9:         insert [v.landmark_id, v.hopcount, nbr.id] to local_hv
10:    else if v.landmark_id = w.landmark_id and v.hopcount < w.hopcount
       - 1, where w ∈ local_hv then                  ▷ nbr has a lower hopcount
11:        set w.parent_id = nbr.id and w.hopcount = v.hopcount + 1
12:    end if
13: end for
14: for each w ∈ local_hv do                                  ▷ refresh local_hv
15:    if w.parent_id = nbr.id then
16:        if w.landmark_id ∉ nbr.hv then             ▷ w is a stale entry
17:            remove w from local_hv, forward clear_entry msg.
18:        else if w.landmark_id = v.landmark_id(v ∈ nbr.hv) then
19:            w.hopcount = v.hopcount + 1
20:        end if
21:    end if
22: end for
```

The *nbrlist* refresh timer is used to detect the missing neighbors due to link quality changes or hardware failure. A neighbor *nbr* will be removed if its beacon has not been received for 10 beacon intervals. For each local hopcount entry $v \in local\_hv$, if $v.parent\_id = nbr.id$, $v$ will be removed and a *clear_entry* messages will be sent. Similarly, for each row $r \in hv\_matrix$, if $r.parent\_id = nbr.id$, $r$ will be removed from *hv_matrix*.

The landmark nodes use *landmark_hv* timer to broadcast the *landmark_hv* message, which consists of the landmark *id* and *local_hv*. All nodes receiving the *landmark_hv* message will insert the landmark hopcount vector to *hv_matrix*. In order to reduce the network traffic, each node will only forward the *landmark_hv* messages received from the parent nodes in *local_hv*. Once *hv_matrix* and *local_hv* have obtained compatible entries, each node will be able to calculate its embedding coordinates independently.

## IV. Simulation Results

We first evaluate performance in simulation so results for large network can be obtained. In Section V, evaluation on a 48 nodes MICAz testbed will be presented. In the rest of this paper, we use *PCA* to denote the routing algorithm leveraging the PCA coordinates. For simulation performance comparison, we implemented Greedy, NoGeo, BVR, LCR and PCA in *ns-2* and a Java based simulator. The output of the Java simulator is verified with *ns-2*. Due to the limited scalability of *ns-2*, the simulation results presented in this section are obtained from the customized simulator. PCA employs the *Jama*[23] library to perform singular value decomposition.

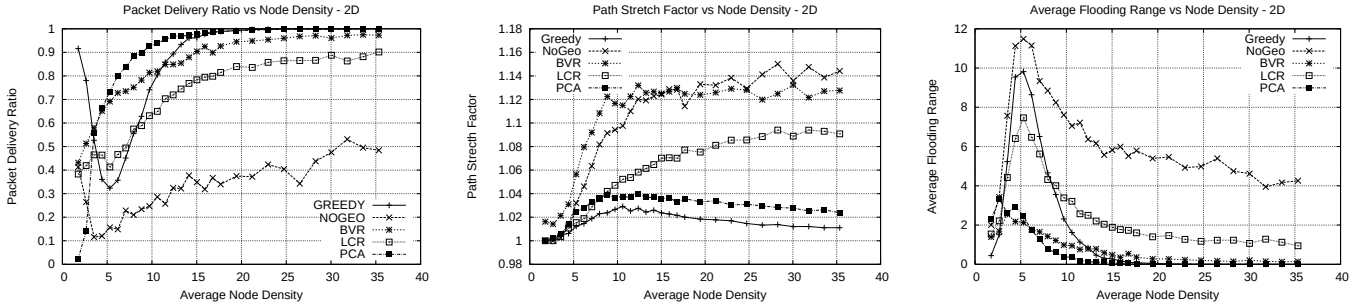
TABLE II
Simulation Parameters

| *Name* | *Value* |
|---|---|
| Deployment Space | 2D: 400m×400m<br>3D: 400m×400m×400m |
| Number of Nodes | 2D: 100∼ 2000, step = 100<br>3D: 1000∼ 20000, step = 1000 |
| Topology | 50 topos/density, uniform random |
| Radio Range | $30m$, unit disk model |
| Connections | 100 connections/topo |
| Routing Beacons | max = 10, candidates: $10\%$(2D), $1\%$(3D) |
| Routing Protocols | Greedy, NoGeo, BVR, LCR, PCA |

Configuration of the simulation parameters are listed in Table II. The simulation area is a square plane for 2D scenarios and a cubic space for 3D scenarios with a side length of $400m$. The number of nodes deployed varies from $100$ to $2000$ for 2D scenarios and $1000$ to $20000$ for 3D scenarios. With a radio range of $30m$, the node density range is $[1.77, 35.34]$. For each node density, 50 random topologies are generated and within each topology, 100 node pairs are chosen to be the source and destination. For BVR, LCR and PCA, $10\%$ and $1\%$ of the nodes are selected to be candidate beacons in 2D and 3D scenarios respectively. The maximum number of routing beacons is set to $10$ as suggested in [5] and [3]. We use the five protocols to route packets for each connection and measure the packet delivery ratio, path stretch and the average flooding range as the performance metrics. As the results for 3D scenarios show similar trend as the 2D ones with merely different magnitude, we will present the results for 2D scenarios only.

### A. Packet Delivery Ratio

In this work, packet delivery ratio is measured as the proportion of packets that can be successfully delivered without flooding. For PCA to be effective, the number of landmarks in the distance matrix should be larger than the dimensionality of the deployment space. In a sparse network with a node density less than $5$, the number of landmarks in a connected component is often below the threshold, leading to a performance inferior to BVR. The PCA coordinates achieve higher packet delivery ratio, as the node density grows above $5$.




(a) Packet Delivery Ratio (b) Path Stretch Factor (c) Average Flooding Range

Fig. 5. Routing Performance Comparison in 2D Space

At the medium node density range of $5 \sim 15$, the packet delivery ratio of PCA is $73.3\% \sim 97.9\%$, significantly higher than that of NoGeo and LCR. BVR employs a backtrack procedure that diverts the packet to the closest beacon from the local minimum position, obtaining a delivery ratio of $69.1\%$ to $90.4\%$. At the highest node density of $35$, the delivery ratios for NoGeo, LCR and BVR reach $48.4\%$, $90.1\%$ and $97.3$, while PCA has the highest delivery ratio of $99.9\%$, nearly equivalent to the performance of greedy forwarding with perfect position information.

### B. Path Stretch Factor

Assuming the routing protocol generates a path of $h_p$ hops and the shortest path has $h_s$ hops, the path stretch $\lambda$ is computed as $\lambda = \frac{h_p}{h_s}$. The protocol with a lower path stretch $\lambda$ achieves shorter routing paths and lower delay.

The greedy protocol achieves the lowest path stretch in all density ranges. The path stretch for PCA is comparable to that of LCR at node densities less than $8$, lower than NoGeo and BVR. As the node density increases above $8$, the path stretch of NoGeo, BVR and PCA starts to stabilize, while LCR's path stretch gradually increases. The converged path stretch values for NoGeo, LCR, BVR and PCA are 1.14, 1.09, 1.12 and 1.02. Thus, for high density 2D networks, the routing paths discovered by PCA is $7\% \sim 11\%$ shorter than the rest.

### C. Scoped Flooding Range

For NoGeo, BVR, LCR and PCA, the primary forwarding procedure may fail to deliver a packet due to anomalies in the coordinates, while scoped flooding can be invoked as a recovery step. As the network traffic grows exponentially during flooding, the protocol with a shorter flooding range introduces less duplicate packets to the network.

PCA and BVR obtain the lowest flooding range among all the evaluated protocols. At the node density around $5.3$, the average flooding range for all protocols reaches the peak as scoped flooding is frequently triggered. The maximum flooding ranges for Greedy, NoGeo, LCR, BVR and PCA are $9.8$, $11.5$, $7.5$, $2.1$, and $2.4$. As the node density increases, all protocols obtain higher delivery ratio and the flooding range starts to decrease. The converged flooding range values for NoGeo, LCR and BVR are $4.25$, $0.95$, and $0.14$, while the flooding range of PCA approaches the minimum value of $0$.

### D. Effects of Dimensionality

While utilizing more components in PCA coordinates provides better routing performance, it also brings higher demands for processing time and memory space in the packet header. Determining the critical dimensionality is important for obtaining a balance between routing performance and overhead. We use empirical results to derive the critical dimensionality value.

TABLE III
PERFORMANCE OF PCA AT CRITICAL DENSITY OF 5

| Metric | 2D Networks | | | | |
|---|---|---|---|---|---|
| Dimensionality | 2 | 3 | 5* | 10 | 20 |
| Delivery Ratio | 52.6% | 62.4% | 70.5% | 73.3% | 73.5% |
| Path Stretch | 1.04 | 1.03 | 1.03 | 1.02 | 1.02 |
| Flooding Range | 4.12 | 3.28 | 2.60 | 2.45 | 2.44 |

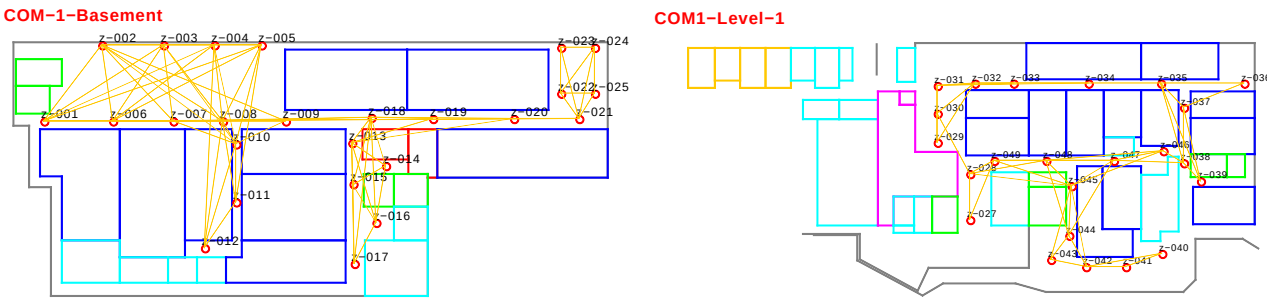
In general, when node density increases, the need for higher dimension reduces (Results are not shown due to space constraint). Therefore, the effect of dimension is more crucial at lower node density. The routing performance of PCA with node density 5.3 under various dimensionalities is summarized in Table III. We choose to highlight node density 5.3 because this is the minimum node density required for any geographical or connectivity based algorithm to work well. For higher node density, equal or less components will be needed.

As shown in the table, routing performance gradually improves as more dimensions or components are utilized. In this 2D deployment, the incremental improvement diminishes rapidly beyond a dimension of 5. As the components are sorted in their significance order, the subsequent components in the PCA coordinates will have even lower influence on routing performance.

To summarize, for 2D networks, the performance of PCA stabilizes once the dimensionality reaches $5$. For 3D networks, the critical dimensionality is around $7$.

## V. TESTBED RESULTS

To explore the practicality of the dimension reduction method for real sensor networks, we implemented the PCA




(a) Node z-001 to z-025 in Basement (b) Node z-027 to z-049 in Level-1

Fig. 6. Deployment Floorplan of the Testbed(inter-floor links not shown). Landmarks: 2, 8, 18, 21, 28, 31, 36, 38, 45

algorithm on MICAz[24] motes with 4KB RAM running TinyOS 2.0.2[25]. The singular value decomposition code is adopted from [26]. The experiments were conducted on a testbed deployed on two floors of an office building, as demonstrated in Fig. 6. The parameter settings are listed in Table IV.

TABLE IV
PARAMETER SETTINGS OF THE TESTBED

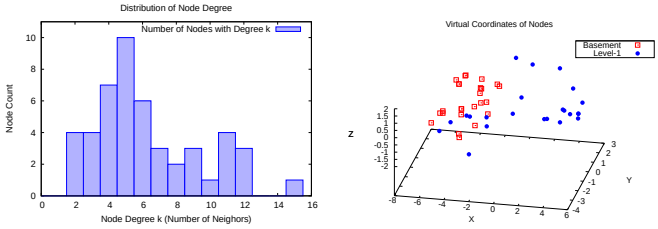
| *Parameter* | *Value* |
|---|---|
| Network Size | 48 MICAz nodes, 303 directional links |
| Num of Neighbors | avg = 6.31, min = 2, max = 15 |
| Transmission Power | 31 (max, 0 dBm) |
| 802.15.4 Channel | 26 (default) |
| Number of Landmarks | 9 (4 in basement, 5 in level-1) |
| Intrinsic Dimension | 3 (25 in basement, 23 in level-1) |


(a) z-003 at Basement (b) z-027, z-028 at Level-1

Fig. 7. Images of Nodes deployed in the Testbed

The testbed contains 48 nodes and 303 directional links. All nodes transmit at the maximum power level of 31. We deploy nodes z-001 to z-025 at the *basement* floor and nodes z-027 to z-049 at the floor *level-1*. Nine nodes were chosen as the landmarks, four in the basement and five in level-1. Fig. 7 gives a snapshot of the deployment. The distribution of node degree is displayed in Fig. 8(a), where the average number of neighbors at each node is 6.31.

Network initialization took five minutes for neighbor discovery and hopcount propagation. Once the landmark hopcount matrix and the hopcount vector are established, each node computes its coordinates individually as described in Section III-A. Only the first three components are used for the embedding in order to reduce the overhead in the packet header. The coordinates plotted in Fig. 8(b) show that nodes



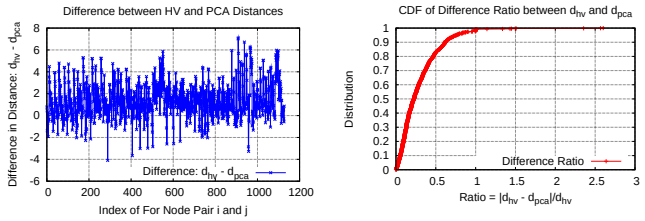

(a) Distribution of Node Degree (b) Network Embedding

Fig. 8. Node Degree and 3D Embedding Graph of 48 Nodes

on different floors clearly form two clusters, implying that the PCA coordinates can successfully retain the locality feature.

### A. Distance Metric Comparison

To compare the Euclidean distance computed from the hopcount vectors and the PCA coordinates, we calculate the inter-node distances in both ways. For two nodes $A$ and $B$ with hopcount vectors $[a_1, a_2, \ldots, a_n]$ and $[b_1, b_2, \ldots, b_n]$, the hopcount vector distance $d_{hv}$ is computed as $d_{hv} = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$. The PCA distance $d_{pca}$ between $A$ and $B$ is the Euclidean distance computed from their PCA coordinates $(x_A, y_A, z_A)$ and $(x_B, y_B, z_B)$. We plot the difference between these two distances $\delta = d_{hv} - d_{pca}$ in Fig. 9, where $50\%$ of the distances computed from the PCA coordinates are within a deviation of $0.25$ from the corresponding full hopcount vector distances.




(a) Difference: $d_{hv} - d_{pca}$ (b) CDF of Difference Ratio

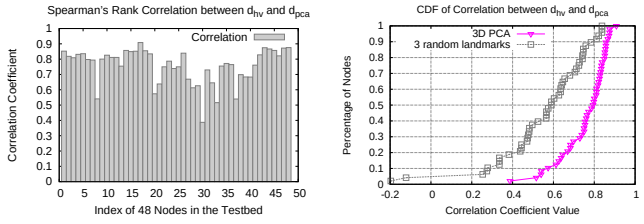
Fig. 9. Comparison between Distances: $d_{hv}$ and $d_{pca}$

We use the *Spearman's Rank Correlation* metric to quantify the effectiveness of using PCA coordinates to replace hopcount vectors. Given two ranking lists $X = [x_1, x_2, \ldots, x_n]$ and

$Y = [y_1, y_2, \ldots, y_n]$, the Spearman's Rank Correlation Coefficient $\rho$ between $X$ and $Y$ can be computed by Formula 4. The value of $\rho$ varies in the range of $[-1, 1]$. A higher coefficient value indicates a higher consistency between the two lists.

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}, \text{ where } d_i = x_i - y_i \qquad (4)$$

For each node $N_i$, we compute its hopcount vector distance $d_{ij}$ to each node $N_j (j \in [1, n])$ and form a distance vector $D = [d_{i1}, d_{i2}, \ldots, d_{in}]$. According to the distance $d_{ij}$, we assign a rank $r_{ij} (r_{ij} \in [1, n])$ to node $N_j$, such that if $d_{ij} < d_{ik}$, $r_{ij} < r_{ik}$. We use $R^i_{hv} = [r_{i1}, r_{i2}, \ldots, r_{in}]$ to denote the ranking list of hopcount vectors at node $N_i$. We compute another ranking list with the PCA coordinates and name it $R^i_{pca}$. For each node $N_i$, the correlation $\rho_i$ between $R^i_{hv}$ and $R^i_{pca}$ is computed as in Equation 4.




(a) Correlation at 48 Nodes (b) CDF of Correlation

Fig. 10. Spearman's Rank Correlation between $d_{hv}$ and $d_{pca}$

The correlation coefficient values at each node are depicted in Fig. 10(a). The correlation lies in a range of $[0.39, 0.91]$, with an average equal to $0.76$ and a standard deviation of $0.017$. As shown in Fig. 10(b), $80\%$ of the nodes have a correlation above $0.65$, indicating that the node locality characteristics estimated from the PCA coordinates is highly consistent to that of the full hopcount vectors using all 9 landmarks.

For comparison, Fig. 10(b) shows the CDF of distance correlation using 3 randomly chosen landmarks as is done on LCR (data is averaged of 5 different data sets). It is clear that with the same amount of overhead, the randomly chosen landmarks significantly deviates from the geometry of the original graph.

### B. Routing Performance

To compare the routing performance, we employ greedy forwarding to find a routing path between two nodes based on $d_{hv}$ and $d_{pca}$ respectively. We measure the number of paths discovered and the path length $L_{hv}$ and $L_{pca}$. The path stretch $\lambda = \frac{L_{pca}}{L_{hv}}$ will imply the performance degradation by using the lower dimensional coordinates instead of the full hopcount vector for routing. The results are provided in Table. V.

With 48 nodes in the network, we can form 2256 pairs of source and destination. The $d_{hv}$ distance metric discovered 1048 paths and the $d_{pca}$ metric discovered 998 paths. Only 581 of the paths are common to both metrics. In terms of packet delivery ratio, $d_{pca}$ found $95.2\%$ of the paths discovered by $d_{hv}$. The path length of applying $d_{hv}$ and $d_{pca}$ varies in the range of $[1, 11]$, where $d_{hv}$ has an average of $3.85$ hops. The average path length of $d_{pca}$ is $3.94$, a slight increment of $2\%$. The path stretch of $d_{pca}$ over $d_{hv}$ ranges from $0.17$ to $4$; thus, either metric has found some paths shorter than the ones discovered by the other. The overall path stretch factor of $d_{pca}$ is $1.12$, representing an average increment of just $1.3$ hops in the worst case.

TABLE V
ROUTING PERFORMANCE EVALUATION OF $d_{hv}$ AND $d_{pca}$

| | No. of Paths | Path Length | Path Stretch over $d_{hv}$ |
|---|---|---|---|
| $d_{hv}$ | 581+467=1048 | [1, 11], avg = 3.85 | 1 |
| $d_{pca}$ | 581+417=998 | [1, 11], avg = 3.94 | [0.17, 4], avg = 1.12 |

The path lengths are displayed in Fig. 11(a) and the routes discovered by $d_{hv}$ and $d_{pca}$ are marked in Fig. 11(b) and Fig. 11(c). The route map of $d_{pca}$ in Fig. 11(c) clearly indicates that nodes from z-013 to z-020 form a cluster inaccessible from the rest part of the network, causing most routing failures. This is due to the lack of resolution in the input hopcount vectors. It is possible to alleviate the problem by applying pairwise transmission power control[27] to create a small scale multihop topology in these nodes or introducing additional landmark nodes in this cluster. Meanwhile, the route map of $d_{hv}$ in Fig. 11(b) is less indicative for network performance diagnosis.

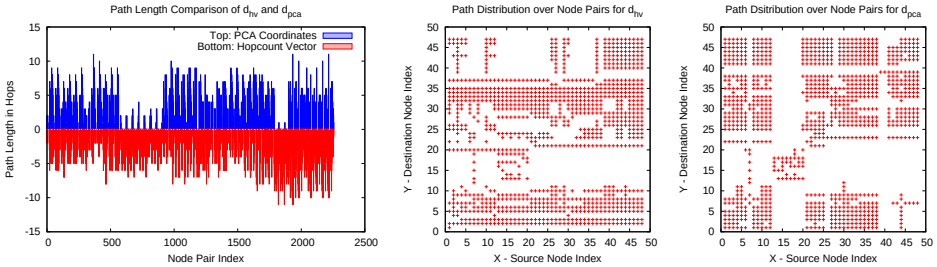
### C. Packet Overhead

The control overhead in the packets mainly comes from the hopcount vectors or coordinates used to address the destination. To use the full hopcount vector of the destination, each packet must carry the hopcount entries to 9 landmarks. Assuming that both *nodeid* and *hopcount* are of `uint_16` type, this constitutes $36$ bytes. The $d_{pca}$ metric relies only on the 3D embedded coordinates, which cost $12$ bytes – just one third of that in $d_{hv}$. It can be further reduced to $8$ bytes, if all nodes are placed on a plane. Although the PCA operation has to maintain a hopcount matrix of size $n^2$($n$ is the number of landmarks), the memory and computational overhead is only required once during the initialization stage. The memory space can be reclaimed once the procedure completes. Therefore, the computational overhead of the PCA coordinates is transient, while the storage and communication overhead of applying full hopcount vectors is persistent throughout the entire network lifetime.

In summary, using dimension reduction method, the routing protocol can work with only 3 components, while maintaining equivalent packet delivery performance with negligible path stretch overhead.

## VI. CONCLUSION

In this paper, we evaluated the technique of applying dimension reduction method for connectivity-based routing. By extracting the underlying dimension information from the sampled connectivity graph, the resulted coordinates can




(a) Path Length for Node Pairs (b) Paths found by $d_{hv}$ (c) Paths found by $d_{pca}$

Fig. 11. Distribution of paths discovered by $d_{hv}$ and $d_{pca}$. A + at position $(x, y)$ indicates a path is found from node $x$ to node $y$.

significantly improve routing efficiency. We implemented the PCA algorithm on MICAz nodes and conducted experiments on a medium scale testbed. With a 3D embedding of 9 landmarks, the distance computed from the virtual coordinates can closely approximate the distance computed by the full hopcount vector. The experiment results show that the dimension reduction algorithm can effectively reduce the routing overhead to make connectivity-based routing more applicable for real sensor network deployments, without compromising the routing performance significantly.

The current connectivity-based routing still leaves the following aspects for further investigation. A better distance metric for Lipschitz embedding may exist, which is more tolerant to local minimum conditions. An analytical approach is more desirable than empirical methods for evaluating the quality of various landmark sets. Embedding the network by the hopcount distance may generate unstable coordinates, as the hopcount vectors may vary due to the fluctuation of link quality. Higher preference should be assigned to more reliable links in order to reduce oscillation in the coordinates.

## REFERENCES


[1] G. Tolle, N. Turner, K. Tu, and P. Buonadonna. A macroscope in the redwoods. In *Proc. of the 3rd SenSys*, pages 51–63, 2005.

[2] J. Y. Li, J. Jannotti, D. Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *Proc. of the 6th MobiCom*, pages 120–130, 2000.

[3] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Cullerand S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensor networks. In *Proc. of the 2nd NSDI*, pages 329–342, 2005.

[4] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[5] Q. Cao and T. Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. In *Proc. of the 25th IEEE RTSS*, pages 349–358, 2004.

[6] Y. Zhao, B. Li, Q. Zhang, Y. Chen, and W. Zhu. Efficient hop id based routing for sparse ad hoc networks. In *Proc. of the 13th IEEE ICNP*, pages 179–190, 2005.

[7] G. R. Hjaltason and H. Samet. Properities of embedding methods for similarity searching in metric space. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(5):530–549, 2003.

[8] Y. Chen, K. H. Lim, R. Katz, and C. Overton. On the stability of network distance estimation. *ACM SIGMETRICS Performance Evaluation Review*, 30(2):21–30, 2002.

[9] R. Zhang, Y. Charlie Hu, X. Lin, and S. Fahmy. A hierarchical approach to internet distance prediction. In *Proc. of the 26th IEEE ICDCS*, pages 73–80, 2006.

[10] S. Srinivasan and E. Zegura. An empirical evaluation of landmark placement on internet coordinates schemes. In *Proc. of the 13th ICCCN*, pages 335–340, 2004.

[11] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *Proc. of the 9th MobiCom*, pages 96–108, 2003.

[12] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. of the 2004 SIGCOMM*, pages 15–26, 2004.

[13] B. Leong, B. Liskov, and R. Morris. Greedy virtual coordinates for geographic routing. In *Proc. of the 2007 ICNP*, pages 71–80, 2007.

[14] M. Costa, M. Castro, A. Rowstron, and P. Key. Pic: Practical internet coordinates for distance estimation. In *Proc. of the 24th ICDCS*, pages 178–187, 2004.

[15] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proc. of the 4th ACM IMC*, pages 278–287, 2004.

[16] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. Kolaczyk, and N. Taft. Structural analysis of nework traffic flows. In *Proc. of ACM SIGMETRICS*, pages 61–72, 2004.

[17] P. Xi, C. Shu, and M. Rioux. Principal components analysis of 3-d scanned human heads. In *Proc. of the 34th SIGGRAPH (poster)*, 2007.

[18] H. Lim, J. C. Hou, and C. Choi. Constructing internet coordinate system based on delay measurement. In *Proc. of the 3rd ACM IMC*, pages 129–142, 2003.

[19] L. Tang and M. Crovella. Virtual landmarks for the internet. In *Proc. of the 3rd ACM IMC*, pages 143–152, 2003.

[20] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1st edition, 1993.

[21] J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and embedding using small sets of beacons. In *Proc. of the 45th FOCS*, pages 444–453, 2004.

[22] J. Edward Jackson. *A User's Guide to Principal Components*, chapter 3, pages 72–75. Wiley-Interscience, 1st edition, 1990.

[23] J. Hicklin, R. F. Boisvert, and *et al. Jama: Java Matrix Package*. The MathWorks and NIST, http://math.nist.gov/javanumerics/jama/, 1.0.2 edition, July 2005.

[24] Crossbow Technology Inc. *MICAz - Wireless Measurement System*, 2008. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAZ_Datasheet.pdf.

[25] Tinyos 2.0.2. http://www.tinyos.net/.

[26] C. Bond. Singular value decomposition source code, 2000. http://www.crbond.com/download/misc/svd.c.

[27] S. Lin, J. Zhang, G. Zhou, L. Gu, T. He, and J. Stankovic. Atpc: Adaptive transmission power control for wireless sensor networks. In *Proc. of the 4th SenSys*, pages 223–236, 2006.